Abdullah et al. LGURJCSIT 2024

ISSN: 2521-0122 (Online)
ISSN: 2519-7991 (Print)

LGU Research Journal of Computer Science & IT

doi: 10.54692/lgurjcsit.2024.081519

Vol (8): Issue (1), January – March 2024

Formal modelling and verification of autonomous reasoning based flight simulation system

Abdullah[1*], Hafiz Mahfooz Ul Haque[2], Nida Hafeez[3]
[1]Department of Computer Science, Bahria University Lahore Campus, Lahore, Pakistan.
[2]Faculty of IT & CS, University of Central Punjab, Lahore, Pakistan.
[3]University of Science and Technology of China, China.

Email: abdullah.bulc@bahria.edu.pk

ABSTRACT:

The emerging trends towards intelligent computing have drastically changed the human lifestyle in every facet of their lives. Due to the swift escalation of smart systems and emerging technologies, human life has become much more dependent and even more addicted to fulfilling their desire using smart and tiny resource-bounded gadgets. This revolution has evolved towards autonomous decision support systems. Autonomous decision support systems can acquire information autonomously, reason the information, and adapt behavior accordingly. As these systems are deployed in a highly decentralized environment and exhibit complex adaptive behavior, however, the inconsistent nature of information may raise different challenging issues. This paper presents a multi-agent environmentally-aware framework for modeling and reasoning flight management systems. This system has a sound reasoning mechanism to execute and monitor flight control activities while considering liveness and safety-critical constraints. We use the UPPAAL model checker to formally analyze the system's behavior and verify its correctness properties.

KEYWORDS: Autonomous Reasoning, Multi-agent, Model Checking, UPPAAL

1. INTRODUCTION

Recent years have witnessed the significant contributions of Unmanned aerial vehicles (UAVs) with great interest in industrial usage and wide utilization carrying small objects in remote areas where the surrounding environment is not well supported for human beings. UAVs have been anticipated as manned aircraft equipped with smart sensors, communication equipment, cameras, and intelligent decision-making capability to take the right action at the right time and in the right place. However, these systems are highly sensitive in nature and may cause the loss of huge investment in case of failure. The research community has introduced formal model checking and verification techniques to test the system well before its actual launch. Model checking has been widely used by the industry and scientific community in different domains such as healthcare, safety-critical systems, and aviation systems, for formal modeling and verification to determine whether the system meets the desired specifications or not. The ultimate purpose is to verify the correctness behaviors in terms of hardware, software, and a joint venture of interactive cross-platform devices and systems. Literature has revealed a significant number of formal modeling techniques deployed in a variety of systems including avionics, healthcare, defense systems, etc. In [1], Fremont et.al. have proposed a rigorous design model of autonomous safety-critical systems. For formal analysis, the VERIFAI toolkit has been used for modeling, falsification, debugging, and overall evaluation process. A case study of aircraft taxiing systems has been modeled and probabilistic reasoning-based systems are deployed to track the runway centerline traffic flow system. Authors in [2], have discussed UAV simulations in multi-agent systems along with their challenging issues including autonomy, security, validation, and verification of the systems. In another work [3], the design flow of a flight control system has been presented using distributed devices connected using a digital communication channel. The system was verified by the B model to specify the logic model of the system and verify its properties. In this work, we propose an autonomous flight control system that can perform different flight control missions and verify the correct behavior of the system. For formal analysis, model checking and formal verification, the case study has been simulated using UPPAAL model checker with the accentuation on properties like safety, liveness, robustness, and deadlock. The proposed flight control system is better than the conventional systems in the sense that we incorporate formal verification properties to ensure safety, liveness, robust properties, and deadlock resistance. To realize this goal, we utilize the power of formal analysis tools, i.e. UPPAAL model checkers, that include model checkers and formal verifications. The rest of the paper is structured as follows: Section II presents the related work considering smart flight management systems. In Section III, we present a formalism for a smart flight management system along with its architectural specifications. In Section IV, we model the case study in UPPAAL model checker and Section V specifies the system behavior and verifies its correctness properties. We finally conclude in Section VI.

2. RELATED WORK

Literature has revealed a significant number of intelligent systems along with autonomous reasoning techniques. Autonomous reasoning-based intelligent assistive systems facilitate systems and users to achieve their goals [4-6]. In [6], authors have presented a formal ATC system model for regulating aircraft flow on two runways using Hierarchical Timed Color Petri Nets, demonstrating feasibility and correctness. While focusing on FCFS scheduling, studies investigate advanced techniques, real-time data integration, and collaborative decision-making to improve efficiency and scalability in different runway operations under varying traffic situations [6]. Skorupski in [7], has proposed the conceptual airport traffic operational model using timed, colored, and stochastic Petri nets with an emphasis on airside capacity, operational delay, and safety issues. The simulation tests shed light on the impact of initial landing formations as well as prospective improvements to the takeoff and landing procedures. The proposed system has been elaborated with the example of the airport with one runway and multiple takeoffs and landing operations. The CPN tool has been used to simulate the experiment taking-off and landing operations simultaneously. Su et al. have proposed a Coloured Petri Net to model and simulate the aircraft's operational activities such as directing aircraft movement on the ground surface and parking along with constraints considering taxiing estimation system. It creates static and dynamic Petri net models, which incorporate taxiing velocity constraints and simulate conflict-free taxiing. They have modeled and simulated the experiment at Toulouse airport to incorporate real-time data for dynamic simulations. The study validates the model by comparing it to real-world flight data, paving the path for improved airport surface operating simulations [8]. In another approach, dynamic modeling of the aircraft handling process has been developed to simulate the operational system of aircraft. The ground-handling process activities are performed and results are generated in the form of graphs along with their dependencies. The operational workflow has been modeled using algorithms and the robustness of the system has been analyzed to observe the intensity [9]. Robert et al. [10], have proposed an approach to monitor the flight performance of different electric vertical take-off and landing aircraft (eVTOLs). This approach is suitable for air taxi operations however, their battery parameters are insufficient due to limited battery mass and high energy densities. The methodology describes power modeling across flight phases, exposing vectored thrust, lift and cruise, and multicopter eVTOL ranges of 115 km, 70 km, and 50 km. While informative, shortcomings include oversimplific-ations in transition stages and a lack of consideration for external influences, raising concerns for real-world application and future battery developments. Another work reported in [11,12], presented a fully simulated air taxi control process that is integrated as per designed models to observe aircraft performance, routing, crew scheduling, and maintenance management. The findings highlighted key performance metrics as well as sensitivity analyses.

3. PROPOSED FORMALISM FOR SMART FLIGHT MANAGEMENT SYSTEM

In this section, we present a state-of-the-art formalism for a smart flight management system. The design flow of the system utilizes automated assistive reasoning processes when modeling the system to formally specify the requirements and verify the correctness flow of the system. The system has a series of actions (steps) to compute the requirement matrix, analyze the required set of computational resources along with traditional resources, evaluate the safety critical systems' properties, delegate tracing and debugging plan, perform verification and validation of specified test plans, and configuration of a complete simulation of the system. The system's simulation initiates with the acquisition of the computational requirements considering resource utilization for the distributed systems' concurrent processing. Then the system analyzes the formal specifications using a formal specification tool to clarify the system's requirements and remove ambiguities, inconsistencies, and incompleteness. As the formal specifications act as base-level design for the software development process, there is a need to use the model-driven specification approach to model the system and the property-oriented state fulfill the desired requirements that are established during the modeling at the previous phase whereas the validation process evaluates the outcomes assessments to ensure compliance with predetermined functional configurations. In the system, we perform statistical model checking [12], which is a simulation-based technique to evaluate the quantitative as well as qualitative properties. Using these properties, the system evaluates whether the desired properties hold, and satisfy in the prescribed format or not. In case of a deadlock situation or conflicting information, the system then triggers error traces for traversing the extensive execution in state transition flow to analyze the behavior in the specified states and a counterexample might need to be executed to traverse back to check whether the system satisfies the desired properties at a specific time interval and this process continues until the desired goal is achieved or the system execution process completes. The formal depiction of the smart flight management system reasoning process can be seen in Figure 1. To formally specify the system, we use Z notation [13], for the example scenario of the use cases of a smart flight management system. Z specification language is based on a set theory and developed by a programming research group at Oxford University comp-

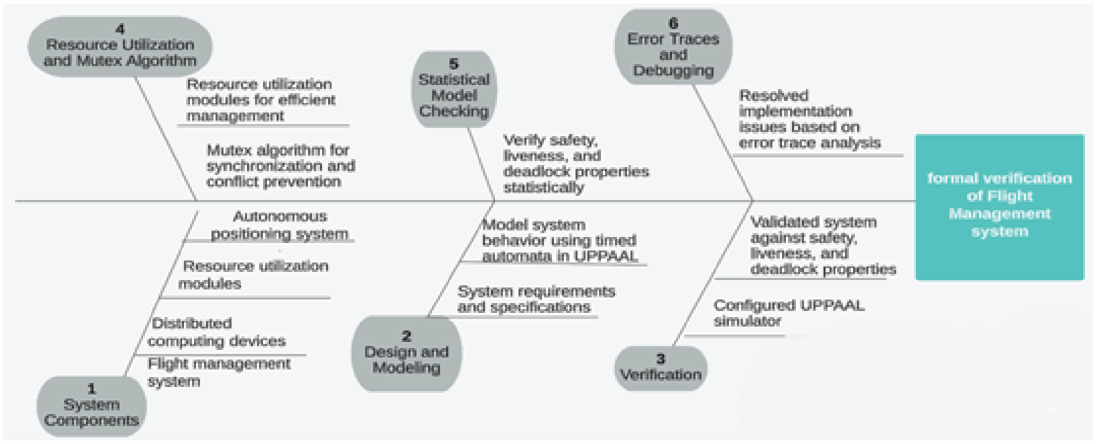
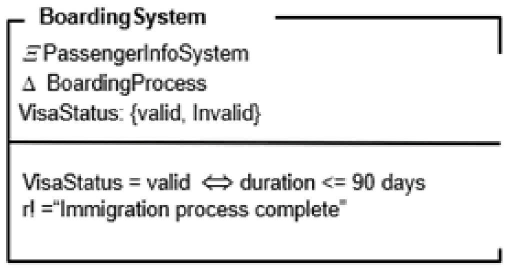

4 Resource Utilization and Mutex Algorithm
Resource utilization modules for efficient management
Mutex algorithm for synchronization and conflict prevention
5 Statistical Model Checking
Verify safety, liveness, and deadlock properties statistically
6 Error Traces and Debugging
Resolved implementation issues based on error trace analysis
formal verification of Flight Management system
Autonomous positioning system
Resource utilization modules
Distributed computing devices
Flight management system
1 System Components
Model system behavior using timed automata in UPPAAL
System requirements and specifications
2 Design and Modeling
Validated system against safety, liveness, and deadlock properties
Configured UPPAAL simulator
3 Verification

Figure 1: Proposed Methodology for Multi-agent Flight Control System

specification specifies the system behavior in terms of properties. Executable specification specifies the system's architectural specification and graphical depictions to formally model temporal reasoning based on concurrent reasoning processes. Concurrent and temporalized specifications explicitly define system behavior in terms of states and the set of events with guards. The verification process determines whether the required set actions triggered at a uting lab in the 1970s. Since then, it has gained international standards for Z notation under the ISO guidelines and is considered as the most widely used formal specification language. It conceptualizes the system in the form of mathematical notation known as schema. Schema structure allows a building block of static and dynamic executions of the use cases and validates the specification in terms of consistency and completeness. Simplistically, system specification are designed using schema which has a set of entities and predicates to express the objects and relationship among objects.



BoardingSystem

Ξ PassengerInfoSystem

Δ BoardingProcess

VisaStatus: {valid, Invalid}

VisaStatus = valid ⇔ duration <= 90 days

rl ="Immigration process complete"

Figure 2: Z Specification of Boarding System

To illustrate the use case of a boarding system, it is represented by a schema provided in Figure 2. The schema named BoardingSystem is one of the important use cases in the flight management system. There are two segments in Figure 2. The first segment shows that the system may acquire the required information from two different modules named as PassengerInfoSystem and BoardingProcess. The operator attached to PassengerInfoSystem is represented by the Greek letter Xi which means PassengerInfoSystem is an interlinked system with BoardingSystem however data extraction and verification can be performed in the BoardingSystem module but changes cannot be applied in the PassengerInfoSystem. Similarly, the operator "Delta" is applied to the BoardingProcess module which means that as the passenger verification and boarding process completes, the system updates in the BoardingProcess module as well. In another way, the BoardingProcess module is dynamic whereas PassengerInfoSystem is static for the boarding system module. This ensures the authentication process as per user access privileges. The architecture of the proposed system is divided into four layers as shown in Figure 3. The first layer is known as Layer 1, which is responsible for managing communication flow to/from data sources such as databases that may either be centralized or distributed in nature. Wireless sensor networks (WSN) and other resources including sensors and other hardware components. WSN actively collects real-time data from various segments of the aircraft and then feeds this data into the database for detailed processing. Layer 2 is responsible for managing connectivity among different processing units and components using Wi-Fi modules and gateways for exchanging information. The controller is a central element of the control layer which precisely executes predefined policies and rules facilitating efficient data processing and decision-making. The flight-control framework incorporates controllers, sensors, and multiple modules. The controller acts as the central hub of the flight control framework and is responsible for handling to/from messages and control directions. In addition, gateways serve as interfaces that enable the control layer to be networked with the infrastructure layer and ensure a harmonious communication exchange. Layer 3 is the operational layer which deals with the simulation of core functionalities such as boarding, departure, and landing management processes. Information derived from layer 1 can be manipulated to improve autonomous decision-making capability within operational activities. In addition, the information extracted from data sources is interpreted specifically for the designated task however in such cases data privacy and data preservation cannot be tolerated at all to avoid inconsistency and incompleteness issues. The fourth layer is called the simulation and management layer, which combines the UPPAAL simulator and verifier. The UPPAAL simulator takes central responsibility for simulating and verifying real-time scenarios of the proposed system and ensures the responsiveness of the system under various conditions.

4. FORMAL MODELLING OF SMART FLIGHT MANAGEMENT SYSTEM

We use UPPAAL [14,15] to formally specify the system and verify the system's correctness properties. UPPAAL has been considered to be the most effective and optimistic model checker because it combines symbolic techniques along with an on-the-fly algorithm to reduce the complexity when modeling the system's behavior. UPPAAL model checker can be used for timed automata based on Computational Tree Logic (CTL). It allows users to model the system, and analyze and simulate the system behavior dynamically in a real-time environment. It is based on timed automata, so it simulates system behaviors in terms of a state transition model and is expressed by a clock value as the system transits to the next state. To illustrate the use of the proposed formalism, a smart flight management case study has been developed. The core aim is to provide smooth flight management operations using autonomous decision support systems. The case study has been modeled and simulated in the

UPPAAL simulator. UPPAAL provides a userfriendly graphical user interface including three different segments namely; editor, simulator, and verifier. The basic declaration and conditional statements are written in the editor, process execution flow in the form of simulation can be designed in the simulator, and properties are defined in the verifier to analyze the behavior of the system's correctness properties. UPPAAL has a client-server architecture where its model-checking engine acquires the desired requirements according to prescribed declarations and models the system's processes to analyze and simulate the behavior in terms of the state transition model. Using query language, the state formula and path formulae determine whether the desired property holds in the specified model or not. For concurrent processing, the system quantifies the paths to ensure the reachability, safety, and liveness properties. We model a smart flight management case study using the UPPAAL model checker. The proposed system has been modelled along with its processes such as the boarding management process along with its local and global declarations in the editor. The process configuration is performed at the design time of the system according to the process execution flows. Each process in the system is initiated using templates. According to the case study of the proposed system, processes consist of six modules that handle different aspects of airport operations. For example, the landing management module handles the process of requesting and managing aircraft landing operations such as communication to/from the control tower. The emergency landing module is responsible for handling emergency landing requests due to different hazardous situations. The departure module is connected to the controller for the takeoff and departure of the planes. In addition, this module is responsible for managing resource utilizations and billing issues as well. Maintenance, security, and flight boarding modules are connected with corresponding modules to perform assigned tasks. Figure 4 shows the landing management module which has 4 possible actions such as landing request, communication with the control tower, checking runway availability, and landing operation. Whenever a flight intends to land, the system generates a request for landing operation to the control tower. If the request is not recognized then the system cancels the request and the request is regenerated. If the air control tower accepts the request but denies an immediate response, then it will wait for the response until the landing request is approved. After getting acceptance from the control tower, then the system checks the runway availability. If the runway is not available, then it will go to turn around time and keep tracking path. If the runway is available then the system allows for landing operation. After a successful landing operation, the control tower updates the system.

The depiction of the departure module has been simulated in Figure 5. As per the system's description, five different operations are deployed in this module such as departure request, air traffic control system, billing clearance and runway availability. Whenever the aircraft is ready for departure, a departure request is generated to the air traffic control system. The request is evaluated based on different aspects such as takeoff clearance, runway availability, weather situation, etc. If the request is approved, then the air traffic control system sends clearance for the departure operation, otherwise, the system is in a wait state until the response is received from the control system. As shown in Figure 6, the Boarding module covers the set of actions such as security checks, antinarcotics, customs, immigration and boarding procedures, etc. The first step is a security check whether the system checks the passenger's details in different steps. All other modules are simulated similarly.

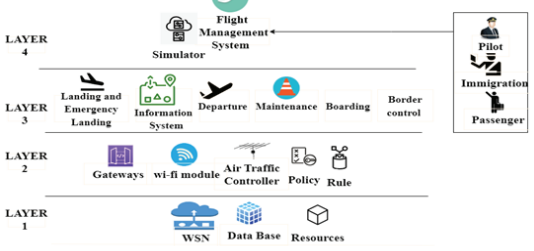


Figure 3: Layered Architecture of the proposed system

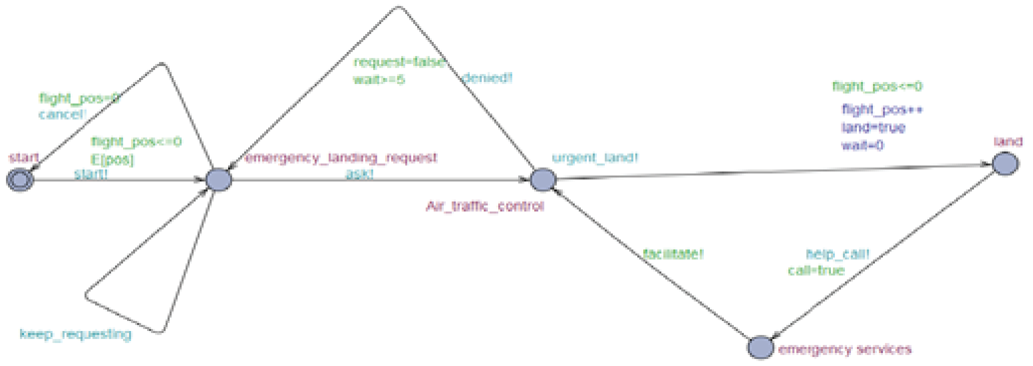


Figure 4: Modelling and simulation of the landing module

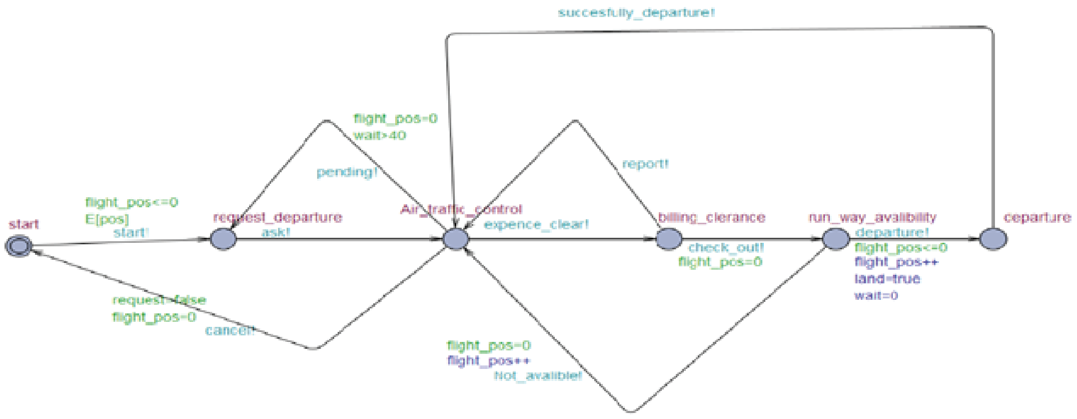


Figure 5. Modelling and simulation of Departure Module

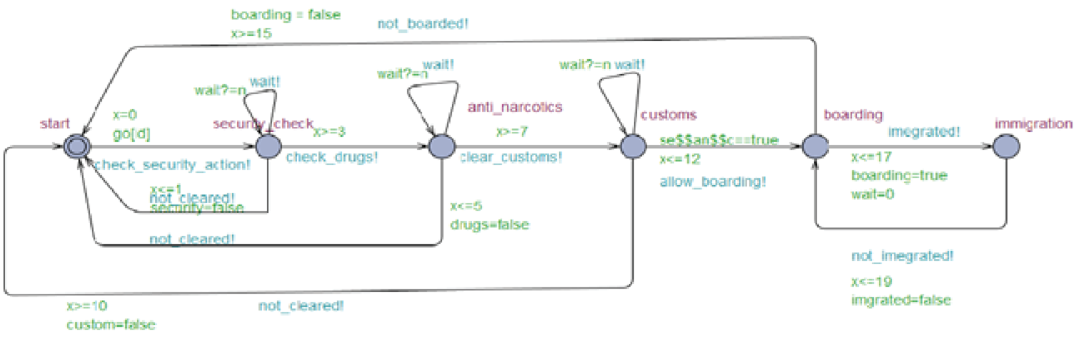


Figure 6. Modelling and simulation of boarding module

5. FORMAL VERIFICATION OF PROPOSED SYSTEM USING UPPAAL MODEL CHECKER

We use the UPPAAL verifier to formally verify the correctness properties of the system. Formal verification properties are derived based on the formal specification design of the proposed case study. When modeling the system, verification properties are specified to check whether the desired properties are satisfied or not. We check safety properties, liveness properties, reachability states, guarantee of service, mutual exclusion, and robuſtness properties. Safety property ensures that inconsiſtent execution muſt not happen in the sense that the process of flight management muſt be executed as per designed guidelines. To specifically check the reachability, we write property "A[]p" which means a formula p remains true in all reachable ſtates. In the case of a few reachable ſtates, we use an exiſtential quantifier, we apply a formula E<>p which ſtates that a formula p remains true in some ſtates. Upon successful execution of the UPPAAL verifier, the syſtem displays the message "property is satisfied". The syſtem's correctness properties along with their response time are illuſtrated in Table 1 and their depiction is shown in Figure 7.

E<> ((A pos! =0 && C! =0 && A(parked)= True) or ((A pos=0 && C=0 A && A(parked)= false) and (A pos! =0 && C! = 0&& E=true && A(departure)=True) or (A pos! =0 && C! = 0&& E=false && A(departure)=false) or (G=true && A1(maintained=True) and (G! =true && A1(maintained=false) or ((p&&Q) -->R =true && P(R)=True) and ((p&&Q) -->R! =true && P(R)=false) or (i&&j&&k&&L=true && H (M)=True) and (i&&j&&k&&L! =true && H (M)=false))

Verification/kernel/elapsed time used: 0s/0s/0.003s. Resident/virtual memory usage peaks:7204KB/26404KB. Property is satisfied.

A<> ((A pos! =0 && C! =0 && A(parked)= True) or ((A pos=0 && C=0 A && A(parked)= false)

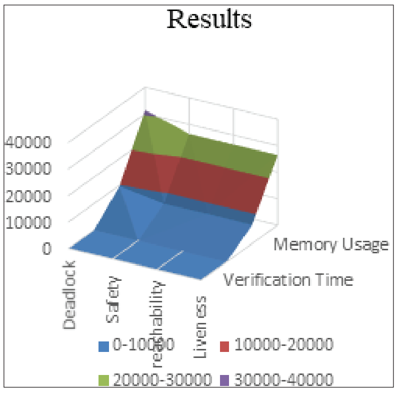


Figure 7: Formal Verification Property **Satisfaction Graph**

and (A pos! =0 && C! = 0&& E=true && A(departure)=True) or (A pos! =0 && C!= 0&& E=false && A(departure)=false) or (G=true && A1(maintained=True) And (G! =true &&A1(maintained=false) or ((p&&Q) -->R =true && P(R)=True) And ((p&&Q) -->R! =true && P(R)=false) or (i&&j&&k&&L=true && H (M)=True) And (i&&j&&k&&L! =true && H (M)=false))
Verification/kernel/elapsed time used: 0s/0s/0.006s
Resident/virtual memory usage peaks: 7001KB/26472KB.Property is satisfied.
A [] not deadlock
Verification/kernel/elapsed time used: 1.557s/0.060s/1.786s. Resident/virtual memory usage peaks: 9,680KB/31,472KB. Property is satisfied.

Table 1: Verifying System's correctness properties

Property	**Tool and Version**	**Verification Time**	**Kernel Time**	**Memory Usage**	**Peak virtual memory**	**Satisfied**
Deadlock	UPPAAL version 4.1.19	1.786 seconds	0.060 seconds	9,680KB	31,472KB	yes
Safety	UPPAAL version 4.1.19	0.006 seconds	0 seconds	7001KBs	26472KB	yes
Reachability	UPPAAL version 4.1.19	0.003 Seconds	0 seconds	7204KB	26404KB	yes
Liveness	UPPAAL version 4.1.19	0.006 seconds	0 seconds	7001KB	26472KB	yes

In case of few reachable ſtates, we use exiſtential quantifier, we apply a formula E<>p which ſtates that a formula p remains true in some ſtates. Table 1 explains The table provided outlines a Flight

Management System (FMS) with modules designed for different functions within airport operations. These modules are integrated into a resource planning framework, utilizing a mutex algorithm to optimize resource allocation and ensure efficient coordination among various tasks. Table 1 overarching framework for resource allocation and coordination across all modules. It ensures that resources are efficiently distributed among the different modules based on their needs and priorities.

6. CONCLUSION & FUTURE WORK

In this paper, we proposed a multi-agent reasoning-based smart flight management system that provides real-time dynamic assistance for daily routine activities. The core aim is to develop an autonomous decision support system while considering safety and liveness constraints. We modeled the case study in the UPPAAL model checker, simulated the system behavior, and verified correctness properties. In future work, we propose a state-of-the-art smart flight management system using ambient intelligence that can provide a predictive reasoning-based autonomous decision support mechanism.

REFERENCES

[1] Fremont et al., “Formal analysis and redesign of a neural network-based aircraft taxiing system with VerifAI,” In *Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part I 32,* pp. 122-134, Springer International Publishing, 2020.

[2] Y. Mualla et al., “Agent-based simulation of unmanned aerial vehicles in civilian applications: A systematic literature review and research directions,” *Future Generation Computer Systems, 100,* pp. 344-364.

[3] N. A. El-Araby et al., “Implementation of formally verified real time distributed systems: Simplified flight control system,” in T*he 2011 International Conference on Computer Engineering & Systems*, pp. 25-32, 2011.

[4] G. Czibula et al., “IPA-An intelligent personal assistant agent for task perfor- mance support," *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing. IEEE*, pp. 31-34, 2009.

[5] T. Y. Leong and C. Cao, “Modelling medical decisions in dynamol: A new general framework of dynamic decision analysis,” *MEDINFO'98,* IOS Press, pp. 483-487, 1998.

[6] M. I. Fakhir et al., “Formal Modeling and Analysis of Air Traffic Control System Using Petri Nets,” *VAWKUM Transactions on Computer Sciences*, 11(2), pp. 35–48, 2023.

[7] J. Skorupski, “Airport traffic simulation using petri nets,” In *Activities of Transport Telematics: 13th Inte rnational Conference on Transport Systems Telematics, TST 2013, Katowice-Ustroń, Poland, October 23–26, 2013, Selected Papers 13,* pp. 468-475, Springer Berlin Heidelberg, 2013.

[8] Z. Su and M. Qiu, “Airport surface modelling and simulation based on timed coloured petri net,” *Promet-Traffic&Transportation,* 31(5), pp. 479-490, 2019.

[9] A. Kierzkowski and T. Kisiel, “A simulation model of aircraft ground handling: case study of the Wroclaw airport terminal,” In *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology–ISAT 2016–Part III,* pp. 109-125, Springer International Publishing, 2017.

[10] R. Brühl et al., “Air taxi flight performance modeling and application,” In P*roceedings of the USA/Europe ATM R&D Seminar, Online,* (September, 2021).

[11] P. A. Bonnefoy, “Simulating air taxi networks,” In *Proceedings of the Winter Simulation Conference,* pp. 10-pp, IEEE, (December, 2005).

[12] A. Legay et al., “Statistical Model Checking.,” In *Steffen, B., Woeginger, G. (eds) Computing and Software Science Lecture Notes in Computer Science(),* vol 10000, Springer, Cham, 2019.

[13] J. M. Spivey, *Understanding Z: a specification language and its formal semantics,* Vol. 3, Cambridge University Press, (1988).

[14] K. G. Larsen et al., “UPPAAL in a nutshell,” *International journal on software tools*

for technology transfer, 1, pp. 134-152, 1997.

[15] J. Vain et al., :On the benefits of using aspect-orientation in UPPAAL timed automata," In *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions)(ICTUS),* pp. 84-91, IEEE, (December, 2017).