Jarrar *et al. Complex Adapt Syst Model* (2020) 8:3
https://doi.org/10.1186/s40294-020-0069-7

Complex Adaptive Systems Modeling

RESEARCH Open Access

Formal approach to model complex adaptive computing systems

Abdessamad Jarrar[1*], Abderrahim Ait Wakrime[2] and Youssef Balouki[1]

*Correspondence: Abdessamad.jarrar@gmail.com
[1] Faculty of Sciences and Technologies of Settat, Informatics Imaging and Modeling of Complex Systems Laboratory, University Hassan First, Settat, Morocco
Full list of author information is available at the end of the article

Abstract

Complex adaptive systems provide a significant number of concepts such as reaction, interaction, adaptation, and evolution. In general, these concepts are modelled employing different techniques which give an inexplicit vision on the system. Therefore, all concepts must be carefully modelled using the same approach to avoid contradiction and guarantee system homogeneity and correctness. However, developing a computing system that includes all these concepts using the same approach is not an easy task and requires a perfect understanding of the system's behaviour. In this paper, we contribute as stepwise towards proposing an approach to model the most important concepts of complex adaptive systems while ensuring homogeneity and the correctness of models. For this aim, we present five standard agent-based models formalizing agent properties, reaction, interaction, adaptation, and evolution. These models are adapted to all cases of complex adaptive systems since they include an abstract description of these concepts. To implement our approach formally, we choose the Event-B method due to the strong assurance of bugs' absence that it guarantees. Besides, it supports horizontal and vertical refinement which facilitates the specification process. Furthermore, the approach of this paper addresses the very abstract level of modelling which expand the use of this approach to other formal methods and tools.

Keywords: Formal methods, Complex adaptive systems, Event-B, Agent, Interaction, Reaction, Evolution, Adaptation

Introduction

Developing computer systems that benefit from complex adaptive systems concepts is a significant and challenging task in the context of computer science engineering. The development of such a system will allow its elements to interact, react, adapt and even evolve autonomously (Kharchenko et al. 2017). The implementation of these capabilities correctly will make the dream of making computers more independent and intelligent than human come true. However, the achievement of this dream is constrained by hardware and software limits as well as the safety requirements. For example, developing robots that can serve humans should be performed while taking into consideration the capability of analysing the surrounding environment and commands understanding besides the safety of the served human (Siciliano and Khatib 2019). The first step to

implementing complex adaptive systems is modelling it correctly to deal with its complexity and avoid failures. One of the most powerful design tools in computer engineering is formal methods based on theorem-proving due to their strong assurance of bugs' absence.

In this paper, we present an approach to develop correctly a complex adaptive system formal model while ensuring homogeneity and correctness. In particular, we focus on developing a standard model that plays the role of a starting point in the process of developing any complex adaptive system. This model includes the essential concepts of complex adaptive systems that can be enriched by adding detail depending on the typical requirements of the system.

Formal methods are techniques based on the mathematical language used for specifying and verifying systems. Utilization of formal techniques can extraordinarily expand our comprehension of the system by uncovering inconsistencies, ambiguities, and deficiency that may be harder to detect using standard methods. The formal method used for system-level modelling and analysis is Event-B, which is based on set theory notation. Event-B uses refinement to represent the different abstraction levels of the system while performing proofs to guarantee consistency of refinements. This method has been effectively used in different systems where no bugs were detected. For instance, the driverless meteor in Paris worked correctly and no bugs were identified after the theoretical proofs, neither at the functional approval (Boudi et al. 2019). This achievement encourages Alstom and Siemens Transportation Systems to use this method to develop future metros (Vistbakka and Troubitsyna 2018). In addition, several approaches have been proposed that use Event-B to provide a correct system (Wakrime et al. 2018a, b). Therefore, we also used Event-B to develop the concepts presented in this paper.

In many works, researchers tried to apply formal methods to verify the correctness of complex systems where they applied different concepts such as the process algebra CSP (Bartels and Kleine 2011), SOTA (Abeywickrama and Zambonelli 2012), MAPE-K (Iglesia and Weyns 2015), SMARTOS (Giese 2016), and LTL (Sadraddini and Belta 2017). In these works, researchers proved that the application of formal methods highly improve the security and correctness of complex systems. Unlike all these works that presents ready-made formal models, the correct-by-construction approach proposed in this work presents a systematic way that guides engineers in using a formal method to develop any complex adaptive system. Furthermore, this approach uses the basics of first-order predicate logic and set theory which make it easy and convenient for users of different scientific backgrounds.

In order to illustrate the proposed approach, we present a case study of modeling an air traffic control system as a complex adaptive system. The presented approach is a generalization of the proposed one in our previous work (Jarrar and Balouki 2018). The previous work deals with the application of Event-B to verify and analyze an air traffic control system; whereas, this work present a more general approach that can be applied in the case of any complex adaptive system.

The rest of the paper is structured as follows. Section 2 presents the state of art, related works, and we give some background on formal methods and validation tools used in this paper. In Sect. 3, we give a summary of the proposed approach in 5 steps, after that, we present the main contribution in term of a methodology for modeling each concept

of a Complex Adaptive System: agent, reaction, interaction, evolution, and adaptation. In Sect. 4, we illustrate the use of our approach through a case studied of an air traffic control system ATC. Section 5 focuses on the validation and verification of the model generated in the case study to ensure the validity of our approach. Finally, we conclude in Sect. 6.

Background

In this section, we specify the technical terms and concepts related to the complex adaptive systems and formal method Event-B. In addition, we introduce the related works to position our contribution.

Complex adaptive systems

Nowadays, computing systems are becoming more and more complex and adaptive through the inspiration of concepts from the natural complex adaptive systems. Many concepts were inspired when observing systems such as social insect colonies, embryo development as well as the immune system. On the other hand, many other systems in nature such as the biosphere and the human brain are tried to be simulated as complex adaptive systems. Furthermore, the human being, the stock exchange, the business world, the social group are also considered of the complex adaptive systems.

Recently, the scientific community has opted for an adaptive approach to the development of computing systems. The integration of this approach provides a set of benefits such as autonomous adaptation and evolution of the system elements called agents. Adaptation of an agent is based on detecting the existence of a problem occurred due to environmental change and then trying to adopt a suitable strategy to avoid chaos. Whereas evolution relies on the control of the state of the system to identify the behaviours of the components that allow the system to evolve and strengthen it and at the same time impoverishing the behaviour that negatively influences evolution. Figure 1 bellows gives an overview of the complex adaptive system structure:

Formal method: Event-B

To overcome the high risk of failure in complex systems, it is extremely important to perform preliminary modeling based on a certain theory. The main purpose of these theories is explaining, predicting, and understanding some characteristics of the system before construction. These theories are used in different field of science such as Maxwell's equations in electrical engineering and theory of material's strength in civil engineering. In computer engineering, formal methods are less considered, therefore, there are engineers who do not know any theory building computing systems (Clayton and Radcliffe 2018). These systems often have bugs that may cost millions to fix. Hence, the use of formal methods is highly recommended for verifying and specifying software in order to highly guarantee bugs' absence.

Event-B is a system-level modelling formal method that provides a set of features such as analysis and the correct-by-construct approach performed by theorem proving (Vistbakka and Troubitsyna 2018; Abrial 2010). Modelling in Event-B is introduced as a set of successive models that are made of two parts: Context (the static part) and Machine (the dynamic part). The context includes the definition of carried sets, constants, and

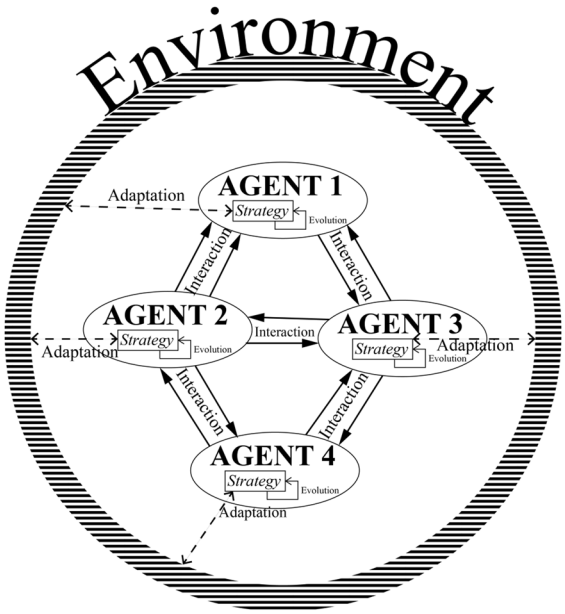

Fig. 1 Structure of complex adaptive systems

axioms. The machine is made up of variables that describe the state of the systems. The states of the systems are constrained by mathematical formulas called invariants. The states transitions are performed by events. The events are made up of actions that are the elementary operation allowing the change of variables values. The events may have some necessary conditions to trigger, these conditions are called guards.

Event-B supports a very practical technique of modelling, which is refinement. Refinement relies on creating an abstract model including the basic elements of the system; after that, more details are added to build successive models that are more and more concrete. This technique makes modelling 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 2 below illuminates of the process of this modelling process:

In order to guarantees the well-performance of this process, a set of proofs called proof obligations should be discharged. These proofs ensure the consistency of models and the invariant preservations as well as the non-contradiction of refinements. The most important proof among these is the invariant preservation which guarantees that all events preserve a certain invariant. More formally, let I a certain invariant, A refers to the set of axioms, c refers to constants, while s are the carried sets, v and v' are respectively the before and after values of variables. The invariant preservation proof is formalized in (1).

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

Almost all proofs are discharged automatically using an eclipse based platform called Rodin (2017). In the case of other few ones, a set of predicates may be added to guide the interactive prover.

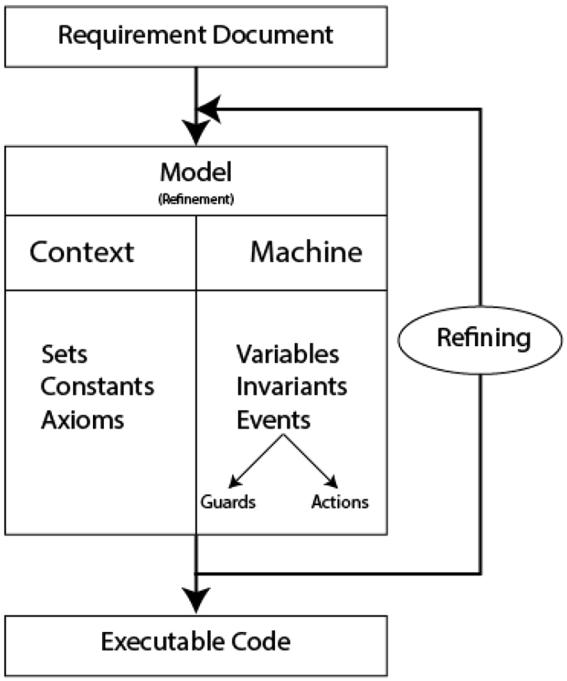

Fig. 2 Process of modelling

Related works

Recent works about complex adaptive systems

Given the complexity that can be increased explosively of this kind of systems compared to the increase in the number of components, their engineering must be carried out with great attention to minimize the error rate and avoid the chaos that these errors may generate. The majority of these errors are caused by issues related to the design phase and the use of standard methods to develop such complex systems.

To overcome these problems, many works addressed methods to model and study the behaviour of these systems before the construction phase. The most used approach to model an Adaptive Complex System is that of agent-based modeling. This approach has been defined by Volker (Grimm et al. 2005) as a class of computer models for simulating the actions and interactions of autonomous agents (individual or collective entities such as organizations or groups) to evaluate their effects on the environment system as a whole. It combines elements of game theory, complex systems, emergence, computer sociology, multi-agent systems and evolutionary programming. Monte Carlo methods are used to introduce chance.

This method has been used in many works to simulate the behaviour of different systems such as the retail electrical energy markets with demand response (Dehghanpour et al. 2018), urban crimes development (Groff et al. 2019), and the emergence of bio-jet fuel supply chain in Brazil (Moncada et al. 2019). These works present a graphical and numerical result of a simple agent-based modelling simulation performed through mathematical models. However, the verification of complex behaviour is constrained by the performance of the machine used for simulation and still always test for finite states; therefore, the verification does not perfectly ensure the absence of failures.

Formal methods for complex adaptive systems

The use of formal methods in the development of complex systems is highly recommended due to the high assurance of bugs' absence that they provide. However, in order to use them, a very clear understanding of the system behaviour is a requirement which makes their use in the case of Complex Adaptive Systems a difficult task. Therefore, only a few works attempt to use formal methods in order to model and verify complex adaptive system. The authors of Bartels and Kleine (2011) study the specification and verification of adaptive systems seen as a subclass of reactive systems using the process algebra CSP. This allows them to use a set CSP tools for the verification such as CSP-Prover and ProB. They also introduce an approach for the implantation of a specified system. In Abeywickrama and Zambonelli (2012) DB Abeywickrama and al. present Model Checking Goal-Oriented Requirements for Self-Adaptive Systems. To this end, authors show how to perform a goal-level model checking analysis of complex adaptive systems by SOTA (State of the Affairs) as a general goal-oriented modelling framework. The effectiveness of their method is exposed by transforming the conceptual SOTA model into an operational one. On the other hand, Christopher and al give an overview in Rouff et al. (2012) of an approach for the verification of adaptive software systems. This approach is a result of the combination of different fields of science: stabilization science, high-performance computing simulations, compositional verification and traditional verification techniques, and operational monitors. In Iglesia and Weyns (2015), the authors propose a promising approach to handle designing software systems that have to deal with dynamic operating conditions through self-adaptive dynamic realized by a MAPE-K (Monitor-Analyze-Plan-Execute plus Knowledge). They validate their approach using formal specified MAPE-K templates that encode design expertise for a family of self-adaptive systems. Authors of Giese (2016) also present formal models and analysis for self-adaptive cyber-physical systems. They analyze the challenges face self-adaptive cyber-physical systems and outline their results through a generic approach based on extensions of graph transformations systems called SMARTOS. Sadraddini and Belta (2017) is considered as one of the latest work in this domain in which authors use formal methods for adaptive control of dynamical systems. They develop a method to control discrete-time systems with initially unknown parameters from LTL (linear temporal logic) specifications. They also show how to compute adaptive control strategies for finite and infinite systems using formal methods.

Another approach has been presented by Niazi and Hussain (2010), authors proposed a formal agent-based simulation framework (FABS). This framework is based on formal specification to describe wireless sensor networks used for sensing the environment of complex adaptive system. The approach is applied to a boids model of self-organized flocking of animals monitored by a random deployment of proximity sensors. Zafar (2016) presented a formal specification of take-off procedure using Vienna Development Method-Specification Language VDM-SL complex adaptive systems modeling in order to overcome communication failures that cause delays and collisions. The presented model is developed by a series of refinements and made up of (i) a static part that includes the graph-based model, aircrafts, controllers, taxiways and runways, whereas, (ii) the dynamic part describe the take-off algorithms. The same formal method along with the graph theory is used for modeling wireless sensor and actors networks

(WSANs) earthquake disaster mitigation and management system by Zafar et al. in Zafar and Afzaal (2017). Although these approaches are formally well presented, still these work are 'ad-hoc' for specific cases of complex adaptive systems and wireless sensor networks. On the other hand, our approach is a more general and can be applied in any case of complex adaptive system modeling and it is independent of the used technology which extend its applicability.

Unlike all the works presented in this section that presents ready-made formal models, our approach presents a systematic standards way to guide in generating of formal models while guaranteeing the homogeneity of models and the use of basic mathematical concepts. For this study, we use a formal method called Event-B that present a set of tools that can be used for the verification of models such as proof obligations and model checking through animation. These tools ensure the correctness of models and thus the resulting implemented system is correct by construction.

Proposed approach

The purpose of this section is to present our contributions namely formal approach to model complex adaptive computing systems using Event-B.

Summary of approach

Our approach is as follow:

1. Filtering the desired concepts needed in the case study. During this stage we specify among the possible concepts those which interest us (reaction, interaction, adaptation, and evolution).
2. Filtering and formalizing the related requirements from the requirement document. These requirements will be inserted in step 4.
3. Adapt the standard model with the system requirements by renaming the elements of the model and adding appropriate properties and types. This step will generate and an abstract typical initial model.
4. Refine the model as much as needed by formalizing and adding the remaining typical requirements as well as the establishment of proof obligations. At the end of this step, we get a typical concrete model.
5. Construct the system according to the concrete typical model which will be guaranteed to be correct by construction.

Figure 3 below illustrates the outlines of the proposed approach:

Formal approach to model complex adaptive computing systems using Event-B

Agent

Agents can be defined as the smallest elements of a complex adaptive system, these agents have the capability to interact with each other in order to adapt, evolve and solve problems (Niazi 2017). The main purpose of the use of agents is that they can act separately and independently as well as the ability to add and remove easily agents without stopping the system. Thus, we propose the following Definition 1 for an agent:

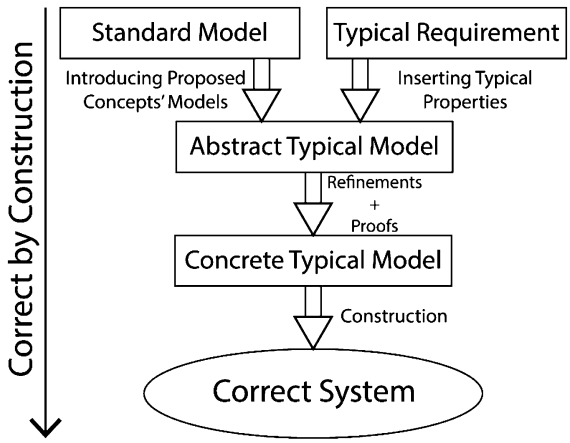

Fig. 3 Proposed approach

Definition 1 An agent is the smallest element of a complex adaptive system that can act independently.

These agents are defined using a set of properties such as types and strategies that determine their behaviour in certain circumstances. Although these properties may seem very simple, the significant number of agents in the same CAS beside the variety of their properties arise explosively the difficulty of predicting system behaviour. Also, a simple change of a single property of a certain agent may cause a completely new complex and potentially novel behaviours of the system. Therefore, properly defining the agents and their properties is the most critical part of modelling a complex adaptive system (Akram and Niazi 2018).

For this purpose, we start with the following simple formalization shown in Listing 1 and 2 that captures the agents' main properties:

CONTEXT
SETS
AGENTS
CONSTANTS
LOCATIONS
AXIOMS
$axm : LOCATIONS = \mathbb{N} \times \mathbb{N} \times \mathbb{N}$
END

Listing 1. Context of agents locations

MACHINE

...

INVARIANTS

Inv : Agent ∈ AGENTS

Inv : Location ∈ AGENTS→LOCATIONS

Listing 2. Location/agents typical invariants

We define a set AGENTS that refer to all possible agents that may exist in the system currently, in the past or even in the future; therefore, the AGENTS set can be modelled as carried set and can be seen as a type of variables.

The location of an agent can affect many aspects of how it operates which makes formalizing it extremely important (Roundy et al. 2018). We introduce a constant LOCATIONS denoting the natural coordinate space N^3 which formalize altitude, longitude and latitude. Then, we "link" agents with their locations by means of a total function from the set of AGENTS to the set of LOCATIONS.

When one or more agents share the same characteristics, it is possible to associate to these agents a type (Abar et al. 2017). Groups of agents of the same type are defined as subsets of the AGENTS set. For instance, to define 3 types of agents we use the following formalism in Listing 3:

CONTEXT

SETS

AGENTS

Type1

Type2

Type3

CONSTANTS

LOCATIONS

AXIOMS

axm : LOCATIONS = ℕ×ℕ×ℕ

axm : partition(AGENTS, Type1, Type2, Type3)

END

Listing 3. Types of agents

The partition predicate is an easy way to enumerate sets. Mathematically, the partition predicate is defined in (2):

$$partition(S, x, y) \Leftrightarrow x \cup y = S \wedge x \cap y = \emptyset \tag{2}$$

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

Each agent has a number of properties that describe it such as colour, speed, size, etc. These properties may change during the system lifetime in order to adapt, react or evolve (Mittal and Risco-Martín 2017). Therefore, we propose a modelization of these properties in term of a total function from the set of agents to the set that includes all the possible values of a certain property. This function can be denoted *AgentPropertyi* (this notation is necessary when the type of property values is not primitive). For instance, in order to describe the colour of an agent, we propose a function named "AgentColour" defined from the set AGENTS to the set COLOURS that include all possible colours that an agent may have. This method simplifies the modification of the colour of an agent by only modifying the value of AgentColour (agent) (Burns et al. 2017). In general, for a certain property *i* we define the set of all its possible values in the context as a carrier set denoted *PROPERTYi* as shown in Listing 4. Then, we define the total function in Listing 5 below:

CONTEXT
SETS
PROPERTYi
CONSTANTS
Value1
Value2
Value3
...
AXIOMS
...
axm : PROPERTYi={Value1, Value2, Value3, ...}
END

Listing 4. Definition of properties set

MACHINE
...
INVARIANTS
...
Inv : AgentPropertyi ∈ AGENTS→PROPERTYi

Listing 5. Agent properties total function

Reaction

Due to the unpredictable change of environment, each agent should have a strategy in term of stimulus/response that indicates what to do in which circumstances (Matheson and Thompson-Schill 2019). A stimulus occurs by changing a certain factor (the

effect of the environment) that affect agent behaviour such as temperature, wind speed, light, etc. Sometimes, the agent should be able to react to a certain stimulus in order to remain functioning correctly (surviving) (Durniak et al. 2017). Therefore, for some stimulus, we should assign a response to guarantee the appropriate reaction. The appropriate response for a certain stimulus is up to the engineer to determine, while this is a typical problem—each problem has its own stimulus/response logic. The following state diagram in Fig. 4 describes the reaction process:

We propose in this paper an abstract formalization of this stimulus/response combination according to the following Definition 2:

Definition 2 The reaction is adopting an appropriate response for an external stimulus.

The approach proposed in this paper to formalize the strategy is by means of two sets: STIMULUS and RESPONSES. Similarly to the AGENTS and PROPERTYi sets, STIMULUS and RESPONSES refer to all the possible stimulus and responses; hence there formalization in B language is similar to the previous ones.

To formalize the association between each stimulus of an agent with an appropriate response, we present the following total function in Listing 6:

MACHINE

...

INVARIANTS

...

Inv : AppropriateResponse $\in AGENTS \times STIMULUS \rightarrow RESPONSES$

Listing 6. Formalization of Stimulus/response associations.

In order to formalize the fact that agents in a CAS are able to react, we use an event that occurs each time a stimulus appears. In this event and for a certain agent that was affected by a certain stimulus, an appropriate response should take place and reconfigure the agent properties. This is in other words: reacting to stimulus.

The reaction event can be presented as follows in Listing 7:

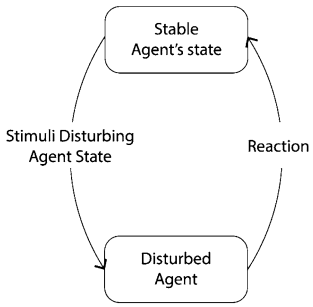

Fig. 4 Reaction process

```
Agent_reacting
ANY
    agent
    s
WHERE
    grd1 : agent ∈ AGENTS
    grd2 : s ∈ STIMULUS
THEN
    act1 : AgentProperty1(agent) ≔ Property1Response( agent ↦
AppropriateResponse(agent ↦ s))
    act2 : AgentProperty2(agent) ≔ Property2Response( agent ↦
AppropriateResponse(agent ↦ s))
    ...
END
```

Listing 7. Agents reaction event

PropertyiResponse is a function that determines for a certain agent the appropriate new value of the property *i* based on a specific response. The definition of this function is as follows in Listing 8:

```
Inv : PropertyiResponse ∈ AGENTS×RESPONSES→PROPERTIESi
```

Listing 8. Appropriate properties values determination function.

Interaction

In order to develop the responses to stimuli, agents should enrich consistently their knowledge base. The most efficient way to build their knowledge base is by interaction/communication and sharing experiences (Conklin et al. 2019; Sadri et al. 2019). The interaction between two agents can be described as illustrated in the following Fig. 5:

The interaction in a complex adaptive system is constrained by either the physical location or logical localization. Therefore, the interaction of agents can be defined as follows:

Definition 3 Interaction is the exchange of information between agents in a complex adaptive system.

An agent can primarily interact in a direct way with the agents surrounding it (its neighbors). Hence, we should associate to each agent a number of neighbors which will at the same time define the structure of the whole system (notice that in some cases, other properties may be added such as the distance between two agents). This can be formalized by means of the following function in Listing 9:

```
Inv : Neighbors ∈ AGENTS→ ℙ(AGENTS)
```

Listing 9. Neighbors function

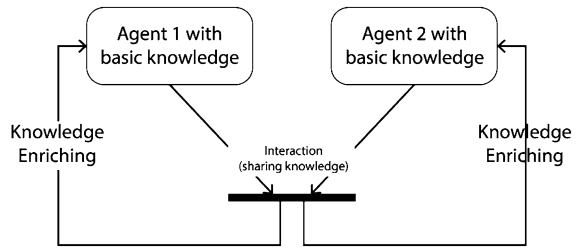

Fig. 5 Interaction process

P is the power set function which refers to the set of all subsets of AGENTS.

The interaction of agents implies that each agent has a set of data called agent knowledge (Grillitsch et al. 2019). This knowledge helps the agents to make decisions, share knowledge, adapt and evolve. Such a concept is very typical hence we will present an abstract formalization of this concept in term of a set of data associated with each agent. Listing 10 below presents the knowledge function as a total function from the set agents to the power set of data.

$$Inv : Knowledge \in AGENTS \rightarrow \mathbb{P}(DATA)$$

Listing 10. Knowledge function

where DATA is a carried set of data.

While we had presented the basic elements of interaction (neighbors and knowledge), we can formalize the interaction between neighbors agents in term of an event that occur each time two agents interact or exchange data in order to learn (notice that the interaction as an agent affecting physically its neighbor is seen in our approach as a stimulus that was treated in the previous section). The agent interaction event is formalized in Listing 11 as follows:

Agent_interacting
ANY
agent
neighbor
data
WHERE
grd1 : $agent \in AGENTS$
grd2 : $neighbor \in Neighbors(agent)$
grd3 : $data \in Knowledge(neighbor)$
THEN
act1 : $Knowledge(agent) \coloneqq Knowledge(agent) \cup \{data\}$
END

Listing 11. Interaction event

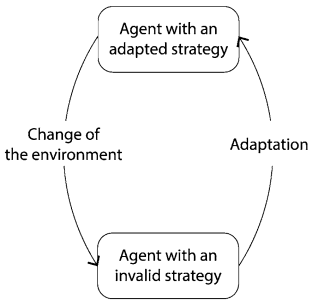

Fig. 6 Interaction process

Adaptation

This concept was inspired from the biological adaptation which is defined as the process of adapting to a certain environment to survive. In the same way, the adaptation of an agent in complex adaptive systems is the change of strategy in order to survive and guarantee the correct performance of its tasks (Aldrich et al. 2019). The following state diagram in Fig. 6 illustrates the adaptation process in a complex adaptive system:

We can define adaptation as follows in Definition 4:

Definition 4 Adaptation of an agent is changing its strategy to conform a change of environment.

The adaptation is necessary when the state of an agent is deteriorated because of the environment change; this means that a number of specific stimuli can be observed indicating this change of environment. Therefore, a new strategy should be adopted to adapt to the new environment which can be formalized as the total function in Listing 12 that

$$Inv : AdaptedStrategy \in P(STIMULUS) \rightarrow (AGENTS \times STIMULUS \rightarrow RESPONSES)$$

Listing 12. Adapted strategy function

determine for each environment (set of stimuli) the strategy that should be adopted:

In order to formalize the adaptation of an agent, we present the following event that triggers for a certain agent when some stimuli are observed (Rozantsev et al. 2019). To make the event more accurate, we add two guards: the first ensure that the current strategy is not the adapted one (grd3); while the second ensures the deterioration of the agent state (grd4). When all the guards are verified, the current strategy is changed by changing the AppropriateResponse. This is formalized as shown in Listing 13 below:

```
Agent_Adapting
ANY
    Agent
    Stimuli
WHERE
    grd1 : agent ∈ AGENTS
    grd2 : Stimuli ∈ P(STIMULUS)
    grd4 : AppropriateResponse≠ AdaptedStrategy(Stimuli)
    grd3 : state(Neighbors(agent) ∪ {agent}) < Previous_state(agent)
THEN
    act1 : AppropriateResponse := AdaptedStrategy(Neighbors(agent) ∪ {agent})
END
```

Listing 13. Adaptation event

Evolving

The evolution concept as illustrated in definition 5 was also inspired from the evolution theory in biology that can be defined as an improvement in the heritable biological characteristics of a population over a long period of time in successive generations (Hall and Strickberger 2008).

Definition 5 Evolution in a complex adaptive system is developing gradually by improving agents strategies.

This evolution can be measured by the improvement of the whole system state. However, it is claimed in the complex adaptive system literature that no agent can visualize the state of the whole system; still, each agent can see its neighbors. Thus, we assume that agents are able to measure the state of the subsystem of their neighbors (the agent itself included). This ability will help the agent to observe the effect of a change in its strategy on the subsystem and decide either it will help the system to improve or deteriorate. In addition to the ability to measure the neighbors' subsystem state, an agent should be able to memorize the before-change state to compare it with the after-change state; and memorize also the old strategy in order to reuse it if needed (the case of deterioration).

Figure 7 summarizes the proposed evolving approach:

The proposed evolution process is based on 3 phases:

- Evolution attempt starting: in this phase, the agent proposes a random new strategy by changing at least one stimulus/response combination and memorizes the previous strategy alongside the neighbors' subsystem state.
- Deciding phase: during this phase, the agent measures the current state of the neighbours' subsystem and then compares it with the previous state in order to decide if the change is improving or deteriorating the system. This phase should occur after

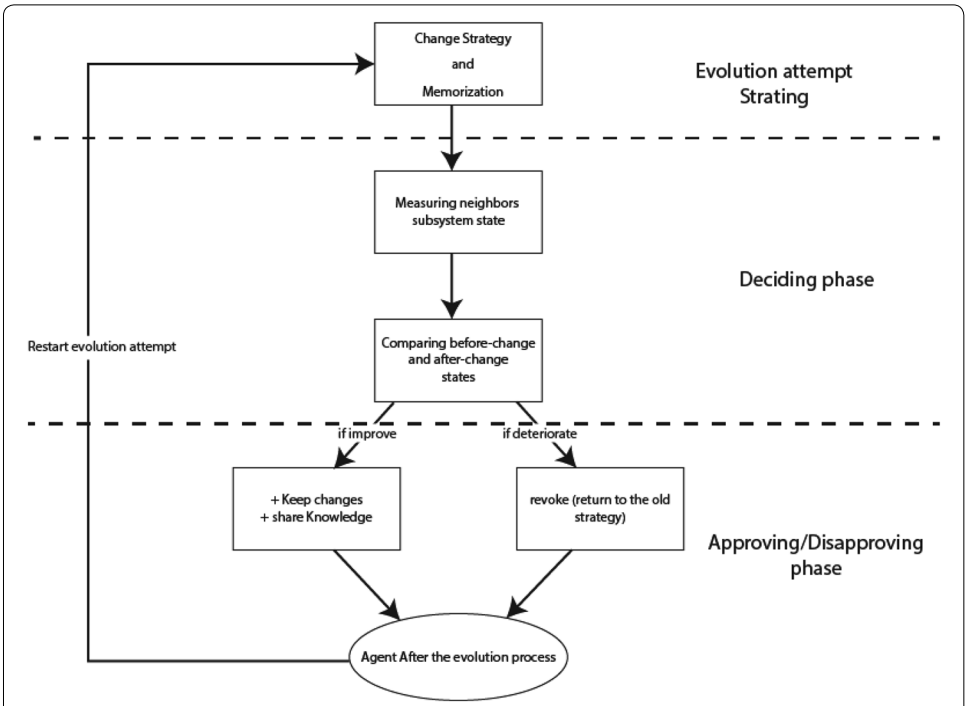

Fig. 7 Evolution process

a certain time from the first one to give the agent time to observe the after-change state of the subsystem.

- Approving/disapproving phase: lastly, the agent may approve the change of strategy and evolve or disapprove changes and readopt the old strategy. If the evolution is approved then the agent will share this experience with others so that they may try the same thing. Otherwise, the agent readopts the old strategy that was previously memorized during the *evolution attempt starting* phase.

Notice that this process is based on the generation of random strategies; thus, the system should start with a predefined acceptable strategy. Formally, this can be defined in the initialization event (the first event that triggers at the beginning of the system lifetime).

To translate our proposed process of evolution into Event-B, we need to start with the basic elements and abilities that an agent needs to be able to evolve. Firstly, we introduce the total function Previous_strategy defined from the AGENTS set to ($AGENTS \times STIMULUS \rightarrow RESPONSES$) that allows the agent to memorize the previous strategy. Secondly, the total function Previous_state defined from AGENTS to natural numbers that represent the before-change neighbours' subsystem state in term of a natural number; this number increases if the system improves and decreases if it deteriorates. Finally, we present the State function that formalizes the assumption that an agent is able to measure the state of a subsystem. This state function is defined from the power set of AGENTS to a certain natural number. The formalization of the functions used during evolution is presented in Listing 14 below:

$Inv : Previous\_strategy \in AGENTS \rightarrow (AGENTS \times STIMULUS \rightarrow RESPONSES)$

$Inv : Previous\_state \in AGENTS \rightarrow \mathbb{N}$

$Inv : State \in P(AGENTS) \rightarrow \mathbb{N}$

Listing 14. Evolution functions

The first event that triggers during *evolution attempt starting* phase is the memorization event that forms a sort of backup data before starting the evolution attempt. The second event in this phase allows the agent to change its strategy. However, during modelling, we figure out that if two neighbours are attempting to evolve the process may fail because in this case the after-change state that an agent will observe does not reflect exactly the effect of the change in its strategy; rather than that, it reflects the effect of the change of both strategies. To avoid this problem, we add a new function that its value is equal to 1 if an agent is currently in the process of evolution and 0 if not. This function can be formalized as shown in Listing 15 below:

$Inv : Evolving\_state \in AGENTS \rightarrow \{0,1\}$

Listing 15. State of evolution function

This function will help us to ensure before starting the evolution attempt that all the neighbours of the agent are not currently attempting to evolve.

Additionally, the process of evolution is related to the concept of time. The formalization of this concept proposed in this approach is inspired by the time pattern proposed by Dominique Cansell et al. (2007). This formalization is based on formalizing the time as a natural number that increases over time. The value of this variable will represent the time when the last time the memorization event trigger which is the evolution attempt starting moment. Thus, the memorization and changing strategy events can be formalized as shown in the following Listing 16 and 17:

Memorization
ANY
Agent
Current_time
WHERE
grd1 : agent $\in$ *AGENTS*
THEN
act1 : Previous_state(agent) $:=$ *state(Neighbors(agent)* $\cup$ *{agent})*
act2 : Previous_strategy(agent) $:=$ *AppropriateResponse*
act3 : time $:=$ *current_time*
END

Listing 16. Memorization event

Change_strategy
ANY
Agent
strategy
WHERE
grd1 : agent ∈ AGENTS
grd2 : strategy ∈ AGENTS×STIMULUS→RESPONSES
grd3 : ∀a· a∈ Neighbors(agent) ⇒ Evolving_state(a)=0
grd4 : Evolving_state(agent) = 0
THEN
act1 : AppropriateResponse ≔ strategy
act2 : Evolving_state(agent) ≔ 1
END

Listing 17. Strategy change event

One last thing should be mentioned in this first phase, since the strategy function is taking the agent as a parameter then it can access to all its properties and most importantly its knowledge. This allows the agent to evolve considering its previous experience and without repeating the same attempt several times.

The decision of either the new strategy is helping the system to evolve or deteriorate is done after a certain time (denoted *separation_time*) to allow the agent to observe the effect of the new strategy (Son et al. 2019). After that, the agent measures the state of the neighbours' subsystem and compares it with the previous state and then decides either the new strategy should be approved or disapproved. For this purpose, we present the following two events in Listing 18 and 19:

Approving_change
ANY
Agent
Current_time
WHERE
grd1 : agent ∈ AGENTS
grd2 : Evolving_state(agent) = 1
grd3 : current_time > time + separation_time
grd4 : state(Neighbors(agent) ∪ {agent}) > Previous_state(agent)
THEN
act1 : Knowledge ≔ (λa·a∈Neighbors(agent) | Knowledge(agent) ∪ StrategyToKnowledge(AppropriateResponse ↦ Approved)) ∪ (λa·a∉Neighbors(agent) | Knowledge(agent))
act2 : Evolving_state(agent) ≔ 0
END

Listing 18. Approving change event

Disapproving_change
ANY
Agent
Current_time
WHERE
grd1 : agent ∈ AGENTS
grd2 : Evolving_state(agent) = 1
grd3 : current_time > time + separation_time
grd4 : state(Neighbors(agent) ∪ {agent}) ≤ Previous_state(agent)
THEN
act1 : AppropriateResponse ≔ Previous_strategy(agent)
act2 : Knowledge ≔ (λa·a∈Neighbors(agent) | Knowledge(agent) ∪ StrategyToKnowledge(AppropriateResponse ↦ Disapproved)) ∪ (λa·a∉Neighbors(agent) | Knowledge(agent))
act3 : Evolving_state(agent) ≔ 0
END

Listing 19. Disapproving change event

In both events, the second guard helps to ensure that the agent is in the evolution process; at the same time, the third guard guarantees the minimum separation time. The decision about the strategy is done by means of the fourth guard which compare the current and previous state of the subsystem. If the subsystem is evolving then the approving event triggers which share this experience –the new strategy helps the agent to evolve- and terminate the evolution process. As you may notice, the agent should be able to translate the experience into data that can be added to the current knowledge of neighbours. Therefore, we introduce in Listing 20 the StrategyToKnowledge function that translates a strategy and decision into data:

Inv : StrategyToKnowledge ∈ (AGENTS×STIMULUS→RESPONSES) ×{Approved,Disapproved} → P(DATA)

Listing 20. Converting strategy to knowledge function

In the approving event, the *StrategyToKnowledge* function takes Approved as the second parameter while in the disapproving event it takes Disapproved. Sharing the experience of evolution or deteriorating will give a high priority of trying the new strategy in future evolution attempts of neighbours in the evolution case, and ensure high avoidance possibility of the strategy in the case of deteriorating.

Case study

This section presents a case study to extrapolate results of our contributions and provide a means for understanding our formal approach with greater clarity.

Air traffic control in an airport vicinity

Air traffic control (ATC) is a service provided currently by controllers located in the control tower. The main responsibility of the controllers is organizing and expediting the air traffic flow while preventing collisions and minimizing delays (Fact Sheet 2010). This service is performed due to the structure of the communication network between the aircrafts and the control tower. Figure 8 below illustrates this structure:

Therefore, if the control tower has a technical issue or natural disaster, the aircrafts will not be able to react autonomously. For this reason, we propose a complex adaptive system structure for ATC system making aircraft able to react to the different kind of external stimuli such as Headwind, Crosswinds and Tailwinds. Aircrafts should also be able to interact and exchange data over a network in order to provide a more general vision about the surrounding environment. Based on this vision, the aircrafts may need to adapt by changing its way of flying and reacting (it strategy) which highly improve their performance to carry out their tasks. This interaction and adaptation will allow the aircrafts to build a rich knowledge base that permits them to perform attempts to evolve. However, automated evolution is never guaranteed which raises the risk of deterioration; therefore, aircrafts should be able to observe the result of their attempts to prevent this risk.

The purpose of this case study is presenting a brief example of how to apply the proposed approach in the air traffic control system. This work is based on our previous work in this domain (Blok et al. 2018; Jarrar and Balouki 2018a, b; Jarrar et al. 2017).

Requirement document

Since the main purpose of this work is guiding and building the first step of modelling correctly complex adaptive systems, we need to reorganize and reformulate the requirement document in a more formal way. JR Abrial proposes in Abrial (2010) an approach of presenting the requirement document in a more suitable form based on presenting it along two axes: Fun that refers to the functional requirements of the system; and Env that refer to environment assumptions and non-functional requirements.

The requirement document presented here focus on the future vision of the most popular organizations in the domain of airspace systems: International Civil Aviation

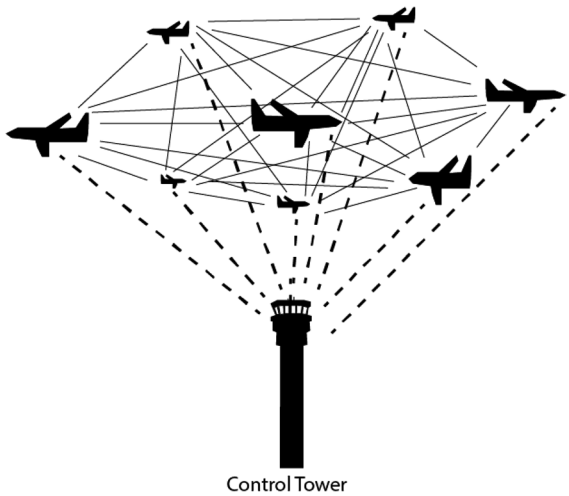

Fig. 8 Air traffic control structure

Organization ICAO, Federal Aviation Administration FAA, and National Aeronautics and Space Administration (NASA) (In Focus 2018; Fact Sheet 2010; Department of Transportation Federal Aviation Administration 2017; National Aeronautics and Space Administration NASA 2009). Their vision focuses on making the aircrafts smart and minimizes as much as possible the participation of human beings to avoid errors. The proposed requirements document is as follows:

The aircrafts of the system are intelligent and able to react independently	Fun 1
Aircrafts are either fixed wings or rotorcrafts	Fun 2
Each aircraft has a manufacturer	Fun 3
Aircrafts always have vertical and direction angle as well as linear speed	Fun 4

Figure 9 illustrates the vertical and direction angle of an aircraft:

Different types of winds may be faced during flights: Headwind, Crosswinds and Tailwinds	Env 1
Aircrafts are able to react in the case of Headwind, Crosswinds and Tailwinds	Fun 5
Aircrafts are able to interact and exchange data over a network	Fun 6
Aircrafts should keep a minimum separation distance to prevent collisions	Fun 7
Each aircraft has its own knowledge base	Fun 8
The environment of the system is unpredictably and frequently changeable	Env 2
Aircraft are able to adapt in case of environment changes	Fun 9
Aircrafts are supported with a strategy that allows it to only evolve and prevented it from deterioration	Fun 10

Formalization of air traffic control

Formalizing agents

In the air traffic control system, the agents can be considered as the aircrafts and they have different properties. In this case study, we propose the following properties: type, manufacturer, speed, vertical angle, directional angle and location (Fun 4).

We introduce two types of aircrafts: fixed-wing and rotorcrafts (Fun 2). We also present the following manufacturer: Airbus, Boeing, Bombardier, Cessna, and Cirrus (Fun 3).

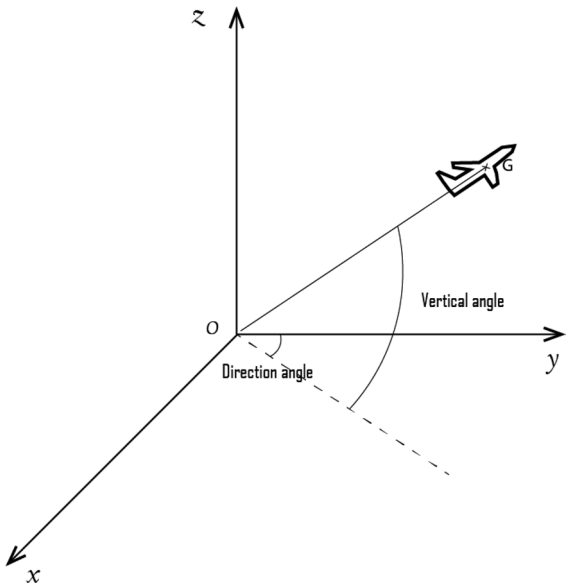

Fig. 9 Direction and vertical angles

The context of the system is formalized as follows in listing 21:

CONTEXT

SETS

AIRCRAFTS

MANUFACTURER

CONSTANTS

Fixed_wings

Rotorcrafts

LOCATIONS

Airbus

Boeing

Bombardier

Cessna

Cirrus

AXIOMS

axm 1 : LOCATIONS $= \mathbb{N} \times \mathbb{N} \times \mathbb{N}$

axm 2 : Fixed_wings $\in \mathbb{P}$*(AIRCRAFTS)*

axm 3 : Rotorcrafts $\in \mathbb{P}$*(AIRCRAFTS)*

axm 4 : partition(AIRCRAFTS, Fixed_wings, Rotorcrafts)

*axm 5 : partition(MANUFACTURER, {*Airbus}, {Boeing}, {Bombardier} , {Cessna}, {Cirrus*})*

END

Listing 21. Aircrafts Locations/Types/Manufacturers modelisation

In the machine, we present the set of all functions that formalize the association agent-property (Jarrar et al. 2017). This can be formalized as shown in Listing 22:

INVARIANTS

Inv 1 : aircraft ∈ AIRCRAFTS

Inv 2 : Location ∈ AIRCRAFTS→LOCATIONS

Inv 3 :AircraftManufacturer ∈ AIRCRAFTS→MANUFACTURER

Inv 4 : AircraftSpeed ∈ AIRCRAFTS→ ℕ

Inv 5 : VerticalAngle ∈ AIRCRAFTS→ 0..360

Inv 6 : DirectionalAngle ∈ AIRCRAFTS→ 0..360

Listing 22. Configuration functions

Formalizing reaction of aircrafts

In this section, we will present an example of how to model aircraft reaction in case of strong wind. The wind is one of the main elements that affect an aircraft's flight; therefore it is one of the most important stimuli that should be considered when modelling aircraft reaction. The headwind is preferred by pilots for landing and taking off due to its benefits of using less runway and low ground speed at touchdown. On the other hand, Crosswinds and tailwinds are more difficult to deal with (Env 1), and aircrafts should adapt to the situation either by changing speed, angles or even location (Fun 5). To summarise, the wind may affect three properties: speed, vertical angle, and directional angle (iFACTS 2018).

To formalize the aircrafts reaction, the first elements that should be defined are STIMULUS and RESPONSES. All possible stimuli and responses should be formalized as shown in Listing 23 below:

CONTEXT

SETS

...

STIMULUS

RESPONSES

CONSTANTS

...

Headwind

Crosswinds

Tailwinds

R_Headwind

R_Crosswinds

R_Tailwinds

AXIOMS

...

axm 4 : partition(STIMULUS, {Headwind}, {Crosswinds},{Tailwinds})

axm 5 : partition(RESPONSE, {R_Headwind}, {R_Crosswinds}, {R_Tailwinds})

END

Listing 23. Context including Stimulus and responses

In the machine, the AppropriateResponse function should be added beside the different function that allows adaptation of angular and location properties; these are LocationResponse, VerticalAngleResponse, and DirectionalAngleResponse. These functions are presented as invariants in Listing 24:

Inv 7 : AppropriateResponse $\in AIRCRAFTS \times STIMULUS \rightarrow RESPONSES$

Inv 8 : LocationResponse $\in AIRCRAFTS \times RESPONSES \rightarrow LOCATIONS$

Inv 9 : VerticalAngleResponse $\in AIRCRAFTS \times RESPONSES \rightarrow 0..360$

Inv 10 : DirectionalAngleResponse $\in AIRCRAFTS \times RESPONSES \rightarrow 0..360$

Listing 24. Reaction functions

Finally, the reacting event can be formalized as shown in Listing 25:

Aircrafts_reacting
ANY
a
s
WHERE
grd1 : a ∈ AIRCRAFTS
grd2 : s ∈ STIMULUS
THEN
act1 : Location (a) ≔ LocationResponse*(a ↦ AppropriateResponse(a ↦ s))*
act2 : VerticalAngle (a) ≔ VerticalAngleResponse*(a ↦ AppropriateResponse(a ↦ s))*
act3 : DirectionalAngle (a) ≔ DirectionalAngleResponse*(a ↦AppropriateResponse(a ↦ s))*
END

Listing 25. The reaction of aircrafts event

Formalizing interaction between aircrafts

The first thing that should be presented to guarantee the interaction between aircraft is modelling the structure of aircrafts network (Fun 6); this can be formalized by means of the neighbors function. Besides, we introduce the knowledge function that formalizes the knowledge base of each aircraft (Fun 8). For air traffic management systems, aircrafts should always maintain a minimum distance between them which makes the distance between aircrafts also an important factor in the system (In Focus 2018). Therefore, we present an additional function formalizing the distance between two aircrafts. This can be formalized in Listing 26 as follows:

Inv 11 : Neighbors ∈ AIRCRAFTS→P(AIRCRAFTS)
Inv 12 : Knowledge ∈ AIRCRAFTS→P(DATA)
Inv 13 : Distance ∈ LOCATIONS×LOCATIONS → ℕ

Listing 26. Interaction of aircrafts functions

The knowledge of aircraft is building by exchanging data over the network with neighbors. This can be formalized as shown in the Listing 27 below:

Aircraft_interacting
ANY
a
neighbor
data
WHERE
grd1 : a ∈ AIRCRAFTS
grd2 : neighbor ∈ Neighbors(a)
grd3 : data ∈ Knowledge(neighbor)
THEN
act1 : Knowledge(a) ≔ Knowledge(a) ∪ {data}
END

Listing 27. Interaction of aircrafts event

Finally, the distance and location functions can be used to guarantee a minimum separation distance between aircraft during flying in the airport airspace (Fun 7). If two aircrafts are keeping this distance, collision will be strongly avoided as well as wake turbulence. The minimum distance is fixed and we denoted it Min_distance constant. To ensure that the minimum distance will be kept the following invariant presented in Listing 28 must be preserved:

$$Inv14: \forall a,b \cdot a \in AIRCRAFTS \wedge b \in AIRCRAFTS \Rightarrow distance(location(a) \mapsto location(b)) \geq Min\_distance$$

Listing 28. Separation distance invariant.

Formalizing Adaptation of aircrafts

In order for an aircraft to adapt (Fun 9), it is necessary to define for each change of environment ($P(STIMULUS)$) a certain strategy ($AIRCRAFTS \times STIMULUS \rightarrow RESPONSES$) that is more suitable to the new environment. This is illuminated in Listing 29 below:

$$Inv\ 15: AdaptedStrategy \in P(STIMULUS) \rightarrow (AIRCRAFTS \times STIMULUS \rightarrow RESPONSES)$$

Listing 29. Adapted strategy for aircrafts

The event responsible for adapting aircrafts is denoted Aircraft_Adapting. This event is constrained by the condition that the current state is worse than the previous one, and the condition that the current strategy is not the adapted one. In this case,

the Aircraft_Adapting event presented in Listing 30 will switch the current strategy to the adapted one defined by the AdaptedStrategy function.

```
Aircraft_Adapting
ANY
   a
   Stimuli
WHERE
   grd1 :  a ∈ AIRCRAFTS
   grd2 :  Stimuli ∈ P(STIMULUS)
   grd3 :  AppropriateResponse≠ AdaptedStrategy(Stimuli)
   grd4 :   state(Neighbors(a) ∪ {a}) < Previous_state(a)
THEN
   act1 :  AppropriateResponse ≔ AdaptedStrategy(Stimuli)
END
```

Listing 30. Aircrafts adaptation event

Formalizing the evolution process

Before starting the formalization of the evolution process (Fun 10), we present the Previous_strategy and Previous_state in order to make a backup of the aircraft state while we cannot ensure if the evolution will succeed or not. Besides, we introduce the state function that formalized a feature that an aircraft may have easily; this feature is the collection of information of its surrounding aircrafts and evaluates the state of the neighbours' subsystem state. Also, a total function will be needed to recognize either an aircraft is in the process of evolution attempt or not; this is necessary to avoid multiple evolution attempts of the same aircraft at the same time. The functions need for aircrafts evolution as presented in Listing 31 below:

```
Inv16:Previous_strategy∈AIRCRAFTS→(AIRCRAFTS×STIMULUS→RESPONSES )
Inv 17 : Previous_state ∈ AIRCRAFTS→ ℕ
Inv 18 : State ∈ P(AIRCRAFTS)→ ℕ
Inv 19 : Evolving_state ∈ AIRCRAFTS→ {0,1}
```

Listing 31. Aircrafts evolution functions

The first event that triggers in the process of evolution is the Memorization event presented in Listing 32. This event allows the backup of before-change state and strategy as well as the time when the process started.

```
Memorization
ANY
  a
  Current_time
WHERE
  grd1 : a ∈ AIRCRAFTS
  grd2 : Current_time ∈ ℕ
THEN
  act1 : Previous_state(a) ≔ state(Neighbors(a) ∪ {a})
  act2 : Previous_strategy(a) ≔ AppropriateResponse
  act3 : time ≔ current_time
END
```

Listing 32. Aircrafts memorization event

The next event is called Change_strategy; this event changes the strategy of a certain aircraft that has no neighbor in the process of evolution as shown in Listing 33. However, if two neighbors are changing their strategies at the same time, the effect of these strategies will not be clear while these two neighbors will share some neighbors that will be affected by both strategies changes. Therefore, we allow the change of strategy for only aircrafts that do not have any neighbor in the process of evolution.

```
Change_strategy
ANY
  a
  strategy
WHERE
  grd1 : a ∈ AIRCRAFTS
  grd2 : strategy ∈ AIRCRAFTS×STIMULUS→RESPONSES
  grd3 : ∀x· x∈ Neighbors(a) ⇒ Evolving_state(x)=0
  grd4 : Evolving_state(a) = 0
THEN
  act1 : AppropriateResponse ≔ strategy
  act2 : Evolving_state(a) ≔ 1
END
```

Listing 33. Changing aircraft strategy event

After a certain time denoted separation_time, the aircraft may decide if the change should be approved or not based on the before and after-change states. If the before change state is better than the after change state then the Approving_change event will trigger, otherwise, the one who will trigger is the Disapproving_change event. When the Approving_change trigger, it enriches the knowledge of the evolving aircraft and this knowledge will be shared with its neighbors by means of interaction; and then it terminates the process of evolution by associating to the evolving state the value zero as

shown in Listing 34 (Department of Transportation Federal Aviation Administration 2017). In the same way, if disapproving_change trigger then the previous strategy backed up during the memorization event will be readopted; this is shown in Listing 35. Also, the knowledge of the aircraft will be enriched to avoid trying the same strategy over and over. These two events can be formalized as belows:

Approving_change
ANY
a
Current_time
WHERE
grd1 : a ∈ AIRCRAFTS
grd2 : Current_time ∈ ℕ
grd3 : Evolving_state(a) = 1
grd4 : current_time > time + separation_time
grd5 : state(Neighbors(a) ∪ {a}) > Previous_state(a)
THEN
act1 : Knowledge ≔ (λx·x∈Neighbors(a) | Knowledge(a) ∪ StrategyToKnowledge(AppropriateResponse ↦Approved)) ∪ (λx·x∉Neighbors(agent) | Knowledge(a))
act2 : Evolving_state(agent) ≔ 0
END

Listing 34. Approving evolution of aircraft event

Disapproving_change
ANY
a
Current_time
WHERE
grd1 : a∈ AIRCRAFTS
grd2 : Current_time ∈ ℕ
grd3 : Evolving_state(a) = 1
grd4 : current_time > time + separation_time
grd5 : state(Neighbors(a) ∪ {a}) ≤ Previous_state(a)
THEN
act1 : AppropriateResponse ≔ Previous_strategy(a)
act2 : Knowledge ≔ (λx·x∈Neighbors(a) | Knowledge(a) ∪ StrategyToKnowledge(AppropriateResponse ↦Disapproved)) ∪ (λx·x∉Neighbors(a) | Knowledge(a))
act3 : Evolving_state(a) ≔ 0
END

Listing 35. Disapproving evolution of aircraft event

Before starting the proof verification, one last event should be added. This event is called the INITIALISATION event; it is an event without guards that what happen at the beginning. In general, this event initializes all the variables in the proposed model. The Listing 36 below presents the initialization of all the variables:

INITIALISATION

BEGIN

act1 : aircraft :∈ AIRCRAFTS

act2 : location :∈ AIRCRAFTS→LOCATIONS

act3 : AircraftManufacturer :∈ AIRCRAFTS→MANUFACTURER

act4 : AircraftSpeed :∈ AIRCRAFTS→ ℕ

act5 : verticalAngle :∈ AIRCRAFTS→ 0··360

act6 : DirectionalAngle :∈ AIRCRAFTS→ 0··360

act7 : AppropriateResponse :∈ AIRCRAFTS×STIMULUS→RESPONSES

act8 : LocationResponse :∈ AIRCRAFTS×RESPONSES→LOCATIONS

act9 : VerticalAngleResponse :∈ AIRCRAFTS×RESPONSES→0··360

act10 : DirectionalAngleResponse :∈ AIRCRAFTS×RESPONSES→ 0··360

act11 : Neighbors :∈ AIRCRAFTS→ℙ(AIRCRAFTS)

act12 : Knowledge :∈ AIRCRAFTS→ℙ(DATA)

act13 : Distance :∈ LOCATIONS×LOCATIONS → {Min_distance}

act14 : AdaptedStrategy :∈ ℙ(STIMULUS)→(AIRCRAFTS×STIMULUS →RESPONSES)

act15 : Previous_strategy:∈AIRCRAFTS→(AIRCRAFTS × STIMULUS →RESPONSES)

act16 : Previous_state :∈ AIRCRAFTS→ ℕ

act17 : State :∈ ℙ(AIRCRAFTS)→ ℕ

act18 : Evolving_state :∈ AIRCRAFTS→ {0,1}

act19 : time:=0

act20 : Separation_time :∈ ℕ

act21 : StrategyToKnowledge :∈ (AIRCRAFTS×STIMULUS→RESPONSES) × {Approved,Disapproved} → ℙ(DATA)

END

Listing 36. INITIALIZATION Event

Verification and validation

In this section, we are interested in experiments to validate our work. To do this, we used a verification using proof obligations and verification using ProB animator.

Verification using proof obligations

In order to guarantee the correctness of the proposed model in the case study, we use the Rodin platform to perform proof obligations. Table 1 presented below illuminate statistics about the established proofs:

This table is generated automatically by Rodin and presents the number of proofs generated including manual (6) and automatic (109) proofs. Most of these proofs are Invariants preservation verifications which guarantee that all the invariants are maintained verified during the system lifetime. Therefore, the construction of a system based on the proposed model will generate a correct by construction system.

Verification using ProB animator

The validation of invariants preservation guarantees that all the before and after event states verify always all the invariants, which is a very critical validation requirement. Still, this validation is not enough to ensure the correctness; however, in some cases, we may arrive at a state where not guard is verified. In these cases, all events will be prevented from triggering since their guards are not verified. This is why another type of validation is needed which is the Deadlock Freedom.

In order to guarantee the deadlock freedom, two methods are provided: Invariant preservation of one additional invariant, and model checking through animation (Bozga et al. 2019). The first method is based on adding an invariant that include the disjunction of all the model events guards. This method guarantees the deadlock-freedom through ensuring that at least one of the guards is verified no matter what is the state of the system. Listing 37 illustrates the invariant of deadlock freedom:

$$
\begin{aligned}
&Inv14: \quad (\exists a,s \cdot a \in AIRCRAFTS \land s \in STIMULUS) \lor \\
&(\exists a, neighbour, data \cdot a \in AIRCRAFTS \land neighbor \in Neighbors(a) \land data \in Knowledge(neighbor)) \lor \\
&(\exists a,stimuli \cdot a \in AIRCRAFTS\ stimuli \in \mathbb{P}(STIMULUS) \land AppropriateResponse \neq AdaptedStrategy(stimuli) \land State(Neighbors(a) \cup \{a\}) < Previous\_state(a)\ ) \lor \\
&(\exists a, Current\_time \cdot a \in AIRCRAFTS \land Current\_time \in \mathbb{N}) \lor \\
&(\exists a, strategy \cdot a \in AIRCRAFTS \land strategy \in AIRCRAFTS \times STIMULUS \rightarrow RESPONSES \land \forall x \cdot x \in Neighbors(a) \Rightarrow Evolving\_state(x)=0 \land Evolving\_state(a) = 0) \lor \\
&(\exists a, Current\_time \cdot a \in AIRCRAFTS \land Current\_time \in \mathbb{N} \land Evolving\_state(a) = 1 \land Current\_time > time + Separation\_time \land State(Neighbors(a) \cup \{a\}) > Previous\_state(a)) \lor \\
&(\exists a, Current\_time \cdot a \in AIRCRAFTS \land Current\_time \in \mathbb{N} \land Evolving\_state(a) = 1 \land Current\_time > time + Separation\_time \land State(Neighbors(a) \cup \{a\}) \leq Previous\_state(a))
\end{aligned}
$$

Listing 37. Deadlock-freedom verification invariant

As you may notice, the invariant is overcharged and contain a significant number of predicate which complicates the establishment of the invariant preservation proof. Therefore, the second method based on model checking through animation is more

Table 1 Rodin report

Element name	Total	Auto	Manual
ATC system	115	109	6
Context	7	7	0
Machine	108	102	6

suitable for this kind of verification. In almost all cases, if there are deadlocks in the model they will be revealed through animating the model several times with a big number of steps. For this task, we use a tool called ProB Animator. ProB is a feature of the Rodin platform that allows the animation, constraint solving and model checking of models written in the B-language (Körner and Bendisposto 2018). We have used this tool to animate our model using one hundred steps and the model always was free of deadlocks. Figure 10 illustrates the result of ProB model checking with a total of 14,568 transitions that visited 4779 nodes. During this check no deadlock or invariant violation was detected which highly guarantees Deadlock freedom in our model. All experiments were conducted on a 64-bit PC, Windows 7 Professional operating system, an Intel Core i7, 2.13 GHz Processor with 4 cores and 4 GB RAM.

To conclude, our approach allows the creation of a model that is verified by proof obligations and animation; as a result, a system constructed based on this model will be correct by construction. Therefore, our approach provides a systematic way to build highly verified formal models of computing complex adaptive systems using the Event-B method.

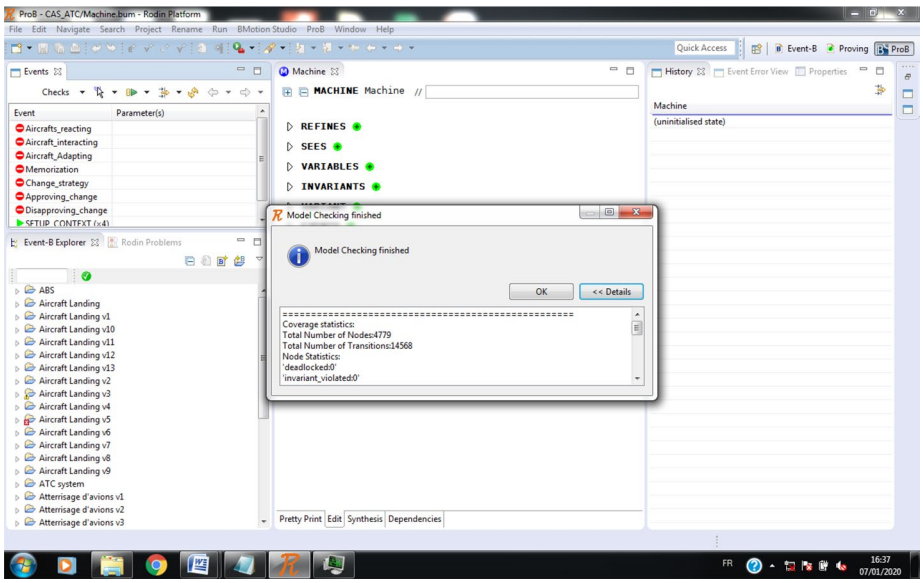

Fig. 10 Deadlock-freedom verification screenshot

Conclusion

We have introduced the steps and reasoning involved in the construction of a model of complex adaptive systems using the Event-B formal method. The main contribution is presenting a methodology for modelling that can be used to develop any complex adaptive systems. We cover the most important concepts proposed by the most well-known works in this domain, which facilitate the development of such a complex system. According to the requirements document, the developers of the system may add more refinements by including their typical requirements to build a customized model suitable to their cases. To illustrated the proposed approach, we have presented a use case of an air traffic control system. The requirement document of this use case was build based on several organizations descriptions (FAA, ICAO, and NASA). After the application of the approach, a model, that includes all these requirements in addition to complex adaptive systems concepts,has been produced. The validation of the model was guaranteed by two types of verifications: proof obligations using Rodin and Deadlock freedom using ProB. Therefore, the correctness of the generated model of the case study is guaranteed which demonstrate the validity of the proposed approach.

Due to the huge number of concepts and characteristics of Complex Adaptive System, the presentation of a methodology to formalize them all in a single work is not an easy task. Therefore, we intend, In future works, to improve the proposed approach by considering the rest of the complex adaptive systems concepts such as populations that refer to collections of agents or strategies, self-similarity, interaction pattern, selection of strategies, emergence and self-organization. Moreover, the combination of this approach with other verification methods such as model checking will highly minimize the probability of failures. Furthermore, we aim to develop a complete recommendation for other standardizations such as QoS and RM-ODP (Jarrar et al. 2017; Jarrar and Al 2019).

Abbreviations

CAS: Complex adaptive system; ATC: Air traffic control system; ICAO: International Civil Aviation Organization; FAA: Federal Aviation Administration; NASA: National Aeronautics and Space Administration; QoS: Quality of service; RM-ODP: Reference Model of Open Distributed Processing.

Acknowledgements

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

Authors' contributions

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

Funding

The study was not funded.

Availability of data and materials

No dataset was used in this work. This work is based on international recommendations.

Competing interests

The authors declare that they have no competing interests.

Author details

[1] Faculty of Sciences and Technologies of Settat, Informatics Imaging and Modeling of Complex Systems Laboratory, University Hassan First, Settat, Morocco. [2] Computer Science Department, Faculty of Sciences, Mohammed V University, Rabat, Morocco.

Received: 23 November 2019 Accepted: 16 January 2020
Published online: 12 February 2020

References

Abar S, Theodoropoulos GK, Lemarinier P, O'Hare GM (2017) Agent based modelling and simulation tools: a review of the state-of-art software. Comput Sci Rev 24:13–33

Abeywickrama DB, Zambonelli F (2012) Model checking goal-oriented requirements for self-adaptive systems. In: 2012 IEEE 19th international conference and workshops on engineering of computer-based systems, IEEE, New York, pp 33–42

Abrial J-R (2010) Modeling in Event-B: system and software engineering. Cambridge University Press, New York

Akram W, Niazi MA (2018) A formal specification framework for smart grid components. Complex Adapt Syst Model 6(1):5

Aldrich J, Garlan D, Kästner C, Le Goues C, Mohseni-Kabir A, Ruchkin I, Voysey I (2019) Model-based adaptation for robotics software. IEEE Softw 36(2):83–90

Bartels B, Kleine M (2011) A CSP-based framework for the specification, verification, and implementation of adaptive systems. In: Proceedings of the 6th international symposium on software engineering for adaptive and self-managing systems, ACM, New York, pp 158–167

Blok AN, Sharpanskykh A, Vert M (2018) Formal and computational modeling of anticipation mechanisms of resilience in the complex sociotechnical air transport system. Complex Adapt Syst Model 6(1):7

Boudi Z, Ait Wakrime A, Collart-Dutilleul S, Haloua M (2019) Introducing B-sequenced petri nets as a CPN sub-class for safe train control. In: Proceedings of the 14th international conference on evaluation of novel approaches to software engineering, SCITEPRESS-Science and Technology Publications, LDA, pp 350–358

Bozga M, Iosif R, Sifakis J (2019) Checking deadlock-freedom of parametric component-based systems. In: International conference on tools and algorithms for the construction and analysis of systems, Springer, Cham, pp 3–20

Burns AJ, Posey C, Courtney JF, Roberts TL, Nanayakkara P (2017) Organizational information security as a complex adaptive system: insights from three agent-based models. Inf Syst Front 19(3):509–524

Cansell D, Méry D, Rehm J (2007) Time constraint patterns for event B development. In: International conference of B users. Springer, Berlin, Heidelberg, pp 140–154

Clayton T, Radcliffe N (2018) Sustainability: a systems approach. Routledge, Abingdon

Conklin SM, Davidson PL, Archambault A, Lee JY, Berg J, Zuo P, Rogan P (2019) U.S. Patent Application No. 10/270,819

Dehghanpour K, Nehrir MH, Sheppard JW, Kelly NC (2018) Agent-based modeling of retail electrical energy markets with demand response. IEEE Trans Smart Grid 9(4):3465–3475

Department of Transportation Federal Aviation Administration (2017) Aeronautical information publication, 24th ed. United States of America, amendment 2

Durniak T, Friedlander RR, Kraemer JR, Linton J (2017) U.S. Patent Application No. 14/752,230

Fact Sheet—FAA & NTSB's "Most Wanted" Recommendations (2010) https://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=11186. Acessed 15 Sept 2018

Giese H (2016) Formal models and analysis for self-adaptive cyber-physical systems. In international workshop on formal aspects of component software, Springer, Cham, pp 3–9

Grillitsch M, Schubert T, Srholec M (2019) Knowledge base combinations and firm growth. Res Policy 48(1):234–247

Grimm V, Revilla E, Berger U, Jeltsch F, Mooij WM, Railsback SF, De Angelis DL (2005) Pattern-oriented modeling of agent-based complex systems: lessons from ecology. Science 310(5750):987–991

Groff ER, Johnson SD, Thornton A (2019) State of the art in agent-based modeling of urban crime: an overview. J Quant Criminol 35(1):155–193

Hall B, Strickberger MW (2008) Strickberger's evolution. Jones & Bartlett Learning, Burlington

iFACTS—Air Traffic Management System (2018) https://www.adacore.com/customers/uks-next-generation-atc-system. Acessed 15 Sept 2018

Iglesia DGDL, Weyns D (2015) MAPE-K formal templates to rigorously design behaviors for self-adaptive systems. ACM Trans Auton Adapt Syst 10(3):15

In Focus: ICAO'S Strategic Objectives (2018) https://www.icao.int/Pages/default.aspx. Acessed 15 Sept 2018

Jarrar A et al. (2017) Modeling aircraft landing scheduling in Event B. In: International conference on information technology and communication systems. Springer, pp 127–142

Jarrar A, Al (2019) Reference model of open distributed processing basic modelling concepts in Event-B. In: Third international conference on computing and wireless communication systems proceeding, ICCWCS 2019, EAI, http://dx.doi.org/10.4108/eai.24-4-2019.2284096

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 (2018b) Formal modeling of a complex adaptive air traffic control system. Complex Adapt Syst Model 6(1):6

Jarrar A, Balouki Y (2018) Towards sophisticated air traffic control system using formal methods. Model Simul Eng 2018

Jarrar A, Balouki Y, Gadi T (2017) Formal specification of QoS negotiation in ODP system. Int J Electr Comput Eng 7(4):2045

Kharchenko V, Kondratenko Y, Kacprzyk J (Eds.) (2017) Green IT engineering: concepts, models, complex systems architectures. Springer International Publishing, Berlin

Körner P, Bendisposto J (2018) Distributed model checking using ProB. In: NASA formal methods symposium, Springer, Cham, pp 244–260

Matheson HE, Thompson-Schill SL (2019) Investigating grounded conceptualization: stimulus-response compatibility for tool handles is due to spatial attention. J Exp Psychol Hum Percept Perform 45(4):441

Mittal S, Risco-Martín JL (2017) Simulation-based complex adaptive systems. In: Guide to simulation-based disciplines, Springer, Cham, pp. 127–150

Moncada JA, Verstegen JA, Posada JA, Junginger M, Lukszo Z, Faaij A, Weijnen M (2019) Exploring the emergence of a biojet fuel supply chain in Brazil: an agent-based modeling approach. GCB Bioenergy. 11(6):773–790

National Aeronautics and Space Administration NASA, NASA Official: Brian Dunbar. Past Projects: Intelligent Flight Control System IFCS (2009) https://www.nasa.gov/centers/dryden/research/IFCS/index.html. Acessed 15 Sept 2018

Niazi MA (2017) Towards a novel unified framework for developing formal, network and validated agent-based simulation models of complex adaptive systems. arXiv preprint arXiv:1708.02357

Niazi MA, Hussain A (2010) A novel agent-based simulation framework for sensing in complex adaptive environments. IEEE Sens J 11(2):404–412
Rodin GXVLM (2017) Rodin. IIC-international review of intellectual property and competition law 48(5):592–598
Rouff C, Buskens R, Pullum L, Cui X, Hinchey M (2012) The AdaptiV approach to verification of adaptive systems. In: Proceedings of the fifth international c* conference on computer science and software engineering, ACM, New York, pp 118–122
Roundy PT, Bradshaw M, Brockman BK (2018) The emergence of entrepreneurial ecosystems: a complex adaptive systems approach. J Bus Res 86:1–10
Rozantsev A, Salzmann M, Fua P (2019) Beyond sharing weights for deep domain adaptation. IEEE Trans Pattern Anal Mach Intell 41(4):801–814
Sadraddini S, Belta C (2017) Formal methods for adaptive control of dynamical systems. In 2017 IEEE 56th annual conference on decision and control (CDC), IEEE, New York, pp 1782–1787
Sadri AM, Hasan S, Ukkusuri SV (2019) Joint inference of user community and interest patterns in social interaction networks. Soc Netw Anal Mining 9(1):11
Siciliano B, Khatib O (2019) Humanoid robots: historical perspective, overview, and scope. Humanoid robotics: a reference, pp 3–8
Son CHC, Kim B, Seo J (2019) Evolution map based on advance invention, process, and case studies. J Int TRIZ Assoc Matriz 1:61–73
Vistbakka I, Troubitsyna E (2018) Towards integrated modelling of dynamic access control with UML and Event-B. arXiv preprint arXiv:1805.05521
Wakrime AA, Ayed RB, Collart-Dutilleul S, Ledru Y, Idani A (2018a) Formalizing railway signaling system ERTMS/ETCS using UML/Event-B. In: International conference on model and data engineering, Springer, Cham, pp 321–330
Wakrime AA, Gibson JP, Raffy JL (2018b) Formalising the requirements of an E-voting software product line using Event-B. In: 2018 IEEE 27th international conference on enabling technologies: infrastructure for collaborative enterprises (WETICE), IEEE, New York, pp 78–84
Zafar NA (2016) Formal specification and analysis of take-off procedure using VDM-SL. Complex Adapt Syst Model 4(1):4
Zafar NA, Afzaal H (2017) Formal model of earthquake disaster mitigation and management system. Complex Adapt Syst Model 5(1):10

Publisher's Note