CHAPTER 6

Example: Interface for Air Traffic Controllers

6.1 Introduction

This chapter introduces a single large example of using the properties and architecture which have been discussed in earlier chapters. The example chosen is an Air Traffic Control (ATC) Support System. The concrete meaning of the abstract properties introduced in earlier chapters will be discussed in that context, showing how one property interacts with other properties. The relevance of this to the design is shown by examples, which are followed by a discussion of possible architectures for the ATC Support System.

A realistic example of this kind is inherently complex. Different individuals within the overall system potentially fulfill multiple roles. Both real time and safety-critical aspects of the system need to be considered. In this respect the example provides a challenging test for the merits of the design approach proposed in this book. It also serves to illustrate situations in which considerations of the functional domain may need to over-ride user interface engineering considerations.

6.2 The Air Traffic Service

The global task of Air Traffic Management is provided by individual national Air Traffic Services containing many hundreds of people carrying out support, maintenance, technical, controling and other activities. The major real time activity in this overall system provides Air Traffic Control services to aircrew (pilots of civil, military and private aircraft) during flight planning and en route.

The main purpose of the Air Traffic Control system is to ensure flight safety: aircraft must be able to fly from take-off to landing without fear of collision. Given this strict constraint the ATC system aims to offer optimal flight paths to the pilots who are the ATC users: aircraft should be able to take off when planned, fly at a speed desired by the operator at an altitude suitable to the desired flight path – all dependent on other traffic, weather conditions and constraints offered by the particular type of aircraft being used.

C. Gram et al. (eds.), *Design Principles for Interactive Software*

In order to achieve these goals, the Air Traffic Control Centres involved in the flight planning and en route control of aircraft need to collect considerable quantities of data about each planned flight – time and place of departure, time and place of arrival, proposed route, height, speed etc. Based upon this data the duty Air Traffic Controllers (ATCos) make decisions about the need for route alteration due to planned traffic, informing the pilots when accepting the flight plan before take off.

The most important part of Air Traffic Control is exercised while aircraft are en route. Changes in weather, other aircraft emergencies etc. all have to be dealt with in real time: duty controllers assisted by an ATC Support System monitor the minute-to-minute situation in the air, giving advice and requests to aircrew to avoid conflicts between aircraft flight paths.

The system being considered in the remainder of this chapter is the ATC Support System needed to help the human controllers manage the vast amount of data available and needed at some time or other in order to make optimal decisions while carrying out controling activities.

The order of presentation of the requirements from the support system takes into account some simplification necessary for the purposes of this example. A design would normally begin by reviewing controllers' activities in an existing system in order to permit a direct comparison to be made between the new design and existing practice. For brevity this is omitted and an abbreviated description provided.

The complete ATC Support System interacts with many individuals fulfilling a variety of different roles. This example concentrates on the tasks performed by and for the en route controllers themselves. In practice, however, each individual controller must adopt many roles over time – sometimes even more than one at once – advising pilots, ensuring safety, communicating with other controllers about air traffic movements etc. These multiple roles are indicated in the descripton of controllers' tasks given below.

The descriptions given make use of a simple framework frequently employed in software engineering – first describe the goals and then the inputs and outputs available to assist in achieving the goals.

6.2.1 A Controller's Tasks

Flight Management

The basic task of an en route controller (an ATCo) consists of monitoring flights in progress. In an ideal world this would mean checking that every aircraft follows its pre-planned (accepted) route. Controllers, however, have to face perturbations as well as poor planning. As soon as a potential conflict (which may lead to a dangerous situation) occurs, the controller must take action to reroute aircraft to avoid the conflict. In order to do

that, controllers often have to choose an altered (sub-optimal) route. This means that they have to build (or ask the Support System to build) a new subset of routes. Once these are determined to be satisfactory the controller must then undertake the correct actions (e.g. requesting change of course to pilots) to adopt the revised plan.

Control Co-ordination

No one controller can manage all airspace. The world's air space is divided into regions roughly in line with national boundaries or other major geographic division. Within each region the sizes of individual control sectors are designed as far as possible to even out controller load. Each controller on duty monitors all flights originating in, passing through or terminating in the single sector assigned – while the aircrafts are in that sector. The size of control sectors is such that most aircraft in transit (i.e. not merely carrying out local flying) will usually cross several sectors between take-off and landing. If the controller of the originating sector for a flight takes an action to modify the planned route then controllers of all subsequent sectors will need to deal with the modified route. Such changes are, of course, cumulative from sector to sector as modifications are found to be necessary. This introduces a new task for controllers: co-ordination and negotiation with other controllers either in the same Control Centre or in a neighbouring one. To simplify this co-ordination, airspace is generally organized into airways so that flights crossing sector boundaries do so only at pre-defined points. Negotiations between sector controllers are then about the time and altitude of such a crossing rather than its position.

Focusing on this en route Air Traffic Control and supposing that take offs and landings are managed in dedicated sectors (usually referred to as Terminal Control Areas or Terminal Control Zones), it may be assumed for the purposes of this example that a controller's duties consist of these two subtasks: negotiating with other controllers and managing flights within the sector – which includes monitoring and building new trajectories. In practice in a busy Air Traffic Control Centre these two subtasks are often performed by a pair of controllers working together.

6.2.2 Available information

The data used for monitoring airspace is synthesized from different sources. First, flight plans are used: though the static information they provide may become obsolete, they make it possible to make assumptions about the future movements of a plane. Then, radars provide real time information (refreshed every few seconds) about the positions and altitudes of aircrafts in flight. Every aircraft carries a device called a transponder that uniquely identifies it, so that radar information can be correlated with flight plans.

Finally, even though this is still under development at the time of writing, the existence of a data link between every aircraft and the ground will be assumed.

In addition to this information about flights, the system also has knowledge about the geography of the sector: its borders, the airways in it, and a number of pre-defined 'waypoints', used as references to build trajectories and to communicate with crews and other controllers. Finally, the pieces of information entered by a controller when dealing with aircraft may be made available to other controllers during negotiating.

6.2.3 Actions

A controller takes a number of actions when monitoring flights, building new trajectories, or negotiating with other controllers. Some of these actions are internal to the ATC system, and depend on the interface used. Using the interface, the controller also has to perform a number of actions that have an impact on the real world. Compared to the information available, the controller has very few ways of acting on the situation, because every meaningful action is mediated by other humans, especially crews (pilots are not yet prepared to let controllers take actions that affect the state of their aircraft). This is why most actions are in fact communications. The controller communicates with crews, in order to request them to implement the decisions made when solving problems.

The first action to be taken as soon as an aircraft enters the sector is to open and check the communication link. Then, when it is necessary, the controller asks the crew to change speed, altitude or heading. Finally, when the aircraft is about to leave the sector, the controller tells the crew to contact the controller of the next sector, and makes sure that the new communication link is established.

In the same way, the actions involved in coordination with other controllers are basically communications. Negotiations among controllers generally occur when a controller wants to hand over an aircraft to another controller. If this is not performed automatically by the system, the controller has to establish a communication link, name the aircraft and propose a waypoint, a time and an altitude for the transfer. The other controller can then accept, or propose other solutions. As controllers share the same computing system, these pieces of information are easily available, provided that they are entered into the computer.

6.3 A Simplified ATC Support System

Now that the controller's task and the means available have been described, the new Support System must be designed. This is, of course, too large a task to be done completely in this example; many of the details are not

relevant to this book. For these reasons the support system considered will be simplified; details will only be introduced when useful in illustrating points being discussed.

The primary aim of an ATC system is to enable controllers to handle an important amount of air traffic without jeopardizing the safety of aircrafts. For the purpose of this book, it will be assumed that a reasonable way of achieving this aim is to create a single integrated computer supported system which includes the various existing subsystems (flight plans management, radar display, radio links and so on) and provides a uniform, usable interface for the air traffic controllers.

An ideal support system design suggests that the system should:

- accept flight plans electronically and log them automatically;
- establish communication with the pilot when the aircraft crosses into the sector (the controller of the previous sector having both informed the new sector controller of the entry and requested the pilot to establish communication with the new controller; both pilot and ATC system use a time-out mechanism to indicate a problem to the preceding sector controller if successful communication cannot be established);
- make available to the controller in advance the expected times of aircraft entry to and departure from the sector and the planned flight route through the sector;
- provide an information retrieval and computational support system for investigating putative flight plan alterations ('what if' scenarios), using existing flight plans, permitted 'routes', actual situations in the past, simulations of possible revised routes;
- log chosen flight plan revisions and notify the next sector of the projected consequences, if any;
- notify the controller (and perhaps the pilot) if any anticipated event is not detected when expected (e.g. a failure to establish or maintain communication; a failure to respond to a request).

6.4 External Properties

As may easily be imagined, external properties are crucial to the success of an ATC support system. More generally, the relative weighting of the properties depends on the application domain. Air traffic control is an open, cooperative and very safety-critical application. Many people (e.g. controllers, pilots and supervisors) must cooperate for the ATC system to function at all; because of the nature of the system, the overall system safety must be the overriding concern. This influences strongly the way in which the priorities of the desired properties must be balanced; in several cases domain requirements take precedence over 'nice' software properties.

For each property, with the exception of goal completeness, one or more situations are described, where this property is of significant importance in the system to at least one of the many people involved. In a number of cases, overlaps between properties are identified.

Safety requirements

Because of the application domain of the ATC system, safety requirements override almost every other requirement. This means that the *robustness* of the interaction is very important. Robustness means – among other things – that the system tolerates errors and deviations from 'normal use'. As also noted in subsections below, redundancy in the system and in the dialog assists in deviation detection. In order to be able to tolerate deviation, either the controller, support system or both need to detect potential errors and take appropriate action before the situation gets out of control. The examples noted under observability, insistence and honesty are all cases where the system is tolerating expected deviation of one sort or another. For these reasons, it is essential to examine interaction robustness properties before interaction flexibility properties.

Furthermore, *unexpected* deviation also has to be taken care of. As a general rule, any safety-critical system such as an ATC system, must have anticipated all possible events and have provided actions for each. Yet it is impossible to enumerate all possible events in most real systems – they are usually countable though infinite. This impasse is normally overcome by first defining a *safe default action* which will occur whenever any otherwise unanticipated event occurs ('unanticipated' means that the event is not covered by a specified set of conditions). Later on, as unanticipated problems are observed, explicit tests and responses are added for these specific possibilities.

This method of coping with deviations could be described as Deviation Intolerance since, unless a specific case has been identified, in which case it is not a 'deviation', then a specified action will always be taken. Conversely, it could be thought of as Deviation Tolerance, since all user actions lead to inherently safe operation (but not, necessarily, to what the user would have desired).

Further discussion of safety aspects is found below in several of the subsections on individual robustness properties.

6.4.1 Goal completeness

The principal purpose of the ATC system as a whole is the safe passage of aircrafts from take-off to landing with minimum disruption to the original flight plan and minimum extra cost for operators as secondary aims. The goals of individual humans using the air traffic control system may or may

not be related to these purposes. However, the ATC support system will be complete insofar as it enables the controllers interacting with it to achieve the overall purposes and discourages them from taking actions that are inimical to those purposes. A full judgement on goal completeness is only possible when the system is in operation. It may even be necessary to wait until after the system is taken out of service before the final assessment can be made.

Basically, a task-analytic approach to completeness would not suffice in this safety-critical system. The task analysis results in a task model, and task completeness is demonstrated by showing that all adopted tasks from the model are covered. As pointed out above, however, the real world environment interferes with the ATC system in such a critical manner that task completeness cannot guarantee the safety of the system.

6.4.2 Interaction Robustness

Observability

According to its definition, observability means that a system must make all information that is relevant to the user perceivable. The key word here is 'relevant', especially if information overload is to be avoided: relevance depends on the situation and time. A differentiation can be made here between the capability of observing (i.e. making information available on request) and providing information automatically. Forcing something to be observable may lead to clutter and to important things being overlooked.

An alternative to basing relevance on the user's current tasks is to allow the user to perceive anything he or she can name. 'Name' should be interpreted as meaning 'provide a description for'. If the description is unambiguous then the required information should be provided. If not, the user should be given information about the choices of things that could be made available, and which the designer expected could be useful in the current state.

Some information can be displayed continuously, because it is essential and/or because its display is inexpensive in terms of perceptive load. For instance, the current positions of airplanes are always represented on the display screen because they are essential for managing air traffic. Similarly, the past positions of airplanes over the last minute or so are also represented, because they give a readily assimilable indication of direction and speed, and because they can easily be integrated in the display.

Other kinds of information do not merit continuous display, but need to observable. This can be accommodated by always making search or browse facilities available. This would allow an ATCo to examine any of the information in the system if desired, since anything known to the system which relates to aircraft or to pilots or to the system itself or to other

air traffic controllers is, potentially, relevant to the ATCo in particular circumstances. For instance, the ATCo may wish to check the route that the pilot has entered in the flight management system of the airplane. Observability will allow the ATCo to detect blatant errors in the route, and even to decide whether the pilot will need special care and guidance, because he does not seem to know the area well enough.

The trade-off between observability and browsability (or discoverability) – i.e. searching – can be thought of as the trade-off between providing very focused information, specifically oriented to the current task, and providing additional information because it might be useful. The former strategy may lead to essential information not being available when it is needed, the latter to clutter and information overload. Since in any real system, all the information cannot be observable at all times, it is always possible that the controller will need to 'search' for the information they require. If a very focused task-oriented strategy is adopted for what is made observable, it will be necessary to provide a browse/search mechanism which allows users to quickly find the information they need.

Finally, awareness (or feedthrough) is a special case of observability, as indicated in Section 2.4.1. It is also important in an ATC system. Suppose that a controller has a potential emergency and wants to redirect an aircraft to another sector, but that sector is currently overloaded. If the system has kept the ATCo aware of conditions in the surrounding sectors (perhaps by varying the color of the boundary with the other section displayed on the screen – green might mean lightly loaded, white average load, orange heavy load) then the ATCo is in a position to make an informed choice without needing to search for information.

Conversely, once the potential emergency is notified to the Air Traffic Control system, it must (insistently) be made observable to the ATCos in the surrounding sections (perhaps by making their boundary displays red). In the emergency situation, it is essential to ensure that these ATCos are aware that the emergency exists so that they can be considering whether there is any way in which they can offer help. In this case observability is not enough. Insistence is required in that the system needs to ensure that the information has been observed by obtaining a positive feedback from each of the other ATCos (the only situation where this would not be the case would be where one of the other ATCos was already dealing with an emergency of equal or greater magnitude).

Access Control

As discussed in Chapter 2, there are two different aspects to Access Control. Firstly, it is intended to prevent users deliberately viewing or changing information that the owner of the information did not wish them to be able to access. Secondly, it is there to make it easier for the controller

to use the system by eliminating irrelevant information and/or unnecessary functionality. The latter aspect of access control is closely related to both observability (users should not have to observe information that is not relevant to their role) and honesty (functions should not be offered if they cannot be used). Putting this in a more positive way, the chance of accidental mistakes will be minimized if users are only able to access those items that are relevant to their current task, and are only able to carry out the operations that are appropriate when performing the task (i.e. this form of access control offers role-oriented support for the user).

In the ATC system, an ATCo would not normally need to access information about aircraft which are not currently in their sector and which are not scheduled to enter their sector. Thus the Access Control mechanism should prevent them from getting this information even if they make a search request that it would apparently satisfy (e.g. the access control mechanism should attach to the request 'tell me about aircraft that are scheduled to land in Toulouse at 1400Z' the condition 'an aircraft is only relevant if it is either in my sector now or it is scheduled to pass through it between now and 1400Z'). Similarly, and even more importantly, an ATCo should not be able to make changes to the information relating to aircrafts that ATCo is not controling. Thus a request to amend the route for an aircraft, which is specified by referring to the aircraft's identification number, must be rejected, unless it is under the control of the ATCo at this moment (irrespective of the fact that it may be currently in the air and known to the system, or was recently being controled by this ATCo). However, in some circumstances, it must permit an ATCo to change roles and then be able to access the information that was previously unobtainable, e.g. if a colleague becomes ill on duty, aircrafts may need to be reassigned.

Insistence

A system is insistent if feedback to the user is sustained until some specified user reaction is forthcoming. Note that there is an inherent conflict between the robustness property of insistence and the flexibility property of preemptiveness.

In this air traffic control example, an alert is said to occur whenever something moves outside pre-defined limits possibly specified by the procedures or by the ATCo. System alert should always be insistent, as mentioned above for observability.

Suppose that an aircraft that is due in the sector has not arrived: communications have not been opened and there is no trace on the radar for that aircraft, although the previous sector has passed on information of expected arrival. This could mean that the system in the previous sector did not notify a change in plan. The system would then need to notify the ATCo and log a possible system fault condition so that corrective actions

can be taken. In this case insistence operates on at least two different controllers on different time scales. The ATCo needs to 'find' the aircraft, and the supervisor needs to review the operation of the system. If the ATCo does not respond within a prespecified time alert priority is raised. How this is done depends on the alert and preferences – e.g. a klaxon sounds, or the room supervisor is warned of a possible need to take over. As concerns the supervisor, the error message will be stored and reminded every day, until it is handled.

From time to time the controller may need to temporarily suppress a message in order to deal with a more urgent/dangerous situation. This, however, must only be possible within context-dependent limits, if safety criteria are met. Automatic transfer of 'insistent' messages to other people may be one design solution. Conversely, in an emergency, signals that ordinarily might be insistent may need to be suppressed to allow the ATCo to concentrate on the emergency. There is no need to notify the ATCo of the normal arrival in their sector of an aircraft if it is on time, its flight plan does not overlap with the airspace involved in the emergency, and it requires no ATCo action to allow it to proceed.

Honesty

A system is called honest if representations of internal state elements are designed to be interpreted correctly. Honesty is a fundamental feature of all aspects of the ATC system. For instance, it is hardly acceptable for the system to display the position of an aircraft that did not exist; this is why so much effort is still devoted to the elaboration of algorithms for radar signal processing. The design of the support system can facilitate the detection of accidental dishonesty (e.g. due to system malfunction or user misconception) by providing a high degree of redundant information. This was illustrated in the previous section, where the transfer of an aircraft from one sector to another is accompanied by a message exchange between the systems in the two sectors, a display of the incoming aircraft position on the receiving controller's console and the opening of a communication channel between the aircraft and the ATCo. Various controllers should thus each be able to distinguish between a single device/subsystem failure and a genuine 'problem event' and not be confused by thinking that the system was failing to be 'honest', i.e. failing to make the relevant information available.

Pace tolerance

The definition of a pace tolerant system is a system where the user may control the pace of interaction. Since the Air Traffic Control Support System has to operate in real time, it is not pace tolerant in the way a text editor or some computer games could be. A text editor is generally purely reactive,

and users chose their own pace for typing. Computer action games involve real time, but most of them provide a 'pause' or a 'slow speed' function in order to provide a form of pace tolerance. But an Air Traffic Control system cannot offer such a facility. However, pace tolerance can be introduced in secondary functions. For instance, the system allows the ATCo to tailor the length of time-outs such as the one given in the insistence example above (the missing aircraft), but only within pre-specified limits in order not to compromise safety.

Pace tolerance must take precedence over 'temporal' honesty, for example, in busy periods where an ATCo receives many data and voice messages from other controllers and pilots. If the system is to be 'honest' about time, it must present messages immediately as they arrive. Honesty here must be relaxed by preserving the sequence in which they occur 'in reality'.

Predictability

In a predictable system, users can know its behavior from their knowledge of past interactions and current state. As in most other systems, predictability is, like honesty, a major requirement for the ATC Support System. The system must provide the information that is requested when it is needed and it must carry out requested operations reliably and within an expected time period. For instance, when sending a message to a flight crew, the controller will want that message to be delivered within a well-known amount of time. Or, when filtering useless information away from the controller, the system should make similar decisions for similar situations. Also, as for honesty, the provision of redundancy helps the controller detect potential malfunctions and distinguish these from a simple lack of predictability.

Deviation tolerance

A system is called deviation tolerant if it supports the correction of slips and mistakes. This implies that the system is able to detect 'unwanted' events, such events being sometimes that the user failed to act. This also implies that the system is able to take some appropriate action. This could be to 'take control' (as a kind of forced migration of control), or it may involve presenting warning information to the user that deviated or to other users.

Being a safety-critical system, the support system cannot offer much deviation tolerance for the contents of actions related to real time activities. Deviation tolerance, however, should at least be offered for the structure and the sequencing of actions.

For example, the ATCo fails to respond to a routine notification of the arrival of an aircraft, which is on time and for which there is no anticipated problem; since the ATCo could have suppressed the notification anyway, there is no sense in insisting on a response or of immediately notifying

anyone else. However, the failure to respond is an indication of a potential problem, which should be logged and should 'sensitize' the system to look for the possibility that the ATCo is 'malfunctioning' (e.g. the ATC system might reduce tolerance limits on other expected actions by that ATCo – if a record is kept of the 'normal' time, and the variance on that time, taken by that individual controller to respond to each sort of notification then the tolerance limits might normally be at a 95% level, the maximum time needed to contain 95% of actual responses by that ATCo, but after one failure to respond within the required period, the tolerance might be reduced to a 75% level until there have been, say, three successful responses; a second 'slow' response might cause the tolerance to be reduced to the 50% level and a third failure sets alarm bells ringing, literally, asking the supervisor to go and investigate).

6.4.3 Interaction Flexibility

Device multiplicity

Data is provided to an ATC Support System from many devices and several devices are used to interact with the controllers. ATCos are provided with displays for graphical radar output (or synthesized aircraft position displays), secondary displays for routine information, telephone and radio for communications input and output, sound output for audible warnings, pointing devices for direct manipulation input of commands, and even keyboards if really necessary.

Multi-device capability means that the controller is offered a choice between several modalities of renderings in a number of cases. The key to success that has been noted earlier is that redundancy is essential if potential error situations are to be detected and prevented from becoming actual errors. Multi-device capability is needed to provide the communications redundancy: it is no use having many different possible ways of detecting errors internally, if at the end all messages from the system are displayed on a single screen, and there is no other method of getting a message to the ATCo. Conversely, however many facilities a system could offer a user, if all of them must be chosen using a single input device (be it a mouse or a touch screen or a keyboard) then the system is useless if that input device is broken.

Therefore, multi-device capability in the ATC is needed for safety reasons, not only for operators' convenience. The design must include multi-device capability to ensure that the ATCo can do the job even in the presence of some (hardware) faults.

Representation multiplicity

The discussion in the previous subsection indicates that a 'safe' system (one that is fault tolerant in that it is not dependent on a single output device to present its results) must exhibit multi-channel capability. Also, representation multiplicity adds to the safety, but may make the system more convenient to the user. In a number of cases, the ATCo should be allowed a choice between graphical, tabulated or audible presentation of output.

Consider the following scenario. The controller will need to have information about aircrafts' current positions presented geographically on the display, so as to reason geometrically about the situation. On this radar screen there is not enough room for detailed information about aircrafts, such as the time at which they are expected at a certain waypoint. The ATCo probably also wants that kind of data, and this is why another representation of aircrafts is provided at the same time: card flight strips (current systems) or electronic flight strips (systems now being designed) hold that information in a tabular form.

If both device and representation multiplicity are present, all the information about an object does not need to be presented in one way on one device. For instance, the event of an aircraft entering a sector can be represented as a change of color of its representation of that aircraft, and can also be represented as a brief sound. The use of multiple devices for multiple representations can thus take advantage of the different qualities of different devices.

Human role multiplicity

The ATC Support System must support many different controllers, supervisors, etc. Even a single controller, however, may take several differing roles over a period of time (e.g. establishing communication with an aircraft, performing a 'what-if' simulation, passing information to another ATCo). Indeed, as far as the ATC system is concerned, the ATCo may be performing several roles at the same time if the system supports multithreading, since this permits several roles to logically coexist even though the ATCo can only actually be interacting with one of them at a particular instant.

As has been discussed in the earlier subsections on observability and access control the role which the system ascribes to a particular interaction thread with the ATCo will control what information is made observable and what information can be retrieved (by searching or browsing). It may also change the functionality, which is provided by the system to the ATCo.

On occasions, a particular ATCo will have to perform roles outside his or her normal duties. There is thus a close relationship between this form of human role multiplicity and customizability. On the other hand, where different humans are performing similar but not identical roles, effective

support for human role multiplicity might best be achieved by providing adequate reconfigurability.

Output or input re-use

The second purpose of an ATC Support System, after security, is to enable controllers to manage air traffic efficiently. Therefore, every service offered by the system to save time is very useful. I/O re-use is such a service. Earlier input can be re-used, because an earlier situation reappears, or some output should be forwarded to another controller or to a pilot. Two examples follow.

When an airplane moves from A's sector to B's sector, the relevant flight information is presented to controller B. Later, when the airplane moves on to C's sector, B wants to re-use the flight information to send it to C.

When an alarm occurs on the display, the ATCo shall re-use the output as input to send it to all other relevant ATCos and pilots.

Multithreading

Multithreading means that the user can execute several tasks at a time. Air traffic control is a very demanding activity in terms of multithreading. Every controller receives input from a variety of sources including radar systems while communicating with other controllers and pilots. A controller may be simultaneously working on one or more information searches and what-if simulations, as well as making amendments to existing plans and filing revised flight plans.

For instance, suppose that an ATCo is using the radar display to build a new set of routes for conflicting airplanes. If a pilot calls during that activity, what the controller wants to do is start a new thread of actions in order to store the information given by the pilot, identify the possible new problems it causes, and begin to deal with it. Depending on whether the system provides multithreading or not, the ATCo will be able to start that new thread without losing his previous unfinished work, or will have to choose between losing it or deferring the handling of the call to a later time.

The ability of a system to support browsing or searching for relevant information, referred to earlier in the discussion on observability, could provide another example of multithreading. If the user must explicitly stop the current task and switch to an information-seeking task which must be completed, or abandoned, before the original task can be resumed, then the system is providing only a single thread. If, on the other hand, a user can initiate a search or browse through the information space while retaining the ability to continue with the 'current' task, then multithreading is being provided.

Providing multithreading would also facilitate achieving the next goal,

non-preemptiveness. If multithreading is available then, in any situation where the system requests a response, the user can be allowed to simultaneously choose one or more of the following:

- reply as requested;
- request further information to assist in choosing the response;
- start a different 'conversation', e.g. a replanning simulation exercise (a 'what if').

For example, an ATCo could start a replanning simulation exercise, then look for more information, and finally provide the response as requested.

Non-Preemptiveness

Complete non-premptiveness means that the system must tolerate any permissible event occurring at any time, whether that event be due to user action, communication from another system or communication from another component of the same system (e.g. the radar). Since there are many users and many subsystems, who all perceive themselves as operating independently, it is obvious that an action by one user cannot be synchronized with actions of other users. Furthermore, if the ATCo is to be allowed to ask for further information at any point (see the paragraph on 'honesty', above) and to carry out 'what-if' simulations, then it is necessarily the case that no interaction between one ATCo and the system can be permitted to be pre-emptive.

Reachability

Reachability means that any perceivable state of the system can be attained from any other perceivable state. This property cannot be met in its pure form in air traffic control systems, because ATC is a real time task that models the real world, and some events are irreversible in the real world. This should be reflected in 'future reachability' states in the system. Thus, if an aircraft lands, it is not sensible to want to reach states which represent it as still being in the air when considering 'what-if' scenarios related to the immediate future. However, if the aircraft lands badly, it may be important to consider what went wrong during the descent. In this case it is essential to be able to reach the historical record of the actual states of the system during the descent (this will allow the performance of the system and of the controller to be reviewed as well as that of the pilot).

Reachability should be provided as often as possible, especially in non-real time interactions. As noted above, reachability is given essential support by 'history', even where things have not gone wrong. For instance, if the ATCo has discarded a flight strip, and another ATCo asks questions about that flight, the ATCo will want to retrieve the flight strip. In cur-

rent systems (with card strips), this is done by searching the waste paper basket.

'Full' reachability would let the user move from any perceivable state to any other state, and this should *not* be possible in the ATC Support System for safety reasons. Firstly, only states corresponding to the real world should be reachable. Secondly, system states representing 'dangerous situations' or conflicts must not be reachable (or only when special conditions are met), e.g. a warning should be issued if the controller attempts to simulate a flight replanning in which two aircrafts are flying with too small a separation.

Reconfigurability

Reconfigurability means that the ATCo may change representations and operations. The ATC Support System will be used by many different persons and under many different circumstances. Therefore it should allow for a certain degree of reconfigurability, and a number of examples of this are given above (changing time-out times, altering limits, changing defaults etc.). Each user must be allowed to customize the ATC Support System for their own use on a short term basis (e.g. for this session) or on a longer term.

Another type of reconfigurability is necessary when changing over, especially in evenings. A control sector is usually attended by two or three controllers, but at night a controller alone is enough: there are few enough aircraft that flight managing and coordination can be performed by a single person. This means that displays and input devices have to be reconfigured so as to allow one person to do the work of two. For instance an ATCo will not want to switch from one mouse to another every time he or she has to interact through a console that is usually used by another ATCo. The mouse on this console would then need to be reconfigured from 'one display only' operation to 'all displays' operation.

Adaptivity

Adaptivity means that the system changes representation and accessible operations as a result of surveying the situation and the ATCo's interaction pattern. For instance, the system may be able to know that some group of aircrafts poses more problems than other aircrafts in the sector, because the ATCo keeps interacting with their representation. The support system could then provide a more visible representation for these aircrafts, e.g. by using a brighter color.

This kind of adaptation very easily conflicts with both honesty and predictability, because it changes the behavior of the system. During the operation of a safety-critical system like the ATC, adaptation must be done with extreme care and – when used – with full explanation to the ATCo.

Migratability

The ATC possesses migratability if the ATCo can ask the system to automatically start some tasks usually initiated by the ATCo and, vice versa, if the ATCo can 'take control' over some tasks, which the system used to perform automatically.

The system will have a number of default actions which under normal conditions are sensible and safe actions. But under extreme conditions (heavy load or dangerous situations) the ATCo must be able to 'take control' and possibly change the default action to something else.

For instance, algorithms are being developed to automatically resolve conflicts between aircraft flight paths. These algorithms are still limited, but can be useful to alleviate the ATCo's load. Then the question is how to split the task between the ATCo and the computer. Migratability is important here. For example, a solution would consist in the system deciding that the ATCo is overloaded (adaptation) and proposing to take over. Then, if the ATCo accepts, he or she will want to be able to take over again, either by telling the system to stop solving that problem or by imposing his or her choices over the system's.

6.5 Applying the PAC-Amodeus Model

This section presents a software architecture for the Air Traffic Control Support System. Consider the external specifications that have been used for designing the software architecture. The system supports two workstations per sector. Figure 6.1 schematically presents the user interfaces of the displays of the two workstations. The monitoring tasks supported by each workstation are complementary and performed by two air traffic controllers.

On the first workstation the radar display is not editable. The controller can modify information about an aircraft by using the tool palette, located on the left of the radar screen.

On the second workstation, the controller can modify the flight path of an aircraft on the radar display by direct manipulation. Furthermore the controller can obtain information about an aircraft by selecting it. Consequently visual consistency (see representation multiplicity property described in Section 2.3.2) of the two workstation displays must be maintained (except when one of the controllers is doing some replanning simulation).

Figure 6.2 shows one possible PAC-Amodeus software architecture for implementing the proposed Air Traffic Control system. Figure 6.3 presents the software agents' hierarchy organizing the Dialog Controller.

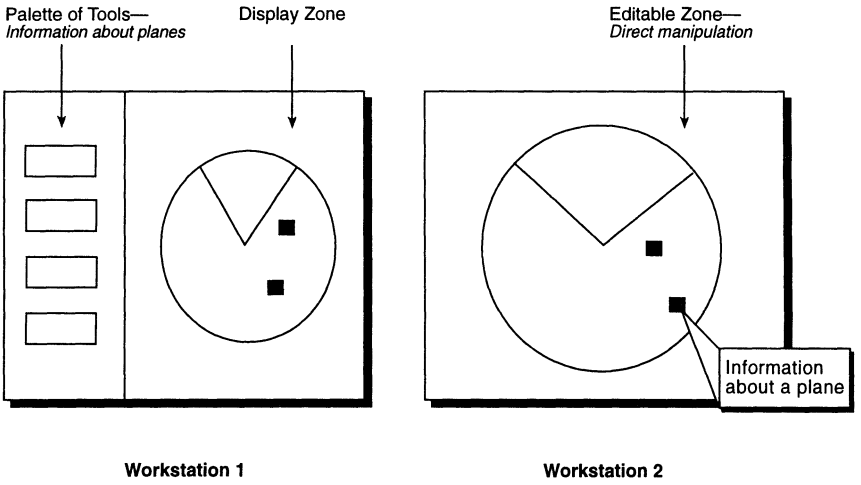

Figure 6.1 *External specification of the two workstations belonging to one sector. Each display shows a radar picture.*

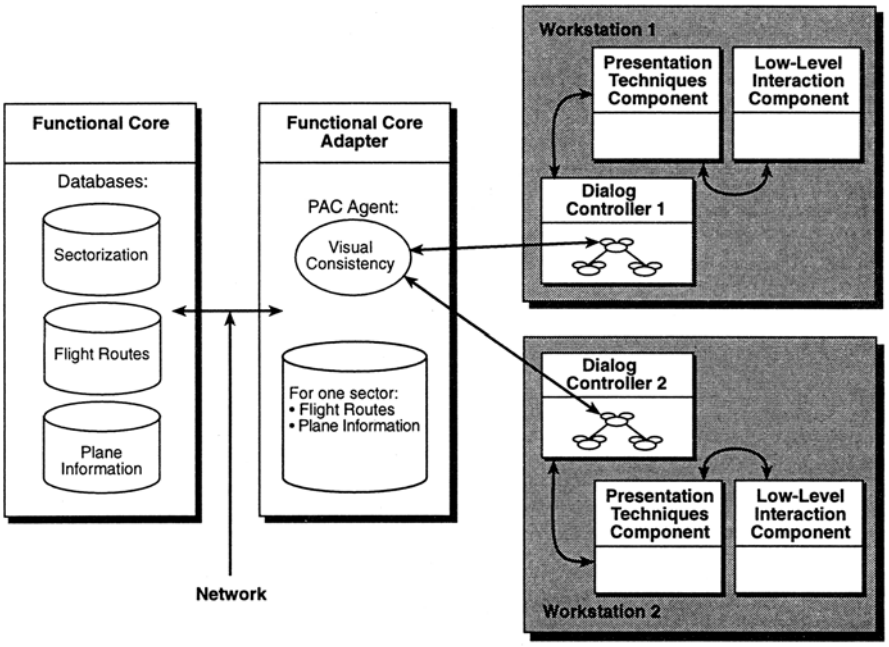

Figure 6.2 *Software components implementing the Air Traffic Control Support System (applying the PAC-Amodeus model).*

At the right-hand side of Figure 6.2, the Low-level Interaction Component (LLIC) denotes the underlying software and hardware platforms. It receives mouse and keyboard events from the user. It also manages the presentation and contains functions to display the radar picture inside a window. The Presentation Techniques Component (PTC) bridges the gap between the Dialog Controller and the LLIC. Neither Dialog Controller (DC) depends on the functions displaying the radar screen, for example.

At the left-hand side of the picture, the Functional Core (FC) maintains and manages the database. The database contains information about:

- sectorization
- flight paths
- aircraft information.

The database is linked to the workstations (two per sector) through the network. To enhance the run time efficiency on each workstation, the information about aircrafts and routes of one sector is duplicated and stored in the Functional Core Adapter (FCA). This option guarantees the stability of the response time because no request is sent through the network. On the other hand this option will increase the number of messages through the network to update the duplicated databases. Moreover the FCA provides for communication between its two adjacent components (i.e. FC and DC) by implementing a communication protocol. It is therefore possible to receive information through the network and to handle user events. This will be managed within the DC, which will be passed network information by the FCA, and receive events from the presentation techniques components.

The hierarchy of PAC agents organizing the DC is presented in Figure 6.3. A PAC agent is composed of three parts (see Chapter 4):

- the abstraction facet
- the control facet
- the presentation facet.

The Dialog Controller (DC) is comprised of PAC agents. There is one DC and thus one hierarchy of PAC agents on each workstation. Figure 6.3 shows the two hierarchies. A dedicated agent within the FCA maintains the visual consistency of the two workstation displays, and is thus linked to the two PAC agent hierarchies.

- The root agents 'Root1' and 'Root2' are in charge of the global control of the interaction with the users. The Presentation facet of each root manages high-level layout and displays ornaments such as frames and separators. The abstraction of each root facet receives information about the flights it should display.
- The Radar agents synthesize the radar pictures. The abstraction parts receive the flight information from the root agent. On workstation 1,

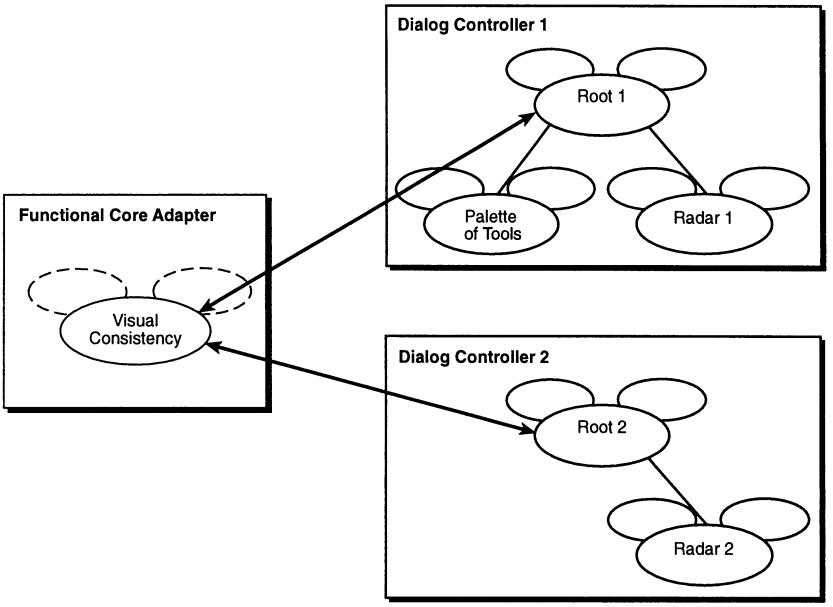

Figure 6.3 *PAC agents organizing the two Dialog Controller components.*

the Radar 1 agent only displays the information. On workstation 2, the Radar 2 agent is more complex and handles the interaction with the user within its zone on screen.

- The 'Palette' agent corresponds to the palette of tools depicted in Figure 6.1.