Intelligent Object-Oriented Software Systems Applying OMT / UML for Airportuary Environments Management

Adilson Marques da Cunha, Válter Strafacci Jr, Lineu Fernando Stege Mialaret

**ABSTRACT.** One of the main problems in airport environments is the lack of an appropriate information system systematic to improve operational efficiency, resource allocation wastefulness and quality service and delays, to reduce customer’s dissatisfaction. A solution for this airportuary environment problem is the development of an Intelligent Software Systems for airportuary environments. This paper describes an investigation for a new methodology tailored from the OMT methodology and extended to meet the UML standard, using I-CASE-E and RAD tools. This new hybrid methodology for Object Oriented Software System is applied for developing Intelligent Airportulary Resource Allocation Software - IARAS prototype. This article also points some major trends in the use of intelligent objects technique, as an important ingredient to integrate Intelligent Systems, enabling Intelligent Software Components - ISC encapsulation and reusability.

**KEY WORDS:** Airport, OMT, UML and I-CASE-E.

1. Introduction

As an essential part of air transportation systems, airports represent resource transfer sites (Ashford et al., 1984). A proper planning, allowing these airportuary resources in terms of right times and places, involving hundreds aircrafts and thousands of passengers simultaneously processed, requires special cares, mainly, concerning time pressures and delays of any kind.

Handy made airportuary operational resources management have been responsible for resource allocation waste, causing services quality losses, delays and customer's dissatisfaction.

Currently, one of the main airports problem has been the lack of an appropriate systematic for the information management system, aiming to improve operational efficiency and supplied quality services, in order to reduce resource allocation waste, delays and the customer's dissatisfaction.

The supplied services improvement is directly concerned with the development of Intelligent Software Systems to aid appropriate managements, using the Information Technology (Hamzawi, 1992).

In this context, methods, tools and techniques for Information Engineering, Software Engineering and Knowledge Engineering must be used to develop quality software for both Process and Product level.

2. Proposed software system methodology
Software Engineering is a field of computer science applying technologies in different layers (Pressman, 1997). Layers contain own processes, methods and tools. Processes combine methods and tools to economically produce reliable software within quality patterns. Methods provide “how to’s” for building software. And tools allow automated support for processes and methods. Inside to layers, there are programming languages, linking software and hardware.

Nowadays, there are more than 50 methodologies for Object-Oriented software system development. Each one having its notation and emphasizing its aspects for software construction. The Unified Modeling Language – UML (Booch, et al., 1999) is a Grady Booch, James Rumbaugh, Ivar Jacobson and other's work consolidation and has represented lately a “de facto” standard in the Software System modeling.

The Object Modeling Technique – OMT was developed by James Rumbaugh et al. and has been considered an Object-Oriented Methodology for Software Development (Rumbaugh et al., 1991). It is a well-known methodology for analysis and design and actually has been applied by scientific community. It is also a backbone for others methods such as OMT++ (Aalto et al., 1994), OMT* (Holz et al., 1995) and Unified Software Development Process (Jacobson et al., 1999), (Pons et al., 2000). Even though it does not provide much support for requirements elicitation (Hasselbring et al., 1998). Requirements are one the most important factor in the Software Engineering development, because expresses the customer's needs. The OMT could be tailored to cover this deficiency.

The proposed OMT / UML Methodology consists of the tailoring of the traditional OMT Methodology and UML standard comprising of 06 (six) Stages: Pre-Analysis, Analysis, System Design, Object Design, Implementation and Tests and Reengineering, emphasizing the quality aspects for Software Systems Development. A first stage was added to OMT/UML methodology, named Pre-Analysis, describing the system aspects that would facilitate its modeling in the Analysis stage.

This Pre-Analysis stage involves the development of the following steps:
- Context Description – this step describes the temporal and situational aspects involving the System in a textual narrative;
- System Objective – this step states the problem to be solved by System that will be developed and alternative chosen solution;
- Title – this step chooses the most appropriate title for the System. At this time a trade name and a acronym for the System is recommended to be chosen to facilitate its future references; and
- Requirements Specification – this step specifies the initial functional, not functional and quality requirements for the System.

It is good to remember that the four steps above were considered consistence amongst themselves.

Together with Integrated Computer Aided Software Engineering Environment I-CASE-E and Rapid Application Development - RAD tools new methodologies and standards represent, nowadays, the state of the art for complex and critical Software Systems development. Figure 1 shows the proposed OMT / UML Methodology.
The following sections describe the airport system features and some fragments of the application of the proposed OMT / UML Methodology tailored to the UML standard and the use of I-CASE-E and RAD tools. A short airport study case is presented in order to show the modeling technique foreseen by the methodology, looking forward Intelligent Airportuary Resource Allocation Software - IARAS.

3. The airport system features

The airport features resource allocation is defined as follows. There are a master timetable of flights and a resource's collection of the airport facilities, including gates (or ramp parking). Every arriving and departing flight must be assigned to gates satisfying various demands. The timetable of flights and resources are not fixed and experienced additions, cancellations and unexpected events may occur. The goal in solving this problem is to produce a schedule of gate's assignments, times processings and resource allocations to all flights, satisfying relevant constraints. These features allocation characterize scheduling problems, which also include job-shop, project management and crew scheduling (Jo et al., 1997). The airport system features could be found in other similar transportation systems such as bus, train and docks terminals.

In practice, professional schedulers manually make the schedules everyday. This requires domain-specific knowledge, experience, heuristics, considerable amount of time, and tedious paperwork to complete the scheduling. Gate allocation is subject to numerous operational constraints, as for example:

- No two aircraft can be allocated to the same gate simultaneously;
- Particular gates can be restricted to admit only certain aircraft types; and
- Airlines and ground handlers prefer to use particular gates or terminals (Henz et al., 2000).

Researchers have been using mathematics and operations research techniques to solve this kind of problem. However, it is very difficult to model constraints and domain knowledge with only
mathematical variables. Serious difficulties related with large-scale practical systems due to real-time operations support had been reported (Gosling, 1990), (Jo et al., 1997).

Many researchers have proposed the use of Artificial Intelligence to solve these problem (Brazile et al., 1988), (Gosling, 1990), (Jo et al., 1997). The implemented artificial intelligence systems are developed using software shells and appropriated specialized languages such as PROLOG and LISP. However, the object-oriented modeling aspects stressed on this paper are not approached in these systems.

4. Describing intelligent software system modeling

The Pre-Analysis Stage describes the main System aspects in order to reduce modeling efforts. The main intermediate software product produced was a Requirement Specifications, including also a Context Description, and the System Objective and Title. The identification of Macro-Functions or Global Use Cases heuristic representing main System functionalities has been also included in the Requirement Specifications work.

In the Analysis Stage, after defining Macro-Functions, the specialization process has continued by inserting Specific Use Cases into each Macro-Function, and by identifying possible Scenarios. In this Stage, the Use Cases involving the Intelligent System functionalities have been also mapped. Figure 2 shows a Use Case Diagram with its Macro-Functions and also those functions involving the Intelligent System behaviors.

In the System context, the more important Conceptual Elements have been used to create Business Class or Objects Prototypes that represent the basic abstractions. They have been identified with the aid of Sequence and Collaboration Diagrams, mapped later on into a Class Diagram to outlook and specify the System static structure.

Figure 2: The IARAS System Macro-Functions.

After this point, the Objects’ dynamic behavior has been mapped, with their non-trivial aspects, normally described in Activity and State Diagrams.

In the System Design Stage, the Physical Architecture mapping has been carried out by means of the Component Diagram distributing objects in physic containers. The distribution of these
containers among processors (hardware) was modeled by means of the Deployment Diagram.
In the Object Design Stage, the same Objects have been specified in terms of its attributes and
operations, getting close to the Implementation Language chosen.

Later on, in the Implementation and Tests Stage, Objects for code mapping has occurred. Then,
within the Reengineering Stage, a refining cycle between framework or synchronized code
generated for I-CASE-E tools has successfully performed, by using the TELELOGIC TAU 4.0
UML Suite, from TELELOGIC AB Enterprise, and the handy made code generated by using
RAD tools.

4.1 Intelligent software component

The Object-Oriented Software Engineering Paradigm has provided the mapping of reality by
means of Objects. The extension of this Paradigm has enabled Business Objects creation, which
has captured and incorporated Business Rules. The mapping of these Rules and its intelligent
processing have taken advantage of Intelligent Object using aggregation, by creating Business
Intelligent Objects.

For example, an Object representing an airport gate has been mapped together with its Intelligent
Object aggregated. These Objects later on have been mapped into a logical and physic structure
of an Intelligent Software Component, which could be reused in other applications, encapsulating
its Business intelligence processing. This type of Component was very similar to the JavaBeans
type of Java Components. This mapping is shown in Figure 3.

![Diagram](image)

**Figure 3: The Intelligent Software Component Mapping.**
5. Conclusion

As it is well known, information management for local airportary resources needs a more appropriate systematic for its evaluation to improve operational decision effectiveness.

This improvement directly relates to the development of a management tool that incorporates the operational necessities of airports into an Intelligent Software System.

The development of this System has followed modern standards of Information Engineering, Software Engineering and Knowledge Engineering, aiming to magnify the system reliability and quality, minimizing errors and faults occurrence probability.

The major finding of this article was the proposed tailoring of OMT Methodology to UML standard. This tailoring has been added two new Stages to the development process. The first one, right at the beginning, named Pre-Analysis, and the second one, right at the end, named Reengineering. The Pre-Analysis Stage has simplified the Requirements Specification work and the Reengineering Stage has also simplified the reusability and synchronization of legacy codes, by means of I-CASE-E and RAD tools. Mainly due to this contribution, the proposed Methodology extends its application potential.

Another important contribution from this article was the introduction of Intelligent Software Components as Business Rules intelligent processors, extending Components reusability.

The continuity of this research work points out to the incorporation of Dynamic Intelligent Software Components, applying the Intelligent Agents Paradigm.

References:


