Towards a Reliable Air Traffic Control (Short Paper)

Minh Nguyen-Duc, Zahia Guessoum
Computer Science Laboratory
Paris 6 University
104, av. Président Kennedy
75016 Paris, France
{minh.nguyen-duc, zahia.guessoum}@lip6.fr

Olivier Marin, Jean-François Perrot, Jean-Pierre Briot
Computer Science Laboratory
Paris 6 University
104, av. du Président Kennedy
75016 Paris, France
{olivier.marin, jean-françois.perrot, jean-pierre.briot}@lip6.fr

Vu Duong
Eurocontrol Experimental Centre
Centre du Bois des Bordes B.P. 15
F-91222 Brétigny-sur-Orge
France
vu.duong@eurocontrol.int

ABSTRACT

Since critical socio-technical systems include people interacting with equipments in workplaces, their intrinsic reliability problems have been concerned with both these two "actors". *Air Traffic Control* (ATC) is going to be such a system in which controllers use a large number of distributed software tools to provide safety ATC services. The reliability of these services relies on the availability of the various tools. Indeed, a partial failure of a tool in use can have tragic consequences. This paper presents a multi-agent approach to this problem. We propose an agent-based decision-aided system that helps controllers in using their multiple software tools in situations where some tools are not available due to technical incidents. We build and test our system in an ATC simulation environment, thus develop an *Agent-Based Simulation* (ABS). Experimental work has demonstrated the significance of our system to air traffic controllers.

Keywords

Agent-based decision-aided system, *Air Traffic Control*.

1. INTRODUCTION

Air Traffic Control (ATC) provides services whose objective is to direct aircraft on the ground and in the air. Its tasks are to keep any aircraft in a minimum distance from another aircraft, to ensure safe orderly and expeditious flow of traffic, and to give information to pilots, such as weather and navigation information.

1.1 New challenges

The forecast growth in air traffic requires the adoption of new technologies and related procedures enabling the safe and efficient provision of ATC services to a larger number of aircraft. This will be made possible by the use of software tools to support air traffic controllers (see the *First ATC Support Tools Implementation* (FASTI) program [10]).

Our work is concerned with the next generation of software systems for ATC. These systems will process some advanced flight data [6], which will be much more complicated than the currently used data. This will increase the controllers' capacity at the expense of a complexification of their task, but also will raise the technical issue of reliability.

On the "human" side, moreover, the integration of sophisticated tools in the controllers' daily work currently faces difficulties, like in any critical socio-technical system [8][11]. On the one hand, controllers have to change their usual, trusted working procedures. The safety and power that the tools are expected to provide will only become effective if the controllers are able to make the most of the functionalities of their tools. And this strongly depends on their familiarity with the tools. On the other hand, the controllers need to feel confident in the reliability of their software tools. In this paper, we report on an experiment with a *Multi-Agent System* (MAS) which aims at building confidence for air traffic controllers.

1.2 Our approach

We argue that the best way to prove a support system's reliability is to show that the ATC system, as a whole, can still provide full traffic control services when errors suddenly appear. Indeed, if the controllers are timely and adequately informed of the incidents, they can accordingly adjust their current control tasks and the following tasks. They can often manage without some of their tools. According to the *Guidance Material for Contingency Planning* [5], this kind of working mode can be seen as a type of *Degraded Mode of Operation*.

Therefore, there exists a need for a decision-aided system that helps the controllers in using their multiple software tools, particularly in situations where some tools are not available, or in other words when technical incidents happen. Our aim is to show that a suitable use of multi-agent technology can help in this respect. To develop such a system, we propose a solution based on software agents (see Section 3).

Cite as: Towards a Reliable Air Traffic Control (Short Paper), Minh Nguyen-Duc, Zahia Guessoum, Olivier Marin, Jean-François Perrot, Jean-Pierre Briot, Vu Duong, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008) – Industry and Applications Track*, Berger, Burg, Nishiyama (eds.), May, 12-16., 2008, Estoril, Portugal, pp. 101-104.

2. AIR TRAFFIC CONTROL

2.1 Future ATC system architecture

The current ATC system is airspace-based. The airspace is divided into many sectors whose size depends on the average traffic volume and the geometry of air routes. There are usually two air traffic controllers to handle the traffic in each air sector: an executive controller who communicates with pilots, and a planning controller who plans his colleague's work. Also, the sectors are regrouped into regions each of which is under control of a control center. For example, the Athis-Mons center is responsible for air traffic control in the Parisian region.

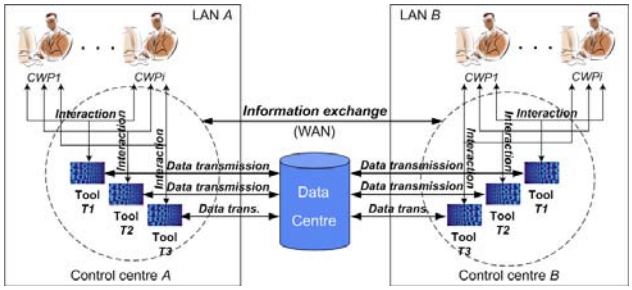

Figure 1. Basic future ATC system architecture.

The general structure of the ATC system sketched above would not be expected to change. But a new architecture would have to support the introduction of distributed software tools. The system will be distributed over local area networks (LANs) in each control center and the wide area network (WAN) between centers. As illustrated by Figure 1, different control centers are connected with a common flight data-processing center through the inter-center network (a WAN). In each control center one (or several) application server(s) host(s) the various software tools in use, *e.g.* *Medium-Term Conflict Detection* (MTCD), *Short-Term Conflict Alert* (STCA), *MONitoring Aid* (MONA), etc. [10] These application servers are connected with the *Controller Working Positions* (CWP) by means of a local network (LAN). The LANs of the control centers are connected via the inter-center WAN.

2.2 Critical situation

In this section, we present an example of a critical situation. We consider two (executive) controllers responsible for two neighboring sectors, at the border between the regions under control of two control centers (named *A* and *B*). They are often in handover situations, *i.e.* they have to transfer the control of aircraft flying from one sector to the other. Occasionally, a network failure occurs at the time when potential conflicts appear: *B* is disconnected from the flight data-processing center. This failure makes the MTCD in *A* unable to detect aircraft flying from the region under control of *B*. However, it still correctly detects the conflicts that only concern the aircraft flying in the region under control of *A*. So we can see it as "locally available". We suppose that the controller in *A* is already aware of the unavailability of his MTCD but does not know its "local availability". He has to detect himself all the potential conflicts, *i.e.* he verifies and follows all the aircraft he suspects. However, since MTCD is still locally available, such an exhaustive verification will unnecessarily increase the controller's workload. In fact, he can still rely on the results given by MTCD for the local conflicts.

3. OUR MULTI-AGENT SYSTEM

3.1 Objectives

To fulfill the need presented in the previous section, a decision-aided system for air traffic controllers is needed. Its missions are to communicate with the controllers, to inform them of the environment state and to show them information of tools' availability. More ambitiously, the decision-aided system would be endowed with the capacity to propose corrective actions to be performed following technical incidents.

Besides, this system helps with mitigating the effects of software faults in a distributed environment. It monitors software components which run on different machines, and keeps an eye on the interactions between the users (*i.e.* the controllers) and these components. To this end, it also has to be distributed. It observes complex data (*e.g.* air traffic data) at the input and output of each computation module of any software tool. More importantly, this system builds up confidence for users of a safety-critical software system. In consequence, it has to guarantee a safety level with respect to the services it offers. All its monitoring services have to run in real-time so that it can inform the users of some change of the software system's state as soon as it happens. Moreover, information it provides need to be not only concise but also adequate, in such a way that the users can determine exactly what to do in response to this change.

In view of these requirements on the decision-aided system to be developed, we propose a MAS solution.

3.2 Agent Design

Since our agents have to take care of the monitoring of software tools and of the communication with the controllers, we design different kinds of agents to perform these two common tasks. We currently use three monitoring agents for each tool (*i.e.* three sentinel agents), and assign an assistant agent to each controller.

1 *Data sentinel agent*: observes the input and output data of a specific software tool and communicates with other agents in order to discover data.

2 *Computation sentinel agent*: observes the input and output data of a specific software tool and communicates with other agents in order to discover computation faults.

3 *Middleware sentinel agent*: receives from the middleware the notifications of faults related to a specific software tool.

4 *Assistant agent*: communicates with other agents in order to determine the automated tools' availability, and informs its controller of this availability; it tracks the actions performed by its controller via his user interface (as

explained in 5.2); additionally, it proposes corrective actions to be executed after technical incidents.

3.3 Distribution of our MAS

We install in a control center a group of coordinated agents that are distributed over the whole corresponding LAN. The agents on the various LANs also communicate over the inter-center WAN, making up a global MAS. Each local group of agents is composed of assistant agents and of monitoring agents. We associate an assistant agent with each *Controller Working Position* (CWP). Each tool instance is observed by monitoring agents. Please note that monitoring and assistant agents may be hosted on any of the machines, or even on additional independent network nodes, as long as they retain the capacity to display information on the controller's screen.

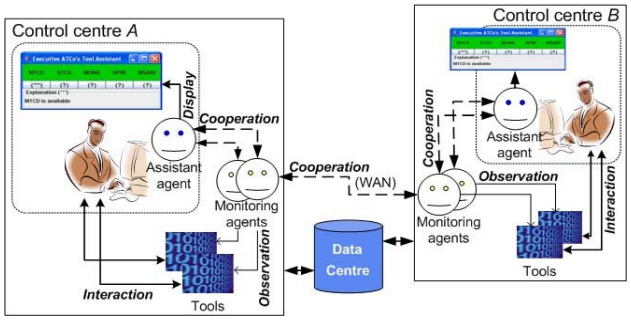

Figure 2. Monitoring and assistant agents.

When an incident occurs, the related tool's monitoring agent first discovers the critical situation by using the data it gathers from the tool's input/output, as well as the information it receives from other monitoring or assistant agents. Then, it transmits information about the tool's state to the assistant agents of the CWPs that use this tool. These assistants display green/yellow/red flags on their controller's screen, thereby indicating the tool's total/partial availability, together with the relevant information, possibly the corrective actions to be performed.

4. SIMULATING OUR SYSTEM

Any novel application to a critical system like ATC has to be tested in simulations before its real world implementation. We hence build and test our MAS into a simulation environment, thus develop an *Agent-Based Simulation* (ABS). We employ the eDEP platform (*Early Demonstration & Evaluation Platform*) [4], which offers not only realistic air traffic data but also a distributed simulated ATC environment. The support tools for air traffic controllers, *e.g.* STCA and MTCD, are implemented in eDEP as independent components which can run on different machines. We also use the DimaX platform [2], which helps with developing reliable MASs.

The integration of our DimaX agents and eDEP components follows the FIPA *Agent Software Integration Specification* [7]. The DimaX platform already includes a generic wrapper agent ready to provide any other agent (*e.g.* a monitoring agent or an assistant agent) with services which allow this latter agent to connect to software components. Special wrappers are then built by extending the generic one. They need to be hosted on the same machine as the components they "wrap".

Figure 3. `CWP_Assistant`'s user interface.

Based on the agent model discussed in 3.2, as a first step we install three monitoring agents for each of software tools, *i.e.* `XXX_DataSentinel`, `XXX_ComputationSentinel`, `XXX_MiddlewareSentinel`, and two wrapper agents, *i.e.* `XXX_ObservationWrapper`, `XXX_GeneralWrapper`. These special wrappers respectively provide XXX[1] observation and general-purpose services to the three other XXX_agents.

We endow the CWP with a `CWP_Assistant` which communicates with other agents in order to determine the automated tools' availability, and shows this availability in its user interface. Figure 3 illustrates the `CWP_Assistant`'s user interface. It uses green/yellow/red flags to show the status of the various tools that run on the LAN. The `CWP_Assistant` observes the controller's actions through observation services provided by a `CWP_InterfaceWrapper`.

5. AN EXPERIMENTAL SCENARIO

5.1 Objective and setup

The first tests of our agents on the ABS, which aim with demonstrating the usefulness of our MAS to the air traffic controllers, runs on the following connected machines: two client machines hosting two CWPs for two controllers belonging to two different control centers (named *A* and *B*), two tool servers hosting two MTCD instances for the two control centers, a data server simulating the common flight data-processing center.

5.2 Scenario

We would like to present here one of the experimental scenarios. We are in a handover situation (as described in 2.2): there are aircraft flying from the region under control of center *B* to the one under control of center *A*. At first, all machines run smoothly and are fully connected. The assistant agents display green labels indicating that the software tools are working at full capacity. The controller in *B* (called *CB*) then makes a flight data change request. Due to some accident, control *B* has been disconnected from the flight data-processing center. Consequently, this request is not sent to the data center.

Now, *CB*'s assistant agent tracks the data change request issued by *CB*, and then notifies the MTCD data sentinel agent in *B* of this request. This agent in its turn informs the MTCD data sentinel agent in *A* through the simulated WAN connection. This second monitoring agent discovers that no such flight data change was received from the data-processing center. This also means that the flight data concerning an

[1] XXX stands for the tool name, *e.g.* MTCD or STCA.

aircraft which is controlled by center *B* are no longer accessible from *A* and therefore unusable for conflict detection. In consequence, the assistant agent of the controller in *A* displays a yellow flag, informing his controller that the software tool is only available locally, *i.e.* it only gives correct results for aircraft under control of center *A*.

Knowing this, the MTCD data sentinel agent in *A* signals back to the monitoring agents in *B* that there was on its side a flight data change request which was not taken into account. This agent notifies the *CB*'s assistant agent of this incident. Finally, the *CB*'s assistant agent then displays a red flag, informing his controller that MTCD is now unavailable (as illustrated by Figure 4).

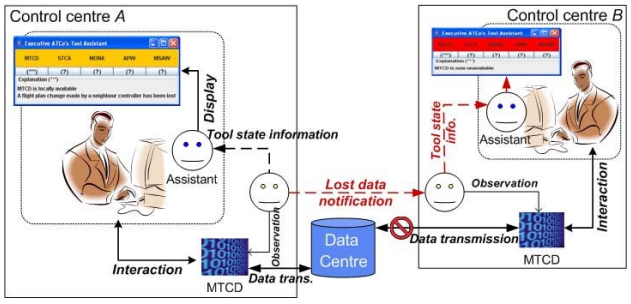

Figure 4. Agents reaction to a network failure (control center *B* is disconnected from the flight data-processing center).

6. VALIDATION

We have used our ABS to demonstrate typical scenarios to experts including both researchers in the field of ATC and professional air traffic controllers. The first series of demonstration sessions was addressed to researchers from Eurocontrol (*European Organisation for the Safety of Air Navigation*). They questioned the kind of information exchanged by agents. They particularly sought to understand the advantages of this exchange in comparison to a simple duplication of lost data. Furthermore, they stressed the need of the involvement of operational points of view in the validation process. The second series of experiments was therefore concerned with professional controllers. At first, they concentrated on the `CWP_Assistant`'s interface and recommended many modifications for it. Although they admitted that adequate information of tools' unavailability would be important for them (if there was any), they still found it difficult to accept eventual technical incidents, to think of incidents and to discuss solutions.

In order to merge the two different visions for our MAS solution to the reliability problem in ATC, we mixed researchers and operational operators (*i.e.* controllers) together in the third series of experiments. With the help of the researchers, the controllers concentrated on the information they require in each situation and gave valuable recommendations for each of scenarios. For example, concerning the one we have presented in Section 5, they suggested that although this scenario was only related to MTCD, in such situation of network failure, all the other tools (*i.e.* STCA, MONA, APW, MSAW) would find themselves in the same state as MTCD.

7. CONCLUSION AND FUTURE WORK

This paper describes the way in which a MAS can help in mitigating the effects of software malfunction in a complex critical system and building confidence for its users, *i.e.* air traffic controllers. Because of safety restrictions, experiments on real traffic control are not allowed. Therefore, we have developed an ABS, by using eDEP, an ATC simulation platform, and DimaX, a multi-agent platform, following the FIPA specifications [7]. Some experiments with ATC specialists, particularly with professional air traffic controllers, have been performed on this ABS.

However, the agents themselves, like any supplementary layer added to a system, bring their own liability to fault. A natural extension of the present work will be to set up mechanism for ensuring a degree of fault-tolerance at the agent level, which would be of a computational, domain independent nature. The possible techniques would include adaptive replication [9] and exception handling [3].

8. REFERENCES

[1] Avizienis, A., Laprie, J-C., Randell, B. 2004. Dependability and Its Threats - A taxonomy. IFIP Congress Topical Sessions.

[2] DimaX project team. 2007 *http://www-poleia.lip6.fr/~guessoum/DimaX/index.html*.

[3] Dony, C., Knudsen, J. L., Romanovsky, A.B. and Tripathi, A. 2006. Advances Topics in Exception Handling Techniques. In Lecture Notes in Computer Science, vol. 4119.

[4] eDEP project team. 2007 *http://www.eurocontrol.fr/projects/edep/*.

[5] European Safety Programme. 2007 Draft of the Guidance Material for Contingency Planning. Technical Report, EUROCONTROL.

[6] EUROCAE project team. 2006 Flight Object Interoperability Proposed Standard (FOIPS) Study. Technical Report, EUROCONTROL.

[7] FIPA. 2001 FIPA Agent Software Integration Specification.

[8] Gregoriades, A., Sutcliffe, A. G. and Shin, J. E. 2002. Assessing the Reliability of Socio-Technical Systems. Proceedings of the 12th Annual Symposium of the International Council on Systems Engineering (INCOSE 2002).

[9] Marin, O., Bertier, M. and Sens, P. 2003. DARX - A Framework for the Fault-Tolerant Support of Agent Software. ISSRE'03, Denver.

[10] Petricel, B. and Costelloe, C. 2007. First ATC Support Tools Implementation (FASTI) Operational Concept. Technical report, EUROCONTROL.

[11] Reiman, T. and Oedewald, P. 2006. Organizational Culture and Social Construction of Safety in Industrial Organizations. In Nordic Perspectives on Safety Management in High Reliability Organizations: Theory and Applications.