MANAGING THE AIRPORT COMPLEXITY: CONTRIBUTION OF THE OBJECT-ORIENTED APPROACH

MICHEL LEMOINE
ONERA Centre de Toulouse
2 avenue E. Belin
31055 Toulouse CEDEX
e-mail: Michel.Lemoine@cert.fr

Keywords: *airport, functional and structural description, object-oriented model, validation*

Abstract

In this paper we emphasise how a computer science technology, namely the Object-Oriented [OO] approach, is able to help modelling large and complex systems such as airports. We will demonstrate how the building of an Information System for the Airport will allow to better manage current and future solutions. First of all we introduce very briefly the 2 used notations: Use Cases and Class Diagrams that respectively allow to model the functional and structural points of view. We show how some application constraints can be taken into account. We then emphasise how these models can be validated by end users such air traffic controllers, pilots, … These models constitute the rigorous and static specification of Decision Support System [DSS] and Collaborative Decision Making systems [CDM] that should be used to introduce new procedures, new regulations, …

1 The AdF project

In the context of an internal ONERA[1] research project started in 1998, we are considering what should be the Airport of the Future [AdF]. Indeed it is now recognised that due to an air traffic increase of 8% a year, the bottleneck is the airport itself.

In the AdF project, we are considering the airport problems from a global point of view. We have restricted them to landing, taking off, taxiing and parking phases.

In a 2nd step we are considering alternative designs that are currently evaluated through the help of DSSs and CDMs which are directly derived from the global views we have been able to build up.

In the following sections we emphasise our ability to model global views of the airport, and how it is possible, using an OO approach, to validate or invalidate the subsequent DSSs and CDMs.

For more information about the AdF project, see [1], and [2].

2 The chosen approach

2.1 The Airport complexity

In an airport, both from an airside and landside point of view, there are many actors. Some are directly interacting with the airport - passengers, pilots, controllers, …, some are more passive - weather forecast, regulations, …. The services - we call them functionalities - provided by the airport are numerous. For instance the Advanced-Surface Movement Guidance and Control System [A-SMGCS] is in charge of taking care of all the vehicles movements on the tarmac. The A-SMGCS interacts both with active and passive actors.

The airport, as a whole, is a very complex system that has not been tackled up to now in the right way.

[1] ONERA stands for *Office National d'Études et de Recherches Aérospatiales*.

Regarding similar complex systems, it has been shown that a possible answer goes through the building up of an Information System, seen as *the collection of technical and human resources that provide the storage, computing, distribution, and communication for the information required by all or some part of an enterprise*.

2.2 The object-oriented approach

In the AdF team computer scientists proposed to model the Information System using the UML notations [3] which are OO. They also suggested to use the Fusion [4] method for the analysis phase.

According to such a method, it is important to develop two kinds of models:

- Use Cases models which allow to identify who are the main actors, and what are the main services the Information System provides.
- Class Diagrams which allow to identify all the information that are manipulated, and their static structure.

These two kinds of models must be validated[2] by end users.

3. Building an Information System for the Airport

3.1 Identifying actors and services.

In Figure 1, there is an example of a Use Case. Here we have considered an hypothetical A-SMGCS.

3.1.1 Actors

The main actors are:

- *Air Traffic Controllers*: they manage the ground traffic of all the vehicles on the tarmac.
- *Vehicles:* they are the interacting actors instead of pilots because there are many kinds of vehicle whereas there are only one kind of pilot / driver.
- *Pilot:* she / he is also an actor but not interacting directly with the AdF system. This results from a design decision. Of course, in another design we could have chosen the *Pilot* as main actor. In the later the point of view could have been different.
- *Weather Forecast:* this is a passive actor only able to send information.
- *Gate Manager*: she / he is responsible for gates allocation.
- *Police*: it is responsible for security.
- *Infrastructure*: it represents the airport as a set of taxiways, runways, buildings, …
- *Sensors*: they are actors as well. They interact with many services and have relationships with the infrastructure.

3.1.2 Services

Regarding the services offered by the A-SGMCS system, we have suggested the 4 traditional functions, as recommended by the International Organisation of Civil Aviation:

- Monitoring
- Routeing
- Surveillance
- Guidance

The 5th functionality presented in the Use Case is relative to warnings that can be set up by police services for security reasons.

Attached to each service are some actors who interact with them, interact meaning exchanging information. It must be noticed that actors are either person, or external systems such as for instance regulations (not here represented), infrastructure, …

3.1.3 Use Case Validation

An important aspect of the OO approach as promoted by Fusion [4] with UML [3] notations, is the ability to validate, long before any software development, the functional and static models.

For validation, as usual, end users are mandatory. Indeed, we can understand a Use Case as a high level functional description of the system under consideration. Thus, a Use Case represents in some way the **What** of a system to be either developed or reengineered.

[2] According to W. Boehm, *Validation* means: *Am I building the right system?*, whereas *Verification* means: *Am I building the system right?*

Regarding the AdF project, the 5 services have been validated by end users because they correspond to very well known services. But, regarding the actors, the end users exhibited that a lot of them were missing. The final list of actors includes more than 30 different actors that we added to our models.

3.1.4 Use Case Summary

The main interest of such Use Cases is threefold:

1. A high level functional description of the system i.e. the set of services it provides: **What** the system does.
2. A clear identification of actors **Who** interact with the services.
3. An easy validation by end users.

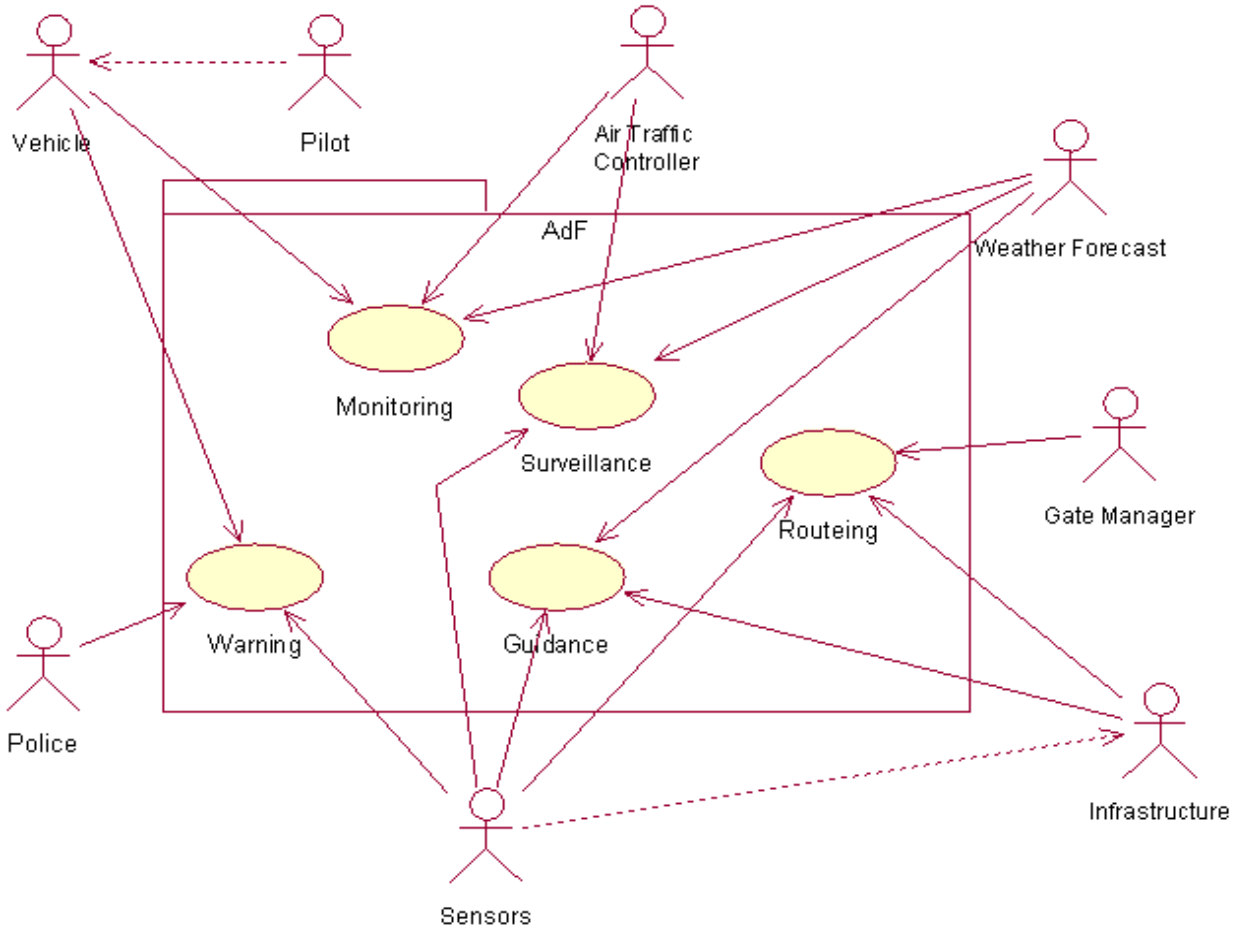

Figure 1: Use Case for the A-SMGCS

3.2 Class Diagrams

In the OO approach, after representing[3] the interactions between actors and services, we must produce what constitutes the core of the OO representation: Class Diagrams that represent both what information are known / manipulated and their relationships.

In the following we are considering 2 diagrams according to 2 points of view of the airport and its air traffic management.

[3] This representation is done through 2 interrelated notations: *Sequence Diagrams* and *Collaboration Diagrams*.

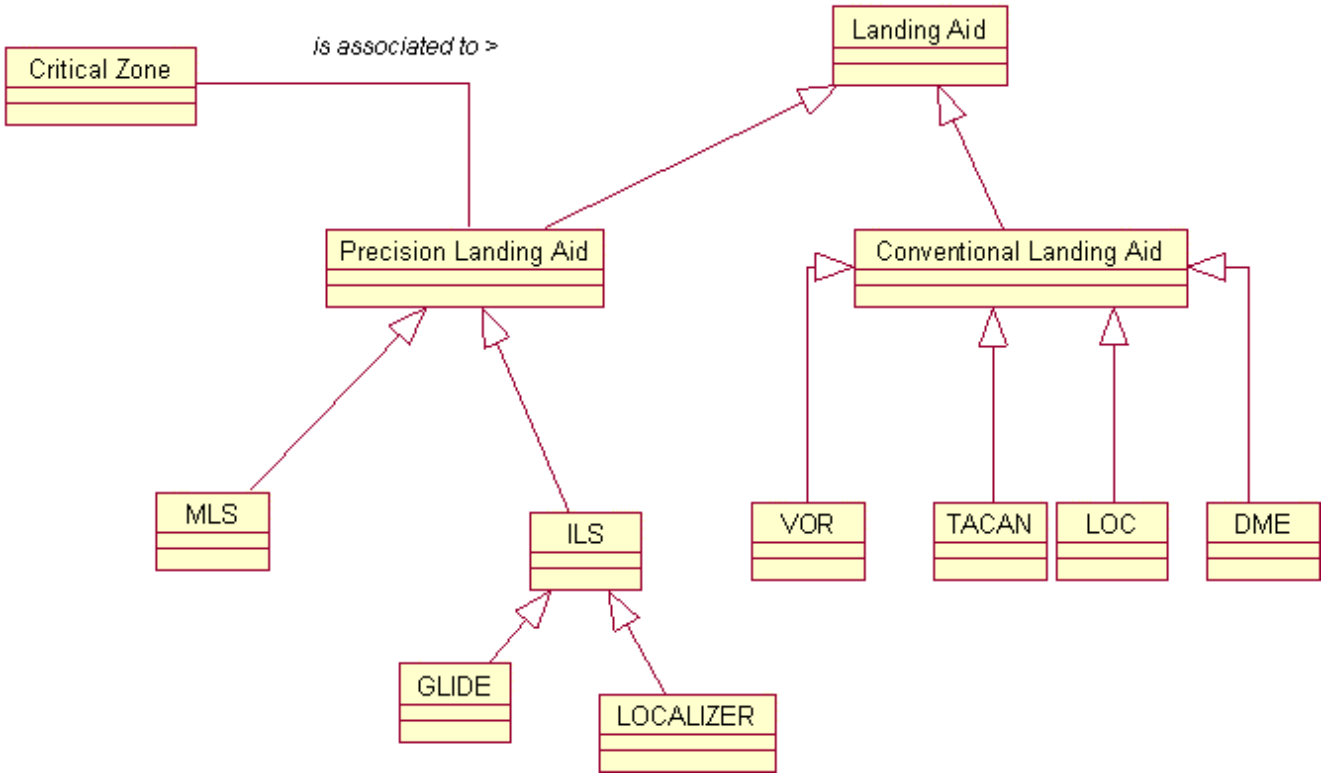

Figure 2: Class Diagram for Landing Aid

3.2.1 Landing Aids

Landing aids are represented in Figure 2. A informal translation of such a Class Diagram is as follows:

1. *Landing Aid:* they are either *Conventional* or *Precision* ones.
2. *Conventional Landing Aid*: it is either a *VOR*, or a *TACAN*, or a *LOC* or a *DME*.
3. *Precision landing aid:* it is either a *MLS* or an *ILS*. To each precision landing corresponds a *Critical Zone*.
4. *ILS*: it s either a *GLIDE* or a *LOCALIZER*.

In this class diagram, we have described landing aids as a classification - or hierarchy. This classification takes advantage of the OO concept of specialisation[4]. In other words, a TACAN is a specialisation of conventional landing aid, which is itself a specialisation of landing aid.

In the above Figure, it must be noticed that a *Precision Landing Aid* corresponds to a *Critical Zone*, i.e. a precision landing aid cannot exist without a critical zone, and vice versa. This association between 2 classes represents a strong link between any pair of precision landing aid and critical zone. We can add, but this is not done here, the cardinality of critical zones and of precision landing aids. These cardinality is a direct translation of some informal constraints such as: to each critical zone is associated at least one or more precision landing aids, but a precision landing aid corresponds to one, and only one, critical zone.

As for Use Cases, a rigorous validation is required. We taught some air traffic controllers, and pilots how to interpret the UML notations for class diagrams. They have immediately been able to read all the class diagrams we have produced, and consequently they have validated / invalidated them. For instance the

[4] The *specialisation* concept is translated by *inheritance* at the OO language level.

definition of the binary relation *is associated to* is wrong from the point of view of cardinality.

For the Figure 2, no important discrepancy appeared. This is mainly due to the fact we have described a classification. It must be noticed that many constraints can be translated by binary association – as is *associated to* in Figure 2 – with the right cardinalities attached to each class appearing in the binary association.

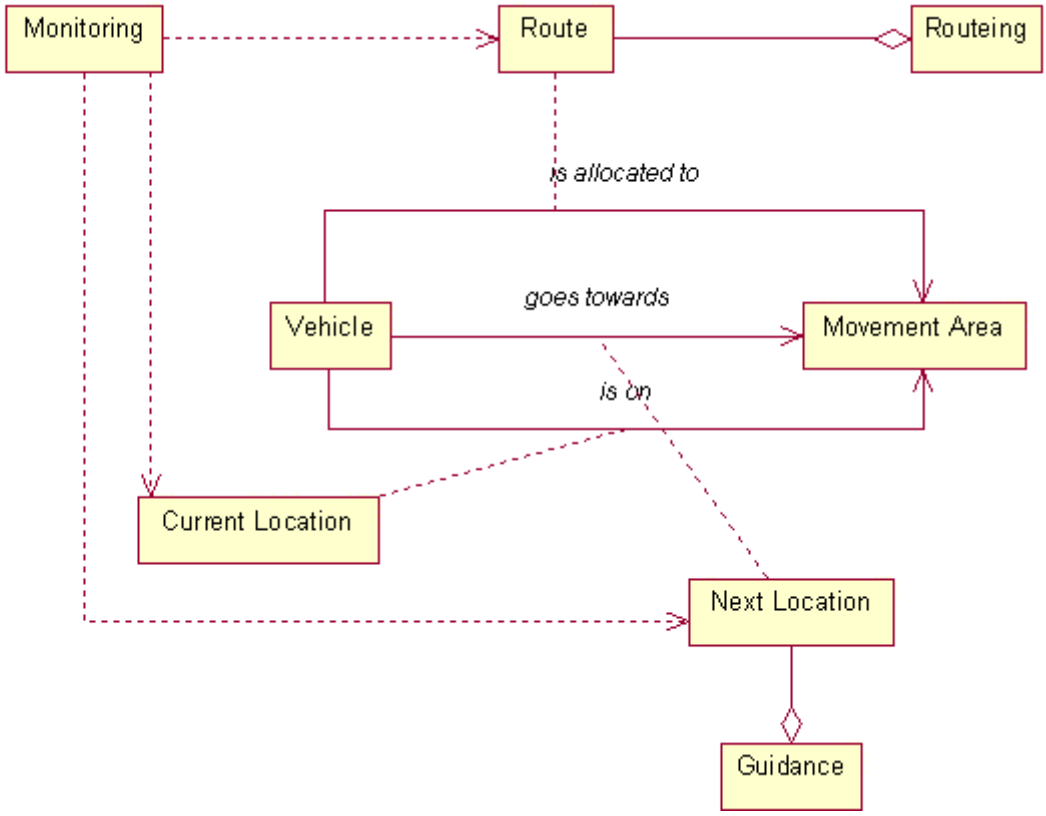

Figure 3: Class Diagram for Guidance, Routeing and Monitoring

3.2.2 Exhibiting conflicts between Vehicle and Movement Area

In Figure 3, we have represented, from a structural point of view, parts of an A-SMGCS.

The above figure should be read as follows:

- *Vehicle*: it represents either an authorised vehicle, or an unauthorised one.
- *Movement Area:* it is the area[5] on which vehicles (authorised or unauthorised as well) can run, or park.
- There are 3 relationships between these 2 classes that may be used for conflict detection.

1. The association *goes towards* represents the next location a vehicle is going to. This association can be seen as the set of a couple *vehicle, movement area*, itself instance of the association class *Next Location.* It must be noticed that an association class is a real class, with its own attributes and its own operations.
2. The *is allocated to* association corresponds, for a given vehicle, to the movement area it should go. As for *Next Location*, the set of couples *vehicle, movement area* are instances of the association class *Route*.
3. The *is on* association represents the movement area where any vehicle is at

[5] There is another class diagram to describe it. It is made of both a hierarchy and an aggregation.

a given time. The corresponding association class is *Current Location*.

- *Guidance:* it is a class, the set of *Next Location* instances. Indeed the vertical diamond represents the notion of composition / aggregation. In other words we have specified that the guidance corresponds to the set of next locations for a vehicle.
- *Routeing*: it is a class representing the set of routes allocated to each vehicle (when they are under control). It must be noticed that at each time step, the next location is unique for a vehicle whereas the route is composed of next locations that can change over the time.
- *Monitoring:* at each time step this class represents what kind of control the system may have. Indeed, knowing where a vehicle is (known by *is on*) and where it should be (known by *goes towards* and / or *is allocated to*) it is easy to detect either Prohibited Area Conflict (the vehicle going to a wrong movement area), or Deviation Conflict (the vehicle following a wrong route), and other conflicts.

This incomplete class diagram helps to understand how we are able to model, even from a static point of view, conflicts. The main point is here we have had the ability to formalise[6] the basic notions taken into account by the Air Traffic Controllers as soon as she / he is managing vehicle on the movement area.

3.2.3 Class Description

The building of use cases and class diagrams is the 1st answer we gave to the AdF project for representing a huge and complex system such as airport. But, even we the best functional and static description of a system, we are not able to tackle all the complexity of the system. Thus we have been obliged to detail as far as possible each class as shown in the next Figure.

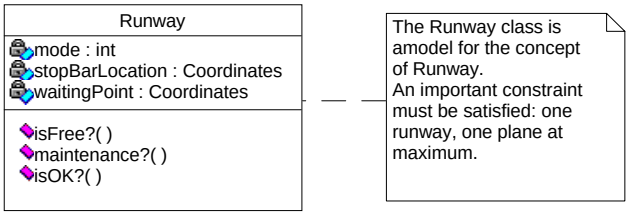

Figure 4: Class description

From a practical point of view we have built such class description, plus a large Data Dictionary as recommended by the Fusion method [4].

4. Conclusion

In this paper we have restricted our presentation to how a OO approach can help modelling, and consequently better understand, a huge and complex system such as an airport.

The current models encompasses more than 100 classes, and more than 50 relations. They have been split in 8 components, related to Landing Aids, Movement Area, Traffic Area, Sensors, Markings (lighting and others), Planes Departure and Arrivals, Airport Infrastructure, Vehicle (all of them). A last component called Central gathers all the 8 components.

Moreover all these diagrams have been validated by the end-users such as pilots, drivers, airport authorities, ATC people, ... These functional and static / structural models are the firm and formal basis for the development of realistic simulators we are currently prototyping as described in [5].

References

[1] http://www.cert.fr/en/dprs/activites/adf/index.html

[2] http://www.red-scientific.co.uk/optas_b.htm

[3] Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. Object Technology Series, Addison-Wesley, 1999.

[4] Coleman D. et al. Object-Oriented Development. The Fusion Method. Prentice Hall Internatial. 1994.

[5] Adelantado M. Experimenting the HLA framework for the ONERA project “Airport of the Future”. Fall Simulation Interoperability, SISO 1999, Workshop, Orlando, USA, 1999

[6] Formalise means here to give a rigorous meaning, not necessarily a full semantics.