Studying the Impact of the Organizational Structure on Airline Operations Control

CHAPTER · DECEMBER 2015
DOI: 10.1016/B978-0-12-397041-1.00006-6

3 AUTHORS, INCLUDING:

António J. M. Castro
LIACC
30 PUBLICATIONS   71 CITATIONS

Eugénio Oliveira
University of Porto
315 PUBLICATIONS   1,734 CITATIONS

All in-text references underlined in blue are linked to publications on ResearchGate, letting you access and read them immediately.
Studying the Impact of the Organizational Structure on Airline Operations Control

Nuno Machado, António J. M. Castro, and Eugénio Oliveira
LIACC/NIADR, FEUP, DEI, University of Porto, liacc@fe.up.pt, http://www.liacc.up.pt/

Abstract. This paper introduces work practice modeling and simulation as a mean to assess and evolve the airline organizational structure performance. It departs from the empirical knowledge conveyed through interviews with airline operators and builds an analytical infrastructure geared towards evaluating the current and hypothetical organizational structures. To better reproduce the operational control challenges faced by airline companies it uses real pre and post operational data containing scheduled flights, delay codes and aircraft and crew rosters. By the end of the research study, the simulation of the same operational scenario across four distinct organizational structures demonstrated improvements up to 15% in disruption handling time and up to 21% in collaborator stress.

Keywords: disruption management, operations recovery, airline operations, multi-agent systems, simulation based

1 Introduction

An organizational structure might be regarded as a set of entities collectively collaborating and contributing toward one common goal. The employees working in 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, makes 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 paper is founded on such observations and aimed at proposing improvements to the operational control within a real airline company. To accomplish such aim, we had to use a real airline company as case study.
TAP, the major Portuguese air carrier, agreed to participate on such 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 on 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 empirical concepts. Finally, we collected a set of metrics, introduced organizational structure modifications and established a quantitative comparison among the latter.

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 at no avail.

Following this, to the best of our knowledge we were the first to simulate the Airline Operational Control Centre (AOCC) organizational structure in order to study its impact in 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 with disruption management.

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 [1].

The activities performed often produce goods, services or data. There are two different approaches when it comes to designing or improving systems: machine-centered and human-centered [2]. The former is usually accomplished through a business process reengineering approach [3] 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 [4]. Unlike the machine-centered approach, which neglects human communication, collaboration, workspaces, problem solving and learning; the human-centered approach analyze human activities, work processes or tasks, comprehensively and chronologically throughout the day [5].

The human-centered work system design approach is based on modeling and simulating work practices: what people actually do, rather their outcomes. This way, it is possible to understand the effects of human behaviors in different places and times, details often omitted in a product-oriented task analysis. In the end, besides the traditional system workflow, 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 \([6]\) 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 programing language, Brahms contains some pre-defined 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 behaviours;
- **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 address through the development of a visualization of the simulated airline.

### 2.2 Airline Operational Control Centre Organization

The main role of the AOCC is to monitor the conformance of flight activity according to the previously defined schedule. The occurrence of some unexpected events might prevent operations to take place as planned, such as aircraft malfunction, crew delays, crew members absence, etc.

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 \([7]\), there are three main AOCC organizations:

- **Decision Centre**: 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 Centre**: all roles share the same physical space and are hierarchically dependent of a supervisor. For small companies we have a Simple Hierarchy Organization. For bigger companies we have a Multidimensional Hierarchy Organization. Figure 1 shows an example of this kind of AOCC organization.

- **Hub Control Centre**: 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 Centre with an hub. If both the aircraft controller and crew
controller roles are physically outside the hub we have an organization called *Integrated Centre* with an hub. The main advantage of this kind of organization is to have the roles that are related with airport operations (customer service, catering, cleaning, passengers transfer, etc.) physically closer to the operation.

![Diagram of Integrated airline operational control centre](adapted from [8]).)

As mentioned, figure 1 shows the traditional *Integrated Operational Control Centre*. As previously stated, the AOCC is composed by groups of workers, each one with its own responsibilities. They must report their activity to a Supervisor, translating a two-level hierarchical system. Figure 1 also represents the activity time-window of the AOCC, it starts 72 to 24 hours before the day of operations and ends 12 to 24 hours after.

The roles more common in an AOCC are, according to Kohl [9] and Castro [7]:

- **Flight Dispatch**: prepares the flight plans and requests new flight slots to the *Air Traffic Control* (ATC) entities (*FAA* in North America and *EUROCONTROL* 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**: responsible for the unplanned services and for the short-term maintenance scheduling. Changes on aircraft rotations may impact the short-term maintenance (maintenance cannot be done at all stations).
– **Passenger Services**: decisions taken on 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. Typical this role is performed on the airports and for bigger companies is part of the HCC organization.

### 2.3 Disruption Management

Disruption Management [9], also known as Operations Recovery, is the process carried out by the AOCC when an unexpected problem prevents a flight to operate as planned.

The first overview of the state-of-the-practice in operations control centers in the aftermath of irregular operations was provided by Clarke [10]. In his study, besides an extensive review over the subject, he proposes a decision framework that addresses how airlines can re-assign aircraft to scheduled flights after a disruptive situation.

Currently, the most thoroughly analysis of the discipline is presented by Kohl et al. [11] where their conclusions are supported by the DESCARTES project, a large-scale airline disruption management research and development study supported by European Union.

Other authors propose more general perspectives regarding disruption management. Yu and Qi [12] 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 are not particularly detailed.

On the other hand, Ball et al. [13] 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 clear 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 based on different methodologies.

An in-depth and comprehensive review over the most relevant studies and methodologies used in disruption management is presented by Clausen et al. [14]. 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.

While 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 [15] and Liu et al. [16].

Moving to crew recovery, the majority of publications formulate the crew recovery problem under assumption that the flight schedule is recovered before the crew re-scheduling decisions are made, thereby following the hierarchical
structure of the disruption recovery in practice. These publications include Wei et al. [17], Guo [18] and Nissen and Haase [19].

For instance, from the list of authors presented in the last paragraph, Wei et al. [17] 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 [19] research is its founding on European reality. They propose a duty-based formulation for the crew recovery problem, which is especially well suited for solving the 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 [8] 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.

### 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 [8]. This is usually known as the Airline Scheduling Problem [20].

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 on 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 the disruption, i.e., 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 scenario we needed to know the entities involved on airline operations. Figure 2 clearly depicts those entities and their geo-location. Squares represent facilities and ellipses computerized systems. Table 1 describes each of the entities’ labels.

With a big picture of the current TAP organizational structure and its components, an in-depth understanding about 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 at each other. Human collaborators interact with the systems by filling forms or reading data. Decisions are carried out at the Operational Control Centre by the specialists and supervisors are required to approve those decisions.

Triggering any of the eight identified workflows is a pre-flight anomaly, e.g., lack of fuel, aircraft malfunction, mandatory security, etc. If the anomaly causes a departure delay, then it is recorded on TAP databases accompanied with a delay code. Just to provide an idea about the number of potential anomalies, the proprietary delay code list of TAP has more than 200 entries, while the IATA, international delay code list has around 80 anomaly types. Each anomaly is usually detected by an airline operator or system, 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 minutes 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 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 on TAP organizational structure are the Aircraft and Crew Specialists, located at the OCC. They must be aware of the problem in order to reason and find the best solution, e.g. replace the aircraft, delay the flight, etc. Figure 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 into the Aircraft Movement System and the informations is propagated instantaneously to the OCC. There, the Aircraft Specialist is hopefully paying
Table 1. Organizational structure concepts.

<table>
<thead>
<tr>
<th>Facilities</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>Crew Terminal</td>
</tr>
<tr>
<td>AP</td>
<td>Aircraft Parking</td>
</tr>
<tr>
<td>CI</td>
<td>Passenger Check-In</td>
</tr>
<tr>
<td>HCC</td>
<td>Hub Control Centre</td>
</tr>
<tr>
<td>LIS</td>
<td>Lisbon Airport</td>
</tr>
<tr>
<td>OCC</td>
<td>Operational Control Centre</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Computerized Systems</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AMS</td>
<td>Aircraft Movement System</td>
</tr>
<tr>
<td>CTS</td>
<td>Crew Tracking System</td>
</tr>
<tr>
<td>DOV</td>
<td>Flight Operations Portal</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Human Collaborators</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>as</td>
<td>Aircraft Specialist</td>
</tr>
<tr>
<td>cms</td>
<td>Crew Members</td>
</tr>
<tr>
<td>cs</td>
<td>Crew Specialist</td>
</tr>
<tr>
<td>fd</td>
<td>Flight Dispatcher</td>
</tr>
<tr>
<td>gs</td>
<td>Ground Supervisor</td>
</tr>
<tr>
<td>hs</td>
<td>HCC Supervisor</td>
</tr>
<tr>
<td>mss</td>
<td>Maintenance Services</td>
</tr>
<tr>
<td>os</td>
<td>OCC Supervisor</td>
</tr>
<tr>
<td>pss</td>
<td>Passenger Services</td>
</tr>
<tr>
<td>ss</td>
<td>Station Supervisor</td>
</tr>
</tbody>
</table>

attention to the screen and becomes aware of the problem. He reasons about 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 on the Operational Control Centre, the OCC Supervisor is required to ratify the decisions proposed by the Specialists, while the Hub Control Centre Supervisor uses the VHF radio again to communicate changes to the Ground Supervisor.

All the activities above require time to perform. TAP was questioned about the duration of such activities and, while 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.
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. Next, we will explain how Brahms greatly improved the experience of modeling the workflows on TAP. Finally, a subsection will also be dedicated to expose some aspects of a visualization module developed to allow a better understanding of the concepts being simulated.

4.1 Background and Overall Simulation Architecture

As referred during introduction, the main goal of this research study was to simulate the operational control on 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 at, thereafter, propose changes that would lead to more efficient activities and workflows.

The empirical observations listed in section 3, made us aware of the reality in 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 featured some business process reengineering capabilities. Meeting this requirements was *Brahms* [6] the Business Redesign Agent-Based Holistic Modeling System.

Although some theoretical information was already presented about *Brahms* on the *Background*, it worths point out some technical information about this system for the purpose of clarify certain options or side activities carried out along this study.

First and foremost, *Brahms* is a Multi-Agent System featuring the BDI, *Beliefs, Desires, Intentions*, architecture \(^1\). While these characteristics are not enough to distinguish it from many other simulation engines, *Brahms* is currently being developed by the *Brahms Team* at NASA Ames Research Center 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* was 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 our airline company but we decided to take the challenge. Another feature lacking in *Brahms* is the ability to visualize the concepts being simulated. While 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 academia or scientific tool, *Brahms* lacks a wide user community, where one may get models, code examples or help. Despite of that, it is thoroughly documented and their creators lead a discussion group to assist early-adopters.

As *Brahms* runs in a closed virtual machine, the first contact with the simulator community intended to inquiry about the possibilities of developing a visualization of a running simulation. While the primary approach would be to interpret a set of output files *post*-simulation, *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 4 depicts the components and architecture beneath the simulation portion of our study. While this section is not meant to be too technical, other

\(^1\) Software model that implements the main aspects of Michael Bratman’s theory of human practical reasoning.
aspects of figure 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 supports our infrastructure was Google
Chrome. Nevertheless, with the 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 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 communica-
tions 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. While the original Pro-
cessing 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, that is, start-
ing, pausing and stop the simulation. The visualization module will be described
on subsection 4.3, so at this point it just matters to understand we are in 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 non-blocking 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 of our simulations only work with this browser.

The NETTY component is too much 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 subsection will be solely concerned the next component, BRAHMS. As we referred, the selected simulation engine to implement our organizational structures 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.

<table>
<thead>
<tr>
<th>Reality</th>
<th>Brahms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Facilities and Locations</strong></td>
<td><strong>area</strong></td>
</tr>
<tr>
<td>ACT</td>
<td>LisbonAirportACT</td>
</tr>
<tr>
<td>AP</td>
<td>LisbonAirportAP</td>
</tr>
<tr>
<td>CI</td>
<td>LisbonAirportCI</td>
</tr>
<tr>
<td>HCC</td>
<td>LisbonAirportHCC</td>
</tr>
<tr>
<td>LIS</td>
<td>LisbonAirport</td>
</tr>
<tr>
<td>OCC</td>
<td>TapOCC</td>
</tr>
<tr>
<td><strong>Computerized Systems</strong></td>
<td><strong>object</strong></td>
</tr>
<tr>
<td>AMS</td>
<td>HCC_AMS</td>
</tr>
<tr>
<td></td>
<td>OCC_AMS</td>
</tr>
<tr>
<td>CTS</td>
<td>OCC_CTS</td>
</tr>
<tr>
<td>DOV</td>
<td>ACT_DOV</td>
</tr>
<tr>
<td>(AS)</td>
<td>Lisbon_AS (World)</td>
</tr>
<tr>
<td><strong>Human Collaborators</strong></td>
<td><strong>agents</strong></td>
</tr>
<tr>
<td>as</td>
<td>AircraftSpecialist</td>
</tr>
<tr>
<td>cms</td>
<td>CrewMember</td>
</tr>
<tr>
<td>cs</td>
<td>CrewSpecialist</td>
</tr>
<tr>
<td>fd</td>
<td>FlightDispatcher</td>
</tr>
<tr>
<td>gs</td>
<td>GroundSupervisor</td>
</tr>
<tr>
<td>hs</td>
<td>HccSupervisor</td>
</tr>
<tr>
<td>mss</td>
<td>MaintenanceMan</td>
</tr>
<tr>
<td>os</td>
<td>OccSupervisor</td>
</tr>
<tr>
<td>pss</td>
<td>PassengerMan</td>
</tr>
<tr>
<td>ss</td>
<td>StationSupervisor</td>
</tr>
</tbody>
</table>

Table 2. Concept mapping between reality and Brahms formalisms.
4.2 The Simulation Module

This section is focused on the BRAHMS component of the simulation architecture (figure 4). We will start by describing the “core”, that is how we used Brahms formalisms and programming language to implement the empirical observation exposed on section 3. As stated, the “JAPI” side is more technical, therefore we will not delve deep into it, solely pointing its use as mean 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 2, intends to clarify our approach concerning such mapping.

Besides presenting a number of associations, table 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 to the reality, the next two contain the name of virtual entities implemented on 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 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 1.1 shows an excerpt of the part of and path formalisms.

Listing 1.1. Excerpt of Brahms area and path definitions

```
area LisbonAirportAP instanceof AP partof LisbonAirport {
}
path LisAP_to_from_LisHCC {
  area1: LisbonAirportAP;
  area2: LisbonAirportHCC;
  distance: 600;
}
```

As it is clear, besides specifying what the area is, we may also specify the relation between areas (part of). Figure 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 seconds) between buildings or areas. The distance, on average, between the Aircraft Parking and the Hub Control Centre is, by feet, 10 minutes, so we must specify it as 600 seconds. As we will see later, if our agents use a vehicle, and thus only spend 2 minutes, the 600 seconds time may be overwritten in the move activity.
Moving on to the next portion of table 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 while they are physically inanimate, they may be used to reason over facts and therefore help humans take decision, e.g., 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 2 the airline systems were distributed across different locations.

Concerning the human collaborators, and 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 2, the “AircraftSpecialist” and the “CrewSpecialist” are members of the same group, the “OCCSpecialists”. This is a very powerful feature in Brahms because when 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 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 put those agents all reading the file and checking if they were responsible to trigger the next anomaly. While it would be correct to do that way, it would not be wiser because it would put too much strain on IO operations to read the same file (or checking 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 another topics worths 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. While 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 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 1.2 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.

Till now we 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 1.2. The Brahms human-centered paradigm

```java
// 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’s add a new one that will be used on our future organizational structure proposals: move between locations.

Brahms supports multiple activities, one of them is the Java activity. The Java activity will not be discussed into detail but it worths mention 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 1.2, which required 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 presence of an anomaly, would pick the radio or phone and communicate such 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 by 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 1.3 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 showed resembles a function where the “with” field is variable.

Still on listing 1.3 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, 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 1.3. The _Brahms_ communicate activity**

```java
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 uttermost 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 1.1), 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 1.3 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 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 2 as there is solely one, the TriggeringObjects that works in a similar fashion than TriggeringAgents.

Till now we described our approach in 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 next, 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 is roughly like calling a static java method.

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 and fourth to stream the ongoing events to the “websockets server”, see figure 4.

To conclude, it worths emphasizing that this section did not aim at thoroughly describe 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 paradigms fit the reality being modeled.

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 work and to better understand the proposed organizational changes.

As it was explained in subsection 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 by two distinct areas, the operational area, very similar to figure 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 to detect the anomaly, and its next activity is to notify the HCC Supervisor, then an arrow is displayed between him and the HCC Supervisor with a visual indicator of a radio communication.

Without being too much 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. While requiring an additional effort, such approach also allows for a flexible display.

Given the need to represent several distinct organizational structure, 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 words goes to the amount of Processing code required to implement solely one action or even 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 lengthly process, the reason why the visualization will always be less expressive than the simulation itself.

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 on the next section.

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 as post flight activity. The pre operational records included the scheduled flights, assigned aircrafts and assigned crew members. On the other hand, post operational 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 amount 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 occurred during flight handling, i.e., 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 worths emphasize the uttermost importance of using real data. In an organizational structure not all the business processes assume the same prevalence,
e.g. 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 week of activity. Although 7317 flights were scheduled to take place that week, due to data incompleteness, e.g. missing databases fields, table referential deficiency, inconsistent data, we were only able to input 1801 flights, 389 of which suffered anomalies.

5.2 Operational Workflow Transformations

The major goal of our study was to assess distinct airline organizational structures. Based on 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 on the previous subsection. When proposing organizational structure modifications we were cautious not to alter the inputs and outputs of the business process, i.e. never change the triggering and deciding agents.

Our first proposal (I) suggested the removal of the HCC Supervisor. After analyzing the actual sequence diagrams, we observed that he usually plays 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 was now required to go to the Hub Control Centre 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 II departed from proposal I and aimed at avoiding the Ground Personnel to go to the Hub Control Centre in order to reach the Aircraft Movement System. This way, we suggested to add 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 they did previously.

In our last proposal, III, we removed the usage restrictions on the AMS found on proposal II 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.

5.3 Metrics

Two metrics were used to assess organizational structure performance: overall disruption handling time and average collaborator stress. While 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, i.e., 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, e.g., the average. It measures the number of hours spent by a collaborator on 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 Results and Conclusion

Considering the scenarios depicted in the previous section, figure 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.

![Fig. 5. Overall disruption handling time (left) and average operator stress (right) across proposals.](image)

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% on collaborator stress.

Figure 6 compares stress across collaborator and proposal (chart column labels described on table 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 proposal “hs” was subtracted and “gs”, “mss” and “os” suffered the highest impact. On 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.

The above results proved that is possible to assess different organizational structure 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 from reality, there is always room for simulation evolution.

Finally, it is important to point out that, although we have used a particular airline company for our study and simulation, it is easy to adapt the system to other airline companies as well as to other domains, since the Brahms language and simulation system are flexible enough for that.

References


