Complex Adaptive Systems Modeling

RESEARCH Open Access

Formal modeling of a complex adaptive air traffic control system

Abdessamad Jarrar and Youssef Balouki*

*Correspondence: balouki.youssef@gmail.com
Computing, Imaging and Modeling of Complex Systems Laboratory, Faculty of Sciences and Technologies of Settat, Hassan 1st University, Settat, Morocco

Abstract
Air traffic control system in airports is one of the most complex systems in the context of air traffic management due to the huge number of requirements. In order to help engineers to develop such complex system we propose a predefined model that includes the essence of air traffic control and the standard requirements. We develop this model using the Event-B formal method which is based on set theory and allows theorems proving. Event-B is also hinged on refinement which means starting with an abstract model and then enriching it in successive steps. Event-B has been successfully applied in several transportation systems and shows no bugs. This encourages us to use it in this critical system to guarantee a strong assurance of bugs' absence and to ensure model correctness. Our approach provides a standard model to start with in order to model any airport control system, which allows engineers to focus on more typical requirements that are not developed here.

Keywords: Formal modeling, Event-B method, Air traffic control, RODIN, Refinement

Introduction

Air traffic control system studied in this paper is a complex adaptive socio-technical system that controls the traffic of all aircrafts in the airport's radar range. The complexity of this system arises due to the number of elements and factors intervene in it such as aircrafts, controllers, runways, taxiways, weather, etc. This complexity makes the system unpredictable which require a real time management and adaptation of a dynamic state. The dynamic state means that during system building we cannot know which aircrafts will enter the airport neither their arrival nor departure time. All these facts makes developing air traffic control system a significant and challenging task in the context of air traffic management.

We describe this complex adaptive socio-technical system using a model based on a complex network. This kind of complex network is often represented using a graph where nodes represent aircrafts (agents) and edges represent the exchange of data over the network. In a complex network we define strategies describing the behaviour of aircrafts in all situations. These strategies should be carefully defined to maintain the stability of the system and avoid any mistake that could leat to a disaster. This high risk can be avoided by using sophisticated techniques such as formal methods during system design.

Formal methods are mathematically based techniques for specifying and verifying systems. Use of those methods can greatly increase our understanding of a system by revealing inconsistencies, ambiguities, and incompleteness that might otherwise go undetected (Clarke and Wing 1996). The formal method used in this paper is Event-B, which is based on set theory as a modeling notation. One of the key features of Event-B is the use of refinement to represent systems at different abstraction levels in addition to the use of mathematical proofs to verify consistency between refinement levels (Abrial et al. 2010; Hoang et al. 2009). This method has been successfully applied to the industry of transportation such as Meteor line 14 driverless metro in Paris where no bugs were detected after the proofs, neither at the functional validation (October 1998). This success encourages Alstom and Siemens Transportation Systems to have a product based strategy and reuse as much as possible existing B models to develop future metros (Lecomte et al. 2007). Therefore, we also used Event-B to develop a standard air traffic control system model.

In this paper, we develop a formal model of air traffic control system in order to help engineers develop alike system. In particular, we focus on developing a standard model that plays the role of a starting model in the process of developing any air traffic control system. This model includes the essence of air traffic management and the essential safety requirements; after that, engineers enrich it by adding more details and requirements depending on the case studied.

Our main contribution is a standard air traffic control system model recommended to be used as a starting model for system development due to its correctness. Being correct means that it verifies all proofs obligations which ensure the absence of contradictions within the elements of the model. It is also proved that it maintains all requirements of the system verified by means of invariant preservation proofs. Therefore, a system constructed based on this model will be indeed correct by construct. This model is very abstract and based on several organizations' standards and recommendations (International Civil Aviation Organization ICAO, Federal Aviation Administration FAA, and National Aeronautics and Space Administration NASA) which ensure its applicability in most airports in the world. Furthermore, the essence of air traffic management requirements are included in order to allow engineers to focus on other typical requirements which will be very useful to ensure the efficiency of developing air traffic control systems.

The rest of the paper is structured as follows. "Background and literature review" section gives some background on related works, Air Traffic Control, and the used method. In "Requirements document" section, we presents the set of requirements considered in this paper. The main content of the paper is "Formal development" section 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 non-functional requirements. "Proving model correctness and result" section presents proof statistics that are generated by Rodin platform. "Conclusion" section concludes the paper.

Background and literature review

Related works

One of the works that uses formal verification to model ATC is presented by Yang et al. (2017). In this work, Yang presents a functional resonance to provide a better understanding of why or how the emergent phenomena appear and develop. For this purpose, they used a formal verification tool SPIN along the paradigm of Functional Resonance

Analysis Method (FRAM). This work contributes to the realization that the hazards caused by functional resonance can be identified, with detailed manifestations about the way that the coincidence of functional variability occurs and ultimately leads to an accident, as well as effective safety measures to damp the resonance. Although the use of the SPIN tool as a model checker is liked by a lot of people because it is exhaustive automated testing, model checker works well for systems with a finite number of states that are predictable and this is not the case here (the location of a aircraft in the radar area have infinite possibilities). On the other hand, our method is based on theorem proving which is harder to use but it provides a complete verification of theorems either the system has a finite or infinite states, which works perfectly in our case.

Similar to the last approach, the Zafar (2016) authors combine a VDM-SL and graph theory to build a formal specification of aircrafts take-off's 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 take-off 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, but it requires further investigation to real-time management that is a major factor in this procedure. On the other hand, Dominique and Neeraj (2014) introduces a formal model of an aircraft landing system. This work 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 (2017). 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.

Overview of the air traffic control system

Air Traffic control is a service provided by controllers located in a control tower. These controllers are responsible for the safety of air traffic in the vicinity of airport; they should organize and expedite the flow of air traffic, prevent collisions, and provide information and other support for pilots (Fact Sheet-FAA & NTSB's "Most Wanted" Recommendations 2010). The system developed in this paper aims at assisting air traffic controllers fulfilling their responsibilities.

To organize and expedite the flow of air traffic, controllers assign landing and taking off clearances according to the first-come-first-served (FCFS) approach. This approach gives a useable scheme for scheduling, however, it does not maximize the profit of the runway. Therefore, we propose our own approach for organizing air traffic flow (see refinement 1 as Graphical abstract, Online).

In order to prevent collisions, air traffic controllers are responsible for enforcing a minimum separation distance between aircrafts. By so doing they avoid wake turbulence which may cause an aircraft to lose its aerodynamic stability. Furthermore, a minimum separation time should also be enforced to avoid air turbulence mostly during landing and taking off (Yu and Bin 2011; In Focus: ICAO'S Strategic Objectives 2018) (see refinement 2 as Fig. 1).

Modeling and refinement in Event-B

Due to the high risk in safety–critical systems, it is highly recommended to base their engineering on a certain theory. For example, electrical engineering is based on Maxwell's equations and Kirchhoff's laws; civil engineering is based on geometry and theory of material's strength. Similarly, software engineering has formal methods which are less considered. Therefore, there are engineers who do not know any theory building software, and the software often has bugs that may cost millions to fix. Now, formal methods can be used for verifying and specifying software in order to highly guarantee bugs' absence.

Event-B is a formal method that provides the correct-by-construct approach and formal verification by theorem proving (Vistbakka and Troubitsyna 2018; Abrial 2010). Models in Event-B are presented based on abstract state machine notion, which presents the model states in term of a set of variables; these states are constrained by invariants. Invariants are the necessary properties that must be preserved during system function. Statuses transitions are described by events, which are a set of actions. Each action

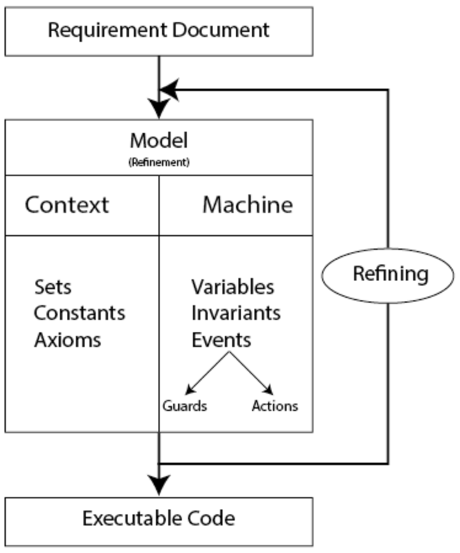

Fig. 1 Process of development in Event-B (Jarrar and Balouki 2018)

changes the value of certain variable. Events may have some necessary conditions to be triggered; these conditions are called guards. Models in Event-B include sets, constants and axioms representing the static part of the model.

One of the main features of Event-B is refinement, which means starting modeling with an abstract model and then enriches it in successive steps by adding more details. This techniques makes modeling easier than trying to model the whole system at once, we focus on a limited number of requirement in each step under the condition of cleverly choose the refinement strategy. Figure 1 presents an outline of the process of modeling in Event-B.

Event-B models consistency, invariant preservation and the correctness refinement—refinements should not contradict—are ensured by discharging a number of verification conditions called proof obligations. For example, to prove that an invariant is preserved by an event, we prove that if an invariant is preserved before the event it will remain preserved after it. Mathematically speaking, let I be the model invariant, A are axioms, c are constants, s are set, v are variables before the event occurrence and v′ are variables after the event. The following logical formula should be proved in order to prove invariant preservation:

$$A(c,s) \wedge I(v,c,s) \vdash I'\left(v',c,s\right) \tag{1}$$

Most of proof obligations are discharged automatically by means of a platform called Rodin. The remaining proofs may be dealt using an interactive prover included in Rodin.

Code generation from Event-B to Java

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 (Méry 2011) presented a method for generating code Java based on Event-B model. Dominique (2011) develop EB2J, a software tool that translates Event-B models into Java code. This tool is developed as plugin using the Eclipse development framework, the input of the code generation tool is a Rodin project file that contains Event-B formal specifications.

The choice of Java is justified by its several benefits; it is robust, reliable, and portable, has a runtime error checking and automatic memory management. The OO-paradigm also forms the basis for software component industry with their need for certification technique. Furthermore, Java is widely used for distributed and network programming, and the potential for reuse in OO-programming carries over to reusing specifications and proofs (Dominique 2011).

Figure 2 illustrates the EB2J architecture.

The generation code approach is based on 4 components:

- Pre-processing: basing on Rodin project, the pre-processing introduces a java context file. Using this context file additional proof obligations are generated, and then the context and concrete machine files are filtered.
- Event-B to Java translator: this translator is using syntax-directed translation to generate Java code from the context and concrete machine files. The generated Java file contains formal code in term of constants, variables, arrays, functions, and event;

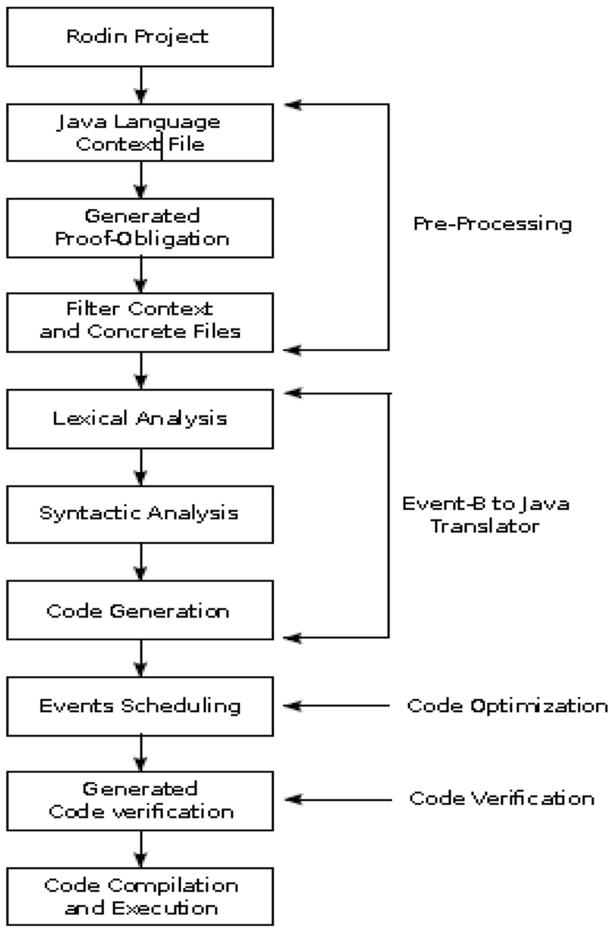
Rodin Project
Java Language Context File
Generated Proof-Obligation
Filter Context and Concrete Files
Pre-Processing
Lexical Analysis
Syntactic Analysis
Code Generation
Event-B to Java Translator
Events Scheduling
Code Optimization
Generated Code verification
Code Verification
Code Compilation and Execution

Fig. 2 The EB2J tool architecture (Méry 2011)

the generation approach is based on generating code from the Event-B specification using lexical and syntactic analysis.

- Code optimization: in this phase, the Java functions translated from Event-B events are synthesized.
- Code verification: the code verification step is used to verify the automated generated code in order to ensure that it satisfies the Event-B model.

The process of these successive steps allows the generation of a proof-based Java code from Event-B specification.

For our approach, EB2J is used as the last step in the process of development. The model developed in this paper is refined by engineers in order to add their typical requirements. The resulting model is developed as a Rodin project that is translated to proof-based Java program.

Requirements document

As we are modelling formally a system, it is recommended to present requirements in a more formal way instead of a simple informal paragraph. We propose presenting the requirement document along 3 axes labelled and numbered: the first is "FUN" that stands for the specific task/functional requirements of the system; the second is "ENV" which deals with the concerning environment and assumptions as for equipment situated around our intended system; the last is labelled "SAF" that deals with the safety requirements that have to be guaranteed by the system.

The requirement document, mainly based on FAA, ICAO and NASA recommendations, is presented as follows:

The airport is equipped with at least one runway	ENV-1
The runway is used for landing and taking-off aircrafts	FUN-1
Runways are equipped with lights indicating their status; these lights are called RunWays Status Lights RWSL	ENV-2
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 (Fact Sheet-FAA & NTSB's "Most Wanted" Recommendations (2010); Department of transportation federal aviation administration 2017; John 2016)	FUN-2
The airport is equipped with at least radar	ENV-3
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. (NASA Air Traffic Management Demonstration Goes Live in Charlotte 2017)	FUN-3
In order to ensure the traffic safety in the runway, the status lights are turned ON whenever the runway is unavailable	SAF-1
At the beginning, the runway is available, runway lights are off, there are not aircrafts in the airport, and no aircraft status is assigned	ENV-4
In order for an aircraft to get landing clearance, the runway must be currently available and the runway lights must be OFF	SAF-2
The take-off clearance requires also that the runway is available and the runway lights are off	SAF-3
The runway is allocated and RWSL are turned red whenever an aircraft is taking-off or landing	SAF-4
A deadline is associated to each aircraft entering the airport	ENV-5
The scheduling method used for aircrafts taking off is First-Come-First-Served	FUN-4
The proposed method for landing is based on real-time scheduling algorithm, which is Deadline Monotonic (DM) (Jarrar et al. 2017). This choice is very important in order to optimize deadlines respecting as much as possible	FUN-5
The system is able to predict if it is possible to maintain deadlines respected or not, therefore a notification feature can be included	SAF-5
The system assigns the highest landing priority to aircrafts with emergency situations such as medical and terroristic threats	SAF-6
An alert system is presented to provide more security for the ATC.	SAF-7
The alert system controls the movement of all aircrafts in the radar range and notifies the controller as soon as something is going wrong	SAF-8
A minimum separation distance should be maintained in order to avoid collisions and wake turbulence	SAF-9
A separation time should be kept between landing and taking off of aircrafts to avoid wake turbulence	SAF-10

Formal development

Our approach is based on complex network paradigm where aircrafts represent agents, and edges represent the exchange of data between aircrafts. This exchange of data occurs through controllers using the proposed control system. We also define the strategy that indicates what to do in which circumstance for each aircraft. Figure 3 emphasizes the approach of seeing the air traffic control system as a complex network.

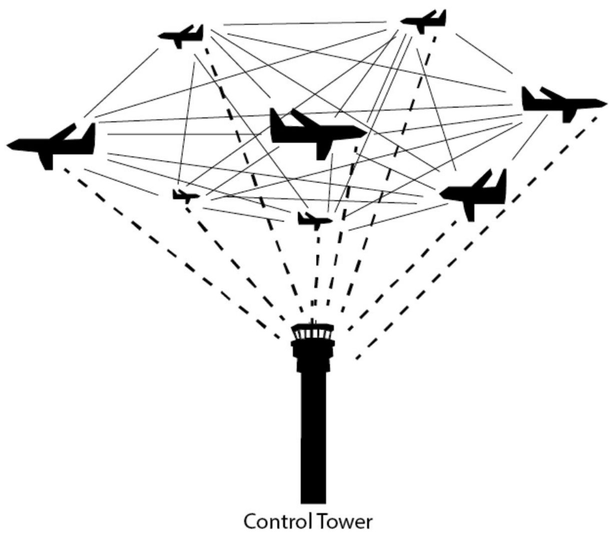

Fig. 3 Complex network

Bussniss rules and requirements modeled in this paper are proposed by the following organizations:

- ICAO (2018) which provides strategic objectives concerning safety, capacity and efficiency, security and facilitation, economic development, and environmental protection;
- FAA (2010), Department of transportation federal aviation administration (2017), John (2016) which has a predetermined number of air traffic manuals, publications, and orders;
- NASA (2017) 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 previously mentioned 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 is able to adapt in case of emergency situations.

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 (Lygeros and Lynch 1829).

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 (Pinol and Beasley 2006).

Initial model: an abstract model of the landing process

In this initial model, we consider the following requirements: ENV-1, ENV-2, ENV-3, ENV-4, FUN-1, FUN-2, FUN-3, SAF-1, SAF-2, SAF-3, and SAF-4 (see "Requirements document" section).

We introduce the essence of the ATC system and the different components taken into consideration (Tomlin et al. 1998). 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 (In Focus: ICAO'S Strategic Objectives 2018). Runways are equipped with lights indicating their status; these lights are called RunWays 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 (Fact Sheet-FAA & NTSB's "Most Wanted" Recommendations 2010; Department of transportation federal aviation administration 2017; John 2016).

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 (Pinol and Beasley 2006).

The first proposed model is made up of two parts: static part and dynamic one (Abrial 2010). The static part is called context and contains carrier sets, constants and associated axioms, whereas the dynamic part (called machine) contains variables, invariants and events. In the first context, we introduce of 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 denots 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. (NASA Air Traffic Management Demonstration Goes Live in Charlotte 2017). When an aircraft enter the airport vicinity, the status blocked is assigned to it. If the aircraft intend to land, it fly toward the VOR area (very high frequency omni directional 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 return to blocked status until getting out of the airport radar range (Yu and Bin 2011; Su and Abrial 2017; NASA Air Traffic Management Demonstration Goes Live in Charlotte 2017). These statuses are expressed as the elements of a carrier set called STATUSES (axm5). Figure 4 illustrates this process and the different statuses (Luo and Yu 1998):

To summarize, the first context is made up of four sets (RW_STATUSES, RWL_STATUSES, AIRCRAFTS, and STATUSES), ten constants (Available, Unavailable, ON,

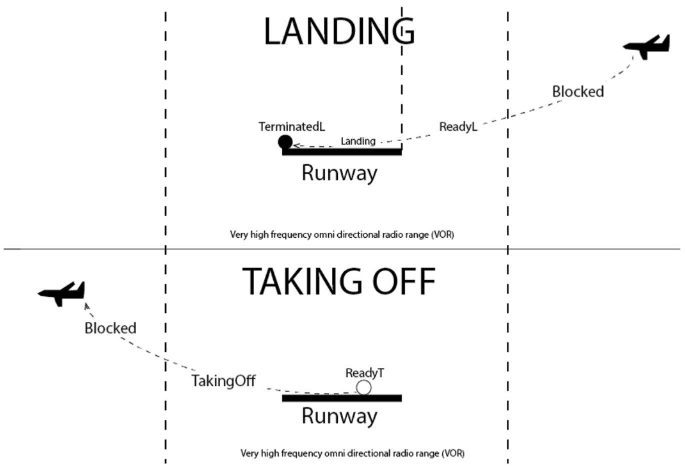
LANDING
Blocked
ReadyL
TerminatedL
Landing
Runway
Very high frequency omni directional radio range (VOR)
TAKING OFF
Blocked
TakingOff
ReadyT
Runway
Very high frequency omni directional radio range (VOR)

Fig. 4 Aircrafts landing/taking off statuses

OFF, Blocked, ReadyL, Landing, TerminatedL, ReadyT, and TakingOff), and five axioms (axm1, axm2, axm3, and axm4). This is expressed as shown in the following box:

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})

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

$$partition(S, x, y) \Leftrightarrow x \cup y = S \wedge x \cap y = \emptyset$$

where x and y are two subsets of a set S.

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 equal 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 usage. These lights help to determine when it is not safe to proceed into or across the runway. Although, The

FAA confirm that the RWSL does not act as a substitution of the ATC clearance, which means that vehicles should not enter the runway without a controller clearance even if the RWSL have gone out (In Focus: ICAO'S Strategic Objectives 2018; Fact Sheet-FAA & NTSB's "Most Wanted" Recommendations 2010; Department of transportation federal aviation administration 2017; John 2016; NASA Air Traffic Management Demonstration Goes Live in Charlotte 2017). 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 to each aircraft in the radar range a significant status. Therefore, the introduction of a variable statusof associating to 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 as follows:

```
VARIABLES:
curr_RW_status, curr_RWL_status, aircrafts_in_airport, statusof
INVARIANTS:
inv1  :  curr_RW_status ∈ RW_STATUS
inv2  :  curr_RWL_status ∈ RWL_STATUS
inv3  :  curr_RW_status=unavailable ⇒ curr_RWL_status=ON
inv4  :  aircrafts_in_airport ⊆ AIRCRAFTS
inv5  :  statusof ∈ aircrafts_in_airport→ STATUSES
```

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 INITIALISATION 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 that the initialization must always be possible. This event is formalized as follows:

```
INITIALISATION
BEGIN
act1  :  curr_RW_status:=available
act2  :  curr_RWL_status:=OFF
act3  :  aircrafts_in_airport := ∅
act4  :  statusof := ∅
END
```

Beside 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 trigger when an aircraft enter the range of the airport radar range. 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. The second guard guarantees that it is not an element of the aircrafts_in_airport set. Similarly, the Entering_VOR is the event associated to 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 Blocked status. Moreover, the Start_Landing event trigger whenever an aircraft get 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 (In Focus: ICAO'S Strategic Objectives 2018; Fact Sheet-FAA & NTSB's "Most Wanted" Recommendations 2010; Department of transportation federal aviation administration 2017; John 2016; NASA Air Traffic Management Demonstration Goes Live in Charlotte 2017).

The Takeoff_Preparing event trigger 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 (Fact Sheet-FAA & NTSB's "Most Wanted" Recommendations 2010). The event triggered at this stage is Start_takingoff; this event allocate the runway for the aircraft and turn lights on under the condition that the runway is not reserved by another aircraft (In Focus: ICAO'S Strategic Objectives 2018). 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 (Department of transportation federal aviation administration 2017; NASA Air Traffic Management Demonstration Goes Live in Charlotte 2017). 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 status of. The proposed approach formalize the events of the initial model in the following boxes:

```
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
```

```
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
```

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

```
Start_takingoff
ANY
aircraft
WHERE
grd1 : aircraft ∈ aircrafts_in_airport
grd2 : statusof(aircraft) = ReadyT
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
```

```
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
```

```
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
```

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 invariants in the next refinement we will be presented.

First refinement: introducing scheduling methods

This first refinement focuses on ENV-5, FUN-4, FUN-5, SAF-5, and SAF-6.

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 (John 2016; NASA Air Traffic Management Demonstration Goes Live in Charlotte 2017). 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 number. The second is a set for aircrafts ready to take off denoted 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.

INVARIANTS
inv1 : $deadline \in aircrafts\_in\_airport \rightarrow \mathbb{N}$
inv3 : $ready\_to\_takeoff\_aircrafts \subseteq aircrafts\_in\_airport$
inv2 : $the\_ready\_to\_takeoff\_moment \in ready\_to\_takeoff\_aircrafts \rightarrow \mathbb{N}$
inv4 : $\forall A \cdot (A \in ready\_to\_takeoff\_aircrafts \Rightarrow A \in aircrafts\_in\_airport \land statusof(A)=ReadyT)$

The currently used method for aircrafts taking off is FCFS where aircrafts take off in the order that they are ready (In Focus: ICAO'S Strategic Objectives 2018; John 2016). 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 at the same time aircrafts are associated to 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 giving take off clearance by means of the following guards in the start_takingoff event:

grd5 : $\forall A \cdot A \in aircrafts\_in\_airport \Rightarrow statusof(A) \neq TakingOff$
grd6 : $\forall A \cdot A \in ready\_to\_takeoff\_aircrafts \Rightarrow$
$the\_ready\_to\_takeoff\_moment(A) \geq the\_ready\_to\_takeoff\_moment(aircraft)$

The first guard requires that there is no other aircraft using the runway to take off (In Focus: ICAO'S Strategic Objectives 2018). 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) (Fact Sheet-FAA & NTSB's "Most Wanted" Recommendations 2010; Department of transportation federal aviation administration 2017; John 2016). Finally, we delete information about the aircraft after give it landing clearance by means of the following two actions:

act4 : $ready\_to\_takeoff\_aircrafts := ready\_to\_takeoff\_aircrafts \setminus \{aircraft\}$
act5 : $the\_ready\_to\_takeoff\_moment := \{aircraft\} \lhd\!\!\!- \; the\_ready\_to\_takeoff\_moment$

Similarly to the take off process, the currently used method for aircraft landing is FCFS (Pinol and Beasley 2006; Vairaktarakis and Aydinliyim 2017). 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, FCFS limits flexibility to air traffic controllers to act in emergency situations (Vairaktarakis and Aydinliyim 2017; Schmidt et al. 2017; NASA Air Traffic Management Demonstration Goes Live in Charlotte 2017). Hence, we propose a use a new approach based on real-time scheduling algorithm, Deadline Monotonic (DM) in our case (Jarrar et al. 2017). 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 some aircrafts will not respect their deadlines (note that we still optimizing deadlines respecting) (Su and Abrial 2017). The second is to prevent the aircraft from entering VOR and redirect it to another runway. This choice is up to controller to decide, the system will only notify him. This notification is done as soon as the aircraft try to enter the VOR; therefore, the following guards in the Entering_VOR event are added:

```
grd3 : (SIGMA({i ↦ ld|∃A· i∈ card(aircrafts_in_airport) ∧ ld=AVERAGE_LD(A) ∧
A∈aircrafts_in_airport ∧ deadline(A) ≤ deadline(aircraft)}) ≤ deadline(aircraft) ∧ Urgents = ∅)
∨ aircraft ∈ Urgents
grd4 : (∀A· A∈aircrafts_in_airport ∧ deadline(A)>deadline(aircraft) ⇒ SIGMA({i ↦ ld|∃a· i∈
card(aircrafts_in_airport) ∧ ld=AVERAGE_LD(a) ∧ a∈aircrafts_in_airport ∧ deadline(a)
≤deadline(A)}) + deadline(aircraft) ≤ deadline(A) ∧ Urgents = ∅)) ∨ aircraft ∈ Urgents
```

For each guards there is two cases, the first one is when there is no emergency landing request (Urgents = ∅, where Urgents is the set of aircrafts requesting emergency landing) (John 2016). 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 follows:

```
CONSTANTS
        BAG
        SIGMA
        AVERAGE_LD
AXIOMS
        axm1   :   BAG={e· e∈ℕ⇸ℕ ∧ finite(e) ∧ dom(e)=1 ‥
card(e)|e}
        axm1 : SIGMA ∈ BAG→ℕ
        axm2 : SIGMA(∅) = 0
        axm3 : ∀e· e∈BAG ∧ e≠∅ ⇒ SIGMA(e)= e(card(e)) +
SIGMA({card(e)}⩤e)
END
```

The second case is when the entering aircraft is requesting an emergency landing (aircraft ∈ Urgents). The Urgents-Cases such as Aeronautical failure, Bad climate, Terrorist attacks, Kidnapping, 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, the VOR and the supervisor must look for a not used runway where the aircraft may be landed. In the second times, those having the lowest deadline are able to be land. This is formalized as guards in the start_landing event as follows:

```
grd5  :  ∀A· A∈aircrafts_in_airport ⇒ statusof(A)≠Landing
grd6  :  ((deadline(aircraft)=min({dl |∃A· A∈aircrafts_in_airport ∧ statusof(A)=ReadyL ∧
dl=deadline(A)})) ∧ Urgents=∅) ∨ aircraft∈Urgents
```

Finally, we present below a new set associated to the aircrafts in the runway which is a subset of the aircrafts_in_airport (inv 5). We introduce also an inv 6 that express that the curr_RW_status is "unavailable" if and only if there is a single airplane in 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 preserve this invariants, and also some actions for adding and removing aircrafts from the aircrafts_in_runway set.

```
Inv 5  :  aircrafts_in_runway ⊆ aircrafts_in_airport
Inv 6  :  aircrafts_in_runway≠∅ ⇒ curr_RW_status ≔ unavailable
```

Second refinement: towards an alert system for a secured ATC

In this last refinement we model safety requirements from 7 to 10 (SAF-7, SAF-8, SAF-9, and SAF-10).

In the previous models, we formalized the functional aspect of the system. Here, an alert system is presented to provide more security for the ATC (In Focus: ICAO'S Strategic Objectives 2018; Fact Sheet-FAA & NTSB's "Most Wanted" Recommendations 2010; Department of transportation federal aviation administration 2017; John 2016; NASA Air Traffic Management Demonstration Goes Live in Charlotte 2017). However, we need to firstly define the following carrier set and constants:

```
SETS
            BRAND
CONSTANTS
        LOCATIONS
        Min_distance
        Separation_Time
AXIOMS
        axm1  :  LOCATIONS = ℕ×ℕ×ℕ
        axm2  :  Min_distance ∈ ℕ
        axm3  :  Separation_Time ∈ BRAND → ℕ
END
```

In this context, we introduce the aircrafts 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 separation time between two successive aircrafts.

The first thing we consider in this last machine is how to integrate time constraint which is not predefined in Event-B. Cansell et al. (2007) presented a pattern to integrate time constraint, this pattern propose integrating the time constraint in two major steps. The first step focuses on defining time variables and the related invariants in addition to the initialization values. In the second three events are included to allow the system to consider time progression.

The pattern proposes the following two variables of time:

- *time* in N models the current time value. The incrementation of this value denotes the time progression. This variable is initialized by zero at the moment of system start that we consider the beginning of time.
- $at \subseteq N$ is the known future active times of the system. Each active time stands for future event activation. In our case, this variable stands for the landing and taking off times that we have modelled previously.

The invariant proposed in this pattern are simple, two of them are typing invariants related to the both time variables. The Third invariant ensures that active times are in the future which means that the time cannot be moved beyond the minimum active time, this should be correct because if time goes beyond one event activation, then we miss the right moment for observing it (Cansell et al. 2007).

To summarize, the first step will be modelled as follows:

```
VARIABLES
        time
        at
INVARIANT
        time ∈ N
        at ⊆ N
        at ≠ ∅ ⇒ time ≤ min(at)
INITIALISATION
        ...
        time := 0
        at := ∅
```

The three events presented in the second step represent different temporal aspects: creation of a new active time (post_time), time progressing (tick_tock), and the last event consider event in the time constraint (process_time).

The post_time event adds a new active time tm in the at set under the constrain that the new active time is in the future (tm is greater than time).

```
Post_time
ANY
        tm
WHERE
        $tm \in N$
        $tm > time$
THEN
        $at := at \cup \{tm\}$
END
```

The second event allows observing time progression; it simply takes a new value of time and assigns it in the current time variable.

```
Tick_tock
ANY
        tm
WHERE
        $tm \in N$
        $tm > time$
        $at \neq \emptyset \Rightarrow tm \leq min(at)$
THEN
        $time := tm$
END
```

The last event allows the system to consider events with time constraints. It has one guard, which is time $\in$ at meaning that the current time is an active then the only action in this event remove that active time to preserve the invariant $at \neq \emptyset \Rightarrow time \leq min(at)$.

```
Process_time
WHEN
        $time \in at$
THEN
        $at := at - \{time\}$
END
```

Now we formalize 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 flies too close behind it, it will lose aerodynamic stability (Yu and Bin 2011; In Focus: ICAO'S Strategic Objectives 2018). 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, two things must be saved to ensure the safety of the next landing: the time of the previous landing and the separation time (Carreño and Muñoz 2005; Umeno and Lynch 2007; In Focus: ICAO'S Strategic Objectives 2018; John 2016; NASA Air Traffic Management Demonstration Goes Live in Charlotte 2017). This is formalized in the Terminating_landing event as follows:

$$\boxed{\begin{array}{ll} act4: & last\_landing\_t := time \\ act5: & at := at \cup \{time + Separation\_Time(brandof(aircraft))\} \end{array}}$$

where, the current time will be saved using the last_landing_t variable and a new active time will be added which is the current time plus the separation time, where the brandof is a total function from aircrafts to brands.

To prove that the system always maintains separation time respected the following guard must be added to the start_landing event:

$$\boxed{grd10: \quad time - last\_landing\_t \geq separation\_t}$$

As well the separation time, a minimum separation distance must be maintained between aircraft during flying in the airport airspace (Narkawicz and Munoz 2015; Fact Sheet-FAA & NTSB's "Most Wanted" Recommendations 2010; John 2016). 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 presented in the previous context. To insure that the minimum distance will be kept the following invariant must be preserved:

$$\boxed{\begin{array}{ll} inv7: & \forall a,b \cdot a \in aircrafts\_in\_airport \land b \in aircrafts\_in\_airport \Rightarrow \\ & distance(locationof(a) \mapsto locationof(b)) \geq Min\_distance \end{array}}$$

The distance function is defined from LOCATIONxLOCATION to natural numbers, it calculate the distance between two aircrafts basing on their locations determinate by means of the location of function. These two functions are defined in the following two invariants:

$$\boxed{\begin{array}{ll} inv1: & locationof \in AIRCRAFTS \rightarrow LOCATIONS \\ inv2: & distance \in LOCATIONS \times LOCATIONS \rightarrow \mathbb{N} \end{array}}$$

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 trigger 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 (Platzer and Clarke 2009; In Focus: ICAO'S Strategic Objectives 2018; Fact Sheet-FAA & NTSB's "Most Wanted" Recommendations 2010; Department of transportation federal aviation administration 2017; John 2016; NASA Air Traffic Management Demonstration Goes Live in Charlotte 2017). 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.

```
Aircraft_moving_Alert_ON
ANY
      aircraft
      loc
WHERE
      grd1  :   loc ∈ LOCATIONS
      grd2  :   aircraft ∈ AIRCRAFTS
      grd3  :   (∃a· a∈AIRCRAFTS ∧ distance(locationof(a)↦loc)<Min_distance) ∨
      ¬ ( Security property 1) ∨ ¬ ( security property 2) ∨ ...
THEN
      act1  :  Alert ≔ TRUE
      act2  :  locationof(aircraft)≔loc
END
```

This alert system highly minimizes human error due the real-time control of all aircrafts movements in the radar range.

This alert system is formalized as below:

```
Aircraft_moving_Alert_OFF
ANY
        aircraft
        loc
WHERE
        grd1  :   aircraft ∈ AIRCRAFTS
        grd2  :   loc ∈ LOCATIONS
        grd3  :     (∀a· a∈AIRCRAFTS ⇒ distance(locationof(a)↦loc)≥Min_distance) ∧ (Security
        property1) ∧ (Security property 2) ∧ ...
THEN
        act1  :   Alert ≔ FALSE
        act2  :   locationof(aircraft)≔loc
END
```

Proving model correctness and result

The RODIN platform is used to prove model correctness (Rodin et al. 2011). Table 1 presents the statistics proofs generated by RODIN (Jarrar and Balouki 2018).

Table 1 Rodin report

Element name	Total	Auto	Manual
Air traffic control	205	184	21
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	68	60	8

The Table 1 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 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.

Conclusion

We discussed the steps and reasoning involved in the construction of a model of take-off and landing at an airport runway using the Event-B modeling language and verified with the Rodin tool. The main contribution is presenting a standard model that can be used to develop an air traffic control system. We cover the most important requirements proposed by the most well-know organizations in this domain, which facilitate the development of such a complex system. According to the requirements document, engineers may add more refinements by formalizing their typical requirements to develop the final model.

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 (Bouarfa et al. 2013), which highly improves system feasibility. Furthermore, we aim to apply standardizations such as QoS (Jarrar et al. 2017) and RM-ODP (Belhaj et al. 2010) 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 omni directional radio range; DM: deadline monotonic; QoS: quality of service; RM-ODP: reference model of open distributed processing.

Authors' contributions

Both authors contribute in collecting information, writing, modeling, and reviewing. Both authors read and approved the final manuscript.

Acknowledgements

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

Competing interests
The authors declare that they have no competing interests.

Availability of data and materials
No dataset was used in this work. This work is based on international recommendation.

Funding
The study was not funded.

Publisher's Note

Received: 3 May 2018 Accepted: 23 August 2018
Published online: 03 September 2018

References
Abrial J-R (2010) Modeling in Event-B: system and software engineering. Cambridge University Press, New York
Abrial JR, Butler M, Hallerstede S, Hoang TS, Mehta F, Voisin L (2010) Rodin: an open toolset for modelling and reasoning in Event-B. Int J Softw Tools Technol Transfer 12(6):447–466
Belhaj H, Balouki Y, Bouhdadi M, El Hajji S (2010) Using Event B to specify QoS in ODP enterprise language. In: Working conference on virtual enterprises. Springer, Berlin, pp 478–485
Bouarfa S, Blom HA, Curran R, Everdij MH (2013) Agent-based modeling and simulation of emergent behavior in air transportation. Compl Adapt Syst Model 1(1):15
Cansell D, Méry D, Rehm J (2007) Time constraint patterns for Event B development. In: International conference of B users. Springer, Berlin, pp 140–154
Clarke EM, Wing JM (1996) Formal methods: state of the art and future directions. ACM Comput Surv (CSUR) 28(4):626–643
Department of transportation federal aviation administration, aeronautical information publication, United States of America (2017) 24 edn, amendment 2
Dominique M, Neeraj KS (2011) EB2J: code generation from Event-B to Java, SBMF. IN: Brazilian symposium on formal methods, São Paulo, Brazil
Dominique M, Singh NK (2016) Modeling an aircraft landing system in Event-B. In: International conference on abstract state machines, alloy, B, TLA, VDM, and Z, ABZ 2014 ABZ: the landing Gear case study. Springer, Berlin, pp 154–159
Fact Sheet-FAA & NTSB's "Most Wanted" Recommendations (2010) https://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=11186. Accessed 20 May 2018
Hoang TS, Kuruma H, Basin D, Abrial JR (2009) Developing topology discovery in Event-B. Sci Comput Program 74(11–12):879–899
iFACTS-Air Traffic Management System (2017) https://www.adacore.com/customers/uks-next-generation-atc-system. Accessed 13 May 2018
In Focus: ICAO'S Strategic Objectives (2018) https://www.icao.int/Pages/default.aspx. Accessed 10 May 2018
Jarrar A et al (2017) Modeling aircraft landing scheduling in Event B. In: International conference on information technology and communication systems. Springer, Berlin, pp 127–142
Jarrar A, Balouki Y (2018) Formal reasoning for air traffic control system using Event-B method. In: International conference on computational science and its applications. Springer, Cham, pp 241–252
Jarrar A, Balouki Y, Gadi T (2017) Formal specification of QoS negotiation in ODP system. Int J Elec Comput Eng 7(4):2045
John S, Duncan (2016) Airplane flying handbook. Department of transportation Federal Aviation Administration Flight Standards Service, FAA-H-8083-38
Lecomte T, Servat T, Pouzancre G (2007) Formal methods in safety-critical railway systems. In: 10th Brasilian symposium on formal methods, pp 29–31
Luo S, Yu G (1998) Airline schedule perturbation problem: landing and takeoff with nonsplitable resource for the ground delay program. In: Operations research in the airline industry. Springer, Boston, pp 404–432
Lygeros J, Lynch N (1997) On the formal verification of the TCAS conflict resolution algorithms. In: Proceedings of the 36th IEEE conference on decision and control. pp 1829–1834
Narkawicz A, Munoz C (2015) A formally verified conflict detection algorithm for polynomial trajectories. In: AIAA Infotech@ Aerospace. p 0795
NASA Air Traffic Management Demonstration Goes Live in Charlotte (2017) https://www.nasa.gov/aero/nasa-air-traffic-management-demo-goes-live. Accessed 20 May 2018
Pinol H, Beasley JE (2006) Scatter search and bionomic algorithms for the aircraft landing problem. Eur J Oper Res 171(2):439–462
Platzer A, Clarke EM (2009) Formal verification of curved flight collision avoidance maneuvers: a case study. In: International symposium on formal methods. Springer, Berlin, pp 547–562
Rodin C, Jastram M, Butler M (2011) User's handbook
Schmidt CV et al (2017) First come, first served: the first-emerging queen monopolizes reproduction in the ant Cardiocondyla "argyrotricha". J Ethol 35(1):21–27
Su W, Abrial JR (2017) Aircraft landing gear system: approaches with Event-B to the modeling of an industrial system. Int J Softw Tools Technol Transf 19(2):141–166
Tomlin C, Pappas GJ, Sastry S (1998) Conflict resolution for air traffic management: a study in multiagent hybrid systems. IEEE Trans Autom Control 43(4):509–521

Vairaktarakis GL, Aydinliyim T (2007) Benchmark schedules for subcontracted operations: decentralization inefficiencies that arise from competition and first-come-first-served processing. Decis Sci 484:657–690

Victor C, Muñoz C (2005) Safety verification of the small aircraft transportation system concept of operations. In: AIAA 5th aviation, technology, integration, and operations conference (ATIO), Arlington, Virginia

Vistbakka I, Troubitsyna E (2018) Towards integrated modelling of dynamic access control with UML and Event-B. arXiv preprint arXiv:1805.05521

Umeno S, Lynch N Safety verification of an aircraft landing protocol: a refinement approach. In: International workshop on hybrid systems: computation and control. Springer, Berlin, pp 557–572

Yang Q, Tian J, Zhao T (2017) Safety is an emergent property: illustrating functional resonance in Air Traffic Management with formal verification. Saf Sci 93:162–177

Yu SP, Bin Cao X, Zhang J (2011) A real-time schedule method for aircraft landing scheduling problem based on cellular automation. Appl Soft Comput J 11(4):3485–3493

Zafar NA (2016) Formal specification and analysis of take-off procedure using VDM-SL. Compl Adapt Syst Model 4(5):4