Hindawi
Modelling and Simulation in Engineering
Volume 2018, Article ID 1692432, 13 pages
https://doi.org/10.1155/2018/1692432

Research Article

Towards Sophisticated Air Traffic Control System Using Formal Methods

Abdessamad Jarrar and Youssef Balouki

Faculty of Sciences and Technologies of Settat, Computing, Imaging and Modeling of Complex Systems Laboratory, Hassan 1st University, Settat, Morocco

Correspondence should be addressed to Abdessamad Jarrar; abdessamad.jarrar@gmail.com

Received 1 April 2018; Accepted 8 July 2018; Published 10 September 2018

Academic Editor: Jing-song Hong

We propose a general formal modeling and verification of the air traffic control system (ATC). This study is based on the International Civil Aviation Organization (ICAO), Federal Aviation Administration (FAA), and National Aeronautics and Space Administration (NASA) standards and recommendations. It provides a sophisticated assistance system that helps in visualizing aircrafts and presents automatic bugs detection. In such a critical safety system, the use of robust formal methods that assure bugs absence is highly required. Therefore, this work suggests a formalism of discrete transition systems based on abstraction and refinement along proofs. These ensure the consistency of the system by means of invariants preservation and deadlock freedom. Hence, all invariants hold permanently providing a handy solution for bugs absence verification. It follows that the said deadlock freedom ensures a continuous running of a given system. This specification and modeling technique enable the system to be corrected by construction.

1. Introduction

Despite the advancements in technology and science, air traffic management still remains one of the most complex critical safety problems. It is ensured by interaction between human and technical system [1]. Many technical systems were built up and developed to assist controllers to do their job successfully. Although, several of these systems operate very well, it is not possible to ensure that they will not stop working at a certain moment. This possible shortcoming may be due to some design errors. To overcome these errors, this paper develops a formal model of an assistance system for air traffic management. According to the ICAO, FAA, and NASA standards and recommendations, this contribution aims to provide a well-structured model useful in future systems and for reverse engineering.

Few works take, however, different tasks; some of them are interested in "taking off" [2, 3], while others are interested in "landing" of aircraft [4]. This research may claim to be one of the formal modelings of the ATC system which at the same time considers both tasks. It is, however, unreasonable to separate "landing" from "taking off" since these two operations occur in the same airport sharing the same runways although different aircrafts may be involved.

This paper specifies the system's functional and nonfunctional requirements. The functional requirements describe the behavior of the system, whereas the nonfunctional requirements specify some other properties such as security and safety. These requirements are formalized using Event-B, which is a formal method for software and system design. Event-B method is based on both refinement and mathematical language, this is exactly what motivates us to choose this method.

During development, we establish proofs called proof obligations to ensure the correctness of each model before and after refinement. Invariants preservation and deadlock freedom are the most important proof obligations. The combination of these two types of proofs provides "correct by construction" system.

The rest of the paper is structured as follows. Section 2 gives some background on formal methods and validation tools that we use. The main content of the paper is in Section 3,

describing our approach to develop the air traffic control system along three models. The first one includes the essence of air traffic management. The second presents how the system schedules taking off and landing of aircrafts. The last model introduces the nonfunctional requirements. Section 4 presents proof statistics that are generated by the Rodin platform. Section 5 concludes the paper.

2. Background

2.1. Related Works. To avoid undesirable delay in flights at the airports during the departure and arrival processes of aircrafts, Yousaf et al. [3] propose a systematic modeling process using a "VDM++" as a methodology. Aiming to ensure safety and accuracy, the method identifies and anticipates errors at the first levels and stages of system designing. It likewise offers an extremely valuable solution of problem and improves the confidence of the quality of the software. Nevertheless, the "VDM++" is not sufficient and adequately analyzed, so it is considered as an abstract process, and its real concretization is impossible.

Similar to the last approach, Zafar [2] combines a VDM-SL and graph theory to build a formal specification of aircrafts' take-off procedure. This formal specification of graph-based model, taxiways, aircrafts, runways, and controllers is provided in the static part of the model. The state space analysis describing takeoff algorithms is provided by defining optimal paths and possible operations in a dynamic model expediting the departure procedure. The model is developed by a series of refinements following the stepwise development approach. Although this work presents a detailed specification of the departure procedure, it requires further investigation to real-time management that is a major factor in this procedure. On the other hand, Méry and Singh [4] introduce a formal model of an aircraft landing system. This work is considered as a benchmark for techniques and tools dedicated to the verification of behavioral properties of the landing system. However, it neglects the procedure of landing which must be taken into consideration to ensure system safety and focus more on the mechanical system.

The most important work in this area is the UK National Air Traffic Services' iFACTS system [5]. This is also a system developed using the correctness-by-construction paradigm. The system was developed from a formal specification in Z using the SPARK technology. It has been in daily use for some years, managing UK airspace, and it has operated without error. The method used in this paper is a developed version of the Z specification language which provides more possibilities.

In this work, we aim to present a formal modeling and verification of an ATC system, considering aircrafts departure and landing side by side. This model ensures the consistency between the two procedures; however, we see that formalizing taking off and landing separately ignore the fact that these operations occur at the same airport and share the resources (runways, airport airspace, etc.). Therefore, they must be modeled together.

2.2. Formal Methods. Modeling is the process of representing a real-world system as a graphical representation or mathematical equations. It is a phase before building a system that clarifies its structure. Modeling helps also to study and test some system properties to reduce the risk of failures. For example, if we are building a self-driving car system, we can ensure theoretically that these two cars will never collide. Modeling can be done by means of graphical representation or mathematical equations. The first one is easy to read and can visualize the system structure better, which means it can be shown to clients. However, it is not very powerful at verifying bugs absence, and it is inaccurate. On the other hand, mathematical representations need mathematical knowledge to be read and cannot be shown to most clients. It is also harder to establish due to mathematical difficulties such as proofs and accuracy. Despite all this, it is very powerful for detecting errors and ensures a strong assurance of bugs absence. Therefore, we use modeling based on mathematical equations called formal methods.

These methods were originally developed for specifying and verifying the correct behavior of software and hardware systems and have been applied in many system development fields, and many achievements have been made [6]. Modeling a system using a formal method and proving system correctness require mathematical knowledge and accuracy. In some cases, it requires a lot of time to completely model a complex system. However, the strong assurance of bugs absence is very important especially in safety-critical systems such as air traffic management.

There are a variety of formal methods available:

(i) Hoare logic is a formal method for reasoning on computer program correctness where specifications are of the form {P} procedure {Q} [7].

(ii) Petri nets are a graphical and mathematical modeling tool applicable to many systems. It is often used to describe and analyze information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and stochastic [8].

(iii) The Language Of Temporal Ordering Specification (LOTOS) is a formal description language developed by the International Standard Organization (ISO) for open system formal specification. Systems in LOTOS are specified by drawing the temporal relation between interactions establishing the discernible behavior of system [9].

(iv) The Z language is a specification language based on predicates. The specification of invariants and the specification of operations have the form of a predicate [10].

(v) The B method (development of the Z language) is a method of software development based on an abstract machine notation used in the development of computer software (B language) [11].

(vi) Event-B is a formal method for software and system design. This method is based on refinement and

proof obligations, which ensures a strong assurance of bugs absence [12].

In Event-B, a system is developed as sequence of models. These models are enriched in successive steps by adding more details; this is called refinement. In other words, we start with a very abstract model called initial model, and then, we refine it to get more concrete models. These models are made up of contexts and machines. Contexts are the static parts of models; they are presented in terms of sets, constants, and axioms, whereas, machines are the dynamic part of models. In a machine, variables describe the current status of a system; these statuses are constrained by invariants. Invariants are the necessary properties that must be preserved during system function. Status transitions are described by events, which are a set of actions. Each action changes the value of certain variable. Events may have some necessary conditions to be triggered; these conditions are called guards. Figure 1 illustrates the process of development in Event-B.

To ensure system correctness, some properties must be proved. These proofs are called proof obligations. They include invariants preservation, which ensure that all invariants are permanently obeyed during system function. They include also deadlock freedom to ensure that the system will never be in a status where no guard is verified, which means no event can be triggered. Most of these proofs are done automatically by means of the platform Rodin. Rodin is a tool that supports the application of the Event-B formal method. It provides core functionality for syntactic analysis and proof-based verification of Event-B models [14]. In some cases, Rodin may need to be manually guided to prove some properties. This is done by indicating some hypotheses that Rodin must consider. These hypotheses may be ignored by Rodin because he sees that it is not needed for the desired proof. In other cases, we add hypothesis and then Rodin prove the desired property; after that, we prove separately the added hypothesis.

2.3. Code Generation. Although developing a system using formal method reveals future failure and improves security, it is highly desirable to be able to translate this modeling to a code. Most of works in this sense such as [15] presented a method for generating Java code based on Event-B model. In [15], authors develop EB2J, a software tool that translates Event-B models into Java code which can be used in our case to develop a proof-based Java program for ATC management. Figure 2 illustrates the EB2J architecture.

3. Formal Development

This section presents the proposed approach aiming to develop the air traffic control system based on

(i) ICAO [16] which provides strategic objectives concerning safety, capacity and efficiency, security and facilitation, economic development, and environmental protection;

(ii) FAA [17–19] which has a predetermined number of air traffic manuals, publications, and orders;

(iii) NASA [20] standards and recommendations considered as the main constraints in this modeling in order to provide a system with maximum feasibility.

This development is designed progressively by starting with an abstract model that captures the essence of traffic management and integrating more details in successive steps. This activity is called refinement technique. The first refinement introduces the scheduling method used during taking off and landing. The said scheduling method assigns priority of taking off using FCFS (first comes first served) and the priority for landing based on deadline monotonic.

Moreover, this complex system gives the highest priority to emergency situations such as medical and terroristic threats.

The proposed model is based on one runway. However, this model maximizes the use of one runway to land and takeoff aircrafts while maintaining deadlines as much as possible [21].

The second refinement introduces safety properties which strongly avoid issues that may cause serious disasters. For example, a minimum separation landing time must be respected in order to maintain aircrafts' aerodynamic stability [22].

3.1. Initial Model: An Abstract Model of the Landing Process. In this initial model, we introduce the essence of the ATC system and the different components taken into consideration [23]. The first component is the runway which is, according to the International Civil Aviation Organization (ICAO), a rectangular area on a land aerodrome prepared for the landing and takeoff of aircrafts [16]. Runways are equipped with lights indicating their status; these lights are called runway status lights (RWSL). The RWSL system was developed by the Federal Aviation Administration (FAA) to improve air crew and vehicle operator situational awareness. These lights are embedded in the pavement of runways and taxiways and turn red when it is not safe to enter for a certain reason [17–19].

This paper develops a system for ATC to manage aircrafts' traffic in the vicinity of the airport airspace. Hence, it focuses only on status lights' modelization of the runway due to their relation to the airspace traffic management [22].

The first proposed model is made up of two parts: static part and dynamic one [12]. The static part (called context) contains carrier sets, constants, and associated axioms, whereas the dynamic part (called machine) contains variables, invariants, and events. In the first context, we introduce the carrier set RW_STATUSES corresponding to the possible statuses of the runway {available, unavailable} (axm1), as for RWL_STATUSES represents runway lights statuses {ON, OFF} (axm2). The AIRCRAFTS set denotes all possible aircrafts that might exist (currently or in the past or even in the future) which is axiomatized to be finite (axm3).

For each aircraft in the radar range, a significant status is associated which is proposed and introduced to help controllers for distinguishing between aircrafts landing, taking off, entering airport, waiting for landing clearance, etc. [20]. When an aircraft enter the aircraft vicinity, the Blocked

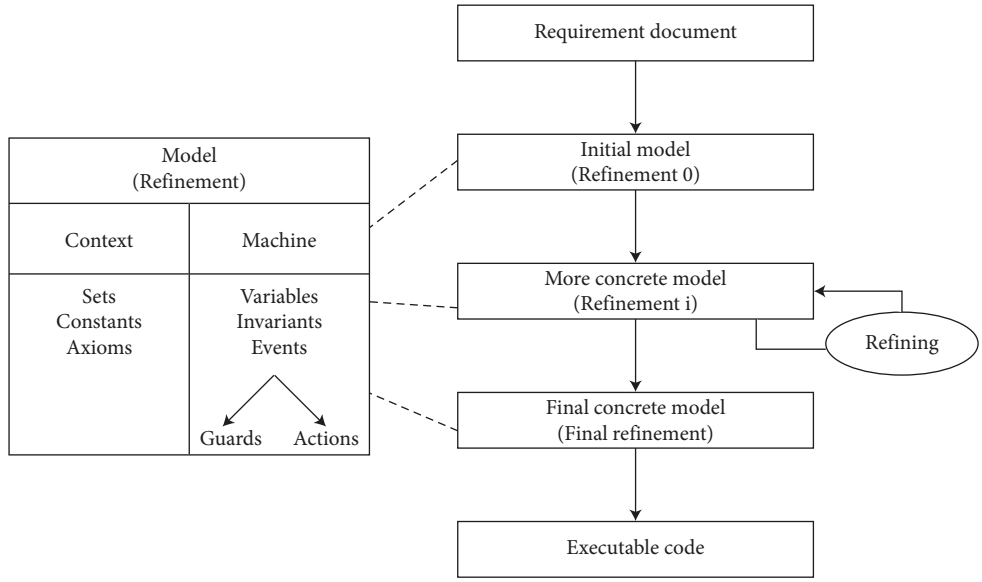

Figure 1: Process of development in Event-B [13].

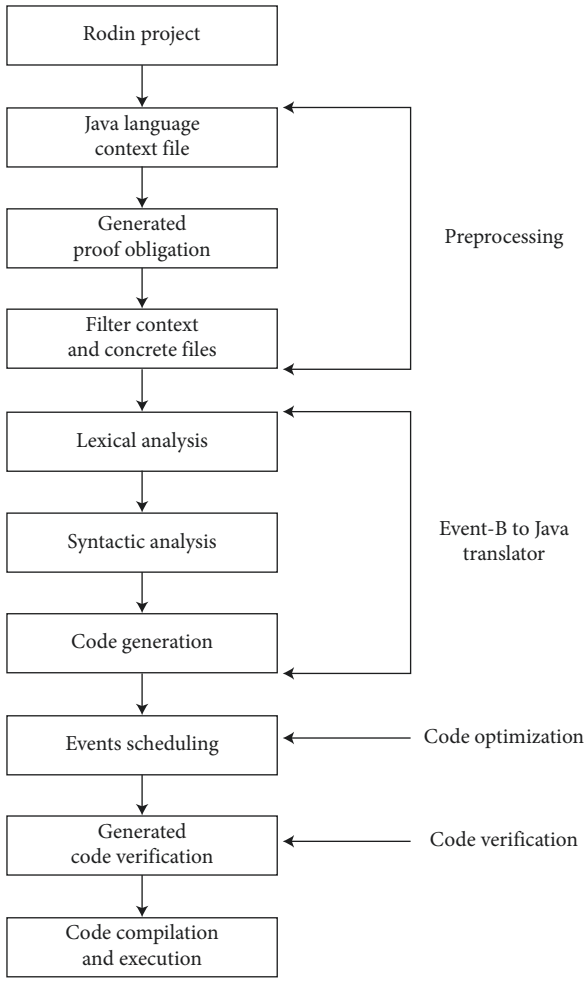

Figure 2: The EB2J tool architecture [15].

status is assigned to it. If the aircraft intend to land, it flies toward the VOR area (very high-frequency omnidirectional radio range) to be qualified to get landing clearance. At this stage, the system assigns to the aircrafts readyL status, which means that it is ready for landing. After getting landing clearance, it is assigned to landing state until finishing landing and passengers' departure; and then, it is considered in TerminatedL status. Likewise, an aircraft in the runway, after passengers' arrival, is considered ready to takeoff and being assigned to readyT status. Immediately upon takeoff clearance confirmation, it is considered in taking off status. Finally, the aircraft leaves out the VOR and returns to the Blocked status until getting out of the airport radar range [20, 24, 25]. These statuses are expressed as the elements of a carrier set called STATUSES (axm5). Figure 3 illustrates this process and the different statuses [26].

To summarize, the first context is made up of five sets (RW_STATUSES, RWL_STATUSES, AIRCRAFTS, and STATUSES), ten constants (Available, Unavailable, ON, OFF, Blocked, ReadyL, Landing, TerminatedL, ReadyT, and TakingOff), and four axioms (axm1, axm2, axm3, and axm4). This is expressed as shown in Box 1.

The partition predicate is an easy way to enumerate sets. Mathematically, the partition predicate is defined as follows:

$$\text{partition}(S, x, y) \Leftrightarrow x \cup y = S \wedge x \cap y = \varnothing. \quad (1)$$

In the dynamic part (machine), we introduce two variables curr_RW_status and curr_RWL_status denoting, respectively, the current statuses of the runway and runway lights (whereas, RW_STATUS and RWL_STATUS represent all the possible statuses). These two variables are defined by means of two invariants inv1 and inv2. Inv1 defines curr_RW_status as an element of the RW_STATUS, which means that curr_RW_status may equally be available or unavailable. Likewise, curr_RWL_status is an element of RWL_STATUS.

In order to ensure the traffic safety in the runway, the status lights are turned ON whenever the runway is unavailable. However, taxiways intersect the runway at many points, and therefore, vehicles must be aware of the runway

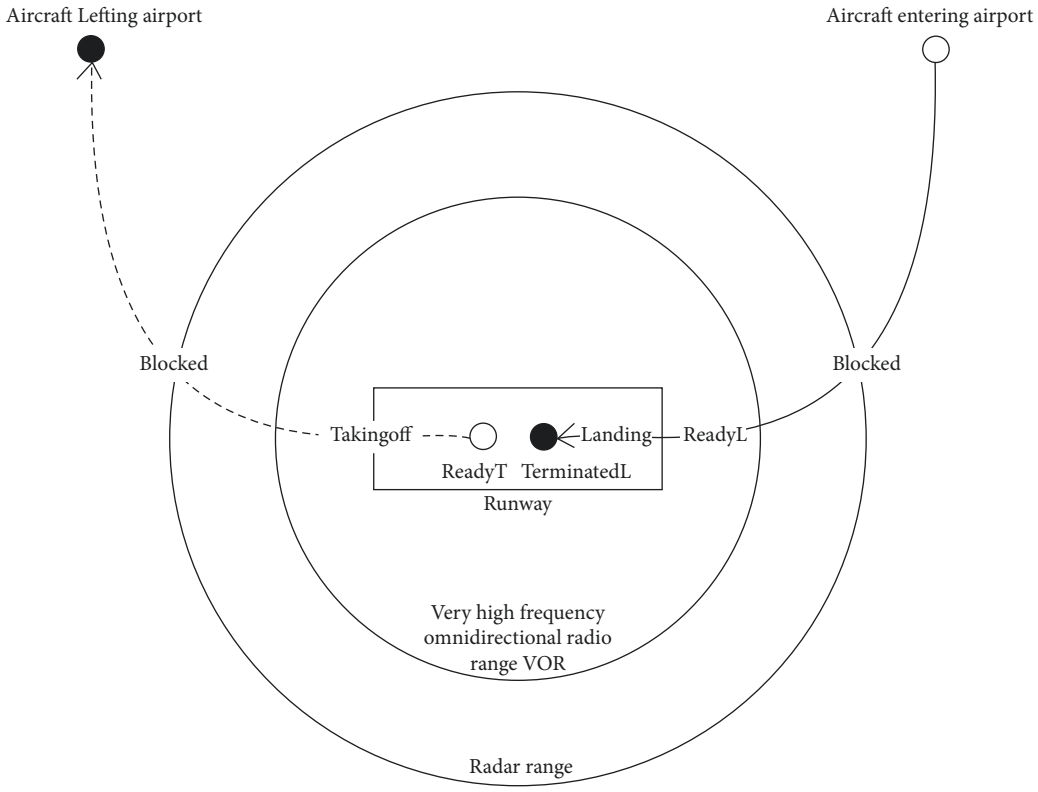

Figure 3: Aircrafts landing/taking off statuses.

SETS

RW_STATUSES, RWL_STATUSES, AIRCRAFTS, STATUSES

CONSTANTS

Available, Unavailable, ON, OFF, Blocked, ReadyL, Landing, TerminatedL, ReadyT, TakingOff

AXIOMS

axm1: partition(RW_STATUSES,{available},{unavailable})
axm2: partition(RWL_STATUSES,{ON},{OFF})
axm3: finite(AIRCRAFTS)
axm4: partition(STATUSES, {Blocked}, {ReadyL}, {Landing}, {TerminatedL}, {ReadyT}, {TakingOff})

Box 1

usage. These lights help to determine when it is not safe to proceed into or across the runway. However, the FAA confirms that the RWSL does not act as a substitution of the ATC clearance, which means that the vehicle should not enter the runway without a controller clearance even if the RWSL have gone out [16–20]. Formally, this is modeled by means of implication between the RWST and runway status (inv3). The proposed approach introduces also a subset of AIRCRAFTS called aircrafts_in_airport denoting the set of aircrafts in the airport (inv4).

As mentioned before, the system associates with each aircraft in the radar range a significant status. Therefore, the introduction of a variable statusof associating with each aircraft its status formalized as a total function from aircrafts_in_airport to the set AIRCRAFTS (inv5). The definition of the variables and the invariants of the initial model are shown in Box 2.

After defining all variables and invariants of the first machine, we present the different machine statuses' transactions described by events. Firstly, we have to define what happens at the beginning. For this purpose, the proposed approach defines the INITIALIZATION event that corresponds to the initial statuses of the system. It assumes initially that the runway is available, runway lights are off, there are no aircrafts in the airport, and no aircraft status is assigned. In addition, the initialization event should not have any guard, since the initialization must always be possible. This event is formalized as shown in Box 3.

VARIABLES:

curr_RW_status, curr_RWL_status, aircrafts_in_airport, statusof

INVARIANTS:

inv1: $curr\_RW\_status \in RW\_STATUS$
inv2: $curr\_RWL\_status \in RWL\_STATUS$
inv3: $curr\_RW\_status = unavailable \Longrightarrow curr\_RWL\_status = ON$
inv4: $aircrafts\_in\_airport \subseteq AIRCRAFTS$
inv5: $statusof \in aircrafts\_in\_airport \longrightarrow STATUSES$

Box 2

INITIALIZATION
BEGIN

act1: $curr\_RW\_status \coloneqq available$
act2: $curr\_RWL\_status \coloneqq OFF$
act3: $aircrafts\_in\_airport \coloneqq \varnothing$
act4: $statusof \coloneqq \varnothing$

END

Box 3

Besides the initialization event, eight more events are introduced: Entering_Radar_Range, Entering_VOR, Start_Landing, Terminating_Landing, Takeoff_Preparing, Start_takingoff, Terminating_takingoff, and Airport_Departing. The Entering_Radar_Range triggers when an aircraft enters the range of the airport radar. An entering aircraft must be added to the set of aircrafts in the airport (aircrafts_in_airport) and assigned to the Blocked status. However, during carrying out the proof obligation for different events, it is discovered that some guards are needed in each event. For the Entering_Radar_Range event, two guards are needed to be added: the first ensures that the entering aircraft is effectively a well-defined aircraft and known by the system and the second guard guarantees that it is not an element of the aircrafts_in_airport set. Similarly, the Entering_VOR is the event associated with an aircraft entering the VOR. This event assigns to an aircraft the status Ready under the condition that it is an element of the aircrafts_in_airport set, and it was in the Blocked status. Moreover, the Start_Landing event triggers whenever an aircraft gets landing clearance. To get that clearance, it must have been in the VOR (which means in Ready status) and an element of the aircrafts_in_airport. Furthermore, the runway must be currently available, and the runway lights must be OFF. After the aircraft landing and passengers' departure, the Terminating_Landing event triggers indicating the end of landing process by assigning the aircraft to the status TerminatedL; therefore, freeing the runway and turning runway's lights off [16–20].

The Takeoff_Preparing event triggers when an aircraft is ready to take off. This means that the aircraft previously finished its landing (it is in TerminatedL status). This event assigns to the aircraft the status ReadyL. After finishing take-off preparation, the aircraft get take-off clearance [17]. The event triggered at this stage is Start_takingoff; this event allocates the runway for the aircraft and turn lights on under the condition that the runway is not reserved by another aircraft [16]. Afterward, the aircraft terminates taking off and leaves the VOR to return to the first status Blocked. The event corresponds to this is Terminating_takingoff; this event has two guards: the first ensures that the aircraft is an element of the aircrafts_in_airport set and the second is that it is in Takingoff status [18, 20]. Ultimately, the Airport_Departing event triggers indicating that the aircraft is leaving the radar range, thus removing it from aircrafts_in_airport. Moreover, the position of the aircraft is deleted by removing it from the total function statusof. The proposed approach formalizes the events of the initial model in Boxes 4–9.

In this initial model, the very basic process of circulation in the airport vicinity is modeled. Therefore, most invariants are simply typing invariants; however, other invariant in the next refinement will be presented.

3.2. First Refinement: Introducing Scheduling Methods. The first refinement is more precise and contains more details; however, it should not contradict with the initial model. Therefore, some consistency proofs are established.

In this refinement, we present how the system manages aircrafts taking off and landing [19, 20]. Therefore, we need to define some additional variables and invariants. The first variable is deadline which is a total function from the aircrafts_in_airport set to some natural numbers. The second is a set for aircrafts ready to take off denoted as ready_to_takeoff_aircrafts. We present also another variable that refers to the moment that an aircraft became ready to land. Finally, we introduce a set for aircrafts requiring urgent landing due to a certain issue (Box 10).

The currently used method for aircrafts taking off is FCFS where aircrafts take off in the order that they are ready [16, 19]. We formalize this by introducing firstly a set of aircrafts ready to take off and a total function returning for each aircraft the moment it is ready to take off. These moments are associated with the same time aircrafts are associated with the readyL status. This is done during the takeoff_preparing event in addition to adding the aircraft to the ready_to_takeoff_aircrafts set. Once having these data about aircrafts, the system adopts the FCFS scheduling for

Start_Landing
ANY
aircraft
WHERE
grd1: aircraft ∈ aircrafts_in_airport
grd2: statusof(aircraft) = ReadyL
grd3: curr_RW_status = available
grd4: curr_RWL_status = OFF
THEN
act1: statusof(aircraft) ≔ Landing
act2: curr_RW_status ≔ unavailable
act3: curr_RWL_status ≔ ON
END

Box 4

Terminating_Landing
ANY
aircraft
WHERE
grd1: aircraft ∈ aircrafts_in_airport
grd2: statusof(aircraft) = Landing
THEN
act1: statusof(aircraft) ≔ TerminatedL
act2: curr_RW_status ≔ available
act3: curr_RWL_status ≔ OFF
END

Box 5

Takeoff_Preparing
ANY
aircraft
WHERE
grd1: aircraft ∈ aircrafts_in_airport
grd2: statusof(aircraft) = TerminatedL
THEN
act1: statusof*(aircraft) ≔ ReadyT*
END

Box 6

Start_takingoff
ANY
aircraft
WHERE
grd1: aircraft ∈ aircrafts_in_airport
grd2: statusof(aircraft) = ReadyL
grd3: curr_RW_status = available
grd4: curr_RWL_status = OFF
THEN
act1: statusof(aircraft) ≔ TakingOff
act2: curr_RW_status ≔ unavailable
act3: curr_RWL_status ≔ ON
END

Box 7

Terminating_takingoff
ANY
aircraft
WHERE
grd1: aircraft ∈ aircrafts_in_airport
grd2: statusof*(aircraft) = TakingOff*
THEN
act1: statusof(aircraft) ≔ Blocked
act2: curr_RW_status ≔ available
act3: curr_RWL_status ≔ OFF
END

Box 8

Airport_Departing
ANY
aircraft
WHERE
grd1: aircraft ∈ aircrafts_in_airport
grd2: statusof(aircraft) = Blocked
THEN
act1: aircrafts_in_airport: =
aircrafts_in_airport\{aircraft}
act2: statusof ≔ {aircraft} ⩤ statusof
END

Box 9

giving take-off clearance by means of the guards shown in Box 11 in the start_takingoff event.

The first guard requires that there is no other aircraft using the runway to take off [16]. The second one ensures that the aircraft that will get take-off clearance is the one with the minimum ready to take-off moment (the one has been ready to take off first) [17–19]. Finally, we delete information about the aircraft after give it landing clearance by means of the two actions shown in Box 12.

$$
\begin{array}{l}
\textit{INVARIANTS} \\
\textit{inv1: } \textit{deadline} \in \textit{aircrafts\_in\_airport} \longrightarrow \mathbb{N} \\
\textit{inv3: } \textit{ready\_to\_takeoff\_aircrafts} \subseteq \textit{aircrafts\_in\_airport} \\
\textit{inv2: } \textit{the\_ready\_to\_takeoff\_moment} \in \textit{ready\_to\_takeoff\_aircrafts} \longrightarrow \mathbb{N} \\
\textit{inv4: } \forall A \cdot (A \in \textit{ready\_to\_takeoff\_aircrafts} \Longrightarrow A \in \textit{aircrafts\_in\_airport} \wedge \textit{statusof}(A) = \textit{ReadyT})
\end{array}
$$

Box 10

$$
\begin{array}{l}
\textit{grd5: } \forall A \cdot A \in \textit{aircrafts\_in\_airport} \Longrightarrow \textit{statusof}(A) \neq \textit{TakingOff} \\
\textit{grd6: } \forall A \cdot A \in \textit{ready\_to\_takeoff\_aircrafts} \Longrightarrow \textit{the\_ready\_to\_takeoff\_moment}(A) \geq \textit{the\_ready\_to\_takeoff\_moment}(\textit{aircraft})
\end{array}
$$

Box 11

$$
\begin{array}{l}
\textit{act4: } \textit{ready\_to\_takeoff\_aircrafts} \coloneqq \textit{ready\_to\_takeoff\_aircrafts} \setminus \{\textit{aircraft}\} \\
\textit{act5: } \textit{the\_ready\_to\_takeoff\_moment} \coloneqq \{\textit{aircraft}\} \lhd \textit{the\_ready\_to\_takeoff\_moment}
\end{array}
$$

Box 12

Similarly to the take-off process, the currently used method for aircraft landing is FCFS [22, 27]. This method is very basic and simple which ease its implementation. However, the aircraft with a low landing speed may increase the waiting duration of other faster ones which affect the total landing duration. In addition, the FCFS limits flexibility to air traffic controllers to act in emergency situations [20, 27, 28]. Hence, we propose the use a new approach based on real-time scheduling algorithm, deadline monotonic (DM) in our case [29]. This approach assigns landing priority to aircrafts with the shortest deadline which offers an effective method for meeting deadlines as much as possible. However, maintaining deadlines respected is not always possible. In some cases, the sum of some high priority aircrafts landing durations is greater than the deadline of an aircraft with a lower priority. In this situation, we have two choices: the first is to proceed landing even if that some aircrafts will not respect their deadlines (note that we still optimize deadlines respecting) [25] and the second is to prevent the aircraft from entering VOR and redirect it to another runway. This choice is up to controller to decide, and the system will only notify him. This notification is done as soon as the aircraft try to enter the VOR; therefore, the guards shown in Box 13 in the Entering_VOR event are added.

For each guard, there are two cases: the first one is when there is no emergency landing request (Urgents = $\varnothing$, where Urgents is the set of aircrafts requesting emergency landing) [19]. In this case, the aircraft entering the VOR should have a deadline greater than or equals the sum (SIGMA function) of all average landing durations (AVERAGE_LD function) of aircrafts in the airport having a deadline lower than the entering aircraft deadline. The SIGMA function and AVERAGE_LD are formalized in the second context as shown in Box 14.

The second case is when the entering aircraft is requesting an emergency landing (aircraft $\in$ Urgents), the Urgents cases like: aeronautical failure, bad climate, terrorist attacks, kidnapping, and threat may affect passenger safety. According to the landing process on DM scheduling, the emergency cases have the highest priority to land in the first time, and the VOR and the supervisor must look for a not-used runway where the aircraft may be landed. In the second time, those having the lowest deadline are able to be landing. This is formalized as guards in the start_landing event as shown in Box 15.

Finally, we present below a new set associated with the aircrafts in the runway, that is, a subset of the aircrafts_in_airport (inv 5). We introduce also an inv 6 that expresses that the curr_RW_status is "unavailable" if and only if there is a single airplane in the state TakingOff or Landing. And such invariant would ensure that no accident may happen on the runway. Associated basic guards must be added in the start landing event to ensure that the system preserves these invariants and also some actions for adding and removing aircrafts from the aircrafts_in_runway set (Box 16).

3.3. Second Refinement: Towards an Alert System for a Secured ATC. In the previous models, we formalized the functional aspect of the system. Here, a model of alert system more security for the ATC is provided [16–20]. However, we need to firstly define the carrier set and constants shown in Box 17.

In this context, we introduce the aircraft brands set. These brands will be needed to determine the minimum separation time between two aircrafts. Besides, we present the LOCATION as a Cartesian product of three natural number sets which refer, respectively, to altitude, latitude, and longitude. Finally, we define a total function (Separation_Time) from the BRAND set to natural numbers formalizing the fact that the separation time between two aircraft landing depends strongly upon the last aircraft brand. This requirement is based on aerodynamic consideration: an aircraft generates a great deal of air turbulence when it flies. If another aircraft

$grd3: (SIGMA(\{i \mapsto ld \mid \exists A \cdot i \in card(aircrafts\_in\_airport) \wedge ld = AVERAGE\_LD(A) \wedge A \in aircrafts\_in\_airport \wedge deadline(A) \leq deadline(aircraft)\}) \leq deadline(aircraft) \wedge Urgents = \varnothing) \vee aircraft \in Urgents$
$grd4: (\forall A \cdot A \in aircrafts\_in\_airport \wedge deadline(A) > deadline(aircraft) \Longrightarrow SIGMA(\{i \mapsto ld \mid \exists a \cdot i \in card(aircrafts\_in\_airport) \wedge ld = AVERAGE\_LD(a) \wedge a \in aircrafts\_in\_airport \wedge deadline(a) \leq deadline(A)\}) + deadline(aircraft) \leq deadline(A) \wedge Urgents = \varnothing)) \vee aircraft \in Urgents$

Box 13

CONSTANTS

BAG
SIGMA
AVERAGE_LD

AXIOMS

$axm1: BAG = \{e \cdot e \in \mathbb{N} \nrightarrow \mathbb{N} \wedge finite(e) \wedge dom(e) = 1 \mathinner{..} card(e) \mid e\}$
$axm1: SIGMA \in BAG \longrightarrow \mathbb{N}$
$axm2: SIGMA(\varnothing) = 0$
$axm3: \forall e \cdot e \in BAG \wedge e \neq \varnothing \Longrightarrow SIGMA(e) = e(card(e)) + SIGMA(\{card(e)\} \lhd e)$

END

Box 14

$grd5: \forall A \cdot A \in aircrafts\_in\_airport \Longrightarrow statusof(A) \neq Landing$
$grd6: ((deadline(aircraft) = min(\{dl \mid \exists A \cdot A \in aircrafts\_in\_airport \wedge statusof(A) = ReadyL \wedge dl = deadline(A)\})) \wedge Urgents = \varnothing) \vee aircraft \in Urgents$

Box 15

$Inv\ 5: aircrafts\_in\_runway \subseteq aircrafts\_in\_airport$
$Inv\ 6: aircrafts\_in\_runway \neq \varnothing \Longrightarrow curr\_RW\_status \coloneqq unavailable$

Box 16

SETS

BRAND

CONSTANTS

LOCATIONS
Min_distance
Separation_Time

AXIOMS

$axm1: LOCATIONS = \mathbb{N} \times \mathbb{N} \times \mathbb{N}$
$axm2: Min\_distance \in \mathbb{N}$
$axm3: Separation\_Time \in BRAND \longrightarrow \mathbb{N}$

END

Box 17

act4: last_landing_t ≔ curr_t
act5: separation_t ≔ Separation_Time(brandof(aircraft))

Box 18

grd10: curr_t − last_landing_t ≥ separation_t

Box 19

flies too close behind it, it will lose aerodynamic stability [16, 24]. For safety purpose, the landing time between two aircrafts should be always greater than the separation time defined by the Separation_Time function. After each landing,

inv7: $\forall a,b \cdot a \in \text{aircrafts\_in\_airport} \wedge b \in \text{aircrafts\_in\_airport} \Longrightarrow \text{distance}(\text{locationof}(a) \longmapsto \text{locationof}(b)) \geq \text{Min\_distance}$

Box 20

inv1: $\text{locationof} \in \text{AIRCRAFTS} \longrightarrow \text{LOCATIONS}$
inv2: $\text{distance} \in \text{LOCATIONS} \times \text{LOCATIONS} \longrightarrow \mathbb{N}$

Box 21

Aircraft_moving_Alert_ON
ANY

aircraft
loc

WHERE

grd1: $loc \in LOCATIONS$
grd2: $aircraft \in AIRCRAFTS$
grd3: $(\exists a \cdot a \in AIRCRAFTS \wedge distance(locationof(a) \longmapsto loc) < Min\_distance) \vee \neg$ *(Security property 1)* $\vee \neg$ *(security property 2)* $\vee \ldots$

THEN

act1: $Alert \coloneqq TRUE$
act2: $locationof(aircraft) \coloneqq loc$

END

Box 22

two things must be saved to ensure the safety of the next landing: the time of the previous landing and the separation time [16, 19, 20, 30, 31]. This is formalized in the Terminating_landing event shown in Box 18.

Here, the curr_t is an event parameter denoting the current time that will be saved using the last_landing_t variable and separation_t which is also a variable denoting the separation time required for the next landing. Finally, the brandof is a total function from aircrafts to brands. In order to guarantee that the separation time is kept, the guard shown in Box 19 must be inserted in the start_landing event.

As the separation time, a minimum separation distance must be maintained between aircraft during flying in the airport airspace [17, 19, 32]. If two aircrafts are keeping this distance, collision will be strongly avoided as well as wake turbulence. The minimum distance is fixed and denoted by the Min_distance constant is presented in the previous context. To insure that the minimum distance will be kept, the invariant shown in Box 20 must be preserved.

The distance function is defined from LOCATION × LOCATION to natural numbers, and it calculates the distance between two aircrafts based on their locations determined by means of the locationof function. These two functions are defined in the two invariants shown in Box 21.

Since the controllers should be aware of the system status in real time, the system should response to each aircraft movement. Therefore, we introduce two new events: Aircraft_moving_Alert_ON and Aircraft_moving_Alert_OFF. One of these two events triggers whenever an aircraft moves, the first one triggers when the movement of an aircraft is not allowed which turn an alert on. Thus, its guards are verified if one of a not-allowed movement happens. On the other hand, the Aircraft_moving_Alert_OFF event triggers when the movement is allowed verifying all security properties which turns the alert off if it is on. This work focuses on the minimum distance property as our example for proving security properties [16–20, 33]. Other security properties may be added by means of disjunction of different properties in the Aircraft_moving_Alert_ON event and the conjunction of these properties in Aircraft_moving_Alert_OFF (Boxes 22 and 23).

4. Proving Model Correctness and Result

The Rodin platform is used to prove model correctness [14]. Table 1 presents the statistics proofs generated by Rodin.

This table measures the size of proofs generated including automatic and manual proofs. Note that there are many proof obligations in the first refinement due to the introduction of scheduling management. In order to guarantee the correctness of this scheduling process, various invariants must be established. Moreover, our formal model

Aircraft_moving_Alert_OFF
ANY

aircraft
loc

WHERE

grd1: aircraft $\in$ *AIRCRAFTS*
grd2: loc $\in$ *LOCATIONS*
grd3: $(\forall a \cdot a \in AIRCRAFTS \Longrightarrow distance(locationof(a) \longmapsto loc) \geq Min\_distance) \wedge (Security\ property1) \wedge (Security\ property\ 2) \wedge \ldots$

THEN

act1: Alert $\coloneqq$ *FALSE*
act2: locationof(aircraft) $\coloneqq$ *loc*

END

Box 23

Table 1: Rodin report (Table 1 is reproduced from Jarrar and Balouki, (under the Creative Commons Attribution License)) [13].

Element name	Total	Auto	Manual
Air traffic control	190	173	17
Initial context	5	5	0
First refinement context	9	9	0
Second refinement context	12	10	2
Initial machine	45	42	3
First refinement machine	66	58	8
Second refinement machine	53	49	4

introduces management functions such as sigma, min, deadline, and average landing durations. According to this report, we conclude that Rodin inference prover was able to establish 91% of proofs, which makes the task of modeling and proving easier. The combination of automatic and manual proofs ensures that the system developed here is correct by construction.

5. Conclusion

We demonstrate formal modeling and verification of an assisting system for air traffic management in airport airspace. We formalize the system's functional and non-functional requirements using Event-B that is based on a mathematical language.

The present contribution deals with the system as a sequence of models in successive steps that add refinement to enrich the initial abstract model in order to achieve a more realistic model. Most of the model correctness proofs are performed automatically by the means of Rodin theorem proving platform that supports the applications of Event-B. The ATC model presented is based on a single runway that take off and land aircraft while maintaining deadlines as much as possible.

Successive refinements such as maintaining minimum separation landing time in order to maintain an aircraft aerodynamic stability and also urgent landing scenarios are integrated.

We have started with an initial model that captures the essence of traffic management and the different elements taken into consideration. The first refinement introduces the scheduling method used during taking off and landing. In the second refinement, we prove safety properties to avoid system failure. Our model is proved correct by construction including invariant preservation and deadlock freedom.

The process presented in this work is based on FCFS during taking off and on deadline monotonic during landing. This system is also able to verify safety properties that must be preserved during system occurrence; these properties are guaranteed by means of invariants and guards.

Because of environmental, political, and geographical constraints, capacity cannot be easily increased by building new airports or runways. Therefore, our model is based on one runway.

In future work, we hope to be able to improve our model by considering the case of several runways in the same airport and several airports. Besides, it is very useful to combine this method with other modeling and simulation techniques such as Monte Carlo presented in [1], which highly improves system feasibility. Furthermore, we aim to apply standardizations such as QoS [34] and RM-ODP [35] in the field of air traffic management.

Abbreviations

ATC: Air traffic control system
ICAO: International Civil Aviation Organization
FAA: Federal Aviation Administration
NASA: National Aeronautics and Space Administration
FCFS: First comes first served
VOR: Very high-frequency omnidirectional radio range
DM: Deadline monotonic
QoS: Quality of service
RM-ODP: Reference Model of Open Distributed Processing.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Authors' Contributions

Abdessamad Jarrar and Youssef Balouki contributed in collecting information, writing, modeling, and reviewing.

Acknowledgments

This research work was supported by Computing, Imaging and Modeling of Complex Systems Laboratory, Settat, Morocco.

References

[1] S. Bouarfa, H. A. Blom, R. Curran, and M. H. Everdij, "Agent-based modeling and simulation of emergent behavior in air transportation," *Complex Adaptive Systems Modeling*, vol. 1, no. 1, p. 15, 2013.

[2] N. Ahmad Zafar, "Formal specification and analysis of take-off procedure using VDM-SL," *Complex Adaptive Systems Modeling*, vol. 4, no. 1, 2016.

[3] S. Yousaf, N. Ahmad Zafar, and S. A. Khan, "Formal analysis of departure procedure of air traffic control system," in *2010 2nd International Conference on Software Technology and Engineering (ICSTE)*, vol. 2, San Juan, PR, USA, October 2010.

[4] D. Méry and N. K. Singh, "Modeling an aircraft landing system in event-B," in *Proceedings of International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z: ABZ 2014: The Landing Gear Case Study*, pp. 154–159, Toulouse, France, 2016.

[5] iFACTS-air traffic management system, 2017, https://www.adacore.com/customers/uks-next-generation-atc-system.

[6] Q. Zhang, Z. Huang, and J. Xie, "Distributed system model using SysML and event-B," in *Proceedings of International Conference on Machine Learning and Intelligent Communications (MLICOM 2017)*, pp. 326–336, Weihai, China, August 2018.

[7] T. Chajed, H. Chen, A. Chipala, M. Frans Kaashoek, N. Zeldovich, and D. Ziegler, "Certifying a file system using crash hoare logic," *Communications of the ACM*, vol. 60, no. 4, pp. 75–84, 2017.

[8] S. Zhang, Y. Ma, C. Meng, and H. Wang, *Formal Verification of Quantum Communication Protocols Using Petri Nets*, arXiv preprint arXiv:1704.07031, Cornell University Library, Ithaca, NY, USA, 2017.

[9] M. U. Ashraf and W. Aljedaibi, "ATAM-based architecture evaluation using LOTOS formal method," *International Journal of Information Technology and Computer Science*, vol. 9, no. 3, pp. 10–18, 2017.

[10] S. Jaidka, S. Reeves, and J. Bowen, "Modelling safety-critical devices: coloured petri nets and Z," in *Proceedings of ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS'17)*, Pittsburgh, PA, USA, July 2017.

[11] T. S. Hoang, H. Kuruma, D. Basin, and J. R. Abrial, "Developing topology discovery in Event-B," *Science of Computer Programming*, vol. 74, no. 11-12, pp. 879–899, 2009.

[12] J.-R. Abrial, *Modeling in Event-B: System and Software Engineering*, Cambridge University Press, New York, NY, USA, 2010.

[13] A. Jarrar and Y. Balouki, "Formal reasoning for air traffic control system using event-B method," in *Proceedings of International Conference on Computational Science and Its Applications*, pp. 241–252, Springer, Melbourne, Australia, July 2018.

[14] C. Rodin, M. Jastram, and M. Butler, *User's Handbook*, Deploy Project, Fontainebleau, France, 2011.

[15] D. Méry and N. K. Singh, "EB2J: code generation from event-B to Java," in *Proceedings of SBMF-Brazilian Symposium on Formal Methods*, São Paulo, Brazil, 2011.

[16] In Focus, "ICAO'S strategic objectives," 2018, https://www.icao.int/Pages/default.aspx.

[17] Fact sheet -FAA & NTSB's "most wanted" recommendations, 2010, https://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=11186.

[18] Department of Transportation Federal Aviation Administration, Aeronautical Information Publication, United States of America, 24th edition, Amendment 2, 2017.

[19] J. S. Duncan, *Airplane Flying Handbook*, FAA-H-8083-38, Department of Transportation Federal Aviation Administration Flight Standards Service, FAA-H-8083-38, Federal Aviation Administration, Washington, DC, USA, 2016.

[20] NASA air traffic management demonstration goes live in Charlotte, 2017, https://www.nasa.gov/aero/nasa-air-traffic-management-demo-goes-live.

[21] J. Lygeros and N. Lynch, "On the formal verification of the TCAS conflict resolution algorithms," in *Proceedings of the 36th IEEE Conference on Decision and Control*, pp. 1829–1834, San Diego, CA, USA, December 1997.

[22] H. Pinol and J. E. Beasley, "Scatter search and bionomic algorithms for the aircraft landing problem," *European Journal of Operational Research*, vol. 171, no. 2, pp. 439–462, 2006.

[23] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: a study in multiagent hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 509–521, 1998.

[24] S. P. Yu, X. Bin Cao, and J. Zhang, "A real-time schedule method for aircraft landing scheduling problem based on cellular automation," *Applied Soft Computing*, vol. 11, no. 4, pp. 3485–3493, 2011.

[25] W. Su and J. R. Abrial, "Aircraft landing gear system: approaches with Event-B to the modeling of an industrial system," *International Journal on Software Tools for Technology Transfer*, vol. 19, no. 2, pp. 141–166, 2017.

[26] S. Luo and G. Yu, "Airline schedule perturbation problem: landing and takeoff with nonsplitable resource for the ground delay program," in *Operations Research in the Airline Industry*, pp. 404–432, Springer, Boston, MA, USA, 1998.

[27] G. L. Vairaktarakis and T. Aydinliyim, "Benchmark schedules for subcontracted operations: decentralization inefficiencies that arise from competition and first-come-first-served processing," *Decision Sciences*, vol. 48, no. 4, pp. 657–690, 2017.

[28] C. V. Schmidt, A. Heimbucher, A. Bernadou, and J. Heinze, "First come, first served: the first-emerging queen monopolizes reproduction in the ant Cardiocondyla "argyrotricha"," *Journal of Ethology*, vol. 35, no. 1, pp. 21–27, 2017.

[29] A. Jarrar, "Modeling aircraft landing scheduling in event B," in *International Conference on Information Technology and Communication Systems*, pp. 127–142, Springer, Berlin, Germany, 2017.

[30] V. Carreño and C. Muñoz, "Safety verification of the small aircraft transportation system concept of operations," in *Proceedings of AIAA 5th Aviation, Technology, Integration, and Operations Conference (ATIO)*, Arlington, VA, USA, 2005.

[31] S. Umeno and N. Lynch, "Safety verification of an aircraft landing protocol: a refinement approach," in *Proceedings of International Workshop on Hybrid Systems: Computation and Control*, pp. 557–572, Springer, Philadelphia, PA, USA, 2007.

[32] A. Narkawicz and C. Munoz, "A formally verified conflict detection algorithm for polynomial trajectories," in *Proceedings of AIAA Infotech@Aerospace*, p. 0795, Kissimmee, FL, USA, January 2015.

[33] A. Platzer and E. M. Clarke, "Formal verification of curved flight collision avoidance maneuvers: a case study," in *Proceedings of International Symposium on Formal Methods*, pp. 547–562, Springer, Oxford, UK, 2009.

[34] A. Jarrar, Y. Balouki, and T. Gadi, "Formal specification of QoS negotiation in ODP system," *International Journal of Electrical and Computer Engineering*, vol. 7, no. 4, p. 2045, 2017.

[35] H. Belhaj, Y. Balouki, M. Bouhdadi, and S. El Hajji, "Using event B to specify QoS in ODP enterprise language," in *Proceedings of Working Conference on Virtual Enterprises*, pp. 478–485, Springer, France, October 2010.