



## 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.

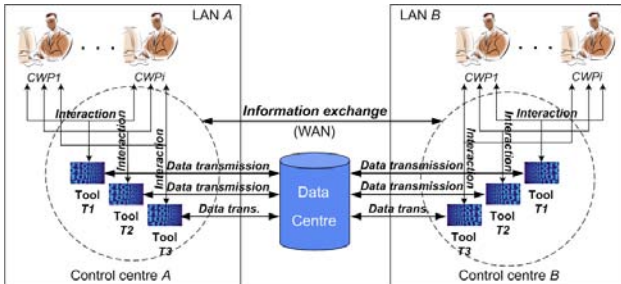


Figure 1. Basic future ATC system architecture.

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* (MTCDD), *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 MTCDD 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 MTCDD 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 MTCDD 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 MTCDD 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.

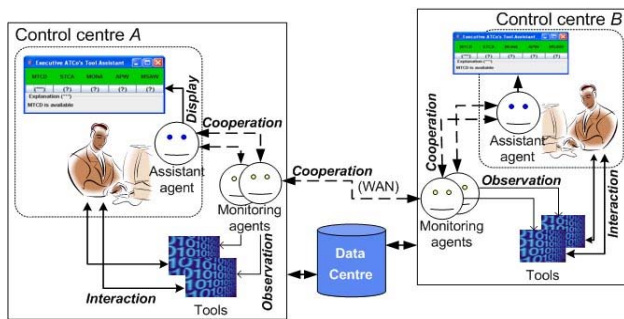


Figure 2. Monitoring and assistant agents.

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*<sup>1</sup> 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

<sup>1</sup> XXX stands for the tool name, e.g. MTCD or STCA.

