DESIGN OF FORMAL AIR TRAFFIC CONTROL SYSTEM THROUGH UML

Shafeeq Ahmad
Azad Institute of Engineering & Technology, INDIA
ahmad_shafeeq@rediffmail.com

Vipin Saxena
Dr. B. R. Ambedkar University, INDIA
vsax1@rediffmail.com

ABSTRACT

In recent years, UML has become most popular among modeling languages and is commonly used to drive the design and implementation of system and software architectures. UML models help to achieve functional and non-functional requirements of system. Furthermore, UML tools have enabled the creation of source code from UML diagrams in order to initiate the programming phase of building software. However, due to lack of clearly defined semantics, it has been challenging to create source code from UML models. The main objective of the paper is to model Air Traffic Control system by the use of UML. An activity of Air Traffic Control i.e. departure process which only covers part of the Air Traffic Control functionality has been considered in this paper. The UML models created using formal naming semantics help them to convert into source code and also help to achieve functional and non-functional requirements. The complexity of Air Traffic Control System is also measured which makes the design simple and visibly understandable.

Keywords: UML model, formal semantics, source code, Air Traffic Control.

1 INTRODUCTION

Nowadays Object Oriented software development process is widely used in the Software Industry. The emergence of Object-Oriented programming has heavily contributed toward a standardized method of modeling known as the Unified Modeling Language (UML). In recent years, UML has become synonym for software modeling and is commonly used to model the software architecture problems. Source code can be easily be generated with the help of different UML diagrams for building the software. To generate the correct source code, the main problem is lack of clearly defined semantics and code generation can only be done if the UML specification is standard, complete, precise, and unambiguous. The present work is based upon the ATC system which is explained below:

1.1 ATC System

ATC system is a service that gives guidance to aircraft, prevents collisions and manages safe & orderly traffic flow. It is a vast network of people & equipment that ensures safe operation of aircrafts. The first air traffic control (ATC) system was originally built in the 1960s; since then, air traffic has increased immensely, and has become increasingly more difficult to maintain safety in the sky. As air travel has become an essential part of modern life, the ATC system has become strained and overworked. The ATC system has been in a process of continuous improvement / change. In the earliest days of aviation, few aircraft were in the skies that there was little need for automated control of aircraft. As the volume of air traffic increased and the control was still fully manual; the system was considered unsafe as human error has been cited as a major factor in the majority of aviation accidents and incidents.

In today's Air Traffic Control system, air traffic controllers are primarily responsible for maintaining aircraft separation. Every aircraft follows several activities during a flight. These activities are shown below in Fig.1:

(i) *Preflight* -This portion of the flight starts on the ground and includes flight checks, push-back from the gate and taxi to the runway.

(ii) *Takeoff* - The pilot powers up the aircraft and speeds down the runway.
(iii) *Departure* - The plane lifts off the ground and climbs to a cruising altitude.
(iv) *En route* - The aircraft travels through one or more center airspaces and nears the destination airport.
(v) *Descent* - The pilot descends and maneuvers the aircraft to the destination airport.
(vi) *Approach* - The pilot aligns the aircraft with the designated landing runway.
(vii) *Landing* - The aircraft lands on the designated runway, taxis to the destination gate and parks at the terminal.

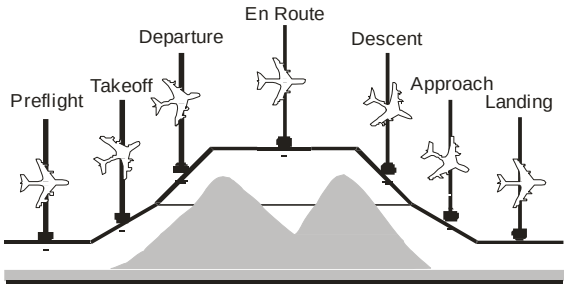

Figure 1: Activities of Aircraft [6]

1.2 Drawbacks of Current ATC System

ATC systems are highly complex pieces of machinery, they employ standard verification and modeling technique to coordinate, distribute and track aircraft as well as weather information. The currently used systems need to employ procedures for improved safety and efficiency which include flexibility, potential cost savings & reduction in staffing. This means that there is a lack of advanced technology and desire to support the controller. Thus there is a need to build ATC system based on a method which can handle increased air traffic capacity/congestion to provide a safety critical interactive system. The following are the major drawbacks of current ATC system:

(i) *Lack of well-defined human/software interface* – The idea of full automation or minimum human intervention of the ATC system still remains unfulfilled. The existing systems do require human interaction as the system only guides but actual decision is taken by the controllers in-charge (ground, local).
(ii) *Need for high maintenance* – Maintenance of the system is also an issue which can cause problem as discussed by Matthew L. Wald [23] about an incidence in which voice communication between the pilot & controllers broke down & the reason behind this was found to be a lack of maintenance.
(iii) *Outdated design/technology* – Obsolete software design and programming language [11] are major barriers to upgrades and efficient software maintenance of the currently used ATC systems because of which improved capacity and efficiency can't be achieved with the current system. The current computer software limits the number of aircraft that can be tracked at any given time, and the dated architecture makes enhancements, troubleshooting and maintenance more difficult. Computer outages, planned or unplanned, are covered by a backup system that cannot handle the same level of air traffic as the main system. The result is significantly limited capacity during backup mode.
(iv) *Mixed communication* – The communication between the controllers & pilot currently is a combination of voice & datalink. The results of test conducted [18] show that the mixed communication leads to slow speed which can be overcome only when the whole communication takes place in a well defined manner .

2 RELATED WORK

The ATC system consists of controllers & technology. The various controllers involved have different tasks assigned to them which have been very well described in [15], [16], [19], [20]. The ATC real-time system [19] is characterized as complex, time-driven and potentially distributed. It is composed from multiple sub-systems, which must cooperate in order to complete its real time targets. ATC employs the approach of Parallel Applications in which the simultaneous execution of some combination of multiple instances of programmed instructions and data on multiple processors are performed in order to obtain results faster. Meilander [4] has proposed a solution for problems arising due to parallel processing used in ATC but its implementation needs reliable software. The reason for system failure discussed in [8] was found to be lack of maintenance. For the efficient software maintenance, [11] emphasizes changes particularly on the technology front which means modernizing the current air traffic control system by replacing the software used in the current ATC systems. The study done by Verma[18] by implementing changes in the procedures, roles and responsibilities of the controllers for redistribution of workload and communications among them, has also stressed on the introduction of new automated technologies for the controllers.

For a critical system like ATC an extremely low probability of failure is needed as if the system fails any related type of hazard can occur. The reports on incidences at non-controlled airports [5] show that

pilots can't entirely rely on vision to avoid collision & also it is necessary to get all the air-traffic related information correctly. So it is crucial that in such large and extremely complex systems the software is designed with a sound architecture A good architecture not only simplifies construction of the initial system, but even more importantly, readily accommodates changes forced by a steady stream of new requirements. This architectural construct can be derived from modeling concepts by using the powerful extensibility mechanisms of UML [1]. The UML model-based approach helps to manage critical systems, since the model support the necessary analysis activities in several ways:

- The formalized structural and behavioral system description gives the necessary basis for criticality analysis.
- Providing behavior is expressed in simple state charts or "English-like" pseudo code, the behavior should at least be explainable to end-users.
- The model gives an excellent basis for fault-tolerance analysis, since the model includes the dependency structure.

The size and complexity of the ATC system demand a considerable initial development effort, typically involving large development teams, that is followed by an extended period of evolutionary growth. During this time new requirements are identified and the system is modified incrementally to meet them. Under such circumstances an overriding concern is the architecture of the software. This refers to the essential structural and behavioral framework [1] on which all other aspects of the system depend. Any change to this foundation necessitates complex and costly changes to substantial parts of the system. Therefore, a well-designed architecture is not only one that simplifies construction of the initial system, but more importantly, one that easily accommodates changes forced by new system requirements.

To facilitate the design of good architecture the domain-specific usage can be implemented using the UML. Object-oriented requirements analysis using modular and decomposable use cases provide to be very powerful method for the analysis of large-scale ATC systems [6]. It is extremely useful to define both the static semantics (i.e., the rules for well-formedness) and the dynamic (run-time) semantics of the ATC system. These semantic rules fully defined and consistent, and in conjunction with an action specification language that is also complete and consistent can be used to specify the details of state-transition (Activity diagram of UML) actions, Interaction (Sequence diagram of UML) and object methods (class diagram of UML) and the resulting models will be executable. Furthermore, if all the necessary detail is included in a model, such model can be used to automatically generate complete implementations.

The role of software architecture is similar in nature to the role architecture plays in building construction. Building architects look at the building from various viewpoints useful for civil engineers, electricians, plumbers, carpenters and so on. This allows the architects to see a complete picture before construction begins. Similarly, architecture of a software system is described as different viewpoints of the system being built. These viewpoints are captured in different model views. UML provides a number of diagram types for creating models. UML does not specify what diagrams should be created or what they should contain, only what they can contain and the rules for connecting the elements. Some UML diagrams can have different uses and interpretations. The result of this has been that behavioral and/or run-time semantics are not well defined for standard UML modeling elements. The semantics problems have made it difficult to achieve model-based programming using standard UML. Hence semantically correct UML models [12], [17] are needed to achieve code. Also the UML models enhance communication as they provide a better way to understand and communicate among analysts, designers, developers, testers and with domain experts for designing a system. Creating and debugging software code has been and continues to be a very labor-intensive and expensive process but having an accurate UML model can ease the work as if any problem comes and modification is required then instead of modifying the code the models can be modified, even the extensibility can be done by adding new constructs to address new development issue.

To meet the non-functional requirements [13] such as modifiability, testability and reliability, the design and analysis of a system at the architecture design level must be done. Non-functional requirements have a critical role in the development of a software system as they can be used as selection criteria to help designers with rational decision-making among competing designs, which in turn affects a system's implementation .Thus the ATC system needs to provide a sufficient amount of dependability and should support a survivability architecture [10].

In this paper we examine the most important modeling constructs used for representing the ATC system handling departure and also describe how they are captured and rendered using UML. The models are based on the work presented by Saxena & Ansari [21].The operations, phases and controllers involved during departure of an aircraft have been described in [2], [3], [7], [9], [10], [14], [15], [19] &

[20] which have been used as base for designing various UML models.

3 PROPOSED APPROACHED

The existing systems are quite complex and inefficient. To adapt to the changing demands of speed and efficiency a reliable software system for ATC is required to be developed. Software architecture based on UML models will help in handling complexities and drawbacks of existing ATC systems and also help to better understand the domain. UML is the de-facto standard visual modeling language which is a general purpose, broadly-applicable tool supported, industry-standardized modeling language which offers an extensive set of diagrams for modeling. The complexity of the problem domain requires extensive efforts for the clarification of the initial problem statement. Moreover, due to the extremely long lifespan of ATC systems, stable and robust analysis models enabling the integration of new operational scenarios are needed which can be efficiently obtained using UML models. In the design of a departure activity of ATC system UML will help to meet safety, reliability, availability, and robustness demands in an environment of steadily increasing air traffic density. The code obtained using the UML models is highly optimized which is one of the main requirements for the design of a cooperative ATC system.

4 DESIGN OF ARCHITECTURE

System architecture is a set of design decisions. These decisions are technical and commercial in nature. To meet the functional and nonfunctional requirements of the above said ATC system it is necessary to model the complete ATC system by the use of UML. Different types of diagrams are designed and described below in brief:

4.1 UML Activity Diagram

Activity diagram describes the workflow behavior of a system. It illustrates the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation of a class, resulting in system state change.

The informal activity diagram for departure of an aircraft is given below in Fig. 2. It simply describes the main activities performed during departure by the controller and the aircraft object. In this diagram some semantic problems are present like if any clearance has not been granted then what will be done is not clear i.e. use of decision box hasn't been done. Further all the activities have been shown in a single step and no refinements have been done to simplify the activities. These loose semantics of the diagram make this model unable to be executable, therefore the formal activity diagram for departure of an aircraft is defined and shown in Fig. 3

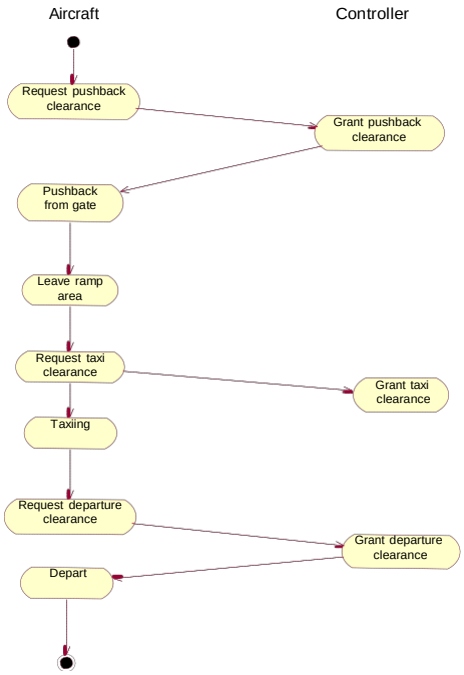

Figure 2: Informal UML activity diagram of departure activity of a flight

Formal activity diagram [fig.3] contains five objects aircraft, Gate_controller, Local_ controller, Ramp_controller & Ground_controller. It describes various activities which are performed by all the objects. While these activities/operations are being done, it also shows in which state the system is like ready for departure, taxiing etc. There is a request by the pilot of the aircraft to the gate controller to assign

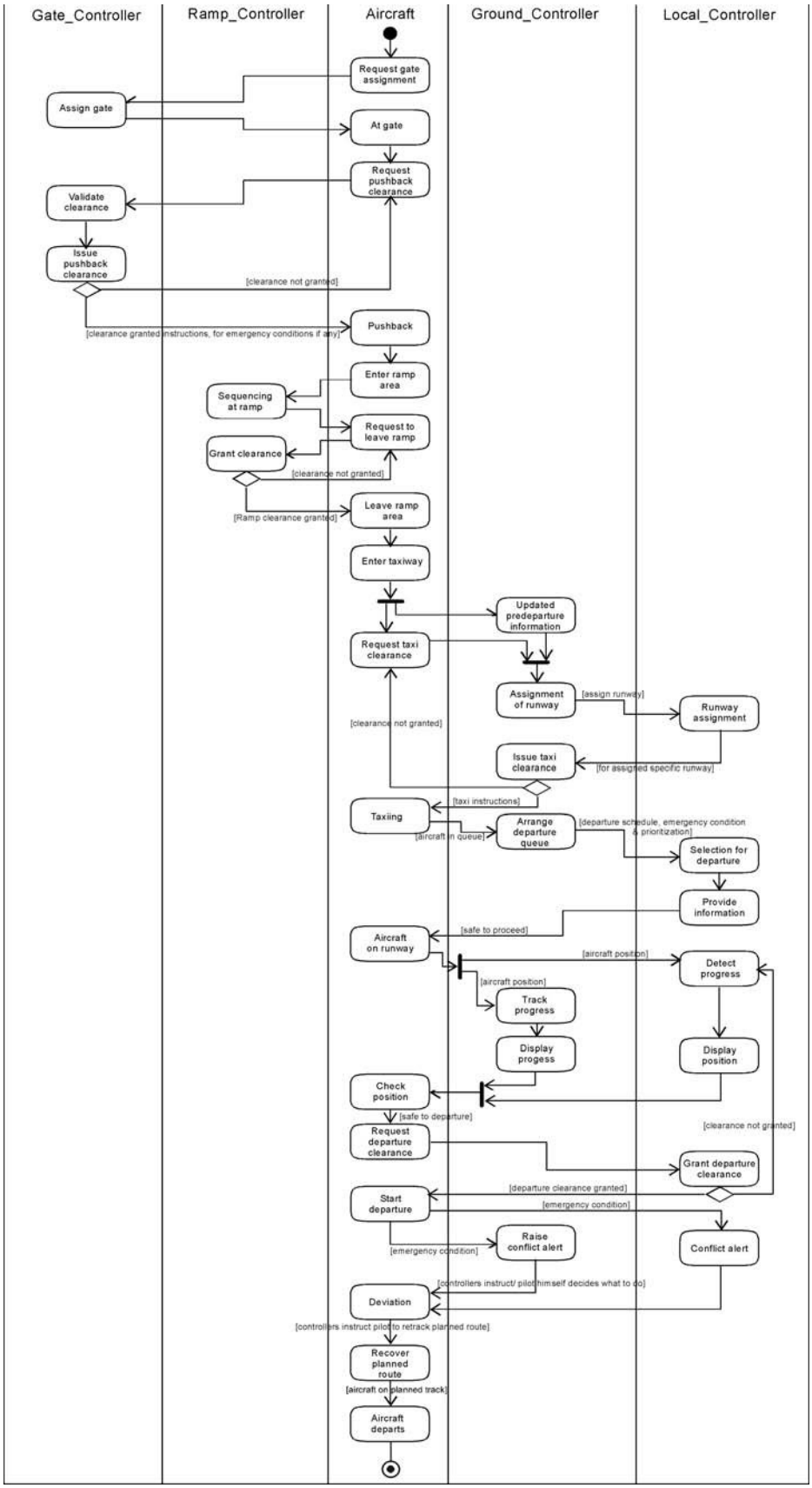

Figure 3: Formal UML activity diagram of departure activity of a flight

it a gate which is the initial state & the final state is named as depart. When the next activity is performed the system is taken to the next state that means a transition causes the change of state. Against the transitions various conditions are listed which actually cause the transition. Also the use of decision box is applied which is required to check whether the various clearances like pushback, taxiing & final departure clearance has been granted or not. If clearance has been granted then only the aircraft starts pushing back or departing else if because of some reason clearance could not be granted then it will again go to the request clearance state where it will be seen & tracked until it is safe for pushback or departure. During the whole departure activity the possibilities of any emergency condition is also taken into consideration and shown in the diagram. Hence the solid and well organized semantics like conditions for transitions, use of swimlanes, fork, join, method calls from objects i.e. able to call methods from other objects, decision condition specification etc. have been used in this diagram which make this model executable.

4.2 UML Use Case Diagram

The use case diagram captures system functionality as seen by users. It shows an outside-in view of the procedures available in the use of the system i.e. all the available functionality of the system is represented at a high level. The UML models can package the most relevant activities in use cases and identify important actors. The UML use case diagram for departure activity of ATC system is shown in Fig.4.

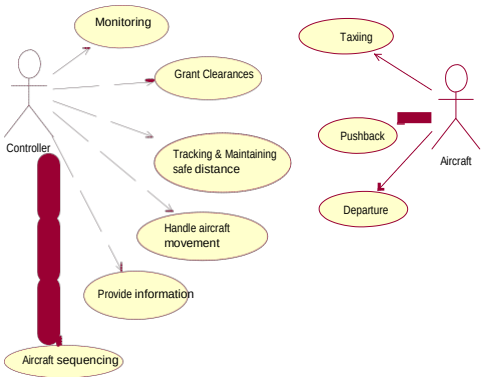

Figure 4: Informal UML use case diagram of departure activity of a flight

This diagram consists of two actors namely controller and aircraft. The use-cases consists represent the functionality of both actors. But this use case diagram provides an informal viewpoint as it lacks concrete, well defined semantics which make it impossible to help it in bringing to a computational consistency, therefore formal use case diagram is drawn for departure activity of a flight and shown below in Fig. 5.

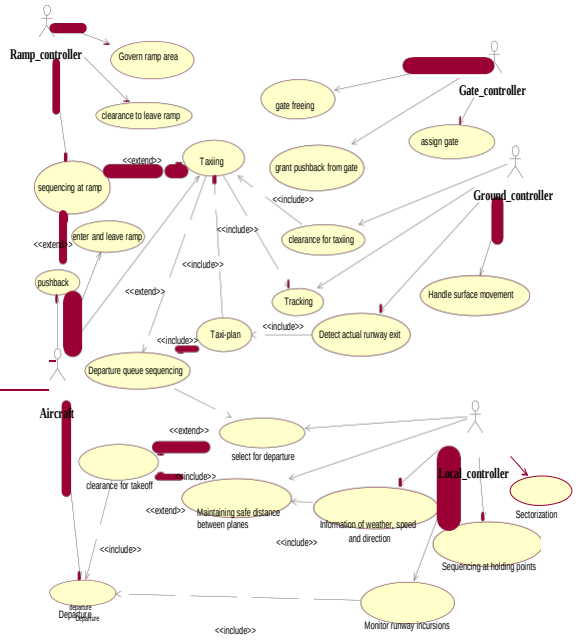

Figure 5: Formal UML use case diagram of departure activity of a flight

In the above model there are all five actors namely Aircraft, Ground_controller, Ramp_controller, Gate_controller & Local_ controller which interact with various use cases. The use cases clearly describe various main functions during departure and whether they are dependent on each other directly or indirectly denoted by extended or included keywords. It explains actually which functions are governed by which actors. This diagram provides the functional basis for creating the remainder of the diagrams needed to arrive at an executable diagram as it has well defined semantics like identifying every actor involved during departure of a flight, specifying all the functions performed by each actor and use of extend or include keyword to identify how one function of an actor is related to another. This formal use case diagram also drives the design of the class diagram, and sequence diagram.

4.3 UML Class Diagram

Class diagram identifies & describes the static structure of the system i.e. the system architecture.

The following Fig.6 shows the informal class diagram of the departure activity of a flight.

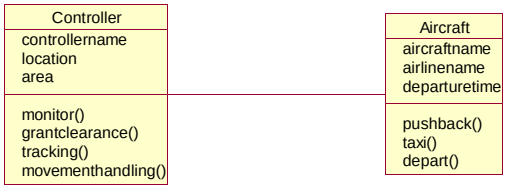

Figure 6: Informal UML class diagram of departure activity of a flight

This diagram gives a general view of the classes involved during departure namely the controller and the aircraft class. In this diagram only the name of the attributes and the operations of the classes have been specified. Even the attribute type or the return type of the methods hasn't been described. These loose semantics make this model impossible to compile and execute. Now let us examine a formal class diagram of departure activity of a flight shown below in Fig.7

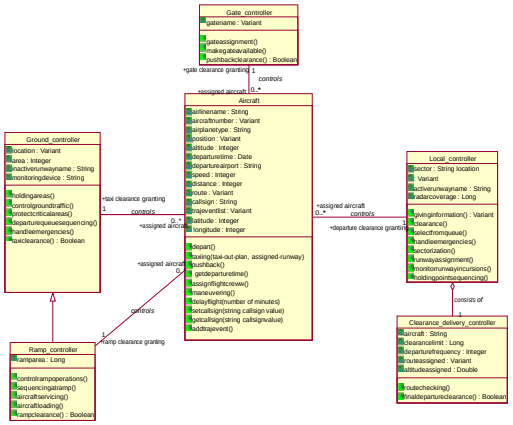

Figure 7: Formal UML class diagram of departure activity of a flight

This class diagram of the departure activity of a flight described in Fig. 7 focuses on the main objects and associations (relation) between them. During departure main objects are Gate_controller, Ramp_controller, Ground_controller, Local_controller and Aircraft itself. The Local_controller in turn consists of the Clearance_delivery_controller which gives the final departure clearance. Also a controller called Ramp_controller is present which is just a type of ground controller which is responsible for operations at ramp (parking) area. All the objects in turn have some distinct attributes (properties) and some operations which are listed in the 2nd and 3rd section respectively of the class diagram. The type of attributes and return type of operations have been clearly specified in the diagram. This class diagram has covered almost all the explicitly defined semantics like class relationships, multiplicities, associations, properly labeled relationships, strict naming of classes, attributes, and methods, explicit parameters/return values which are necessary in order to allow the model to be computationally executable.

4.4 Sequence Diagram

A sequence diagram is an interaction diagram that details how operations are carried out, what messages are sent and when. Sequence diagrams have a temporal focus so they are organized according to time which means what happens when.

The informal sequence diagram of departure activity of a flight is shown below in Fig 8. It simply gives a sequence of messages between the controller

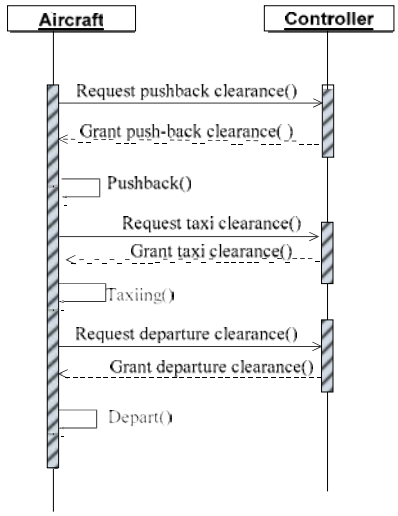

Figure 8: Informal UML sequence diagram of departure activity of a flight

and the aircraft objects. But this model doesn't include the semantics necessary for making it executable as no parameter values are specified, no guards have been used for any decision point, even the class definition is not necessarily present in the called class. Hence this model can not be compiled and executed, therefore a formal sequence diagram is defined & shown in Fig.9.

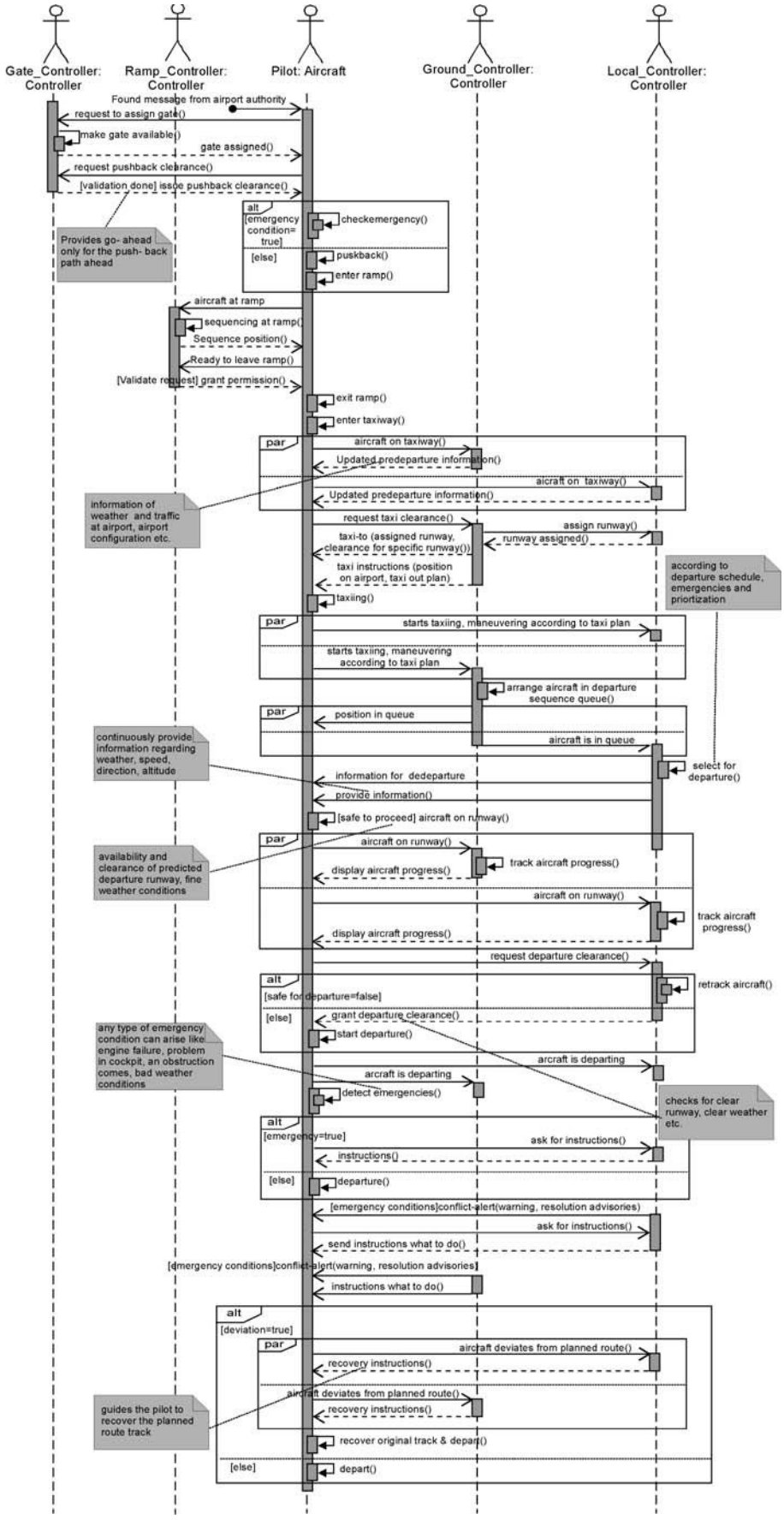

Figure 9: Formal UML sequence diagram of departure activity of a flight

In Fig.9 complete departure activity of a flight can be seen at a glance. The messages are communicated between the five main objects; aircraft, Gate_controller, Ramp_controller, Local_controller and Ground_controller. It clearly describes the interaction between the objects and gives the order or sequence in which the actions take place i.e. in what order messages have been send. The solid lines represent the call messages while the dotted lines represent the messages returned between the objects. The tags are added to clearly specify about the operation. This model consists of all the clear semantics like use of tags, explicit parameter values, explicit return values, combined fragments(alternatives and parallel), guards needed for decision points and all the class definitions exist in the called class. Thus this formal sequence diagram with well defined and solid semantics makes it able to be compiled and computationally executable.

5 CONCLUDING REMARKS

From the above work it is concluded that the described software architecture design process fully covers the departure activity of an aircraft. UML modeling has been used for making the various models (class, use-case, activity & sequence) using informal and formal semantics. These designed models are essential part of architecture of ATC software system and that can help to achieve executable code and other functional and non-functional requirements of software system.

6 REFERENCES

[1] B. Selic, and J. Rumbaugh, “Using UML for Modeling Complex Real-Time Systems”, March 1998. Available: http://www.ibm.com/developerworks/rational/library/content/03July/1000/1155/1155_umlmodeling.pdf

[2] I. Anagnostakis, H. R. Idris, J. P. Clarke, E. Feron, R. J. Hansman, A. R. Odoni, and W. D. Hall, "A Conceptual Design of A Departure Planner Decision Aid", 3rd USA/Europe Air Traffic Management R & D Seminar, June 13-16, 2000. Available: http://dspace.mit.edu/bitstream/1721.1/37321/1/paper68.pdf,

[3] H. Idris, J. P. Clarke, R. Bhuva, and L. Kang, “Queuing Model for Taxi-Out Time Estimation”, Sep 2001. Available: http://dspace.mit.edu/bitstream/1721.1/37322/1/TaxiOutModel.pdf

[4] W. C. Meilander, M. Jin, and J. W. Baker, “Tractable Real-time Air Traffic Control Automation”, International Conference on Parallel and Distributed Computing Systems, pp. 477-483, November 4-6, 2002.

[5] G. Brown, “Remote Intelligent Air Traffic Control Systems for Non-controlled Airports”, Jan. 2003. Available: ww4.gu.edu.au:8080/adt-root/uploads/approved/adt-QGU20040225.084516/ public/02Whole.pdf,

[6] J. Whittle, J. Saboo, and R. Kwan, “From Scenarios to Code: An Air Traffic Control Case Study”, 25th International Conference on Software Engineering (ICSE'03), pp. 490, 2003

[7] Safety Regulation Group, “Air Traffic Services Information Notice”, ATS Standards Department, Civil Aviation Authority, no. 50, Aug. 2004.

[8] M. L. Wald, “Maintenance Lapse Blamed for Air Traffic Control Problem” The New York Times, Sep 2004. Available: http://www.nytimes.com/2004/09/16/politics/16airports.html

[9] M. Axholt, and S. Peterson, “Modelling Traffic scenarios for Realistic Air Traffic Control Environment Testing”, Linkoping University Press, Department of Science & Technology, Linkoping University Sweden, Nov. 2004.

[10] R. D. Lemos, C. Gacek, and A. Romanovsky, “Architecting Dependable Systems II”, Springer-Verlag Berlin Heidelberg, 2004.

[11] ATCA Air traffic Control Association, “Air Traffic Control Computer Modernization is En Route”, Feb 2005. Available: http://www.atca.org/news.asp

[12] Object management Group, "Unified Modeling Language: Superstructure", Version 2.0, Formal/05-07-04, Aug 2005, Available: http://www.omg.org/docs/formal/05-07-04.pdf

[13] L. Dai, and K. Cooper, “Modeling and Analysis of Non-functional Requirements as Aspects in a UML Based Architecture Design”, Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN’05) , IEEE 2005.

[14] European Organisation For The Safety Of Air Navigation (Eurocontrol), “WP 5.2 – Add NFR To Use Cases And Interaction diagrams NFR Mapping Report”, June 2006.

[15] P. A. Bonnefoy, "Airport Operations ", MIT, Air Traffic Control, MIT International Center for Air Transportation, Sep 2006

[16] R.J. Hansman, “Air Traffic Control Overview”, MIT Department of Aeronautics and Astronautics, MIT ICAT, Available: http://ocw.mit.edu/NR/rdonlyres/Aeronautics-and-Astronautics/16-72Fall-2006/11CD67DE-29C1-4AB9-A7DD-004BB1897CB9/0/ lec1.pdf

[17] R. Campos," Model Based Programming: Executable UML With Sequence Diagrams ", A Thesis Presented to The Faculty of the Computer

Science Department, California State University, Los Angeles, June 2007

[18] S. Verma, T. Kozon, V. Cheng, and D. Ballinger, "Changes In Roles/Responsibilities of Air Traffic Control Under Precision Taxiing", 26th IEEE/AIAA Digital Avionics Systems Conference, Oct 2007

[19] Wikipedia, "Air traffic control", October 2007. Available: http://en.wikipedia.org/wiki/Air_traffic_control

[20] L. Brim, " Fundamentals of Air Traffic Control", ParaDise Seminar, Feb 2008, Available: http://www.fi.muni.cz/paradise/Seminar/2008-spring/0225-prezentace.pdf

[21] V.Saxena, and G.A.Ansari, "UML Models of Aircraft System", ICFAI Journal of Systems management, Vol. 6, No.2, PP. 68-74, May 2008