Aircraft maintenance information system design and verification

Yinling Liu

N°d'ordre NNT : 2019LYSEI133

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
(Nom Etablissement)

Ecole Doctorale N° ED 512
(Informatique et Mathématiques)

Spécialité / discipline de doctorat :
Génie industriel

Soutenue publiquement le 13/12/2020, par :
(Yinling LIU)

Conception et Vérification du Système d'Information pour la Maintenance Aéronautique

---

Devant le jury composé de :

GZARA, Lilia	Maître de conférences	Grenoble INP	Rapporteure
ARCHIMEDE, Bernard	Professeur des universités	ENI Tarbes	Rapporteur
PANETTO, Hervé	Professeur des universités	Université Lorraine	Examinateur
TANG, Dan	Professeur des universités	Université de Chengdu des Technologie des l'Information	Examinateur
CHEUTET, Vincent	Maître de conférences	INSA Lyon	Directeur de thèse
WANG, Tao	Maître de conférences	Université Jean Monnet	Co-directeur de thèse
ZHANG, Haiqing	Maître de conférences	Université de Chengdu des Technologie des l'Information	Co-encadrante de thèse

Abstract

Operational support is one of the most important aspects for aircraft maintenance, which concentrates on delivering a portfolio of services to have maintenance implemented with high levels of efficiency, reliability and affordability. Operational support of civil aircraft has the characteristics of multi-point, wide range, a miscellaneous set of things, long period, and large investment. One of the major difficulties of operational support is that there is no platform integrating all maintenance-related processes of aircraft maintenance together in order to reduce maintenance costs and improve service level with the continuing airworthiness.

It is therefore necessary to build an autonomous system for aircraft maintenance where all of the maintenance information can be collected, organised, analysed and managed to support decision-making. To do so, an innovative methodology has been proposed, which involves modelling, simulation, formal verification, performance analysis with regard to the autonomous system mentioned. Three issues have been addressed in this doctoral thesis.

Issue one: The design and simulation of an autonomous system for aircraft maintenance

Operational support is a key issue for aircraft maintenance, which aims to improve operational efficiency and reduce operating costs under the premise of ensuring flight safety. Although many works have emerged to achieve this aim, they mostly address the concept of maintenance systems, the relationship between stakeholders and the loop of maintenance information separately. Hence, the cooperation between stakeholders could be impeded especially when urgent decisions should be made, relying on historical data and real-time data. In Chapters 3 and 4, we propose an innovative design of an autonomous system supporting the automatic decision-making for maintenance scheduling. The design starts from concept formulation of the system, to information transitional level interface, and ends with an instance of system module interactions. The underlying architecture illustrates the high-level fusion of technical and business drives; optimizes strategies and plans with regard to maintenance costs, service level and reliability. An agent-based simulation system is developed as a proof to illustrate the feasibility of the system principle and algorithms. Besides, the simulation experiment analyzing the impact of maintenance sequence strategies on

maintenance costs and service level has demonstrated the algorithm functionality and the feasibility of the proposed approach.

Issue two: Model checking on simulation systems

Model checking is an effective way to verify the behaviours of an agent-based simulation system. Three behaviours are analysed: operational, control, and global behaviours. Global behaviours of a system emerge from operational behaviours of local components regulated by control behaviours of the system. Previous work principally focuses on verifying the system from the operational point of view (operational behaviour). The satisfactory of the global behaviour of the system has not been investigated regarding control behaviours. Thus, in Chapter 5, we propose a more complete approach to verifying both global behaviours and operational behaviours of systems. To do so, firstly these three behaviours are formalized by automata-based techniques. The meta-transformation between automata theories and Kripke structure is then provided, in order to illustrate the feasibility of the model transformation between the agent-based simulation model and the Kripke structure-based model. In the following, a mapping between the models is proposed. Subsequently, the global behaviour of the system is verified by the properties extracted from the control behaviour and the operational behaviour is checked by general system performance properties (e.g. safety, deadlock freedom). Finally, a case study on the simulation system for aircraft maintenance has been carried out. A counter-example of signals sending between Flight agent and Plane agent has been proposed by NuSMV model checker. Modifications for the NuSMV model and the agent-based simulation model have been performed. The experiment results show that 47.37% of cancelled flights have been changed to be serviceable.

Issue three: The performance analysis of simulation systems

The performance of the simulation system proposed is analysed via investigating its capability of fault classification and prediction on aircraft maintenance. Effective Fault Classification and Prediction (FCP) for aircraft maintenance helps improve the efficiency of maintenance. The traditional ways of the FCP mainly rely on the analysis of mechanical characteristics of components and the reliability-related data. Obtaining the fault-related data is usually prior to performing the analysis of fault diagnoses. However, acquiring the data especially the sensitive data in terms of security and confidentiality is extremely difficult. Therefore, in Chapter 6, we propose an approach of combining Agent-Based Simulation System (ABSS) with the Fuzzy-Rough Nearest Neighbour (FRNN) approach, in order to implement the FCP for aircraft maintenance without the sufficient data. Towards this end, a framework for integrating the FRNN approach into ABSS is proposed. The concept and architecture models of FRNN-ABSS are used to describe the FRNN-ABSS system. In the following, random and sequence strategies are designed for the FCP of the engine and an algorithm is developed to accomplish the integration of the FRNN approach and ABSS, aiming at

automatically performing fault diagnoses. Finally, the experiments analysing the impact of different strategies on maintenance costs and service level have been conducted. The results show that the approach proposed achieved success for random and sequence strategies: 9.3% and 2.5% of maintenance costs have been saved; 4.17% and 12.5% of delayed flights have been changed into on-time flights.

Résumé

Le soutien opérationnel est l'un des aspects les plus importants pour la maintenance aéronautique. Il vise essentiellement à fournir un portefeuille de services permettant de mettre en œuvre la maintenance avec un niveau élevé d'efficacité, de fiabilité et d'accessibilité. Le soutien opérationnel pour les avions civils se caractérise par, par exemple: de nombreux points de vue avec un spectre large, une grande quantité d'objets différents, une longue période et un besoin important en investissements. L'une des principales difficultés du support opérationnel est qu'il n'existe pas de plate-forme intégrant tous les processus de maintenance des avions afin de réduire les coûts et d'améliorer le niveau de service avec le maintien de la navigabilité.

Il est donc nécessaire de mettre en place un système autonome de maintenance des avions dans lequel toutes les informations de maintenance peuvent être collectées, organisées, analysées et gérées de manière à faciliter la prise de décision. Pour ce faire, une méthodologie innovante a été proposée, qui concerne la modélisation, simulation, vérification formelle et analyse des performances du système autonome mentionné. Trois axes ont été abordés dans cette thèse.

Premier axe: conception et simulation d'un système autonome pour la maintenance aéronautique

Le soutien opérationnel est un élément clé de la maintenance des avions, qui vise à améliorer l'efficacité opérationnelle et à réduire les coûts d'exploitation tout en garantissant la sécurité des vols. Bien que de nombreux travaux aient été réalisés pour atteindre cet objectif, ils abordent majoritairement les concepts de systèmes de maintenance, les relations entre les parties prenantes et les boucles des informations de maintenance séparément. Par conséquent, la coopération entre les parties prenantes pourrait être entravée surtout lorsque des décisions urgentes doivent être prises, en s'appuyant sur des données historiques et des données en temps réel. Aux Chapitres 3 et 4, nous proposons une conception innovante d'un système autonome prenant en charge la prise de décision automatique pour la planification de la maintenance. Cette conception commence par la formulation du concept de système, l'interfaçage des niveaux de transition des informations, et se termine par une instanciation des interactions des modules du système. L'architecture sous-jacente illustre la fusion à haut niveau de motivations techniques et commerciales; optimise les stratégies

et les plans en matière de coût de maintenance, de niveau de service et de fiabilité. Un Système de Simulation à Base d'Agents (SSBA) est développé comme preuve pour illustrer la faisabilité du principe du système et des algorithmes. En outre, l'expérience de simulation analysant l'impact des stratégies de séquence de maintenance sur les coûts de maintenance et le niveau de service a démontré la fonctionnalité de l'algorithme et la faisabilité de l'approche proposée.

Deuxième axe: vérification de modèles sur des systèmes de simulation

La vérification de modèle est un moyen efficace de vérifier les comportements d'un SSBA. Trois comportements sont analysés: les comportements opérationnels, de contrôle et globaux. Les comportements globaux d'un système émergent des comportements opérationnels de composants locaux régulés par des comportements de contrôle du système. Les travaux précédents portent principalement sur la vérification du système du point de vue opérationnel (comportement opérationnel). Le comportement global satisfaisant du système n'a pas été étudié en ce qui concerne le comportement du contrôle. Ainsi, au Chapitre 5, nous proposons une approche plus complète de la vérification des comportements globaux et des comportements opérationnels des systèmes. Pour ce faire, ces trois comportements sont tout d'abord formalisés par des techniques basées sur des automates. La méta-transformation entre les théories des automates et la structure de Kripke est ensuite fournie, afin d'illustrer la faisabilité de la transformation de modèle entre le modèle de simulation à base d'agent et le modèle basé sur la structure Kripke. Dans ce qui suit, une correspondance entre les modèles est proposée. Ensuite, le comportement global du système est vérifié par les propriétés extraites du comportement de contrôle et le comportement opérationnel par les propriétés de performances générales du système (par exemple, sécurité, liberté de blocage). Enfin, une étude de cas sur le système de simulation pour la maintenance des avions a été réalisée.

Suite à de premiers tests, des contre-exemples fournis par le vérificateur de modèle NuSMV ont été utilisés pour améliorer les performances du modèle SSBA, notamment au niveau de l'envoi de signaux entre l'agent de vol et l'agent Plane. Les résultats de nouvelles simulations suite à ces modifications montrent une réduction de 47,37% des vols annulés par rapport au modèle précédent.

Troisième axe: l'analyse de la performance des systèmes de simulation

Les performances du système de simulation proposé sont analysées en examinant sa capacité de classification et de prévision des défaillances lors de la maintenance des avions.

La classification et prévision efficaces des pannes (CPP) pour la maintenance des avions contribue à améliorer l'efficacité de la maintenance. Les méthodes traditionnelles de CPP reposent principalement sur l'analyse des caractéristiques mécaniques des composants et des données relatives à la fiabilité. L'obtention des données relatives aux pannes est généralement préalable à l'analyse des diagnostics de pannes. Cependant, acquérir les données, en particulier les données sensibles en termes de sécurité et de confidentialité est extrêmement difficile. Par conséquent,

au Chapitre 6, nous proposons une approche consistant à combiner un SSBA avec une approche «Fuzzy Rough Nearest Neighbor »(FRNN), afin de mettre en œuvre le CPP pour la maintenance des avions avec des données manquantes. Pour ce faire, un cadre pour l'intégration de l'approche FRNN dans SSBA est tout d'abord proposé. Des modèles de concept et d'architecture de FRNN-SSBA sont utilisés pour décrire le système FRNN-SSBA. Dans ce qui suit, des stratégies aléatoires et séquentielles sont conçues pour le CPP du moteur et un algorithme est développé pour réaliser l'intégration de l'approche FRNN et du SSBA, visant à réaliser automatiquement des diagnostics de défaillance. Enfin, des expériences analysant l'impact de différentes stratégies sur les coûts de maintenance et le niveau de service ont été menées. Les résultats montrent que l'approche proposée a porté ses fruits pour les stratégies aléatoires et séquentielles: respectivement 9,3% et 2,5% des coûts de maintenance ont été économisés, et 4,17% et 12,5% des vols retardés ont été transformés en vols à l'heure.

List of Publications

Peer reviewed journals

Y. L. Liu, T. Wang, H. Q. Zhang, V. Cheutet. **An improved approach on the model checking for agent-based simulation system**, Software and Systems Modeling. (**Minor revision**)

Y. L. Liu, T. Wang, H. Q. Zhang, V. Cheutet, G. Shen. **The design and simulation of an autonomous system for aircraft maintenance scheduling**, Computers & Industrial Engineering. https://doi.org/10.1016/ j.cie.2019.10 6041

Y. L. Liu, T. Wang, H. Q. Zhang, V. Cheutet. **Simulation based fuzzy-rough nearest neighbor fault classification and prediction for aircraft maintenance**, Journal of Simulation. https://doi.org/10.1080/17477778.2019.1680261

Peer reviewed conferences

Y. L. Liu, T. Wang, H. Q. Zhang, V. Cheutet (2018, July). **Information Systems Simulation for Performance Evaluation-Application in Aircraft Maintenance.** In IFIP International Conference on Product Lifecycle Management (pp. 789-799). Springer, Cham.

Y. L. Liu, T. Wang, H. Q. Zhang, V. Cheutet. **Towards Standards Analysis and Application in Process of Aircraft MRO**, IFAC World Congress, Toulouse, France, July 10-14, 2017.

Remerciement

Le temps passe trop vite! Mes trois ans de recherche ont été comme un rêve. Les trois ans m'ont changé beaucoup. Je suis passé de faire rien à tout faire; de rarement voyager à voyager partout; de ne pas parler français à présenter mon sujet en français; de ne pas savoir comme écrire un article à écrire trois (deux sont publiés ☺☺); de rester silencieux à aimer défendre mes droits légaux; de ne pas connaître la France à connaître son « romantique ». Tous les changements n'ont pas pu se faire sans les aides de personnes au tour de moi.

Tout d'abord, je veux remercier Professur Vincent CHEUTET qui est mon directeur de thèse. Ici, je ne parle pas des efforts qu'il a fait pour superviser mon sujet. Franchement, j'ai apprécié beaucoup sa façon de travailler avec doctorants. Si je me permets d'utiliser un mot pour la décrire, je vais choisir « libre ». Il ne m'a jamais poussé pour faire quelques choses. Une fois j'ai des idées, il m'a toujours soutenu. Il a aussi donné ses idées. Il n'y avait donc pas de raison pour laquelle je ne pouvais pas bien avancer ma recherche. En effet, il m'a poussé à apprendre le français. Sinon je ne peux qu'écrire anglais ou chinois maintenant 😎. Je veux aussi remercier Professeur Tao WANG qui est mon co-encadrant. Il avait beaucoup de patiences avec moi. Parfois, je n'ai pas compris quelques choses. Il m'a bien expliqué en chinois. Il se peut que chinois est plus compréhensible pour moi. Non, je rigole 😉. En fait, c'était des exemples vivants dans les expériences qu'il avait eu qui m'a fait compris toute de suite. Je veux également remercier Professeur Haiqing ZHANG qui est ma co-encadrante. Elle était plus jeune mais plus stricte que les autres. Aujourd'hui, j'ai finalement compris qe'elle souhaitais que je sois le meilleur doctorant.

Ensuite, mes remerciements vont aller aux mes collègues qui sont Corentin Le Hesran, Minh Phuoc Doan, Zied el Hajri, Alnour RIBAULT, EMINE Sid'Ahmed, Anderson Fabian Mosquera Varela, Oscar Augusto Tellez Sanchez, Touzout Faycal, Pealat Clément, Martin Alice. Vous êtiez toujours gentils avec moi. Vos présences m'ont rendu la période heureuse.

Après, je veux remercier ma famille pour tout ce qu'ils m'ont fait. En particulier, je devrais remercier ma femme Qian YANG. Elle m'a suivi et soutenu tout le temps. Elle s'est occupé de tout pour que je puisse bien concentrer à la recherche.

A la fin, je veux remercier mon pays Chine qui m'a financié pendent mes études.

Contents

List of Tables

List of Figures

Nomenclature

ABMS	Agent-Based Modelling and Simulation
ABSS	Agent-Based Simulation System
AD	Airworthiness Directive
AI	Artificial Intelligent
AMOs	Aircraft Maintenance Organizations
AR	Augmented Reality
ATA	Air Transport Association of American
CAA	Civil Aeronautics Authority
CM	Corrective Maintenance
CRA	Customer Requirement Agent
CTL	Computational Tree Logic
DMC	Direct Maintenance Cost
FAA	Federal Aviation Administration
FIFO	First In First Out
FSM	Finite State Machine
GIS	Geographic Information System
ILS	Integrated Logistic Support
IMC	Indirect Maintenance Cost
IS	International Specification

LSA	Logistic Support Analysis
LTL	Linear Temporal Logic
MRO	Maintenance, Repair and Overhaul
MSG	Maintenance Steering Group
MTBF	Mean Time Between Failure
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
OEM	Original Equipment Manufacturing
PM	Preventive Maintenance
PSS	Product-Service System
QCA	Quality Control Agent
RST	Rough Set Theory
SB	Service Bulletin
SCA	Supply Chain Agent
SE	Systems Engineering
SSA	Service Strategy Agent
STA	Service Task Agent
TSB	Transport Safety Board of Canada
UCD	Use Case Diagram
UML	Unified Modeling Language

Chapter 1

Introduction

This chapter intends to provide a general overview of our research project. The indispensable elements for explaining this project will be introduced. First of all, the background on operational support for aircraft maintenance will be discussed. A detailed description of the problem will then be presented. The scope of the research will also be defined. In the following, research objectives will be illustrated, in order to achieve the research goal. In the end, major contributions derived from research objectives will be explored.

1.1 Background

1.1.1 Aircraft Maintenance Perspective

The term “aircraft maintenance" seems to be very large and abstract. It is because this term presumably reminds us of a broad set of activities performed by technicians and the extremely complex structure of aircraft. Since the aircraft is considered to be strictly confidential, less of its definition is provided in the academic domain. Here, we take the definition of this term from Wikipedia.

Aircraft maintenance is the performance of tasks required to ensure the continuing airworthiness of an aircraft or aircraft part, including overhaul, inspection, replacement, defect rectification, and the embodiment of modifications, compliance with airworthiness directives and repair.

From this definition, we observe that the principal aim of aircraft maintenance is to restore or maintain faulty parts of an aircraft so that the aircraft will remain at the same airworthy level. As for stakeholders, keeping aircraft serviceable and reliable can directly increase profits. Establishing an airworthy maintenance or inspection program also makes it possible to let aviation authorities issue an airworthiness certificate. Maintenance generally consists of PM (Preventive Maintenance) and CM (Corrective Maintenance). CM implies maintenance activities are unscheduled, normally caused by the occurrence of faults on the component. PM was originally provided by aircraft manufacturers. As the improvement on the performance of parts and testing techniques, PM is primarily designed by a synthesis analysis of the maintenance period proposed by manufacturers and the real-time states of parts. Maintenance has different levels, including line maintenance, light maintenance and heavy maintenance (TAT Group 2010), which can be illustrated as follows:

- **Line maintenance** includes pre-flight inspections, transit checks, daily checks (visual inspection, fluid levels, general security and cleanliness of the flight deck, emergency equipment), weekly checks, and on-call assistance.
- **Light maintenance** refers to A-checks (check certain service items requiring special tools, once every 65 flights hours), cabin maintenance, and engine changes.
- **Heavy maintenance** involves activities such as C-checks (the whole aircraft is inspected, takes 1-2 weeks, once a year), D-checks (most demanding and comprehensive check, takes 3 weeks - 2 months, once every four years), stripping and painting, and cabin refurbishment.

The aircraft is one kind of safety-critical systems whose parts are strictly qualified. But it cannot avoid inherently degrading. Even though interventions like PM or CM are effective to reduce the failures of aircraft components, they are costly and cause lateness, damage and hazards. For example, pursuant to the TSB (Transport Safety Board of Canada) mandatory incident reporting requirements (TSBC 2019), 833 incidents were reported in 2016, with the top contributors: declared emergency (37%), risk of collision or loss of separation (17%), and engine failure (13%).

Maintaining complex systems often exceeds the expected cost (Randall, Pohlen, and Hanna 2010). Therefore, more efficient maintenance for aircraft is highly demanded.

1.1.2 Operational Support for Aircraft Maintenance

Maintaining aircraft is considered as a potentially fast-growing market. This market owned 135.1 billion dollars in 2015, which is projected to grow by 4.1% per year until 2025. The number of servicing aircraft is expected to be around 28,000 with 4% annual growth rate (Michaels 2010).

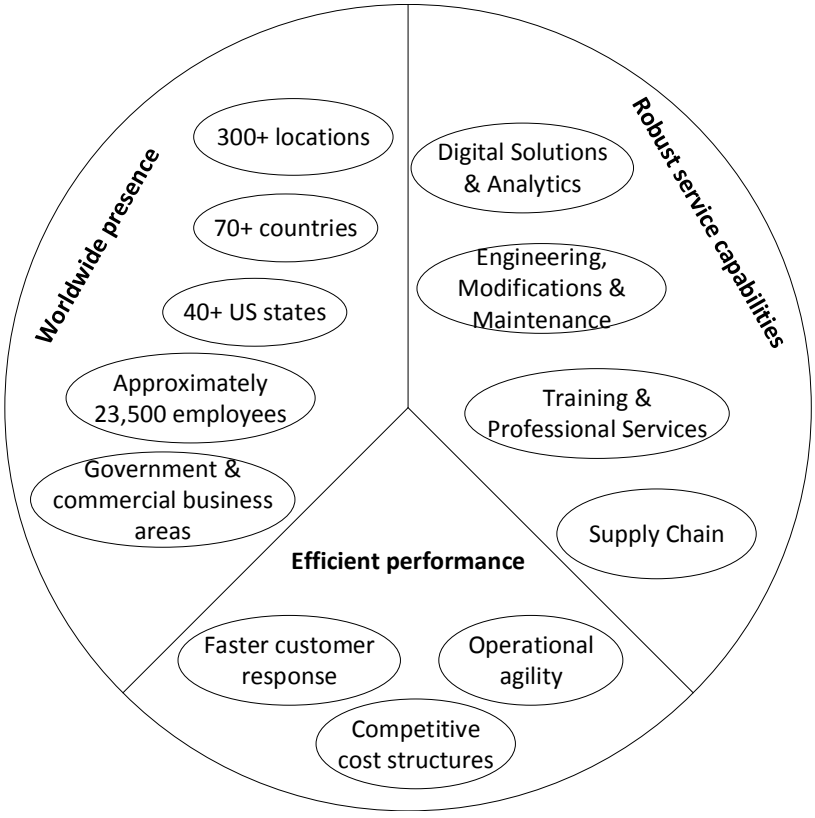

Figure 1.1: The operational support from Boeing, adapted from Boeing (2019)

Operational support is one of the most important aspects for aircraft maintenance, which concentrates on delivering a portfolio of services to have maintenance implemented with high levels of efficiency, reliability and affordability. Operational support of civil aircraft has the characteristics of multi-point, wide range, a miscellaneous set of things, long period, and large investment. For example, the operational support from Boeing is shown in Fig.1.1. The business processes are very complex and the data is collected from multi-aspects in the life cycle of civil aircraft. Different information systems on aircraft maintenance should be integrated to support robust services. A well structured collaborative work environment should be provided, in order to achieve the high

efficiency of aircraft maintenance. Since the system Boeing uses is a purely commercial system, little information for the details about maintenance systems can be accessed; on the other hand, it can prove that our research is highly concentrated in industry.

1.1.3 Complexity in Operational Support

Maintenance businesses for aircraft involve flight scheduling, maintenance strategies and planning, repair, part supply and a number of stakeholders including: OEM (Original Equipment Manufacturing) , suppliers, airlines, the MRO (Maintenance, Repair and Overhaul) , and airworthiness authority, etc. Due to the nature of operations, the collaboration between stakeholders is ephemeral, meaning that the network of actors is unique and their collaboration takes places at the very specific and limited time of the product life-cycle. The collaboration is also distributed, i.e. is rhymed by a set of inter-related decisions that are distributed in the partner network. Such collaboration is synchronised as well because some of the events happen, conforming to a strict sequence. For example, suppliers are independently solicited for relevant parts when the parts are out of stock. When necessary parts and auxiliary equipment for repairing some faulty components are ready, MRO operations will be scheduled. Maintenance managers will define strategies as soon as the maintenance is requested. The major difference between general plant and machinery maintenance, and aircraft maintenance is that aircraft maintenance is mandated and monitored by regulatory authorities like FAA (Federal Aviation Administration), CAA (Civil Aeronautics Authority), etc. Thus, they are the individuality of stakeholders, the complex maintenance regulations, and distributed, synchronized and ephemeral cooperation between stakeholders that clearly make the system on operational support for aircraft maintenance complex (Liu et al. 2019b).

Since the number of commercial aircraft rapidly grows, maintenance costs are requested to be decreased, and service level (the rate of on-time flights) is expected to be further ameliorated, the complexity in maintenance strategy and maintenance task scheduling will probably escalate; and inevitably, a more refined, autonomous and global decision-making capability will be demanded. Recently, despite some advanced technologies like AR (Augmented Reality), Robot or AI (Artificial Intelligent), aircraft maintenance still largely depends on the economical time-based preventive maintenance practice and manual analysis. The deterioration of components and unplanned maintenance activities make maintenance further complex. PM is programmed to prevent failures aircraft from taking place, in order to minimize air transport service disruption. Less effective corrective maintenance is easier to cause “chain reaction" in the network of air transport service. Clearly, aircraft maintenance is becoming a data-rich maintenance issue with a considerable number of data streams and operating databases (Dinis, Barbosa-Póvoa, and Teixeira 2019). Additionally, delayed flights, delayed incidents, broken components, costs, requested parts and safety recommendations, etc., are generated every day. One of the important things for maintenance is to capture relevant information for analysis (Viles et al. 2007). Therefore, **the principal**

challenge is about how to provide efficient operational support for aircraft maintenance under this complex environment, in order to reduce maintenance costs and improve service level with the continuing airworthiness.

1.2 Problem Description

An innovative approach is highly demanded to build an autonomous system of aircraft maintenance where maintenance information is well managed to support automatic decision-makings, in order to effectively decrease maintenance costs and improve maintenance efficiency. Many works have emerged in this area. They mostly address the concept of maintenance systems (Duffuaa et al. 2001; Durazo-Cardenas et al. 2018), the relationship between stakeholders (Ward et al. 2010) and the loop of maintenance information (Zhang et al. 2015; MacKenzie, Miller, and Hill 2012). However, to the best of our knowledge, no one combines these three aspects together, in order to build a new autonomous maintenance system. This may give rise to some deficiencies in the system. Developing a system without its architecture design makes it difficult to truly understand its essence and key properties, which in turn affects concerns such as the feasibility, utility, completeness, and maintainability of the system (ISO 2011). Designing an architecture model of systems without an executable system (or simulation system) makes it hard to demonstrate the feasibility of the architecture design. The problem is categorised into three issues illustrating as follows:

- **The first issue addresses how to build such an autonomous system starting from the requirement analysis, to the architecture design, to the analysis of system modules involving stakeholders, and ending with a concrete simulation system**.
- **The second issue focuses on verifying the satisfactory of behaviours of the simulation system in terms of requirements, in order to improve the trustability of the results from the system**.
- **The last issue emphasizes how to further improve the efficiency of maintenance with the AI**.

This system is then capable of more efficiently transforming the historic and real-time data into global decisions for maintenance scheduling, which finally effectively decreases maintenance costs and improves maintenance efficiency.

1.3 Research Scope

Maintenance is generally studied in the context of production, maintenance of products or the PSS (Product-Service System) (Alrabghi and Tiwari 2015). As for the PSS, products and corresponding

services are developed simultaneously. Since the current research does not involve the development of products, we study the maintenance in the context of maintenance of products.

The maintenance processes for aircraft in this research include flight scheduling, maintenance strategies defining, maintenance planning and scheduling, costs and service estimation, and safety and reliability analyzing.

1.3.1 Flight Schedule

Scheduling flights is the first step to make aircraft serviceable. This step significantly influences other maintenance processes. More flights scheduled may cause maintenance demands increased, which subsequently gives rise to the problem of resource assignment. However, proposing an approach enabling a more efficient flight scheduling is beyond the scope of this research. Thus, we will not study the prevailing approaches for scheduling like heuristic algorithms and mixed-integer linear programming. We consider this step by providing flight schedules close to reality.

1.3.2 Maintenance Strategy

Maintenance strategies are basically divided into scheduled and unscheduled maintenance. The maintenance for aircraft necessitates these two strategies. Scheduled maintenance is predefined while unscheduled maintenance happens when faults on components occur. In addition, the impact on different strategies for the same component is considered. For example, the timer of the scheduled maintenance for component A will be reset as soon as the unscheduled maintenance on this component is just finished. While we do not take further researches on maintenance strategies into consideration. It is beyond the scope of this research to apply more precise maintenance strategies like detective maintenance, condition-based maintenance, opportunistic maintenance, etc., into the maintenance for aircraft.

1.3.3 Maintenance Planning and Scheduling

This process primarily involves the planning and scheduling for the workforce and parts. Individual skills and scheduling the workforce in shifts play an important role in maintenance planning and scheduling (De Bruecker et al. 2018). Workers with different skills take different time to finish the same task. Shift plans involving the starting and ending time, shift succession, team size, etc., significantly influence the process of maintenance. Thus, individual skills and shifts in the workforce are considered. However, as for shifts, we just propose a relatively appropriate one instead of focusing on the optimal shift. Maintenance scheduling is principally based on the urgency of tasks and resource availability.

1.3.4 Cost and Service Estimation

Maintenance costs and service level are the major indicators to evaluate the efficiency of maintenance. In this research, the maintenance cost is involved with DMC (Direct Maintenance Cost) and IMC (Indirect Maintenance Cost). In terms of the IMC, we just consider the downtime cost. Costs like passenger service cost, aircraft servicing cost, advertising, reservation, etc., are out of scope. On the other hand, the service level is evaluated by the degree of delay.

1.3.5 Safety and Reliability Analysis

We derive the analysis of safety and reliability from the data generated from the simulation. True data (simulation data) and target data will be compared, in order to improve maintenance efficiency intentionally. For example, comparison can be performed on data like cycle time, MTBF (Mean Time Between Failure), MTTF (Mean Time To Failure), repair times, etc. Therefore, recommendations can be put forward to analyse safety and reliability, regarding the results of comparison.

1.4 Research Objectives

The scientific goal is to design a reliable autonomous system that can provide efficient operational support for aircraft maintenance. The main research objectives are derived from the three issues highlighted in Section 1.2, which are shown as follows:

- **Issue one: System building**
  - To propose a framework for accomplishing the design of the autonomous system;
  - To identify the requirements in the field of aircraft maintenance;
  - To propose an architecture model for describing the system's essence and key properties;
  - To implement a simulation system based on the architecture model, in order to demonstrate the feasibility of the architecture design.
- **Issue two: System verification**
  - To develop a framework for verifying the autonomous system;
  - To formalize the behaviours of the system;
  - To accomplish the model transformation between the autonomous system model and the formal verification model;
  - To design formal verification specifications, aiming at verifying the satisfactory of the formal model regarding specifications.

- **Issue three: System with data mining**
  - To develop a framework for combining the simulation system with data mining;
  - To build data mining simulation models. in order to integrate data mining techniques into simulation models.

This project aims at building a system to tackle the operational issues of aircraft maintenance. It focuses on not only the design of the system but the verification and practical use of the system. It also gives us with a more complete process of developing an autonomous system, which provides engineers with an alternative approach to building systems.

Manufacturing companies or airlines may be interested in this research. Because one of the ambitions of this research is to integrate all the information systems related to operational support of aircraft maintenance, in order to facilitate information flows of maintenance and improve maintenance efficiency. In fact, the autonomous system proposed can be regarded as a prototype of future real systems. The information used in the system can be understood as the information integrated from several real information systems for aircraft maintenance. The environment and scenarios of the system could give companies a blueprint about what operational support for aircraft will be like in the future.

1.5 Major Contributions

In this thesis, major contributions are concluded into five parts. The first part involves the analysis and application of standards in the process of the MRO. The proposed MRO process with stakeholders helps us understand the complexity of the maintenance process of aircraft. The analysis of maintenance-related standards allows us to comprehend the difference between standards and make the standards be more reachable to others.

Part two includes the design of the autonomous system for operational support of aircraft maintenance. The architecture model proposed clearly shows the essence and key properties of the system. The UML use case diagram links the stakeholders with actions, which provides an example about how different stakeholders cooperate with each other. This may mitigate the conflict between stakeholders caused by the ambiguity of responsibility. More precisely, the contributions of the second part are illustrated as follows:

- Proposing a new framework that fully supports the building of an autonomous system for operational support of aircraft maintenance;
- Proposing an architecture model for the autonomous system based on which the detailed development of components is achieved and a UML use case diagram involving stakeholders is provided.

The contributions of the third part are associated with the implementation of the ABSS (Agent-Based Simulation System) . The concrete ABSS enables us to have a more detailed description of maintenance strategies and schedules, a more in-depth analysis of service level and costs, and a loop of more complete maintenance information. Based on the simulation system, maintenance engineers can run different maintenance scenarios to investigate the impact of the key factors on maintenance costs and service level. For example, the impact of the number and the distribution of maintenance technicians on the maintenance efficiency can be analyzed. Predicting maintenance issues is also possible based on part of real data and the data generated from simulation experiments. Finally, the virtualization of the maintenance process allows us to gain an insight into any maintenance details.

In the following, the contributions are related with the formal verification for the behaviours of the ABSS. The model checking approach proposed helps check the logics of the ABSS in a formal fashion, which significantly demonstrates the satisfactory of the logics of the ABSS. Thus, experiment results based on the ABSS are more convincing. The detailed contributions are explained as follows:

- proposing an improved model checking approach for verifying the ABSS from both the abstract and detailed points of view;
- providing a complete model transformation process from the ABSS to the formal model for verification;
- giving a concrete case study of formal methods in practice and demonstrating the power of formal methods in improving designs and in providing new insights.

The contributions of the last part emphasize the combination of simulation with data mining. The approach proposed is an attempt for the automatic self-improvement for maintenance decision-making and enables us to perform the data analysis without sufficient real data. More specifically, the contributions are shown as follows:

- proposing a conceptual model and a simulation model of FRNN-ABSS (Fuzzy-Rough Nearest Neighbour - Agent-Based Simulation System) have been proposed;
- accomplishing the FRNN approach as Java class;
- giving an algorithm for automatically performing the fault classification and prediction.

1.6 Thesis Structure

The thesis structure is shown in Fig. 1.2, where “C i" is an acronym for “Chapter i". The feedback arrows imply the improvement of systems design can be performed either by formal verification

(C5 ⟶ C3) or by system performance analysis (C6 ⟶ C3). The major content of each chapter is discussed as follows:

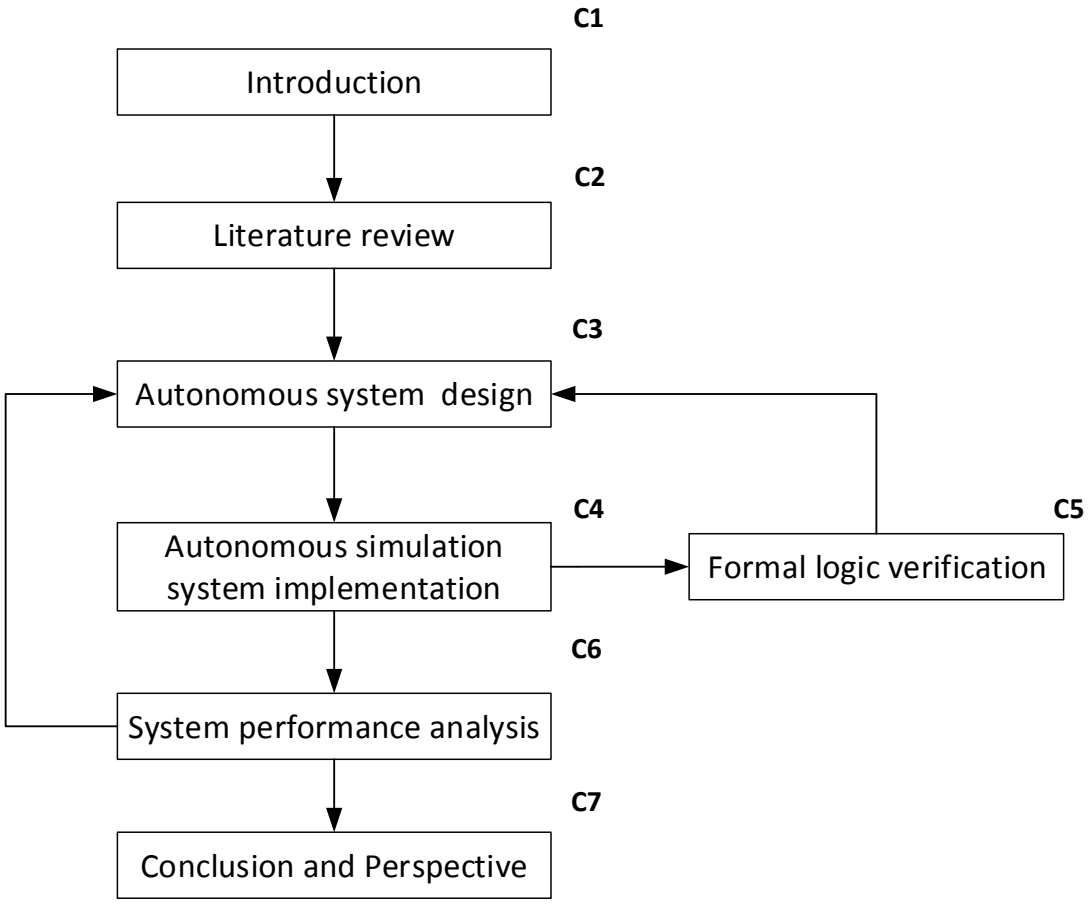

Figure 1.2: The structure of this thesis

Chapter 1 introduces the basic context for aircraft maintenance, major problem, scientific goal and objectives, major contributions and research scope.

Chapter 2 gives a detailed literature review on maintenance businesses, the architecture design for autonomous systems and agent-based modelling and simulation, the model checking for the ABSS, and the framework of the approach proposed.

Chapter 3 firstly describes the requirements in the aviation domain. The architecture model for the autonomous system is then proposed, based on which the lower development of each component in the model is realized. Finally, the system module interaction concerning stakeholders is provided.

Chapter 4 describes the implementation of the simulation system demonstrator in detail. The agent-based simulation model is derived from the architecture model. The development of agents

is associated with inputs, function, and outputs.

Chapter 5 involves the formal verification of the autonomous system. First, the behaviours of the autonomous system are formalized by automata-based models. In the following, the model transformation between the automata-based model and the Kripke-based structure is provided. The verification for the Kripke-based structure of the autonomous system is then carried out based on NuSMV model checker. At last, the agent-based simulation model has been modified with the verification results and comparative experiments have been conducted to demonstrate the efficiency of the modification.

Chapter 6 combines the simulation with data mining techniques, in order to classify the predict faults on aircraft components. The FRNN-ABSS model and the algorithm are proposed, aiming at integrating the FRNN approach into the ABSS. In addition, comparative experiments are given to illustrate the efficiency of the integration.

Chapter 7 concludes the whole work of this research with perspectives.

Chapter 2

Literature Review on Aircraft Maintenance

This chapter gives an overview of maintenance issues on aircraft. First of all, the maintenance process for aircraft including the maintenance program development and the maintenance task executing process is introduced. Secondly, businesses involving with maintenance processes are discussed. Subsequently, the existing maintenance systems are reviewed, in order to gain an insight into the advantages and disadvantages of systems and motivate us to propose an alternative approach to build an autonomous system of aircraft maintenance. The next section analyses the approaches for verifying systems. Finally, a framework of the approach proposed is provided, which explains the conventions, principles and practices for this approach.

2.1 Maintenance Process

The maintenance process is the meeting point of several stakeholders with the aim of maintaining the aircraft in a more efficient way. It is primarily derived from the analysis of related reports, papers and tool Maintenance Pro[1] (Ackert 2010), which is shown in Fig. 2.1. In this figure, round rectangles refer to behaviours happening on specific stakeholders and rectangles mean information transferred between stakeholders. The process is grouped into two phrases. The first phase starts with an aviation authority to the airline, which refers to the definition of the guideline of the maintenance. The second presents how the MRO executes maintenance work. In other words, the former focuses on how to design maintenance work, and the latter concentrates on how to realize maintenance work.

2.2 Maintenance Business

2.2.1 Flight Schedule

Flight schedules are often involved with aircraft maintenance. The existing models of flight schedules principally consider a wide range of factors: *residual maintenance time*, *residual flight time*, *uncertain demands*, *fair working time*, *option legs*, *itinerary-based demands*, and *multiple fare classes*, etc (Yan, Tang, and Fu 2008; Shenneld, Fleming, and Allan 2010; Kozanidis, Gavranis, and Liberopoulos 2014; Doi, Nishi, and Voß 2017). These factors play an important role in maintaining aircraft. For example, the factor of *residual maintenance time* affects decision-makings of repairing strategies. As a result, the whole maintenance process should include flight schedules.

2.2.2 Maintenance Strategy

Effective maintenance strategies can significantly help define maintenance planning and scheduling (Samaranayake and Kiridena 2012). Maintenance strategies are generally categorized into two types: CM and PM (Wang 2002). Tinga (2010) distinguished these two maintenance strategies. For corrective strategies, parts are only replaced or repaired when they fail to function correctly. Maintenance activities are planned to prevent systems from failing by replacing or repairing parts, regarding preventive strategies. The corrective strategy helps fully utilize the lifetime of assets, but it vastly threatens the safety and availability of systems (Mobley 2002).

Aircraft must be maintained with high safety and airworthiness. Air transportation is also full of competition (Kilpi, Töyli, and Vepsäläinen 2009). Over time, a systematic approach has been developed by MSG (Maintenance Steering Group) to define appropriate maintenance strategies for systems, subsystems, and parts, which is based on the logic of “consequence of failure" for a specific

[1] https://www.mtcpro.com/ (accessed in March 2019)

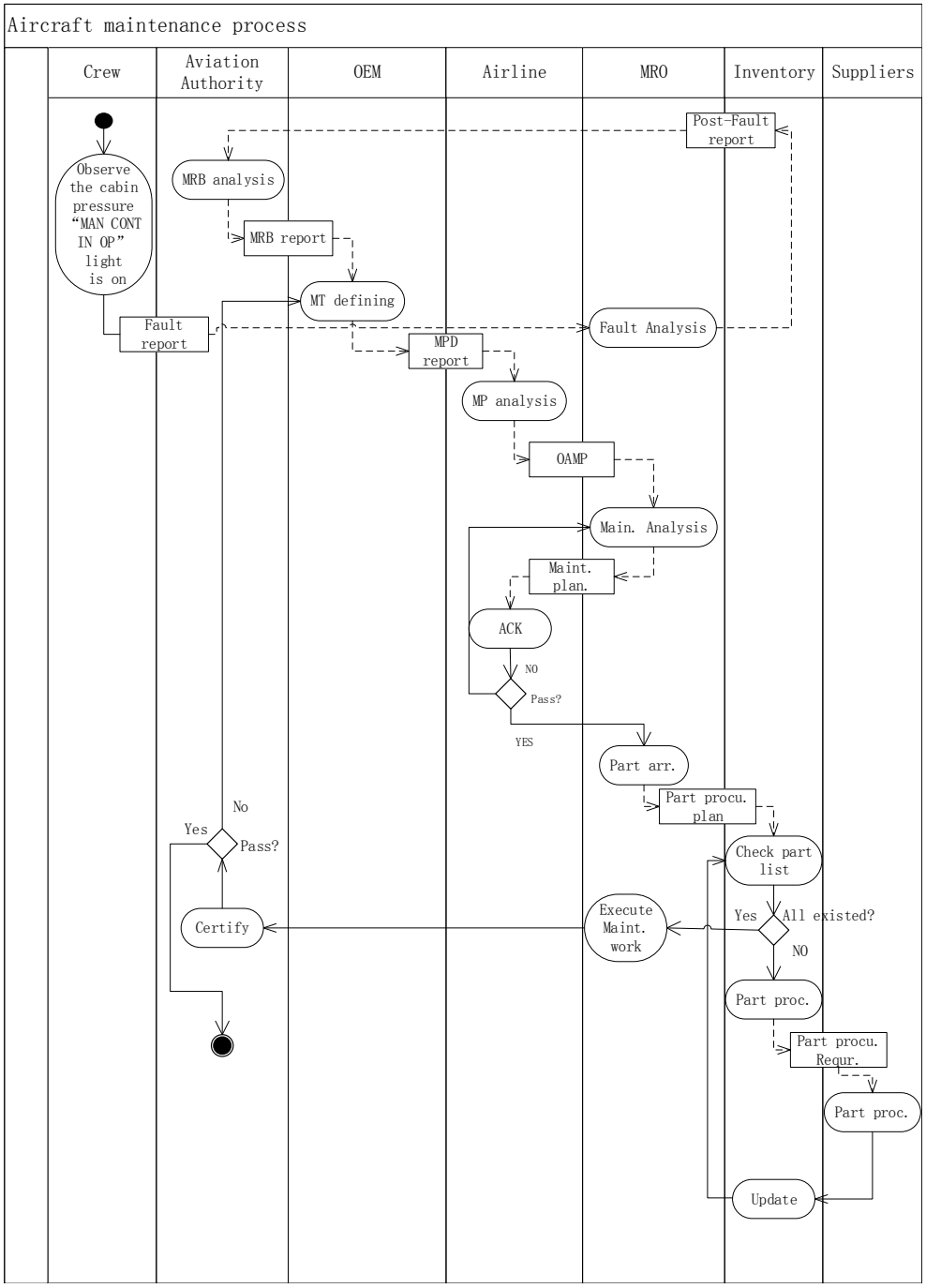

Figure 2.1: The maintenance process for aircraft - MRB (Maintenance Review Board), MT (Maintenance Task), MPD (Maintenance Planning Document), OAMP (Operator Approved Maintenance Planning)

aircraft type (Kinnison and Siddiqui 2012). Subsequently, many further researches on maintenance strategies have emerged. Tinga (2013) discussed the classification of maintenance strategies in detail. Many strategies, including PM, detective maintenance, condition-based maintenance, PM and opportunistic maintenance, etc., have been involved. According to the failure mechanism of target products, some additional works are concentrated on PM and condition-based maintenance (Li, Wang, and Peng 2016; Sheu et al. 2015; Liu, Wu, and Xie 2015; Zhang, Ye, and Xie 2014). As for CM, Verhagen and De Boer (2018) developed both time-independent and time-dependent proportional hazard model incorporating operational factors to reduce the number of unscheduled occurrences. In our work, PM and CM are taken into consideration.

2.2.3 Maintenance Planning and Scheduling

Maintenance planning and scheduling is another issue for aircraft maintenance, which usually involves the planning and scheduling of workforce and parts. As for the workforce, researchers tend to complete factors that affect the scheduling of the workforce and take the uncertainty into consideration. For example, Beliën, Cardoen, and Demeulemeester (2012) improved the workforce scheduling of aircraft line maintenance by incorporating the workforce capacity and non-determinable demand. Van den Bergh et al. (2013) extended the work of Beliën by considering the uncertainty in flight arrivals. Keysan, Nemhauser, and Savelsbergh (2010) studied the tactical and operational planning of scheduled maintenance for per-seat, on-demand air transportation. Safaei, Banjevic, and Jardine (2011) worked for a real maintenance workforce-constrained scheduling problem in a steel manufacturing context considering skill workforce and equipment availability. Quan et al. (2007) proposed evolutionary algorithms for scheduling preventive maintenance tasks with the consideration of minimizing the completion time of maintenance jobs. De Bruecker et al. (2017) introduced a three-stage mixed-integer programming approach for optimizing the skill mix and training schedules via integrating shift construction, skill constraints and training. Scheduling the workforce in shifts is usually much harder than without shifts. On top of the usual coverage constraints, shift planning adds extra constraints about the start time and duration of shifts, shift successions, team sizes, etc. Therefore, the collection of real-time data about the workforce and equipment is helpful for maintenance planning and scheduling.

2.2.4 Cost Model for Maintenance

Maintenance costs for aircraft can be divided into two parts: DMC and IMC. In terms of DMC, Dupuy, Wesely, and Jenkins (2011) stated that it consisted of the cost of maintenance crews, materials, and parts repair and replacement. Kumar et al. (2012) explored the cost of spare parts, materials, personnel, tools and support equipment, facilities and technical data. On the other hand, IMC is the expense which is not related to aircraft operations, such as passenger service cost

(flight attendants, food), aircraft servicing cost (line servicing, control, landing fees), advertising, reservation etc. (Park and O'Kelly 2018).

Some works consider DMC and IMC at the same time. Papakostas et al. (2010) considered elements that affected the maintenance cost are equipment and facility costs, supplies and logistics cost, labour cost, and overhead. Saltoğlu, Humaira, and İnalhan (2016) introduced a new cost model that not only considers the general cost of DMC, overhead but provides an approach to calculate downtime cost.

Additionally, EuroControl provided a model describing a methodology for evaluating true cost flight delays (Cook, Tanner, and Anderson 2004). Based on this model, Ferguson et al. (2013) proposed a new model to allow costs to be updated whenever the basic costs change.

In our cost model, we take only DMC into consideration, because the invisible fees are out of scope in this thesis.

2.2.5 Standard Analysis

The standard is a very useful guideline for work, which is a synthesis of wisdom from experienced engineers, experts and researchers, etc. However, standards are often rather abstract and not keeping on specific issues. So, it is not easy for engineers to make full use of them. Standards are also supposed to be just piles of papers if they cannot be put into use. Thus, it raises the issue about how to use relevant standards to improve the maintenance efficiency for aircraft.

According to ILS (Integrated Logistic Support) specification in the aerospace and defence industry, the standards can be classified into two categories: S-series Suite of Specifications and ATA (Air Transport Association of American) specifications.

From the view of ATA development, a number of standards, including ATA 100, ATA 2100, iSpec 2200, iSpec 2300, iSpec42, Spec 200 and Spec 2000, have been involved. These standards are all originated from ATA 100 and Spec 200, which focus on the technical data standard and material data standard respectively (Lafonsee 2015; Conn, Jones, and White 2008; Conroy et al. 2014).

Even though ATA specifications have made efforts to improve the efficiency of the life cycle of the aircraft, the limitations of ATA specifications become apparent because requirements for more flexible technical information management distribution continue to increase. Since there is no common source database available and no definition of the link with other standards, we just focus on the s-serials suites of specifications.

Multiple ILS specifications are currently available or in the process of development, including:

- S1000D[2], IS (International Specification) for technical publications using a common source database;
- S2000M[3], IS for material management - Integrated data processing for military equipment;

[2] http://s1000d.org (accessed in March 2019)
[3] http://s2000m.org (accessed in March 2019)

- S3000L[4], IS for LSA (Logistic Support Analysis) ;
- S4000P[5], IS for developing and continuously improving preventive maintenance;
- S5000F[6], IS for in-service data feedback;
- S6000T[7] - IS for training analysis and design.

It should be noted that S6000T is still in draft at that date. Without the specification, it is rather difficult to give detailed illustrations. Thus, in the following, we choose to avoid involving it.

In order to analyse s-series suites of specification, a number of factors, including producer, issued time, scope, data model, inter-operability, strength and weakness, have to be taken into consideration. The producer, issued time, scope, strength and weakness will provide a general idea of each standard. The data model and inter-operability mainly address the problem of how these standards arrange their works and cooperate well with each other. Table 2.2.5 gives a detailed comparison among these standards (Liu et al. 2017).

[4] http://s3000l.org (accessed in March 2019)
[5] http://s4000p.org (accessed in March 2019)
[6] http://s5000f.org (accessed in March 2019)
[7] http://s6000t.org (accessed in March 2019)

Table 2.1: The comparisons between the s-series of standards - A (Standard), B (Producers), C (Issued time), D (Scope) , E (Data model), and F (Inter-operability)

A	B	C	D	E	F	Strength	Weakness
S1000D	ASD, AIA, ATA	1989-2012	exchange, distribution and use of technical documents	XML	S2000M, S3000L, S5000F	Strong abilities for data exchange	Lack of certification and details of handling process.
S2000M	ASD, AIA	1992-2015	material management activities	UML class model and UoF	S1000D, S3000L	Strong ability for logistic material management between contractors and the customers	Exist overlap business relationships of S2000M and S1000D. Lack of explanation for some parts of the S2000M. Lack approved toolset.
S3000L	ASD, AIA	2010-2014	all processes & requirements regarding LSA	UML class model and UoF	S1000D, S2000M, S4000P	Strong ability for the selection of LSA candidate items	Lack the completeness of its data model. Lack approved toolset.
S4000P	ASD, AIA	2014 -	preventive management development and continuous improvement	no	S3000L	Strong ability for reviewing the completeness and effectiveness of preventive maintenance tasks	The way of determining ARC (Analysis Relevant Candidates) and non-ARC lacks accuracy. Lack approved toolset.
S5000F	ASD, AIA	2016 -	In-service data feedback	UML class model and UoF	S1000D, S2000M, S3000L	Strong ability for performing a thorough analysis of operation and maintenance performance of a product	Lack of approved toolset.

2.3 Existing Maintenance Systems

Complex systems have been recognized as the systems with numerous components, interactions and interdependencies that are hard to understand, predict, manage, design or change (Magee and de Weck 2004). Durazo-Cardenas et al. (2018) applies this definition to railways maintenance systems, in order to provide the concept of the complex data-rich autonomous maintenance system for railways, which is illustrated as follows:

- Asset degradation triggers the system response;
- A system supported by information sources and expert systems;
- Integrated optimal task sequence;
- Cost factored automatic response;
- Automatic decision making;
- Extremely large data output that is curated fused to produce autonomous actionable information.

As for the maintenance system for aircraft, this definition is rather helpful. However, since the components of aircraft and the characteristics of aircraft maintenance (discussed in Section 1.1.3) are much more complicated than those of trains, some key features are still missing. For example, the maintenance for aircraft is highly regulated by aviation authorities. Maintenance goals and repair schemes are strictly supervised and managed. Maintenance strategies for different components should be defined in detail and be optimized.

This section provides a comprehensive literature review on modelling maintenance systems. Since this paper focuses on the design of the autonomous maintenance system to improve operational support for aircraft maintenance, it excludes the papers without a system's perspective. For example, the topics like utilizing mathematical approaches (Keysan, Nemhauser, and Savelsbergh 2010; De Bruecker et al. 2018) or only introducing algorithms (Diaz, Huertas, and Trigos 2014; Abdelghany, Abdelghany, and Azadian 2017) to improve maintenance scheduling and planning are out of our scope. The synthesis analysis of reviewed papers is concluded in Table 2.2. The papers are analyzed from six aspects: proposes conceptual models, proposes simulation models, involved stakeholders, maintenance-related information, model purposes and the relevance to our research. Furthermore, the relevance is evaluated with different levels: low, medium and high.

According to this table, researchers rarely propose both conceptual models and simulation models at the same time. Fortunately, Chang et al. (2007) gathered these two models together. However, they only concentrated on the issue of maintenance labours. The maintenance process was rather simple. As for involved stakeholders, researchers have almost neglected this aspect.

Although Durazo-Cardenas et al. (2018) have involved stakeholders, they do not provide a simulation model. The majority of maintenance information has been involved by researchers. Most of the reviewed papers consider parts states, resource availability, planning and scheduling information and labours, etc. However, the planning and scheduling of information are just associated with maintenance activities. The scheduling information for the system of interest is still missing. In this thesis, it is the information about the scheduling of aircraft. The lack of this information will lead to the consequence that the impact of the delay for one aircraft on other aircraft can not be investigated. In addition, the papers with low relevance mainly suffer from limited maintenance processes or overly simple model details to support our research. Half of the papers own a medium level of relevance. However, their contributions to our research are limited. Only one paper (Durazo-Cardenas et al. 2018) proposed a detailed conceptual model and discusses stakeholders but it did not involve simulation models. Sahnoun et al. (2015) provided a simulation model, whereas they just delivered a simple maintenance strategy and the design of turbine agents is less of reasonableness. Gary et al. (2018) presented a model where only an analysis of an aggregated level can be performed. However, this model hardly describes the cooperation between stakeholders, which is less of reality. Yang, Djurdjanovic, and Ni (2008) accomplished a discrete-event model and employ a Genetic Algorithm to optimize maintenance schedules. Unfortunately, the details of simulation models can not be observed and only cost factor is considered for optimizing the maintenance schedule.

MacKenzie, Miller, and Hill (2012) (paper 6) and Duffuaa et al. (2001) (paper 10) have a high level of relevance. However, they treat conceptual models and simulation models separately. MacKenzie, Miller, and Hill (2012) provided us with a proper “prototype" of our model. Since no conceptual models and cooperation between stakeholders are analyzed, the simulation model is hardly involved with important insights into the maintenance process for aircraft. On the other hand, Duffuaa et al. (2001) gave us a detailed analysis of maintenance concepts and maintenance flows, however, it can only serve as the theoretic support for the autonomous system design because of no simulation models.

To conclude, the existing simulation models are prone to tackling maintenance issues without a comprehensive understanding of maintenance concepts. They are not able to offer an insight into the maintenance process, which results in simple maintenance strategies (MacKenzie, Miller, and Hill 2012), overly simple algorithms (Sahnoun et al. 2015), only an aggregated level analysis (Gary et al. 2018), etc. While the papers of conceptual models tend to lack a concrete simulation demonstrator to illustrate the feasibility of systems design. Thus, we are motivated to build an autonomous system starting from the requirement analysis and ending with a simulation system. Our simulation system can fully support the analysis of the complete maintenance process, enable a more in-depth analysis of maintenance issues and automatically optimize the maintenance scheduling and the part repairing.

Table 2.2: An overview of related papers - A (proposes conceptual models), B (proposes simulation models), C (involved stakeholders)

Article	A	B	C	Maintenance-related Info.	Model Purposes	Relevance to our research
1.(Durazo-Cardenas et al. 2018)	yes	no	yes	parts degradation state, planning and scheduling info., cost info.	autonomous scheduling maintenance jobs	medium, similar model purposes, similar aspects of architecture designs, no simulation system is involved.
2.(Gary et al. 2018)	no	yes	no	equip. status, labours, maint. type, planning and scheduling info., stock, maint. performance, maint. cost	presents a system dynamics model for the study on dynamic behaviours of maintenance performance and costs	medium, similar in scope, explicit model, only enables an aggregated level analysis.
3.(Sahnoun et al. 2015)	no	yes	no	main. duration, resource availability, degradation level, weather condition, manpower	develops a multi-agent system for selecting the optimal maint. strategy for offshore wind turbines	medium, similar in scope, explicit model, simple maintenance strategy, an overly simple algorithm for maint. scheduling.
4.(Zhang et al. 2015)	yes	no	no	material, parts, knowledge, labours, equipment, failures,	supports the continuous improvement of productivity and reduces the MRO cost	low, the proposed framework is too simple. The model description is quite abstract. No strategies or scheduling is involved.
5.(Samaranayake and Kiridena 2012)	yes	no	no	material, activities, resources, suppliers, labours	develops an integrated framework for the planning and scheduling of aircraft heavy maint.	low, similar objectives, a detailed discussion about planning and scheduling of aircraft heavy maint., only part of maint. process are involved.
6.(MacKenzie, Miller, and Hill 2012)	no	yes	no	average sortie duration, the number of aircraft, manpower, break rates, abort rates, fix rates, planning and scheduling	designs an agent-based simulation model for providing directions about dynamic task priorities and resource assignments	high, complete maintenance process, part of planning and scheduling logic is shared. However, no dynamic strategies for repairing parts are provided. The model is less of reality and no costs are involved.
7.(Ward et al. 2010)	yes	no	no	location, direct outcome of maint., consequences, staff, suggestion, actions	develops a comprehensive and ecological model of the operating system	low, similar model purpose, overly abstract model, principally based on the manual analysis, hard to transfer to our research. (to be continued)
8.(Yang, Djurdjanovic, and Ni 2008)	no	partly	no	machine degradation level, maint. cost, spare parts, maint. resources	proposes an approach for maint. scheduling based on the predicted equipment degradation	medium, similar model purposes, abstract description of the simulation model, hard to reuse.
9.(Chang et al. 2007)	yes	yes	no	total cost, labours, machine failures	investigates the trade-off between maintenance personal staffing level and the throughput of a production line	low, similar in scope, very limited maintenance processes, little applicable details to support model building.
10.(Duffuaa et al. 2001)	yes	no	no	input info., maint. type, planning and scheduling info., spares and equipment info., labours, quality and performance info.	delivers a generic conceptual model for developing a realistic simulation model	high, similar in scope, explicit model, shares the concepts of the model, partly reuses the flow charts, however, no real simulation system is built.

2.4 The Framework of the Proposed Approach

The framework for building the autonomous system is divided into five parts: requirement analysis, autonomous system design, simulation system implementation, formal logic verification and performance analysis, which is shown in Fig. 2.2. Software Engineering and system simulation are involved in this framework. The approach proposed is illustrated in the following.

The first step is to analyze requirements from operational support for aircraft maintenance. Based on these requirements, an architecture schema is designed, which derives component models and a system interactive model. This system has been equipped with the very basic elements allowing operators of aircraft maintenance to globally automatically support maintenance decisions. The next step is to implement an ABSS as a system demonstrator based on which an important insight into the future system can be accomplished. Formal logic verification in terms of the ABSS is then studied, aiming at checking the logics of the system. Finally, we integrate data mining techniques into the ABSS to perform fault location prediction. Therefore, this framework tries to provide a complete methodology to build an autonomous system.

Our framework for developing the autonomous system on aircraft maintenance clarifies the practical value of the work. To begin with, it provides an alternative way to implement the “real system" based on systems design by building simulation systems, which enables us to gain important insights into future systems in an inexpensive way. Secondly, it offers a general approach to build an autonomous system that supports reuse and sharing. System designers in other fields are likely to provide a rigorous architecture design of the system and deliver a powerful demonstrator of the future system, which illustrates the feasibility of the design. At last, this framework proposes a methodology for building digital twins. It not only proposes a prototype system-of-interest but allows us to manipulate the prototype system with part of real data, in order to examine the reaction of the system.

2.5 Summary

The purpose of this chapter is to review the state-of-the-art research in the field of operational support for aircraft maintenance, in order to classify the existing works, highlight research gaps and guide future research. Different aspects, including maintenance process, maintenance business, and maintenance systems, have been addressed in the literature review. The research gap is that maintenance theory model and application model are often treated separately, which leads to less efficiency of application models. Therefore, an innovative framework is proposed, in order to provide an innovative methodology for combining these two models. The detailed description of the approach is discussed in the next chapters.

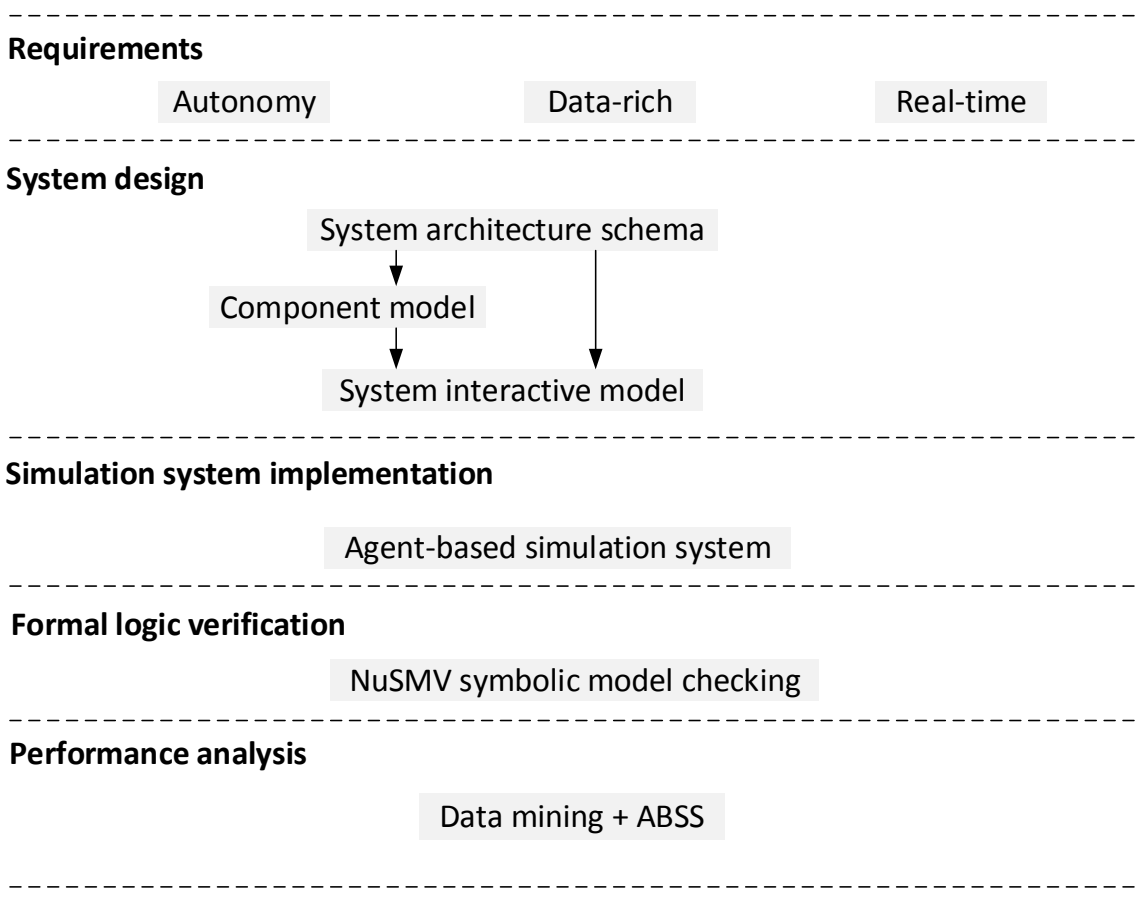

Figure 2.2: The framework of the proposed approach

Chapter 3

Design of An Autonomous System of Aircraft Maintenance

In the last chapter, how to combine design models and application models was identified as the major issue in this thesis. The framework proposed specified the models: the system architecture model and simulation model. In this chapter, we will focus on how to design the architecture model on the autonomous system for aircraft maintenance. To do this, the approach chosen to design the model will be first discussed. The requirement analysis will then be performed, in order to recognise the key features of the autonomous system. In the following, an architecture model for the autonomous system will be presented, showing the major components and connections. The lower development of the architecture model will also be illustrated. At last, the responsibility of stakeholders will be emphasised as well, aiming at avoiding the occurrence of “dead-lock service" (no one really wants to intervene due to the vagueness of responsibility).

3.1 Introduction

The characteristics of systems require systems capable of being broken down into manageable fragments. In line with these principles, a systematic engineering method is introduced. SE (Systems Engineering) is an interdisciplinary field of engineering and engineering management that focuses on how to design and manage complex systems over their life cycles (Blanchard 2004). The *waterfall approach* is a common approach to develop complex systems, which uses a sequence of design, implementation, and test and evaluation phases or steps managed by formal reviews and the delivery of documentation (Royce 1987). The advantages of this approach are illustrated as follows (Waltz and Hall 2001):

- The ability to systematically build large systems by decomposing them into small, manageable, testable units;
- The ability to work with multiple designers, builders, vendors, users, and sponsoring organizations;
- The capability to perform the development over an extended period of time with resilience to changes in development personnel;
- The ability to define and manage risks by identifying the source of potential problems;
- Formal control and monitoring of the system development process with well-documented standards and procedures.

Therefore, SE is a good choice for building the system, in order to have a holistic point of view on the system at the design phase.

Based on SE, the analysis of system requirements is firstly conducted. An architecture model for building the system is then proposed and the lower-level of the system is developed. Finally, the interactions between components with stakeholders are illustrated by the UML (Unified Modeling Language) diagrams.

3.2 Requirement Analysis

A system architecture is a structure of components, relationships, and principles governing the design and evolution of the system over its life cycle (ISO 2011). This definition addresses components and relationships. Understanding the aim of the system is the first step to identify components and relationships. For example, which decisions will be made? What are the characteristics of the system?

Effective engagement with the stakeholders is vitally important to capture the new system requirements. The interview was conducted with a few experienced aircraft maintenance engineers

in Chengdu, China. Because of confidential issues, several general questions were just discussed, which are illustrated as follows:

- Q1: what are the major issues about aircraft maintenance?
- Q2: what is the level of autonomy required?
- Q3: what are the general information needed for maintenance?
- Q4: what are the ways of accessing information?

Table 3.1: The answers on Q1, 2, 3 et 4

No.	Answer
1	AMOs (Aircraft Maintenance Organizations) have multiple IT systems which are not located in the neighbourhood of engineers and aircraft; The real-time progress of aircraft maintenance is highly demanded; AMOs are often not meeting the planned release dates of aircraft maintenance checks.
2	The high level of autonomy required spans several areas: maintenance strategy planning and updating, workers scheduling, resource response, cost and service efficiency.
3	The general maintenance information refers to parts, equipment, preventive scheduling, worker scheduling, inventory, task card, etc.
4	Information systems, telephones, emails are the major ways of communication.

The answers about Q1 to Q4 are summarised in Table 3.1, which is based on a 3-hour interview. According to the answers, we can identify several problems, including “the data fusion of different IT systems is not sufficient", “an autonomous maintenance system for aircraft maintenance is missing" and “A lack of an effective approach to make maintenance strategies and scheduling plans". In terms of autonomous systems, our system is supposed to be capable of choosing to obey a rule or not and making complex decisions. However, implementing the human understanding into the system is not our focus. In addition, the strategy for maintenance enables the maintenance system to be adapted to real-time decision-making. Decision-making should be optimized with the consideration of service level and maintenance costs.

3.3 Architecture Development

3.3.1 Architecture Model

In order to express the data fusion processes in aircraft maintenance, a formal architecture is proposed based on the previous requirement analysis and Llinas and Hall’s data fusion architecture model (Llinas and Hall 1998). Fig. 3.1 illustrates the architecture model for the autonomous system, which consists of six components:

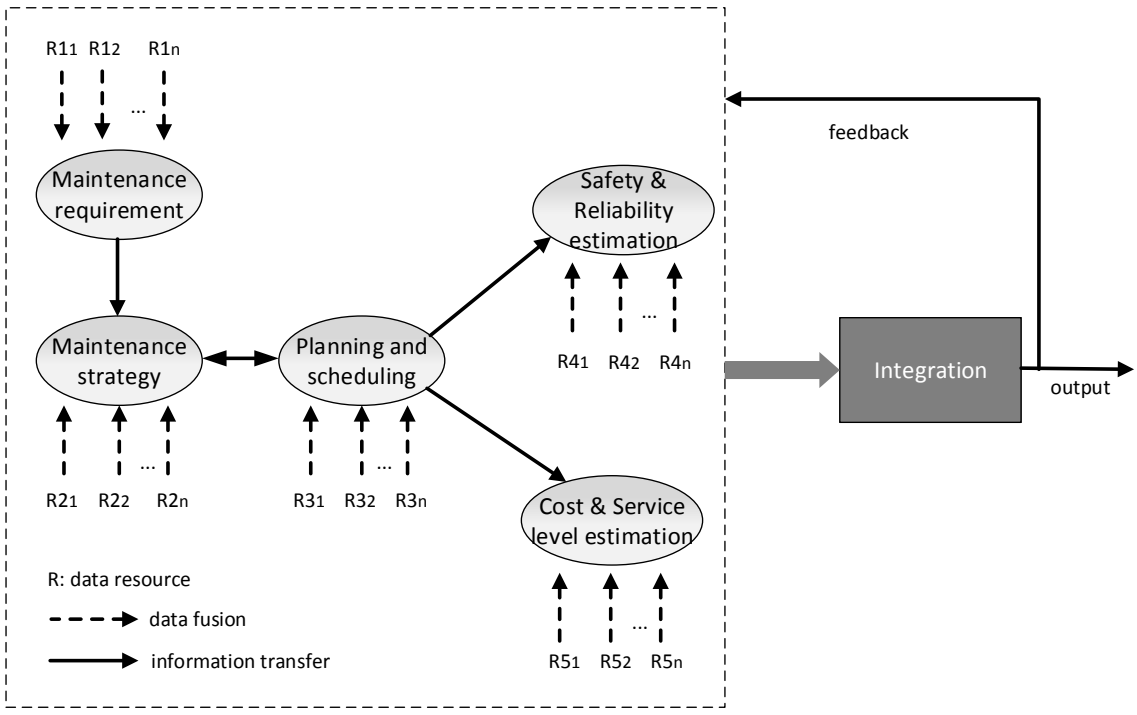

Figure 3.1: The architecture model of the autonomous system, adapted from (Llinas and Hall 1998)

1. *Maintenance requirement*. This component triggers the whole of maintenance activities. The maintenance requirements can be derived from scheduled maintenance, unscheduled maintenance or third-party requests. Thus, the raw data involves the scheduled maintenance data, the sensor data, the alarm, and the third-party maintenance requirement report.

2. *Maintenance strategy*. This component is used to analyze the real-time state of parts in order to determine their corresponding maintenance strategies. The process for data fusion involves the synthesis analysis of the data from real-time inspection reports, sensors and troubleshooting databases.

3. *Planning and scheduling*. This component tries to automatically generate maintenance plans and schedules, including the optimization of the maintenance task sequence with the consideration of fundamental operating maintenance data such as repair time, cost, and resource availability, etc.

4. *Safety & Reliability estimation*. The objective of this component is to collect, analyse, and manage reliability-related data, in order to conduct safety and reliability estimation. The result of estimation will be the data for the integration.

5. *Cost & Service level estimation*. The goal of this component is to perform the estimation of

maintenance costs and service level. The analysis of cost is the data fusion process of all the costs related to maintenance processes such as the repair cost, part cost, downtime cost, etc. The analysis of service level involves the estimation on the waiting time of flights.

6. *Integration*. The aim of this component is to converge the estimation of safety & airworthiness and cost & service level, aviation authority regulation, resource availability into a global fused system output.

Fig. 3.1 shows the relationship between components. The data collected from sensors, information systems or the crew's observation, is transferred to the fusion centre (component), where fusion processes take place in an ordered way. The flow of information between *maintenance strategy* and *planning and scheduling* is bidirectional because these two components may need to cooperate with each other directly due to the needs of maintenance business such as the occurrence of uncertain events. The components inside the dotted rectangle are the core maintenance components. The information from these components is transferred to the integration model where its output delivers a global estimation of impacts on the maintenance strategy, maintenance scheduling, resource, cost, service level and safety & airworthiness on maintenance efficiency. Its feedback also provides recommendations to support the better decision and iteratively refines the maintenance process. This model shows a certain level of variability. For instance, it is possible to be adapted to deal with predictive maintenance. To do so, some information should be prepared and analysed in advance such as real-time states of parts, the historic information of parts and parts degradation curves. Once the decision of maintenance is made, the rest of the issues of making maintenance done makes no big changes.

3.3.2 Concept Models of Components

Concept models of components are developed in this section, which illustrate the static structure of the system. These models are described by UML class diagrams because they are able to easily explain concepts and explicitly capture their relationships. Figs. 3.2 to 3.6 show the details of concepts and relationships of each component.

UML relations used in the diagrams involve *Association*, *Inheritance*, *Aggregation* and *Composition*. Concept *Customer Service* in Fig. 3.2 implies the maintenance requirement from the third party. In Fig. 3.3, concept *Uncertain Activity* means component *Maintenance Strategy* is able to deal with uncertain issues. Concept *Fault Part* indicates this component will finally deliver a list of parts to be repaired. Fig. 3.4 illustrates tasks can be maintenance tasks and the procurement of parts that are out of stock. It should be noted that concept *Replace&Repair* denotes fault parts should be replaced and repaired in the workshop due to the lack of time. In terms of component *Safety & Reliability Estimation* (shown in Fig. 3.5), the concepts of the compliance of safety and reliability imply the degree of conformity to airworthy standards, regarding safety and reliability

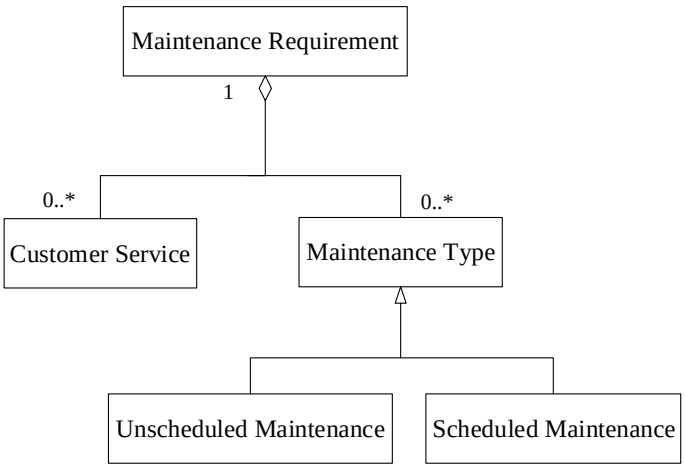

Figure 3.2: Concept model of *Maintenance Requirement*

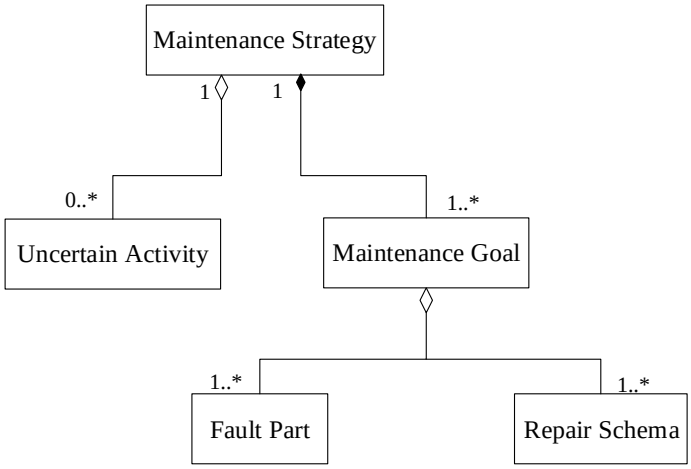

Figure 3.3: Concept model of *Maintenance Strategy*

respectively. As for component *Cost & Service Level Estimation* (shown in Fig. 3.6), indirect costs are involved with concept *Cost*, however, they are not further studied due to the limit of our research scope.

3.3.3 Lower-level Integration Architecture Development

The lower-level of architecture model describes each component in detail. The components, including inputs and outputs, are synthesised in Table 3.2.

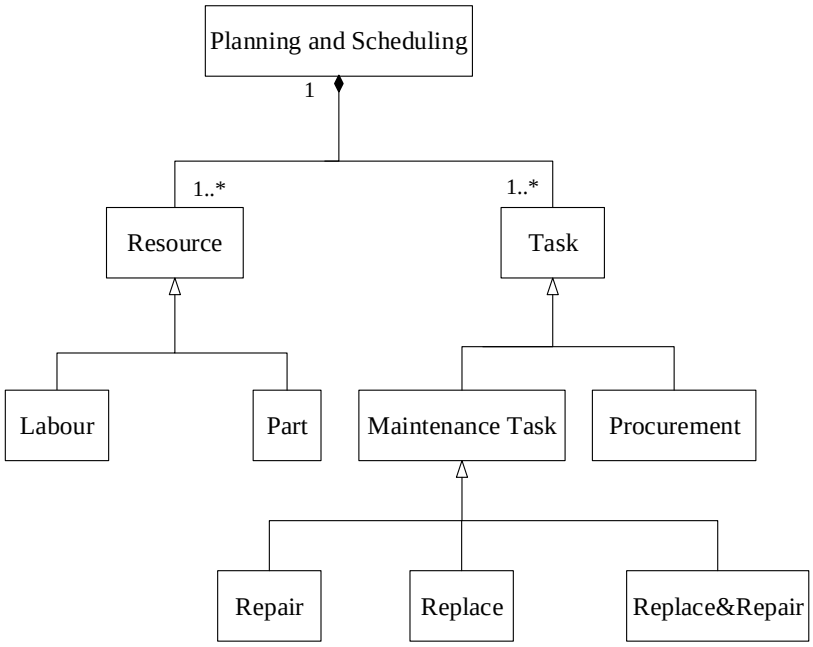

Figure 3.4: Concept model of *Planning and Scheduling*

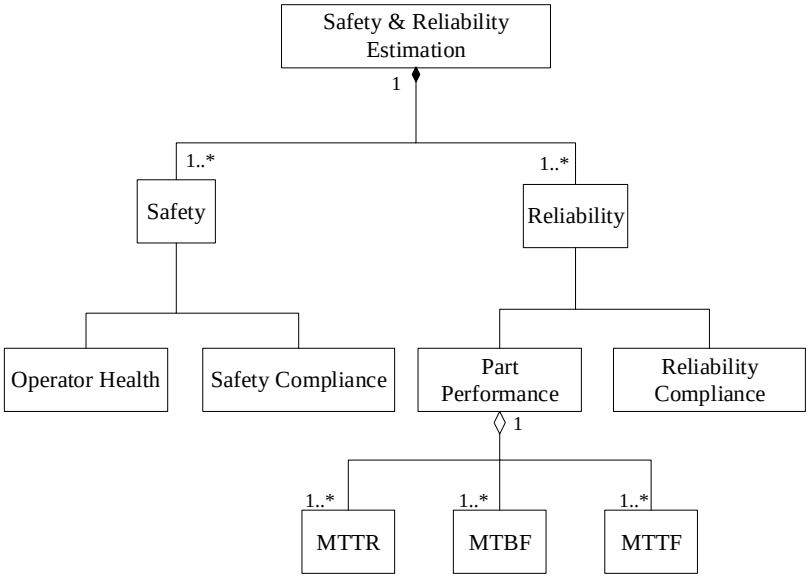

Figure 3.5: Concept model of *Safety & Reliability Estimation*

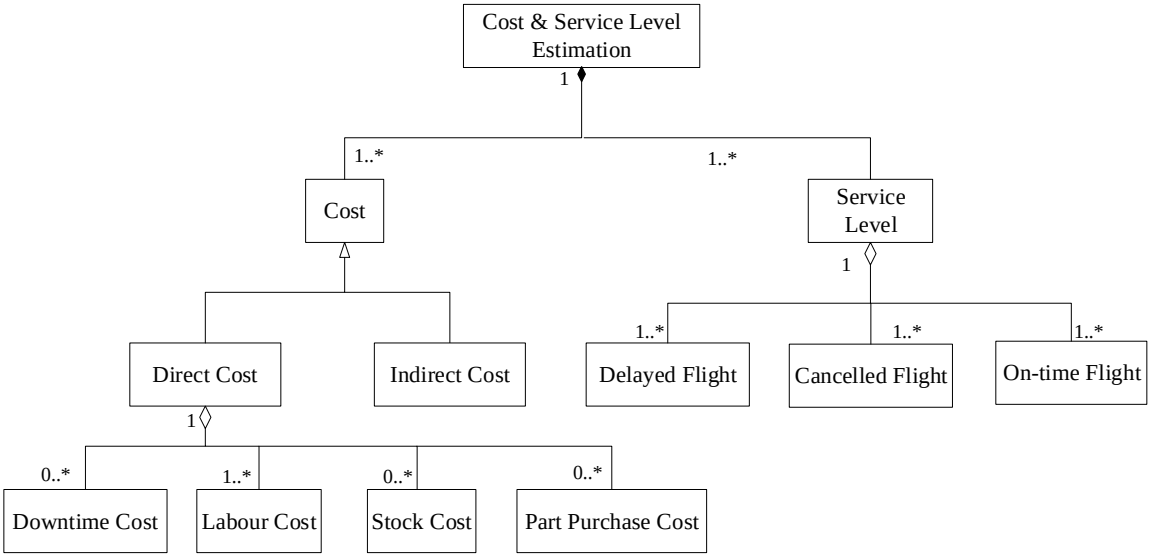

Figure 3.6: Concept model of *Cost & Service Level Estimation*

3.3.3.1 Component 1: Maintenance requirement

The monitoring data, scheduled maintenance, and third-part request are the fundamental inputs to *Maintenance requirement* component. This component fuses these data to determine the basic maintenance request and severity level. The monitoring data can be obtained from sensors, alarms, or observation by crews. The scheduled maintenance for aircraft components is programmed by OEM at the design phase. It consists of not only the timing of maintenance, but maintenance activities, such as procedural instructions for maintenance tasks, procedures for recording the results of inspections, checks, tests, and other maintenance, etc. (Allen 2012). Severity level depicts the degree of impacts of maintenance requirements (such as component faults, scheduled maintenance, etc.) on safety and reliability. The outputs of this component will trigger *Maintenance strategy* component.

3.3.3.2 Component 2: Maintenance strategy

The major aims of *Maintenance strategy* component are to dynamically determine maintenance strategies for parts and deliver maintenance goals and repair schema. The inputs of this component are maintenance requests, maintenance regulations and maintenance timings. The data of maintenance requests implies the whole description of maintenance including maintenance type, location and priority. The maintenance regulation refers to maintenance activities which are mandated to be executed. For example, Operator Approved Maintenance Planning includes numerous

Table 3.2: The table of components

Component	Inputs	Outputs
Maintenance requirement	Monitoring sensor: fault info., location, etc. PM: time. Third-party request: maintenance requirement report.	Maintenance request: type, location. Severity level
Maintenance strategy	Maintenance request: type, location. Maintenance regulation: operator approved maintenance planning. Maintenance timing: remaining useful lifetime, interval, resource.	Maintenance goals and repair schema Maintenance strategies for relevant parts
Maintenance scheduling and planning	Information: remaining useful lifetime, intervals, strategies, labours, parts, equipment. Maintenance historic data: trouble-shooting data. Maintenance cost	Information: job completion, repair time, delay, interval. Tasks scheduling: parts, labors.
Safety & Reliability estimation	Airworthiness directives and service bulletins. Information: the number of faulty parts and corresponding time.	Safety indicator Reliability indicator
Cost & Service level estimation	Information: labour working time, parts, equipment, downtime, interval, etc. Flight information: flight no., aircraft no.	Service level indicator Cost indicator

documents like AD (Airworthiness Directive). The AD provides maintenance management processes in order to ensure the airworthiness level for an aircraft. The data of the maintenance timing is applied to optimize maintenance strategies via analyzing which maintenance can be delayed and dynamically updating maintenance types of parts.

3.3.3.3 Component 3: Planning and scheduling

Once maintenance strategies for parts are determined, the maintenance planning and scheduling will be defined accordingly. *Planning and scheduling* component is able to automatically generate the maintenance task sequence and the scheduling of labours. The input information is used to define the task sequence by analysing the availability of labours, parts and equipment. The remaining useful lifetime and intervals (the difference between the times of the next departure and the current arrival) are the basis for delivering repairing strategies. The historic data of maintenance is utilized when some uncertain events happen during the process of repairing, in order to find solutions. The cost analysis affects ways of repairing. The final scheduling result of maintenance activities is presented by a Gantt chart.

3.3.3.4 Component 4: Safety & Reliability estimation

Safety & Reliability estimation component is dedicated to checking safety issues and analysing system reliability. Checking safety issues principally depends on AD and SB (Service Bulletin). These two documents describe the maintenance management process and repair management process. If one of the steps is not compliant with the standard process, it will cause a safety risk. The system reliability analysis is conducted by analyzing the performance of parts. This component should ensure the safety of aircraft and provide information for product design.

3.3.3.5 Component 5: Cost & Service level estimation

Cost & Service level estimation component concentrates on the estimation of maintenance costs and service level. The input information is concerned with the cost of labour, parts, equipment, downtime, etc. The analysis of service level focuses on the delayed time of flights, which refers to the total repair time and intervals. The efficiency of maintenance is evaluated by a synthesis analysis of maintenance costs and service level.

3.4 System Module Interaction

The system module interaction of the autonomous system of aircraft maintenance is presented by the UML UCD (Use Case Diagram), which is shown in Fig. 3.7. UML UCDs have been widely used in the system module interaction (Skersys, Danenas, and Butleris 2018). There are four basic types of elements in UML UCD (Booch 2005), including:

- A *Subject* is a system with the capability to own *UseCases*;
- A *UseCase* specifies a specific unit of functionality (behavior) that a subject can perform in collaboration with one or more actors;
- An *Actor* is type of role played by an entity that interacts with the associated use cases within boundaries of the specific subject;
- An *Association* is a kind of relationship used to describe the between actors and use cases.

In this figure, modules in the system are depicted by 5 actors: integrator, requirement manager, strategy manager, planning and scheduling manager, cost estimator and quality controller. The interactions between use cases are generally explained as follows:

- The requirement manager generates maintenance requirements by use case “identifies requirement types", which includes three use cases: “identifies component maintenance", “identifies scheduled maintenance" and “identifies customer services". The identified requirement will be used by the integrator to generate strategy requests;

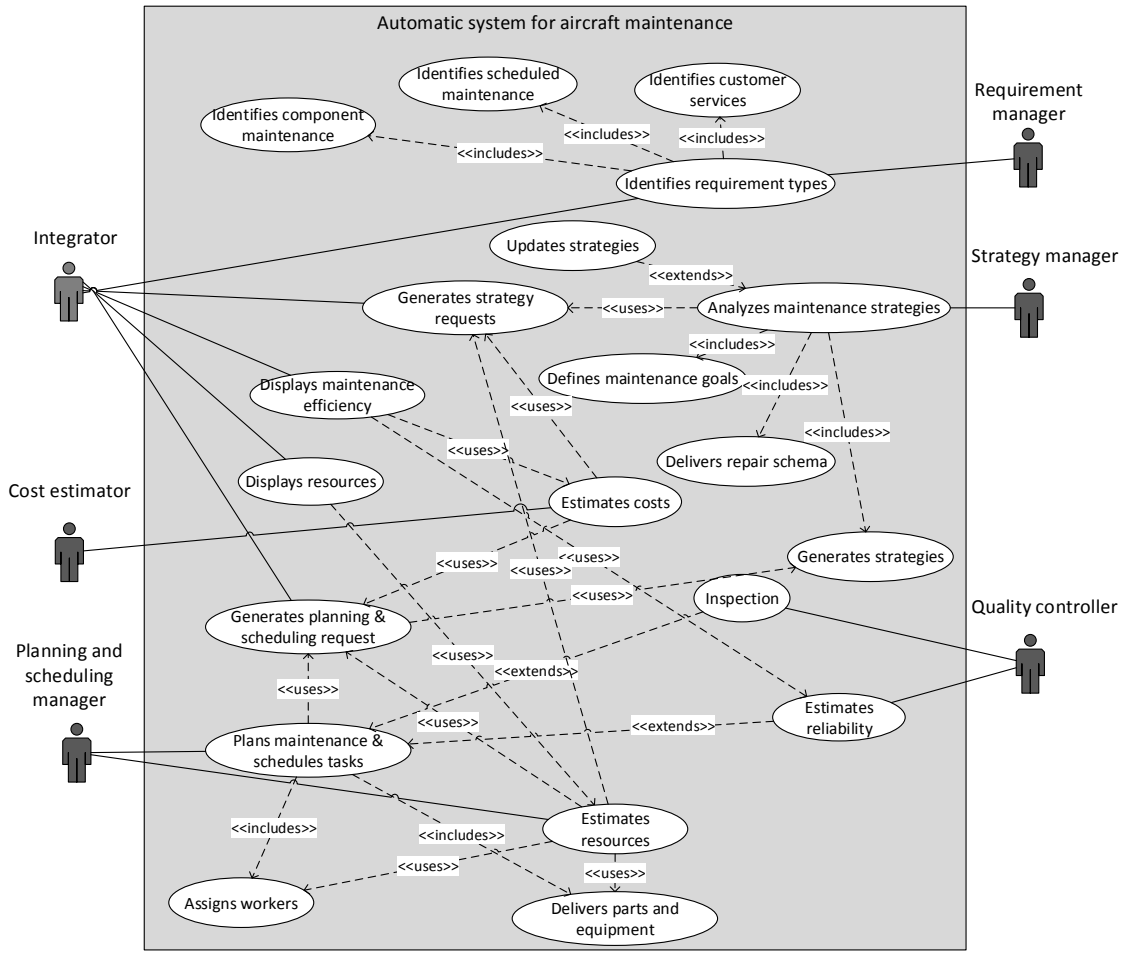

Figure 3.7: Illustrative UML UCD of the autonomous system of aircraft maintenance

- The strategy manager employs “analyzes maintenance strategies" use case. This use case utilizes “generate strategy requests" use case, in order to evaluate resource and maintenance costs. It then involves use cases: “defines maintenance goals", “delivers repair schema" and “generates strategies". Additionally, use case “analyzes maintenance strategies" will be extended as “updates strategies" when original strategies are not suitable. Use case “generates strategies" will be used to generate planning and scheduling requests;

- The planning and scheduling manager is associated with two use cases: “plans maintenance and schedules tasks" and “estimates resources". Use case “plans maintenance and schedules tasks" uses that of “generates planning and schedules request" and it includes two use cases: “assigns workers" and “delivers parts and equipment". At the same time, use case “estimates resources" uses these two use cases as well, which will be used by use case “displays

resources";

- The cost estimator concentrates on use case “estimate costs";
- The quality controller is linked with “inspection" and “estimates reliability" use cases. These two use cases analyze the reliability, safety-related data generated in use case “plans maintenance and schedules tasks". The results will be used to display maintenance efficiency;
- The integrator issues 5 use cases “identifies requirement types", “generates strategy requests", “displays maintenance efficiency", “displays resources" and “generates planning and scheduling request", aiming at connecting different modules and displaying the key maintenance information.

3.5 Discussion

As mentioned in Section 3.2, the key requirement is to build an autonomous system fusing different IT systems on operational support for aircraft maintenance, in order to improve maintenance efficiency and reduce maintenance costs. The architecture model proposed for the system provides a global view about components and their relationships in the system. It allows us to fuse different IT systems involved since different components are associated with one or multiple IT systems and their relationship implies the connections between IT systems. The UML class diagrams of each component detail the static structure of the system. The architecture model associating with class diagrams guides us to implement the lower development of components, which involves the inputs, outputs and functions of components. Hence, the maintenance-related data from IT systems can be installed at the corresponding components properly, which makes it possible to accomplish the transformation from real-time data to global maintenance decisions. On the other hand, different stakeholders are linked with different activities (Fig. 3.7). The responsibility of stakeholders is clarified to some extent. So, it can reduce the occurrence on the issues of passing the buck when problems come.

However, this model did not take the complexity and difficulty of fusing different IT systems into account. For example, different IT systems may have different naming rules. The semantic consistency of IT systems should be investigated in the future work. Some information may be stored in a paper-based way. Security information has not yet been analysed.

3.6 Summary

Aiming at improving operational support for aircraft maintenance, an integrated approach that fuses aircraft’s condition, strategy, planning and cost has been proposed. First of all, an autonomous system analysis framework is provided to illustrate the approach from the global point

of view. Then, a high-level architecture for the autonomous system is developed to put forward the fundamental components and the relationship between them. The black box approach is employed to derive the underlying inputs and outputs of each component with respect to the component functionality. The UML use case diagram is used to illustrate the system module interaction. This helps deal with the complexity of systems design and enables smoother code development.

In the following chapter, we will give a detailed illustration of implementing a simulation system demonstrator regarding the design of the system.

Chapter 4

The Implementation of the Simulation System Demonstrator

The design of the autonomous system of aircraft maintenance has been discussed in the previous chapter. The architecture model, the concept model of components and the system interaction model have been proposed, which gain an important insight into the system. In this chapter, we will build a concrete simulation system as a demonstrator to illustrate the feasibility of the system design. A simulation model will be provided. The model details such as component inputs, algorithms, and component outputs, will be developed. Finally, all the details of the simulation model will be synthesised and implemented in the simulator. The realness will be highly addressed in the simulation system. Manipulations of difference scenarios on the system will also be available.

4.1 Introduction

Implementing a real autonomous system for aircraft maintenance is definitely difficult because a number of information systems are hardly accessible and the evidence for convincing the operators to follow system instructions is rather limited. However, simulation systems enable us to gain valuable insights into future systems in an inexpensive way, especially when the costs, risks or logistics of manipulating the real system of interest are prohibitive.

ABMS (Agent-Based Modelling and Simulation) is a relatively new approach to modelling the dynamics of complex systems and complex adaptive systems (Macal and North 2005). It is generally employed when the complexity of the system being modelled is beyond what static models or other techniques can fully present (Helbing 2012). The complexity of the autonomous system for aircraft maintenance mainly originates from system uncertainty and dynamics. The uncertainty is reflected not only by random mechanisms but by unknown behaviours resulting from agents' interactions. For example, the occurrence of faults on components of aircraft is conformed to a certain probability. The bounds on these uncertainty issues and the implications of potential outcomes should be understood and evaluated. Agent-based simulations provide a flexible and useful mechanism to capture these uncertainties (Heppenstall et al. 2011; Monostori, Váncza, and Kumara 2006). In terms of system dynamics, maintenance tasks change over time, including maintenance sequences for aircraft and maintenance resources scheduling. Different environments will lead to different system behaviours. Agent-based decentralization takes this into account by letting each agent continuously coordinate its actions with other agents, instead of making this agent apply a behaviour prescribed at design-time (Moyaux, Chaib-Draa, and D'Amours 2006). Additionally, ABMS allows us to model real-world systems of interest in ways that beyond the capabilities of traditional modelling techniques, such as discrete event system or system dynamics (Siebers et al. 2010; Ali et al. 2018). Hence, ABMS is a better fit for realizing the autonomous system design.

In this chapter, the model of the ABSS is derived from component models. In the following, the development of the ABSS is discussed in detail, where the inputs, functions and outputs of each agent are provided.

4.2 The Model of the Agent-based Simulation System

In order to derive agents from component models, the relationship between components and agents are presented in Table 4.1. Combining component models and maintenance-related issues (flight, plane, equipment, etc.), eight agents have been created, which can be divided into two groups: basic agents and maintenance agents. The basic agents refer to the Flight, Plane and Equipment. The maintenance agents consist of the Main, CRA (Customer Requirement Agent), SSA (Service Strategy Agent), STA (Service Task Agent), SCA (Supply Chain Agent) and QCA (Quality Control Agent). Fig. 4.1 provides an architecture model of the agent-based simulation system. In this

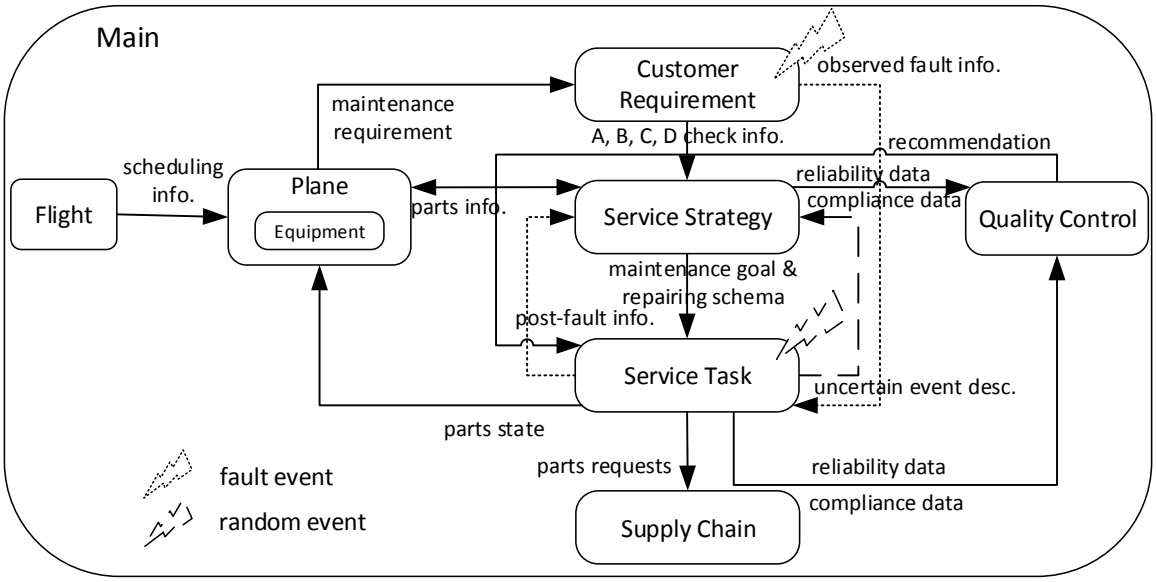

Figure 4.1: The architecture model of the agent-based simulation system

figure, different maintenance information is delivered to different agents. The details of agents are explained in the following.

Table 4.1: The relationship between components and agents

Component	**Agent**
Integration	Main
Maintenance requirement component	CRA
Maintenance strategy component	SSA
Maintenance scheduling planning component Cost and service level estimation component	STA, SCA
Safety and reliability estimation	QCA

4.3 The Development of Agents

The development of agents has been implemented as the following steps: the inputs of agents, the processes of agents, and the outputs of agents. All the relevant data about agents is summarized in Table 4.2.

4.3.1 Main and Customer Requirement Agent

Main is designed as the integrator. It not only links with all the other agents but displays the key information of maintenance. This agent provides a graphical display of flight routes based on a

Table 4.2: The data of agents - DT (Data Type), I (Input data), and O (Output data)

Agent	DT	Functionality	Data	
Flight	I	scheduling	flightNumber:String	aircraftNumber:int
			flightTime:double	arrivalTime:Date
			destination:String	departureTime:Date
			airport:String	interval:double
	O	scheduling	scheduling table	cancelledFlights:int
			delayedFlights:int	ontimeFlights:int
		states	CRA:int;SSA:int;STA:int	SCA:int;QC:int;type:String
Plane	I	planes	aircraftNumber:int	airport:String
			aircraftType:AirPlaneType	
		flights	flightNumber:String	flightTime:double
			destination:String	
	O	maintenance	maintenanceType:String	severity:int
			parts:ArrayList<Equipment>	
Equipment	I	equipment	type:EquipmentType	changeoverTime:double
			serialNumber:int	repairTime:double
			repairingTime:double	PMFrequency:int
			maintenanceStrategy:String	
			equipmentState:EquipmentState	
	O	maintenance	repairType:String	countRepair:int
			timeForFirstRepair:double	currentUseTime:double
			repairingTime:double	
CustomerRequ.	I/O	parts	parts:ArrayList<Equipment>	
		fault info.	faultDescription:String	faultLocation: String
ServiceStrategy	I	planes	aircraftNumber:int	airport:String
			aircraftType:AirPlaneType	interval:double
			parts:ArrayList<Equipment>	
	O	parts	partCheckingList:ArrayList<Equipment>	
		time	faultAnalysisTime:double	
			exceptionalEventSSATime:double	
		strategy	maintenanceStrategy:String	
ServiceTask	I	planes	aircraftNumber:int	airport:String
			aircraftType:AirplaneType	interval:double
			partCheckingList:ArrayList<Equipment>	
		technicians	aircraftTechnicians:ArrayList<Person>	
	O	cost	laborCost:double	partCost:double
			equipmentCost:double	downtimeCost:double
		time	planningTime:double	executionTime:double
			downtime:double	faultIdentificationTime:double
			exceptionalEventSTATime:double	
SupplyChain	I	planes	aircraftNumber:int	
		parts	partBuying:ArrayList<Equipment>	
	O	time	partPreparingTime:double	
		parts	partBought:ArrayList<Equipment>	
QualityControl	I	repairing	timeForFirstRepair:double	countRepair:int
			repairingTime:double	currentUseTime:double
			partCheckingList:ArrayList<Equipment>	
	O	reliability	MTTR:double	MTTF:double
			MTBF:double	Recommendation:String

GIS (Geographic Information System) map, where cities are really located and the flying time can be easily managed in this map. The information on real-time flights scheduling table is stored in the database. In addition, maintenance states of each plane can be observed from this agent, which enables managers to control maintenance actions from the global point of view. Since this agent is the basis for the other agents, there is no input and output for it.

The maintenance sequence optimization is another key issue for Main. In our simulation environment, it is possible for more than two planes to be maintained at the same airport. The *FIFO* (First In First Out) rule is often applied in flight control (Atdelzater, Atkins, and Shin 2000). However, only using the *FIFO* rule to deal with waiting planes cannot be always a good strategy, especially when the high efficiency of maintenance is exceptionally demanded. In fact, this agent allows maintenance designers to investigate the performance of different strategies. For example, some other strategies like *FIFO-Optimized* strategy for waiting planes can be proposed. On the one hand, the least important PM for planes may be delayed to the next destination; on the other hand, the rest of maintenance will be scheduled with the *FIFO-Optimized* strategy. The *FIFO* rule is utilized to avoid the starvation of processes. Optimization operations are based on the *FIFO* rule. Each time when the temp message list is empty, it will copy all the messages from the message list. At the same time, the elements in the message list should be freed. It should be noted that the temp message list is used for optimizing the maintenance sequence for waiting planes and the message list is just employed to collect the messages.

The *FIFO-Optimized* strategy has been implemented in Algorithm 1. The time complexity of this algorithm is $O(n^3)$. In this algorithm, *synthesize* is a function of setting the priority for aircraft. *severity* denotes the delayed times of PM of aircraft. The value of *severity* belongs to [0,2]. In other words, the maximum delayed time (the maintenance should be delayed to the next airport) is two. *resource* refers to the availability of the labours at related airports. The type of *resource* is a Class, which contains attributes such as amount, availability, time, owner, etc. The variable of $MaintenanceType$ is added to synthesize the two aspects mentioned, which is classified into two types: *preventive* and *corrective*. Some key-values are provided in Table 4.3. The priority is defined by a synthesis of $severity$, $MaintenanceType$, $interval$. The evaluation process is illustrated as follows:

1. Define the maximum numbers of waiting aircraft for airport, threshold;
2. Get the real-time value of $severity$, $MaintenanceType$, $labor$ and $interval$ for each related aircraft;
3. Transform the real time of variables to evaluation values by Table 4.3;
4. Calculating the $EvaluationValue$ for each related aircraft by getting the sum of $severity$ and $MaintenanceType$;

5. Get the index of the aircraft with the minimum interval of the maximum $EvaluationValue$;
6. If $labor$ equals $True$, schedule the aircraft of the index;
7. Else wait until $labor$ equals $True$;
8. If the number of aircraft is more than threshold,
    - If minimum of $EvaluationValue$ equals 15 (the maximum of the sum of $severity$ and $MaintenanceType$) or no $MaintenanceType$ equals to preventive , no aircraft will be delayed;
    - Else delay the index of the aircraft with the minimum interval of the maximum $EvaluationValue$ and "$MaintenanceType = preventive$".

In this evaluation process, a few issues should be addressed. First of all, the design of the variable threshold aims to improve maintenance efficiency, i.e., the maintenance of less urgency can be delayed to the next airport. In terms of setting the key-values for variables, the objective is to more easily compare the severity of cases. For example, comparing with the cases of "$severity$ = 1 and $MaintenanceType$ = corrective" and "$severity$ = 2 and $MaintenanceType$ = preventive", the latter case will be given a higher priority. Because the value of $EvaluationValue$ for the latter case is bigger. Since the values of severity and $MaintenanceType$ are very limited, it is possible for different flights to have the same evaluation value. In this case, one of the minimum intervals will be chosen.

Table 4.3: The key-values for $severity$ and $MaintenanceType$

($severity$, value)	(0, 0)	(1, 1)	(2, 10)
($MaintenanceType$, value)	(preventive, 0)	(corrective, 5)	-

On top of that, the estimation of severity and resource and the delay of the least important maintenance are conducted with respect to airports. Because different airports are equipped with different capacities of the resource. This may lead to changing maintenance sequences significantly. Additionally, Algorithm 1 is different from Algorithm 3. The former focuses on scheduling faulty aircraft to repair (i.e. the sequence of "faulty" aircraft needed to be repaired); the latter concentrates on programming maintenance resources for maintenance tasks (i.e. the sequence of maintenance tasks needed to be performed).

CRA sends maintenance requests to relevant agents after receiving signals from Main. If it is a PM, checking information will be sent to SSA. If it is a CM, fault information will be delivered to STA. As this agent principally manages requests and triggers maintenance activities, the input and output are the same.

Algorithm 1: Maintenance sequence optimization (part 1)

```
input  : airportsList, messagesList, tempMessagesList
output : index.
// The index represents the number of the scheduled aircraft to be
   maintained.
// Firstly, the FIFO rule is used to deal with messages.
int threshold, count,index1; double max, value, min, severity, resource, interval;
max = min = severity = resource = value= count = index1 = interval = 0;
String maintenanceType = null;
// The threshold indicates the maximum number of aircraft to be maintained
   each time.
threshold=5;
if Main.receiveMessagesFromPlanes == true then
    messagesList.add(msg);
end
if tempMessagesList.IsEmpty() and !messagesList.IsEmpty() then
    for i = 0 to messagesList.size() do
        tempMessagesList.add(messagesList.get(i));
    end
    messagesList.clear();
end
// to be continued.
```

4.3.2 Flight

Flight is a controller of scheduling planes. It schedules both on-time and delayed flights. The information about flights is stored in the database.

The scheduling strategy is implemented by two events: *scheduling* and *schedulingDynamical*. The *scheduling* event is a cyclic event where the first occurrence time is *departureTime* and the recurrence time is one day. The *schedulingDynamical* is a dynamic event. The occurrence of this event is totally determined by users by indicating the occurrence time. The scheduling strategy is described as the flowchart shown in Fig. 4.2. As a result, the final state of one flight will be scheduled, delayed or cancelled. Since the scheduled information is determined, the “chain reaction" can happen due to some delays of flights. This phenomenon can significantly increase the complexity of maintenance and clearly shows the high-level relevance of real-world maintenance issues of the simulation system. This agent will deliver the scheduling information and maintenance state information to Plane and Main respectively.

```
   // Maintenance sequence optimization (part 2)
   // Then, based on the FIFO rule, prioritize maintenance sequences.
1  for j = 0 to airports.size() do
2      max =0;
3      for k = 0 to tempMessagesList.size() do
4          if tempMessagesList.get(k).equals(airports.get(j)) then
5              severity =tempMessagesList.get(k).severity;
6              interval =tempMessagesList.get(k).interval;
7              maintenanceType = tempMessagesList.get(k).maintenanceType;
8              resource = resourceEstimate(airports.get(j));
9              value = synthesize(severity, resource, interval, maintenanceType);
10             if max < value then
11                 max = value; index = k;
12             end
13             count++;
14         end
           // The least important PM may be delayed to the next destination.
           // min(messagesList,airport) means getting the minimum value of
              aircraft number from messages list at airport j
15         while (count−−) > threshold do
16             for l = 0; tempMessagesList.get(l)==airports.get(j) to tempMessagesList.size() do
17                 if tempMessagesList.get(l).maintenanceType=="Corrective" or
                    tempMessagesList.get(l).severity==2 then
18                     break;
19                 end
20                 index1=min(tempMessagesList,airport.get(j));
21                 tempMessagesList.get(index1).severity++;
22                 tempMessagesList.remove(tempMessagesList.get(index1));
23             end
24         end
25         count=0;
26         send(“Scheduling",tempMessagesList.get(index));
27         tempMessagesList.remove(tempMessagesList.get(index));
28     end
29 end
```

4.3.3 Plane and Equipment

Plane is the major carrier for maintenance. It is a composite agent, which contains a number of Equipment. When all the aircraft are loaded from the database, they will be ready at their corresponding airports. As soon as receiving the messages from Flight, they will fly to their destinations. When they reach their corresponding destinations, the maintenance will take place with a certain probability. According to the report of air transportation organization of Canada from 2007 to 2016 (TSBC 2019), landing can be considered as the period when most of the incidents happen.

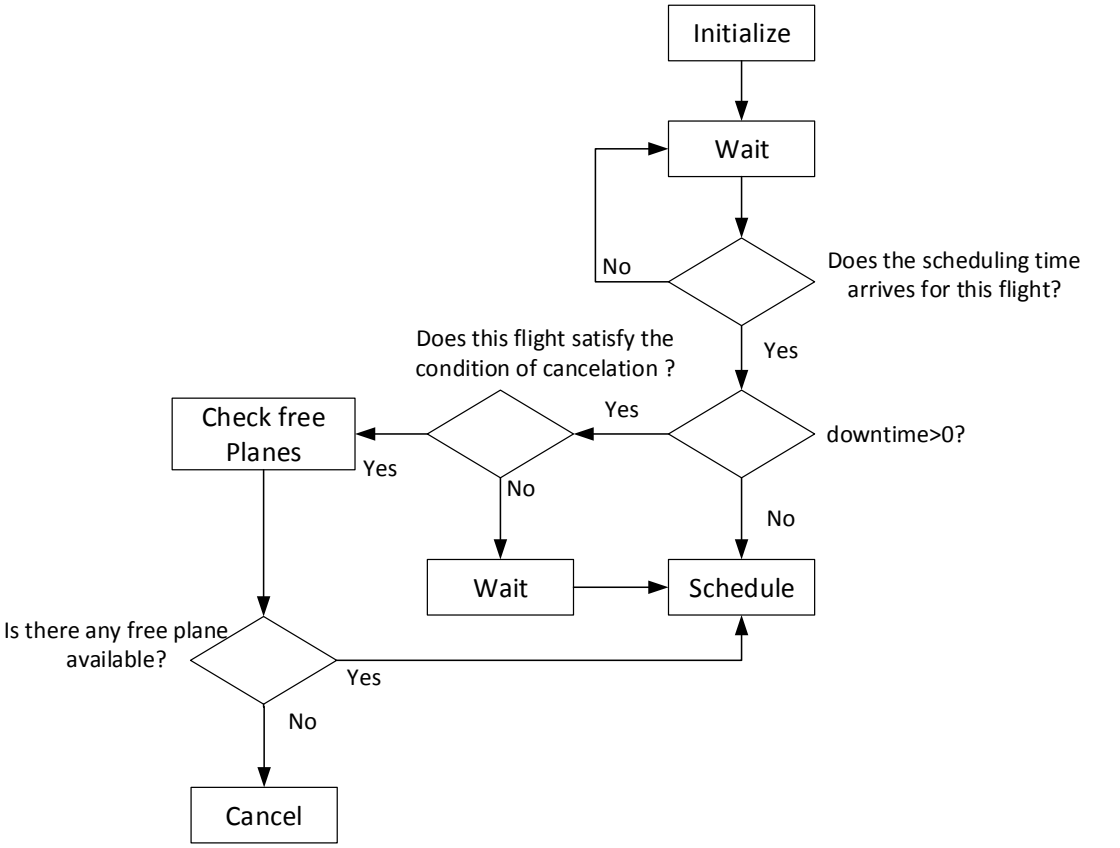

Figure 4.2: The flowchart of flight scheduling strategy

Therefore, in our simulation, the maintenance is assumed to be started from the landing period. After the reparation, they are programmed to fly to next destinations. This cycle repeats every day.

Fig. 4.3 shows equipment states regarding the location of equipment. Failure events transfer the equipment state from *Use* to *Failure*. The state transition from *Use* to *Inspecting* is triggered by PM. If maintenance actions can be done on the plane, the state will pass from *Repairing* to *Use*. On the other hand, the equipment will be repaired in the workshop. In addition, the stock will deliver any available equipment needed. It should be noted that *Repairing* and *Repair* states depict that maintenance activities take place on the spot and at workshops respectively.

During the life-cycle of equipment, recording the current use time of equipment is very important. This time will be the key factor to make a repairing decision. In this simulation environment, as soon as the equipment is loaded from the database, the current use time of this equipment starts. Only when it is placed in the stock, the current use time stops. The definition of *currentUseTime* is

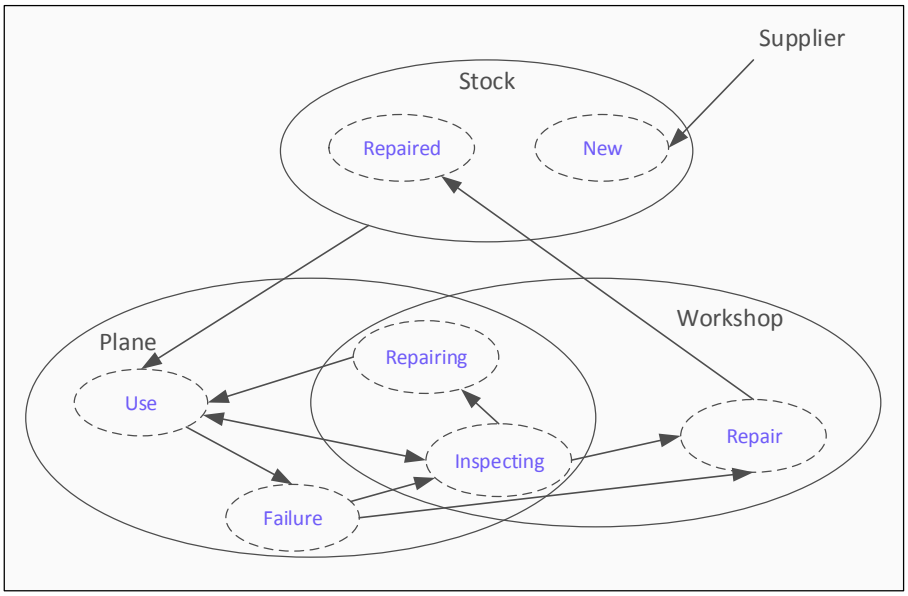

Figure 4.3: The states transition of equipment

illustrated as follows:

$$CUT_a = CT - ST; \tag{4.1}$$

$$CUT_b = CT - ST_i + LUT_{i-1}; \tag{4.2}$$

$$LUT_{i-1} = ET_{i-1} - ST_{i-1} + LUT_{i-2} \tag{4.3}$$

where, $i >= 1$; $LUT_0 = 0$; CUT_b and CUT_a mean parts have been stayed at the stock or not respectively; the indices of equations are shown as follows:

CT (Current Time) the current simulation running time;

CUT (Current Use Time) the current use time of equipment;

ST (Start Time) the time when equipment is leaving from stocks;

ET (End Time) the time when equipment is put into stocks;

LUT (Last Use Time) the last use time of equipment.

Figure 4.4 gives two examples about how to calculate the value of *currentUseTime*. In Fig. 4.4 (a), CUT is equal to t, as repairing time does not influence *currentUseTime* of equipment. In Fig. 4.4 (b), CUT equals $t - ST_i + ET_{i-1} - ST_{i-1} + LUT_{i-2}$.

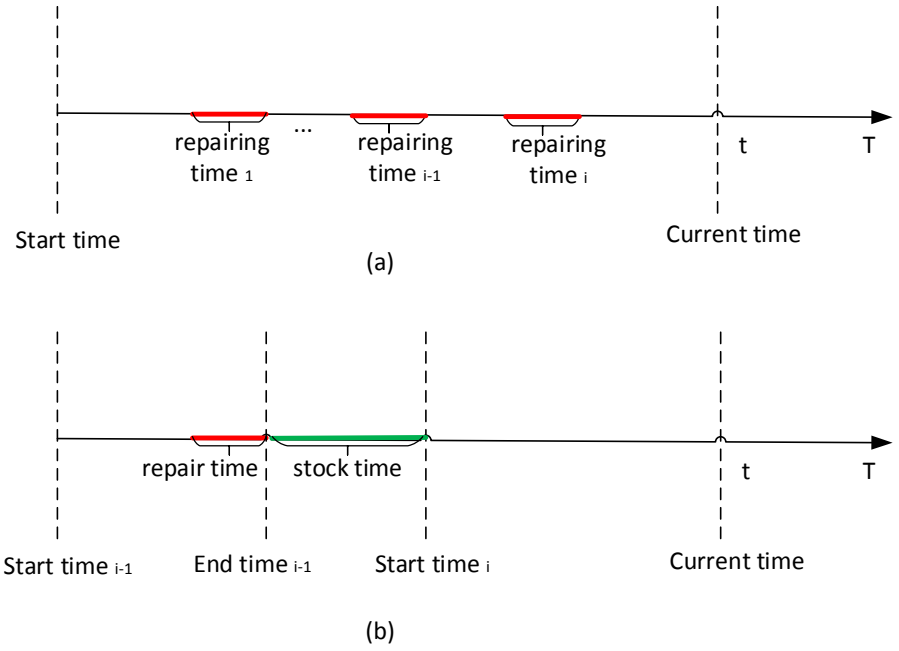

Figure 4.4: The method of calculating *currentUseTime*

The analysis of repairing strategies is realized in Algorithm 2. This algorithm deals with PM and CM. The repairing strategies include “replace", “replace& repair", “repair" and “inspect". The first three strategies are associated with CM. The “inspect" is used for PM. This algorithm is designed for comparing the *currentUseTime* and *changeoverTime* of equipment with respect to *equipmentState*, in order to find the optimal repairing strategy. In addition, the time complexity of this algorithm is $O(1)$ because no loop is involved.

4.3.4 Service Strategy Agent

SSA is intended for dynamically determining maintenance strategies for equipment and delivering maintenance goals and repair schema. PM and CM are analyzed in this agent. If the maintenance type is PM, maintenance actions will be carried out on PM due parts. If the maintenance type is CM, this agent will automatically generate faulty part lists. The selection of parts takes place in a random way. On the other hand, we consider the influence of maintenance strategies on the same equipment. If PM occurs at equipment i, the probability of the fault occurrence at this equipment will be decreased. If CM happens at equipment i, the PM of this equipment should be rescheduled. Based on this idea, the approach of maintenance strategy updating is provided in Fig. 4.5.

Algorithm 2: Repairing strategy analysis

```
input  : lastUseTime, changeoverTime
output : repairType
if this.endTime == 0 then //  this condition means this equipment has never been in stock.
  The time function gets the real time of simulation environment.
    currentUseTime = time(MINUTE);
end
else
    currentUseTime = lastUseTime + time(MINUTE) - StartTime;
end
if currentUseTime > changeoverTime then
    repairType = “replace";
end
else
    if equipmentState is Broken and repairingTime > interval then
        if Inventory is enough then
            repairType = “replace& repair";
        end
        else
            repairType = “repair";
        end
    end
    else if equipmentState is Broken and repairingTime <= interval then
        if changeoverTime - currentUseTime <= 1200 then //  the useful remaining life is less
          than 20h
            repairType = “replace";
        end
        else
            repairType = “repair";
        end
    end
    else //  The maintenance type is PM
        repairType = “inspect";
    end
end
```

4.3.5 Service Task Agent

This agent is dedicated to programming maintenance tasks. The maintenance planning and task scheduling are based on the estimation of resource availability and real-time condition.

Algorithm 3 employs the same strategy as Algorithm 1. The time complexity of Algorithm 3 is $O(n)$. The major differences between them are the maintenance location and key indicators. In Algorithm 1, maintenance objects (planes) are from airports all around the world and key indicators are relatively simple. While in Algorithm 3, maintenance objects are at the same airport. As

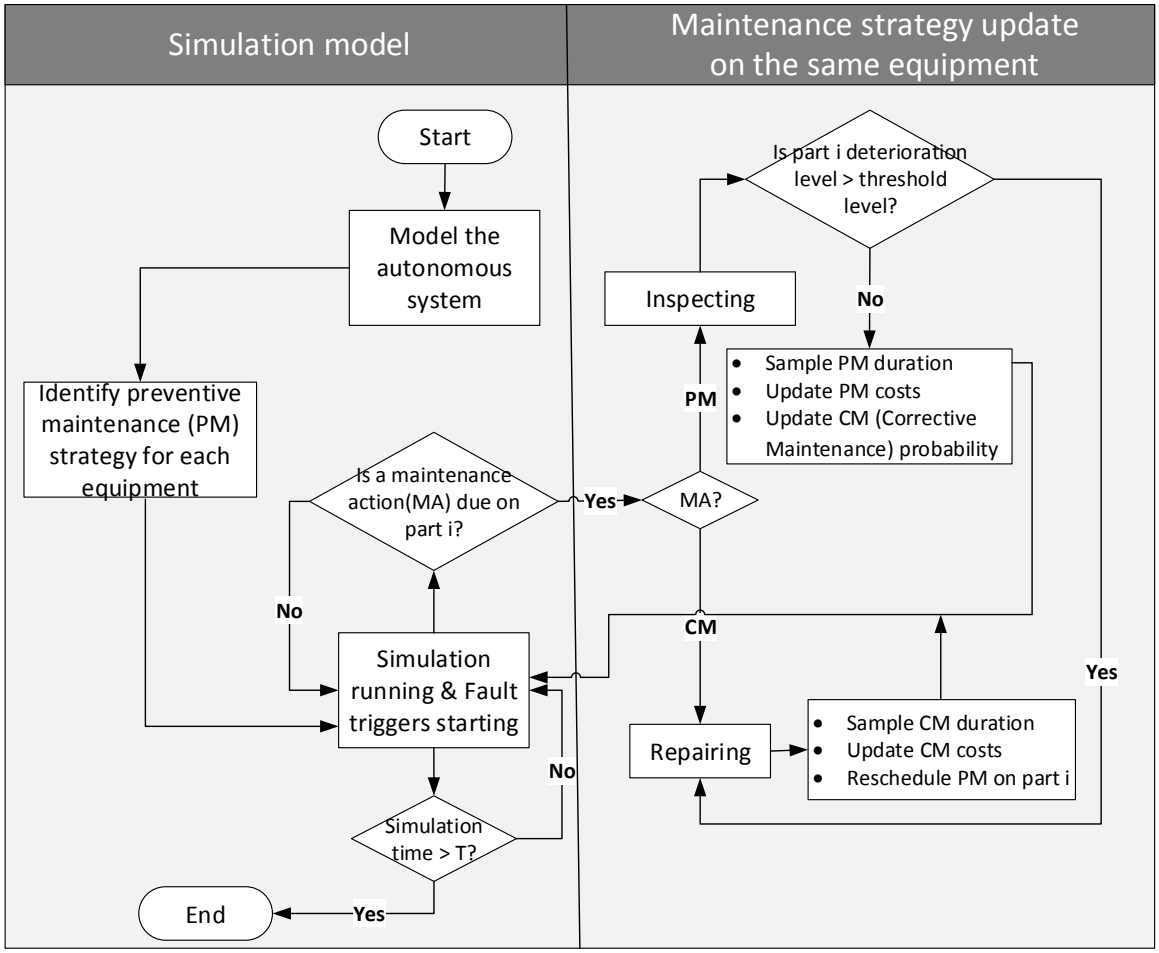

Figure 4.5: An approach of the maintenance strategy updating on the same equipment

the most of details of maintenance tasks have been determined, the maintenance priority can be estimated by multiple indicators.

4.3.6 Supply Chain Agent

SCA addresses how to supply enough parts or components for repairing aircraft. This agent will go through the inventory to check the availability of parts, once they are being requested. Parts in the inventory consist of new parts and returned parts (repaired parts). Therefore, these two kinds of parts are both possible to be used in repairing. As such, one part can serve for different aircraft. Besides, the current use time of this part will continue, starting from the last time of being time, which is mentioned in Fig. 4.4. New parts will be purchased if they are out of stock.

The process for purchasing parts is simulated by dynamic events. Random time will be given

Algorithm 3: Maintenance tasks sequence optimization

```
input  : messagesList, tempMessagesList, aircraftTechniciansList, constraint[], interval.
// The index represents the number of scheduled aircraft to be maintained.
output : index.
// The FIFO rule in Algorithm 1 is reused to deal with messages from SSA. The
   details can be seen at lines 4 - 13, part 1 of Algorithm 1.
// Under the FIFO rule, prioritize the maintenance task sequence.
// XXEstimate(partsList) means synthesis estimation of parts, according to
   the service level, risk, delay, etc., based on historic data and real-time
   data.
1  double max, value, risk, delay, resource, serviceLevel;
2  max = value = serviceLevel = risk = delay = resource =0;
   // The city means the airport of current aircraft.
3  String city;
4  for i = 0 to tempMessagesList.size() do
5      partsList=tempMessagesList.get(i).partsCheckingList;
6      risk = riskEstimate(partsList);
7      serviceLevel = serviceLevelEstimate(partsList);
8      delay = delayEstimate(partsList);
9      availability = resourceEstimate(partsList);
10     value = synthesize(serviceLevel, risk, delay, availability);
11     if max < value then
12         max=value;
13         index = i;
14     end
15 end
   // Assign resource to the selected plane's parts list
16 ArrayList<Person> workingTechnicians =new ArrayList<Person>();
17 city = tempMessagesList.get(index).city;
18 for j = 0 to aircraftTechnicians.size() do
19     Person p= aircraftTechnicians.get(j);
20     if p.airport.equals(city) and p.schedule == 1 and p.free == true then
21         p.free = false;
22         workingTechnicians.add(p);
23     end
24 end
25 tempMessagesList.remove(tempMessagesList.get(index));
```

to indicate how much it will take to obtain parts. So, the input of this agent is the information of aircraft and the parts needed. Its output is the time for purchasing parts and the set of parts bought.

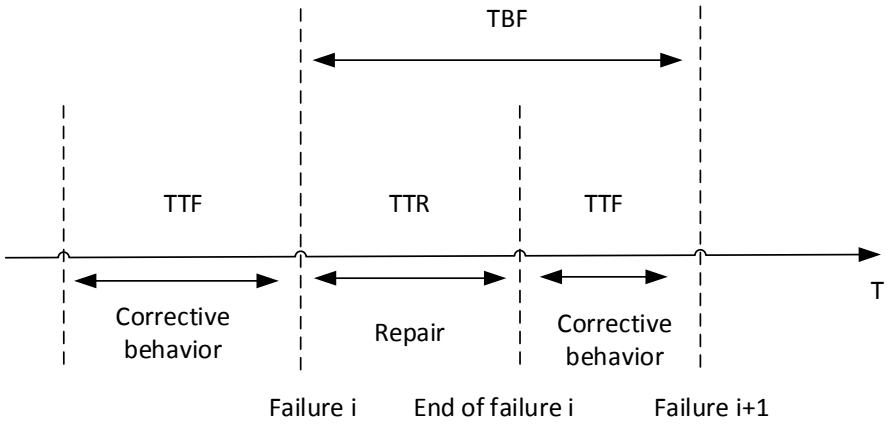

Figure 4.6: The relationship between MTBF, MTTR and MTTF

4.3.7 Quality Control Agent

QCA analyzes the reliability of equipment and checks safety issues depending on AD and SB. Kullstam (1981) proposed a few reliability indicators like MTTF (Mean Time To Failure), MTBF (Mean Time Between Failure) and MTTR (Mean Time To Repair) . MTTF is a general reliability index of estimating mean time to failure. MTTR is a basic measure of the maintainability of repairable items. It represents the average time required to repair a failed component or device. MTBF is the predicted elapsed time between inherent failures of a mechanical or electronic system, during normal system operation. The general definitions of these indicators are illustrated as follows:

$$MTBF = \sum_{(i=2)}^{n} (T_{Failure_i} - T_{Failure_{i-1}})/n - 1; \tag{4.4}$$

$$MTTR = \sum_{(i=2)}^{n} (T_{EFailure_{i-1}} - T_{Failure_{i-1}})/n - 1; \tag{4.5}$$

$$MTTF = \sum_{(i=1)}^{n} (T_{Failure_i} - T_{EFailure_{i-1}})/n - 1; \tag{4.6}$$

where, $T_{EFailure_i}$ implies the time of the end of failure i. In Eq. 4.6, $T_{EFailure_0} = 0$. In addition, NFF (Not Fault Found) rate will be considered as well. It should be noted that NFF means that equipment sends the faulty signal, but after inspecting it, no fault is found. During the process of simulation, all the relevant information will be collected to conduct the reliability analysis. The results are compared with target repair time, target MTBF, and target MTTR respectively, in order to get the correspondent performance. In the following, the process of generation of recommendations is analyzed in details. The performance of MTBF means the performance of the component.

Therefore, combing the performance of MTBF and the real failure causes, some recommendations about changing handling procedures can be proposed. The performance of the NFF rate reflects the real situation of the component quality. When we compare the performance of the NFF rate with an arising pattern, some recommendations about developing a new component or update it can be proposed. For the performance of MTTR, it shows the efficiency of the reparation. Thus, associating the performance of the MTTR with maintainer's ID, some recommendations about strengthening the training can be proposed. At last, linking with the performance of maintenance frequency with training sequence and content, some recommendations about changing the maintenance sequence can be proposed. Furthermore, once the feasibility of these recommendations is demonstrated, an improvement of the maintenance activities can be implemented.

Safety analysis is achieved by utilizing AD and SB. The maintenance management process and repair management process are supervised. Once these processes are violated, a warning will be sent.

Above all, the input data of this agent is the working period, the time of failure, and the repairing period for each equipment. The output data of this agent is the results of MTTR, MTBF, MTTR and safety.

4.3.8 Uncertain Analysis for Aircraft Maintenance

The uncertain analysis is a big and inevitable issue for aircraft maintenance. It is very common to see uncertain events occurring during the process of aircraft maintenance. Uncertain events make operational support for aircraft more complicated and difficult. In our simulation system, we take uncertain issues into consideration. Song et al. (2018) addressed uncertainty issues in resource-constrained multi-project scheduling for an ABSS. Here, "project", in this paper, means an ensemble of tasks. They suggest uncertain events often happen during the execution of projects. Two types of uncertainty are involved, including workability uncertainty and sub-task failure. Our consideration of uncertainty is derived from Song's categorisation because the above-mentioned aspects are involved in our case. Besides, the uncertainty on the ability to tackle fault events in terms of maintenance organisations is considered as well, since the knowledge of maintenance organisations can improve maintenance efficiency. Different types of uncertainty have been achieved in our simulation system, which are shown in the following:

Workability uncertainty How to choose the "faulty" aircraft (scheduled maintenance is due) to be maintained relies on the real situations of aircraft itself and the corresponding airport. The "faulty" aircraft can be delayed, if its severity is not high enough. The availability of resource at the airport also affects the final decision. Thus, the order on whether the maintenance for aircraft should be delayed to the next airport or not is uncertain. This principle can be also adapted to determining the sequence for scheduling aircraft to be maintained.

Resource uncertainty Resource uncertainty refers to technicians and the stock of aircraft components. The assignment of technicians varies from time to time in 24 hours. The decision on the operation of faulty components is dependent on their real situations. Different operations such as replacing, repairing, and waiting, etc., will cause the level of stock changed.

Problem-solving capability Technicians can encounter all kinds of uncertain events. For example, some tools are missing. Or some situations never happened before. These uncertain events have an adverse influence not only on maintenance efficiency but on feelings of technicians (Tsagkas, Nathanael, and Marmaras 2014). This is because technicians will run into difficult situations: The tasks cannot be finished on time if they seek ideas from managers; The tasks may not be delayed if they make their own decisions but it will burden technicians with additional worries. Maintenance tasks can be severely impeded due to the inefficiency of the management of maintenance organisation, which motivates us to propose the process for dealing with uncertain events (Fig. 4.7). The major idea is to precisely define maintenance activities for stakeholders and make full use of information systems. It should be noted that “P solution" in this figure means possible solutions.

4.3.9 Cost and Service Level Analysis

In terms of Maintenance Cost (MC), Downtime Cost (DC), Labor Cost (LC), Stock Cost (SC) and Parts Purchase Cost (PPC) are taken into consideration. The definitions of each cost (Eqs. 4.7 - 4.11) are provided, where *totalRepairTime* (shown in Table 4.4) and *interval* are counted by seconds, and $unitPrice_{DC}$ means the unit price for downtime; *salary(i)* and *time(i)* represent the *i*th aircraft technician's salary counted by hours and the corresponding working time; *number(i)* implies the number of the *i*th kind of parts, $unitPrice_{SC}$ means the unit price for stock and *days* means the number of storing days; and *cost(j)* means the cost of the *j*th kind of parts. In addition, all the costs are counted by dollars.

$$MC = DC + LC + SC + PPC; \tag{4.7}$$

$$DC = (totalRepairTime - interval) * unitPrice_{DC}; \tag{4.8}$$

$$LC = \sum_{(i=1)}^{n} salary(i) * time(i); \tag{4.9}$$

$$SC = \sum_{(i=1)}^{n} number(i) * unitPrice_{SC} * days; \tag{4.10}$$

$$PPC = \sum_{(j=1)}^{n} cost(j); \tag{4.11}$$

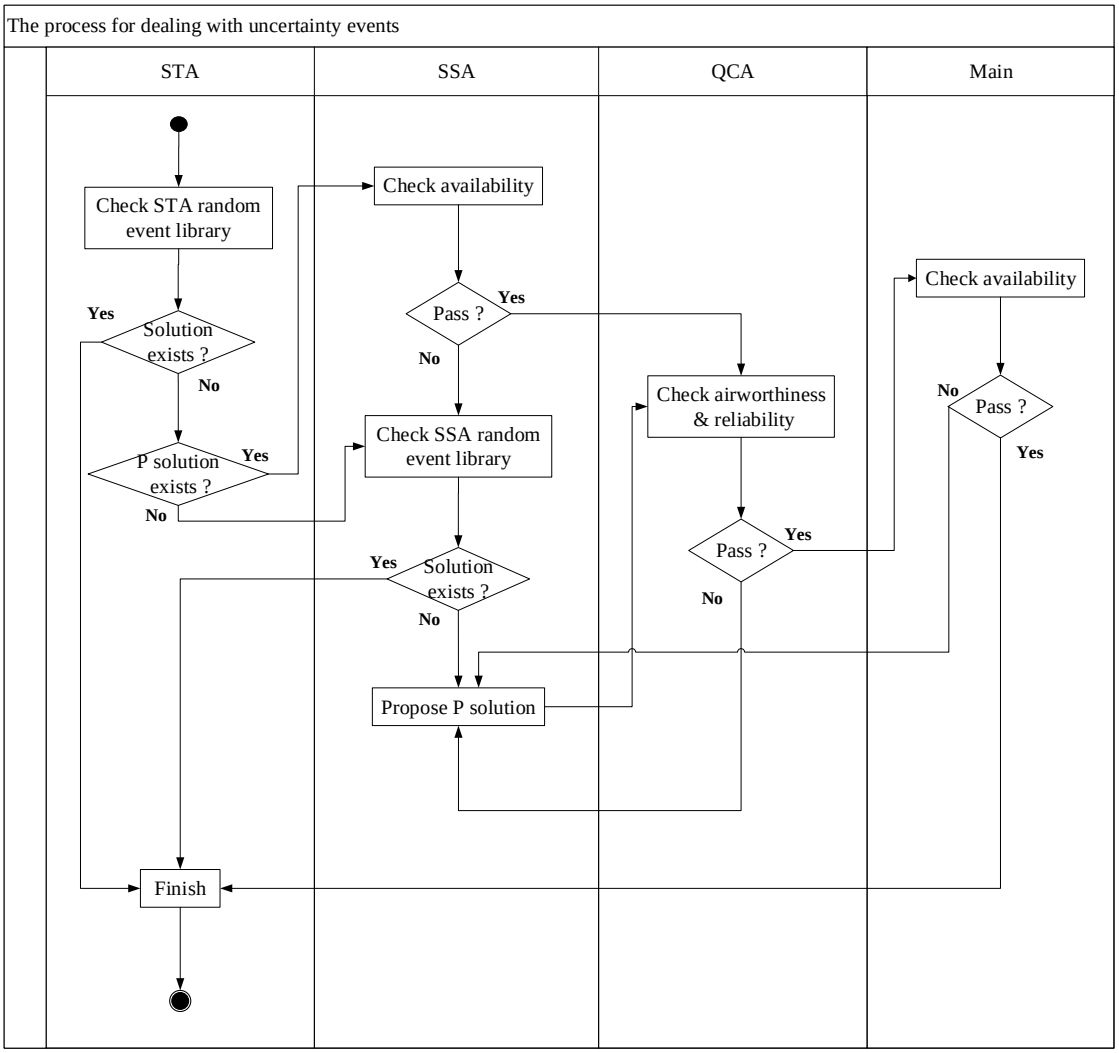

Figure 4.7: The process for dealing with uncertain events

The complexity of MC depends on maintenance scenarios. As shown in Table 4.4, fault events and uncertain events are considered in maintenance scenarios. The analysis of fault events aims to deliver the causes of faults on components. The uncertain event analysis tries to simulate the process of the trouble-shooting during the process of repairing when exceptional issues happen. Fault and uncertain events take place with a user-defined probability. The elements of *totalRepairTime* will be discussed in detail at the last paragraph of Section 5.6.1.1. Additionally, *scheduledTime* in Table 4.4 means the total repair time of PM.

Table 4.4: The compositions of total repair time on maintenance scenarios - UE (Uncertain Events)

Scenario	Total repair time	
PM	informationDelivery, maintenancePlanning	inventoryPrepare, repairing
PM+UE	**exceptionalEventSTA(eSTA)**	**exceptionalEventSSA(eSSA)**, scheduledTime
CM	**faultAnalysis**, scheduledTime	**faultIdentification**
CM+UE	**eSTA**, **eSSA**, scheduledTime	**faultIdentification, faultAnalysis**

The definition of service level is shown as follow:

$$ServiceLevel = totalRepairTime - interval. \tag{4.12}$$

Service level reflects the degree of the delay time of passengers. It is another way to evaluate maintenance efficiency. Obviously, decreasing the maintenance cost all the time is not definitely a good idea. Because it may lead to numerous delays or even unnecessary cancels of flights. In this paper, when the difference between *totalRepairTime* and *interval* is more than 4 hours for one flight, it is supposed that this flight will be cancelled.

4.4 Discussion

The simulation system is the demonstrator to simulate the whole process for aircraft maintenance. Many factors, including strategies, scheduling plans, the cost analysis and the service level analysis, have been addressed in this system.

The analysis of strategies consists of the maintenance sequence optimization strategy (Algorithm 1), the repairing strategy (Algorithm 2), and the maintenance task sequence optimization strategy (Algorithm 3). These strategies ameliorate the efficiency of aircraft maintenance. The maintenance sequence optimization strategy states that the requests from aircraft should be answered in an optimized way. From the experiment results (Section 5.6.3), we can see that the judge of the best strategy relies on the chosen priority and the synthesis of real-time conditions. The strategies of repairing and maintenance task sequence have considered the real-time conditions as well. As for the updating of maintenance strategy, the system dynamically updates the maintenance strategy based on the real-time condition. Alrabghi and Tiwari (2016) provided the interactions among maintenance strategies (shown in Table 4.5), which illustrates how the actions of a given maintenance strategy might affect components in the system in different ways assuming the probability of occurrence of all failure modes does not change. This table enables us to model the complex maintenance strategies for the autonomous system.

Table 4.5: The interactions amongst maintenance strategies

	Might affect other maintenance strategies on the same asset?	Might affect other maintenance strategies on other assets?
Corrective Maintenance	No	Yes
Preventive Maintenance	Yes	No
Opportunity Maintenance	No	No
Condition-Based Maintenance	Yes	No

The planning and scheduling task scheduler uses Algorithm 3 to enable the allocation of maintenance jobs to groups of MRO workers, providing cost-effective plans for dealing with maintenance issues in a simple and timely way.

The cost analysis uses a simple cost-breakdown structure (Eqs. (4.7) to (4.11)) to estimate the maintenance cost; but because of the complexity of the effect of repair time (Table 4.4), multiple values are possible for any given task.

The analysis of service level is another way to evaluate maintenance efficiency. Even though the definition of service level is quite simple, it could make the analysis of maintenance much more complicated. The concept of service level allows aircraft to be delayed or cancelled. Since flying routes of aircraft have already planned, it may cause the “chain reaction". For instance, as the resource of one airport is not well evaluated, it may lead to the resource competition between CM and loose PM. In other words, the loose PM could be delayed to some other airports (details see Algorithm 1). Consequently, more aircraft will have to wait for resources. It will then cause the delay of flights. Hence, the “chain reaction" will happen between Flight and Plane. So, a delay of an aircraft may cause multiple delays of other aircraft. The real-time condition varies from time to time. The planned scheduling could be useless. As a result, quick responses to real-time conditions become vitally important to improve maintenance efficiency.

As above, our simulation system is a successful proof of demonstrating the design of the autonomous system for aircraft maintenance.

4.5 Summary

In order to illustrate the feasibility of the design of the autonomous system, a solid simulation system demonstrator has been accomplished by using *Anylogic*. The autonomous system is implemented in the way of the agent-based simulation system where components are instantiated as agents. The strategies, the planning and scheduling, the cost analysis and service level analysis have also been achieved in the simulation system.

Chapter 5

The Formal Verification of the Autonomous System

The autonomous system has been accomplished in previous chapters. The analysis of the system could be carried on, in order to investigate the performance of the system. However, a system without the logic checking can neither convince users nor assure its correctness in the long run, especially a system applying in the security-critical domain. As a result, we will conduct the research on the formal verification of the autonomous system.

Model checking is an effective way to verify the behaviours of an agent-based simulation system. Three behaviours are analysed: operational, control, and global behaviours. Global behaviours of a system emerge from operational behaviours of local components regulated by control behaviours of the system. The previous works principally focus on verifying the system from the operational point of view (operational behaviour). The satisfactory of the global behaviour of the system has not been investigated regarding the control behaviour. Thus, in this chapter, we propose a more complete approach for verifying both global behaviours and operational behaviours of systems. To do so, firstly these three behaviours are formalized by automata-based techniques. The meta-transformation between automata theories and Kripke structure is then provided, in order to illustrate the feasibility for the model transformation between the agent-based simulation model and Kripke structure-based model. In the following, a mapping between the models is proposed. Subsequently, the global behaviour of the system is verified by the properties extracted from the control behaviour and the operational behaviour is checked by general system performance properties (e.g. safety, deadlock freedom). Finally, two case studies on the simulation system for aircraft maintenance have been carried out. A counter-example of signals sending between Flight agent and Plane agent has been proposed by NuSMV model checker in case one. Modifications for the NuSMV model and agent-based simulation model have been performed. The experiment results

show that 47.37% of cancelled flights have been changed to be serviceable. In terms of the second one, the simulation experiment analyzing the impact of maintenance sequence strategies on maintenance costs and service level has demonstrated the feasibility of the simulation system.

5.1 Introduction

The autonomous system is built based on SE and ABMS, which is finally implemented as an ABSS. Since the basic elements of the ABSS consist of agents, agent relationships, and agents' environment, the modelling of the ABSS is not rather difficult if requirements are well investigated. The modelling process involves the building of individual agents, the definition of the interactions between agents, and the design of the agents' environment. Agents are built individually, the interactions with neighbouring agents are usually clear. However, the behaviours of the ABSS rely not only on the programs inside the agents but also on the agent interactions, which makes it difficult to explore the impacts of the individual agent behaviour on the ABSS especially when the number of agents increases.

Formal verification is a powerful tool to demonstrate the correctness of system behaviours. In this thesis, system behaviours are divided into global behaviours and operational behaviours. Many efforts have been made to this area. However, most of the researches focus on how to extend the formal logic to support the description of the more complex system (Al-Saqqar et al. 2015; Raimondi 2006). Some other works contribute to the subject of model transformation (El Menshawy et al. 2018; Keshanchi, Souri, and Navimipour 2017). Few studies have given attention to the system itself. Control behaviours are often involved when we analyze a system. They are application-independent, which limit operational behaviours (application-dependent) from the abstract level. Understanding the conformity of global behaviours to control behaviours is another effective way to verify whether the system design is satisfactory (Bentahar et al. 2013). Thus, this chapter focuses on how to better verify the behaviours of the ABSS with only the existing formal logics.

Different behaviours are checked in different ways. The global behaviour is extracted from the ABSS, which describes the behaviour of the object of interest (like aircraft) instead of concerning any details. The operational behaviour of the ABSS is based on the real ABSS, where any reachable details (like cooperation between agents) will not be ignored. The differentiation of these two behaviours allows us to verify the behaviour of the ABSS from the abstract and detailed levels. The behaviour of the detailed level is verified by the general system performance properties like safety, liveness, etc. On the other hand, the verification for the behaviour of the abstract level is performed with a control behaviour. In Bentahar's work, the control behaviour was proposed to verify the conformity of the operational behaviour of the composite web services Bentahar et al. (2013). We verify the global behaviour of the ABSS based on the idea of Bentahar's work. The control behaviour helps guarantee that the model of the ABSS considers design requirements and constraints. The satisfactory of the operational behaviour changed in case of issues can be checked by control behaviours. For instance, a new agent has to be integrated into the ABSS to complete its functionality. In this case, the control behaviour remains the same and it can be utilized to verify the conformity of the global behaviour of a new ABSS.

The symbolic model checking technique is chosen to verify global behaviours and operational

behaviours of the ABSS. The problem of model checking is formally expressed by $M \models \varphi$, where M represents the system model, φ is a property, and $\models$ is the satisfaction symbol to check whether the model M satisfies the property φ. If the property is not satisfied by the system, a counterexample is produced. The efficiency of this technique has been proven to automatically verify the multi-agent systems (Al-Saqqar et al. 2015), as it uses less memory than automata-based approaches and it alleviates the "state explosion" problem. Hence, the symbolic model checking is the appropriate technique to verify the design of the ABSS. NuSVM is a symbolic model checker designed to allow for the description of FSM (Finite State Machine) which ranges from completely synchronous to completely asynchronous, and from the detailed to the abstract (Cavada et al. 2019). It supports the verification of the LTL (Linear Temporal Logic) and the CTL (Computational Tree Logic) specifications. The primary purpose of NuSMV input language is to describe the transition relation of the FSM, which is quite suitable for describing the state transition of statecharts in agents. Therefore, we choose NuSMV as the model checker.

The approach proposed is employed to verify the behaviours of the ABSS proposed in Chapter 4. As shown in Fig. 4.1, a number of agents such as Flight, Plane, etc., are involved and the general information between agents is provided as well. As a result, the symbolic model checking is the appropriate technique to verify the design of the ABSS.

To conclude, the symbolic model checking technique is used to check the satisfaction of the global behaviour and the operational behaviour of the ABSS. The main contributions of this research work are illustrated as follows:

- proposing an improved model checking approach for verifying the ABSS from both the abstract and detailed points of view;
- providing a complete model transformation process from the ABSS to the formal model for verification;
- giving a concrete case study of formal methods in practice and demonstrating the power of formal methods in improving designs and in providing new insights.

Based on the verification results, the problem of signals transferring between Flight agent and Plane agent has been discovered, which influences the system operation for a relatively long period of the simulation execution. The simulation results show that the efficiency of the aircraft maintenance has been improved after having modified the behaviour of Flight agent.

5.2 Literature Review

This chapter refers to "global" and "operational" behaviours with the intention to refer to different levels of abstractions of a system model. These terms may cause readers confused because they are not quite often used in model checking in this way. Researchers in model checking often use

terms like software architecture ("global") and programs ("operational") (Khurshid, Păsăreanu, and Visser 2003; Visser et al. 2003; Zhang, Muccini, and Li 2010). We choose not to use them because we restrain our research scope to model checking for the ABSS. Agent is the key concept for the ABSS. The behaviour of agents is often described by statechart diagrams. However, these diagrams express the behaviour of agents in a relatively abstract manner, which are neither codes nor the architecture description. Furthermore, different levels of behaviours have been proposed in model checking (Bentahar et al. 2013; Souri and Navimipour 2014), in order to describe different levels of abstractions of a system model. Therefore, we argue that our terms are appropriate.

5.2.1 Büchi Automata

Automaton theory consists of four kinds of automata, including combinational logic (used in the digital circuit), finite-state machine (a mathematical model of computation), *pushdown* automaton (employs stack) and Turing Machine (a mathematical model of computation) (Automata 2019). The classes of automata can be seen in Fig. 5.1. Büchi automaton is derived from the finite-state machine, which extends the machine to infinite inputs. Generally, Büchi automaton is divided into deterministic Büchi automaton and non-deterministic Büchi automaton. A generalised definition of deterministic Büchi automata is illustrated in the following.

Definition 5.2.1. (Deterministic Büchi Automata) It is a 5-tuple $A =< \Sigma, Q, q^0, F, \delta >$, where:

- Σ is a finite set of messages sending between agents;
- Q is a finite set of states of agents;
- $q^0 \subseteq Q$ is the initial state of A;
- $F \subseteq Q$ is a set of final states;
- $\delta \subseteq Q \bigtimes \Sigma \rightarrow Q$ is a transition function.

Transitions and initial states are the major difference between deterministic and non-deterministic Büchi automata. Non-deterministic Büchi automaton uses a transition relation and a set of initial states instead of the transition function δ and the single initial state q^0 respectively.

Alternating Büchi automaton is another non-deterministic Büchi automaton whose transitions are categorised into existential and universal transitions. Existential transitions mean the next state of automata is just one of the possible states, reading a signal. Universal transitions imply an automaton simultaneously moves to all the next states from its current state.

5.2.2 Computational Temporal Logic

The CTL is a $branching - time$ logic which enables us to define properties considering the non-deterministic and branching evolution of a finite-state machine (Cavada et al. 2019). An infinite

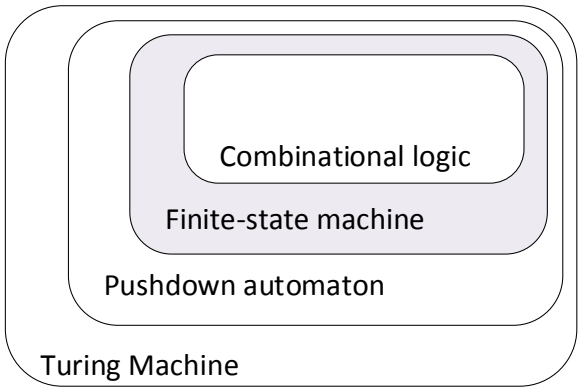

Figure 5.1: Automaton theory, adapted from Automata (2019)

tree can be employed to interpret the evolution of a finite-state machine. The sub infinite tree can describe the non-determinism and branching if the nodes of the tree are treated as states of the finite-state machine. The non-determinism of the transition relation leads to the tree stretching out with branches. The paths from one node to its leaves of the tree are the possible alternative evolution of the finite-state machine from the corresponding state. The idea of the tree allows the CTL to express properties holding for *all the paths* or *some of the paths* of the finite-state machine from a given state. The syntax of the CTL is illustrated as follows:

$\phi ::= \top \mid \bot \mid p \mid (\neg\phi) \mid (\phi \ \wedge \ \phi) \mid (\phi \ \vee \ \phi) \mid (\phi \Rightarrow \phi) \mid (\phi \Leftrightarrow \phi)$
$\mid AX\phi \mid EX\phi \mid AF\phi \mid EF\phi \mid AG\phi \mid EG\phi \mid EG\phi \mid A\,[\phi \ U \ \phi] \mid E[\phi \ U \ \phi]$, where

- p ranges over AP (Atomic Proposition);
- AX, EX, AF, etc., are operators.

The operators for the CTL combines temporal operators and quantification operators, which are shown in the following:

- Quantification operators:
  - $A\ p$: condition p must hold for all the paths starting from a state;
  - $E\ p$: at least one path exists starting from a state where condition p holds.
- Temporal operators:
  - $X\ p$: condition p must hold in the next state;
  - $G\ p$: condition p must hold in all future time instants;
  - $F\ p$: condition p must hold in one of future instants;
  - $p\ U\ q$: condition p must holds at least until condition q holds.

5.2.3 Linear Temporal Logic

Amir Pnueli first proposed LTL to verify the logic of computer programs in 1977 (Pnueli 1977). The LTL is based on a finite set of atomic propositions. The LTL verifies programs by considering each linear path rather than the whole sub-tree. So, it does not have quantification operators, i.e., it is incapable of expressing specifications that hold for some of the paths. Thus, LTL formulas just include logic operators and temporal operators. The two logics have in general different expressive power, but share a significant intersection (Cavada et al. 2019). The LTL fully supports the temporal logic discussed in CTL. Moreover, some LTL specifications can not be expressed by the CTL. For example, F (G p) means every path will eventually reach a point after which condition p always holds. It should be noted that the syntax and operators of the LTL are similar to those of CTL so we just explained the difference between them.

5.2.4 Model Checking for Multi-agent Systems

Operational and control behaviours have been studied in Web services (Sheng et al. 2010; Bentahar et al. 2013; Souri and Navimipour 2014; Navimipour et al. 2015). The control behaviour was proposed by Bentahar et al. (2013) to verify the operational behaviour of composite web services. In their work, the control behaviour defines a general design of web services and guides and monitors the execution progress of the operational behaviour. The operational behaviour implies the business logic of a web service operation. The detailed formal definition of behaviours and the transformation process were provided. However, their verification approach is limited. The major problem may be that it can not check the operational behaviour where states transitions are defined with conditions. For example, the operation like $PlanePrepared \xrightarrow{\text{“invoke”}} PlaneScheduled$ can not be checked. Because the LTL or CTL (Pnueli 1977) specification representing this kind of execution sequences can not pass in NuSMV model checker. Thus, we think that their approach is suitable to verify the global behaviour where fewer details are involved and no condition exists.

Many efforts have been made to verify the logic of agents for multi-agent systems. Logic has been extended to support the model checking for complex systems. Al-Saqqar et al. (2015) extended the CTL logic with knowledge and commitments to verify the multi-agent systems with respect to knowledge and social commitments. Meski et al. (2014) extended the LTL with the epistemic component for multi-agent systems. Raimondi (2006) extended the temporal logic to multi-modal logics for time, knowledge, correct behaviour, and strategies, in order to develop techniques and tools for the formal verification of the multi-agent system. These approaches are principally contributed to improving the capability of model checking. Researchers tend to focus on the logic itself for model checking. However, few of them try to concentrate on better understanding the system, in order to improve the efficiency of model checking. In this chapter, we have not extended any logic, we have decided to check the ABSS by analyzing the global behaviour and operational behaviour of the system.

Model transformation is usually employed in model checking. Model checking often automatically verifies systems by model checkers. Many powerful model checkers such as SPIN[1], PRISM[2], UPPAAL[3] and NuSMV[4] are widely used. The modelling languages supported by model checkers are not usually appropriate to model the systems that need to be checked. Thus, model transformation techniques are involved in model checking. For example, Bordini et al. (2006) automatically transformed multi-agent systems into Promela or Java, in order to use the associated Spin and JPF[5] model checker to verify system. Keshanchi, Souri, and Navimipour (2017) transformed the labelled transition system into NuSMV code to verify the genetic algorithm for the task scheduling in the cloud environment. El Menshawy et al. (2018) transformed RTCTL^{cc} into RTCTL (real-time CTL) where RTCTL^{cc} expresses qualitative and quantitative commitment requirements. Another example is provided by Bentahar et al. (2013), they presented the process of transforming a finite state machine into a Kripke model. Similar to our work, however, their transformation process was not involving the messages.

Model checking is capable of verifying and demonstrating system behaviours. However, industrial applications of model checking are rare. Researchers have a tendency to use less concrete examples. For example, Bentahar et al. (2013) used a ticket reservation system to illustrate their approach. This system just described the operation behaviour of the ticket reservation from a global point of view. No details about reservation processes were involved. Keshanchi, Souri, and Navimipour (2017) adopted the genetic algorithm to explain the feasibility of model checking. Only a single module was referred to. Al-Saqqar et al. (2015) modelled NetBill protocol to demonstrate the efficiency of their method. A real case was analyzed by El Menshawy et al. (2018). They chose the landing gear system in Boniol and Wiels (2014) as their case study, which was a real and industrial case. They corrected their NuSMV model after having found a counter-example by specifications. However, they did not discuss the reason why the original model was not correct and the comparison experiments were not provided regarding the original model and the corrected model.

As a conclusion, we are motivated to propose a more complete approach of model checking and our method is explained based on a real simulation system of aircraft maintenance. So, the improvement of the simulation system and comparison experiments are performed, which provides an example of an application of formal methods.

5.3 The Formalization of the Autonomous System

In this section, the specification of the ABSS is formalized as the global behaviour, operational behaviour and control behaviour. Control behaviours express the general behaviour of any process

[1] http://spinroot.com/spin/whatispin.html (accessed in March 2019)
[2] http://www.prismmodelchecker.org/ (accessed in March 2019)
[3] http://www.uppaal.org/ (accessed in March 2019)
[4] http://nusmv.fbk.eu/ (accessed in March 2019)
[5] https://github.com/javapathfinder/ (accessed in March 2019)

related to aircraft maintenance. However, global operational behaviours are dependent on applications. These behaviours have been investigated for isolated or composite web services (Maamar et al. 2009; Yahyaoui, Maamar, and Boukadi 2010; Bentahar et al. 2013). Different methods like Büchi automaton and labelled transition system, were used to formalize the behaviours. Büchi automaton extends a finite automaton to infinite inputs, which meets the requirements for the continuous invocation of plane's service. Thus, we use Büchi automaton to describe the behaviours of the ABSS.

5.3.1 Global Behaviour and Operational Behaviour

The global behaviour (Fig. 5.2) is expressed as the behaviour happening in the life cycle of planes from the global point of view. The operational behaviour (Fig. 5.3) is employed to describe the inner behaviour of the component. The definitions of behaviours are illustrated as follows.

Definition 5.3.1. (Global Behaviour) The global behaviour of an agent-based system is a 5-tuple $GB =< \Sigma_{gb}, Q_{gb}, Q^0, F_{gb}, \Delta_{gb} >$, where:

- Σ_{gb} is a finite set of messages sending between agents;
- Q_{gb} is a finite set of states of agents;
- $Q^0 \subseteq Q_{gb}$ is a set of initial states;
- $F_{gb} \subseteq Q_{gb}$ is a set of final states;
- $\Delta_{gb} \subseteq Q_{gb} \bigtimes \Sigma_{gb} \bigtimes Q_{gb}$ is a transition relation.

Definition 5.3.2. (Agent Operational Behaviour) The operational behaviour of an agent is a 7-tuple $AOB =< \Sigma_{aob}, \Sigma^{in}, \Sigma^{out}, Q_{aob}, Q^0, F_{aob}, \Delta_{aob} >$, where:

- Σ_{aob} is a finite set of messages of the agent, including the empty message $\emptyset$;
- $\Sigma^{in} \subseteq \Sigma_{aob}$ is a set of messages inside the agent;
- Σ^{out} is a set of messages outside the agent;
- Q_{aob} is a finite set of states of the agent;
- $Q^0 \subseteq Q_{aob}$ is a set of initial states;
- $F_{aob} \subseteq Q_{aob}$ is a set of final states;
- $\Delta_{aob} \subseteq Q_{aob} \bigtimes \Sigma_{aob} \bigcup \Sigma^{out} \bigtimes Q_{aob}$ is a transition relation.

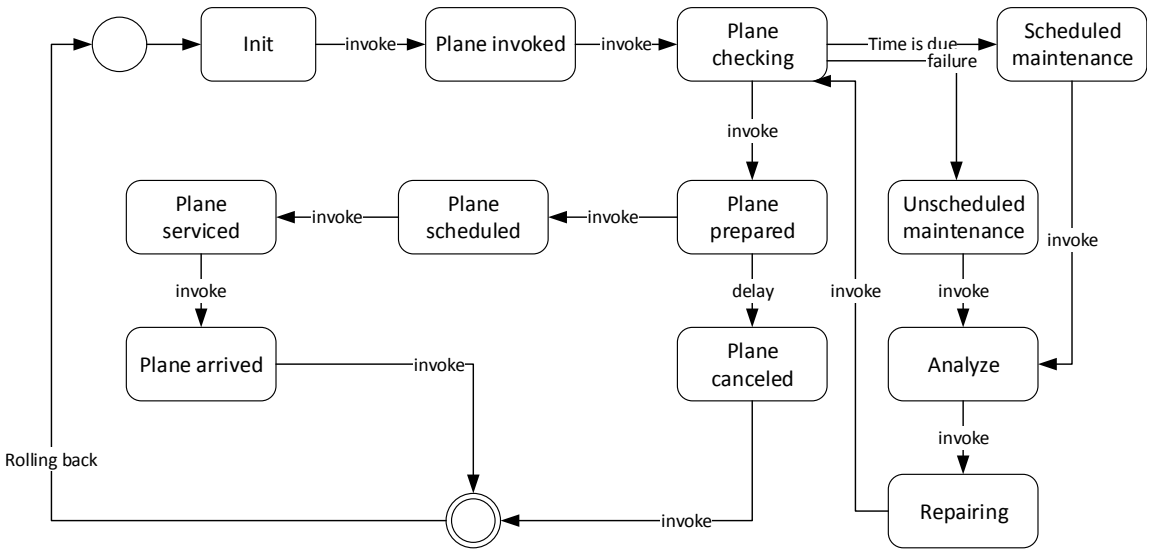

Figure 5.2: The global behaviour of the ABSS for aircraft maintenance

Definition 5.3.3. (Valid Conversation) A conversation in AOB over a sequence of messages $\Sigma_0, \Sigma_1, ..., \Sigma_n$ from Σ_{aob} is a sequence $Q^0 \xrightarrow{\Sigma_0} Q^1 \xrightarrow{\Sigma_1} Q^2 ... Q^{n-1} \xrightarrow{\Sigma_{n-1}} Q^n$ such that $\forall i \geq 0, Q^i, Q^{i+1} \in Q_{aob}, (Q^i, \Sigma_i, Q^{i+1}) \in \Delta_{aob}, Q^0 \in Q_0, Q^n \in F_{aob}$. The conversation is said to be valid iff it visits infinitely an element of F_{aob}.

As the forms of global and operational behaviours are similar, we just provide an example of the operational behaviour. Fig. 5.3 represents the operational behaviour of Flight agent. The elements of AOB over Flight agent are explained as follows:

- Σ_{aob} = {STA.delay, Plane.ready, Plane.receive, “null", (“schedule", plane)};
- Σ^{in} = {STA.delay, Plane.ready, Plane.receive};
- Σ^{out} ={(“schedule",plane)};
- Δ_{aob}= {(Init, STA.delay, Checking), (Checking, Plane.ready, Ready), (Ready , (“schedule", plane), Scheduling), (Init, (“schedule", plane), Scheduling), (Scheduling, Plane.receive, End), (End,“null", Init)}.

One conversation in the AOB of Flight agent can be:
$Init \xrightarrow{(“schedule”,plane)} Scheduling \xrightarrow{Plane.receive} End.$

It should be noted that dashed, dotted and solid arrows imply the messages of receiving from other agents, sending to other agents, and sending to the interior respectively. The messages of sending or receiving are always associated with destination agents. For the message of receiving from other agents like “STA.delay", it is composed of the name of agent (e.g. STA) and the signal

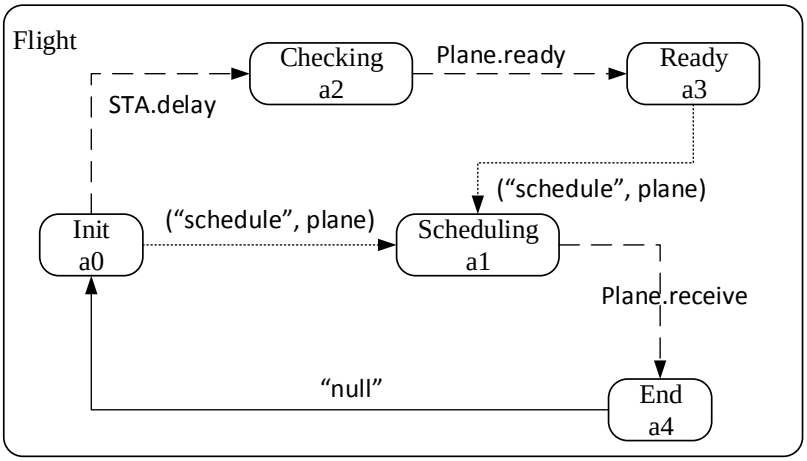

Figure 5.3: The operational behaviour of Flight agent

(e.g. “delay"). In terms of the message of sending to other agents like “(“schedule", plane)", it consists of the signal (“schedule") and the agent (Plane).

5.3.2 Control Behaviour

Control behaviours constrain the operation sequence of system behaviours. The conformity to the control behaviour reflects the correctness of the system behaviours. Since the control behaviour is formalized by Büchi automaton as well, we do not repeat its definition. Here we just introduce the content of the control behaviour.

Fig. 5.4 illustrates the control behaviour of the ABSS, which is adapted from Bentahar et al. (2013). They provided a control behaviour of composite web services. Since the control behaviour is the application-independent specification of business logic, it is possible to adapt control behaviours of one domain to another. In terms of web services, the concept of “compensate" implies the original execution prices will be refunded when the service provider can not provide the committed service or deliver the ordered commodity (Liu, Ngu, and Zeng 2004). However, as for aircraft maintenance, we have not involved this issue. Thus, we delete this behaviour. In addition, “Aborted" refers to one flight has to be cancelled because of a long delay.

To illustrate the efficient design of the global behaviour of the system, we check the conformity of the global behaviour of the system to the control behaviour. The conformity is defined as the relationship between the conversations of the global behaviour and the control behaviour executions. According to Bentahar et al. (2013), the type of the conformity can be: weak or strong. The former implies each conversation c in the global behaviour is simulated by an execution e in the control behaviour, which can be denoted as $c \rightsquigarrow e$. The latter highlights each valid execution e in the control behaviour is simulated represented by a possible conversation c in the global behaviour

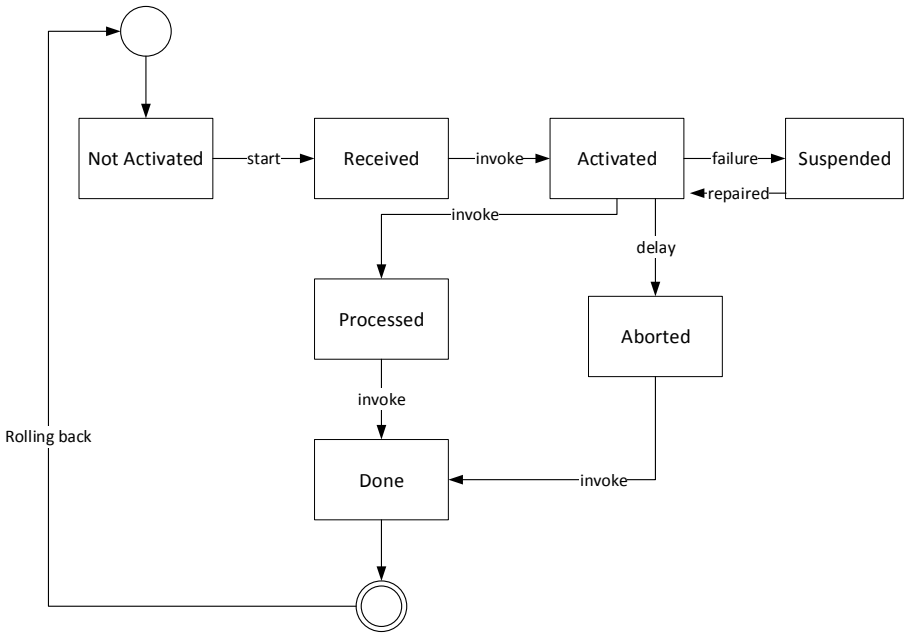

Figure 5.4: The control behaviour of the agent-based system

$(e \rightsquigarrow c)$.

Based on the two types of conformity, all the conversations in Fig. 5.2 should be mapped onto valid executions in Fig. 5.4. On the other hand, all the valid executions in Fig. 5.4 should be mapped onto the conversations in Fig. 5.2. Since the control behaviour is application-independent and more abstract, the mapping is not necessarily state-to-state and transition-to-transition. For instance, Conversation 1 in the global behaviour and Execution 1 in the control behaviour (* means repetition or loop) are shown as follows:

$Conversion1 = (Init \xrightarrow{\text{“invoke”}} PlaneInvoked \xrightarrow{\text{“invoke”}} PlaneChecking$
$\xrightarrow{\text{“TimeIsDue”}} ScheduledMaintenance \xrightarrow{\text{“invoke”}} Analyze \xrightarrow{\text{“invoke”}} Repairing$
$\xrightarrow{\text{“invoke”}} PlaneChecking \xrightarrow{\text{“invoke”}} PlanePrepared \xrightarrow{\text{“delay”}} PlaneCanceled)*.$

$Execution1 = (NotActivated \xrightarrow{\text{“start”}} Received \xrightarrow{\text{“invoke”}} Activated \xrightarrow{\text{“failure”}} Suspended \xrightarrow{\text{“repaired”}}$
$Activated \xrightarrow{\text{“delay”}} Aborted \xrightarrow{\text{“invoke”}} Done)*.$

The mappings between state-to-state (S-S) and transition-to-transition (T-T) are explained in Table 5.1. Verifying this conformity one by one manually is quite tedious even if a small number of states are involved. The number of possible conversations grows dramatically with the increase in the number of states. So, an automatic verification approach is proposed in the next section.

Table 5.1: The mapping between *Conversation* 1 and *Execution* 1

	Conversation 1	*Execution* 1
S-S	$Init$	$NotActivated$
	$PlaneInvoked$	$Received$
	$PlaneChecking, PlanePrepared$	$Activated$
	$ScheduledMaintenance, Analyze, Repairing$	$Suspended$
	$PlaneCanceled$	$Aborted, Done$
T-T	$Init \rightarrow PlaneInvoked$	$NotActivated \rightarrow Received$
	$PlaneInvoked \rightarrow PlaneChecking$	$Received \rightarrow Activated$
	$PlaneChecking \rightarrow ... \rightarrow Prepared$	$Activated \rightarrow Suspended \rightarrow Activated$
	$PlanePrepared \rightarrow PlaneCanceled$	$Activate \rightarrow Aborted \rightarrow Done$

5.3.3 The Adaptability of Büchi Automaton

A typical agent-based model has three elements (Macal and North 2010):

- A set of agents, their attributes and behaviours;
- A set of agent *relationship* and methods of interaction: an underlying topology of connectedness defines how and with whom agents interact;
- The agents' *environment*: agents interact with their environment in addition to other agents.

From this definition, the agent-based system not only concentrates on the behaviour of individual agents but addresses the interactive behaviour between agents.

The agent-based system is usually formalized by IS (Interpreted System) to reason about knowledge and temporal properties (Fagin et al. 2004). The formalism is illustrated as follows (Al-Saqqar et al. 2015) :

- A set of n agents $\mathfrak{A} = \{1, 2, ..., n\}$ such that each agent i is described by:
  - A non-empty set of local states L_i. The local state of agent i is represented by $l_i \in L_i$. Each local state of an agent represents the complete information about the system that the agent has at a given moment;
  - A set of local actions Act_i to account for the temporal evolution of the system;
  - A local protocol function $P_i : L_i \rightarrow 2^{Act_i}$ to identify the set of enabled actions that could be performed in a given local state;
  - A local evolution function τ_i that determines the transitions for an individual agent i between its local states and it is defined as follows: $\tau_i : L_i \bigtimes Act_i \rightarrow L_i$.
- The set of all global states in the system $G \subseteq L_1 \bigtimes ... \bigtimes L_n$: a subset of the Cartesian product of all local states of n agents.

  - A global state $\mathfrak{g} \in G$ is a tuple $\mathfrak{g} = (l_1 ... l_n)$ that represents a "snapshot" of the system;
  - The local state of agent i in the global state $\mathfrak{g}$ is represented by the notation $l_i(\mathfrak{g})$.

- $I \subseteq G$: the set of initial global states for the system;
- The global evolution (transition) function: it can be defined as follows: $\tau : G \bigtimes ACT \rightarrow G$, where $ACT = Act_1 \bigtimes ... \bigtimes Act_n$ and each component $a \in ACT$ is a joint action, which is a tuple of actions (one for each agent);
- A set Φ of atomic propositions and a valuation function $\mho$ for those propositions $\mho : G \rightarrow 2^{\Phi_p}$.

This definition implies that the consideration of the agent level (detailed level) and the global level (abstract level) is necessary for the description of the agent-based system. The formalization of states and transitions from both levels are required. As for our definitions, Definitions 5.3.1 and 5.3.2 are corresponding to the global level and the agent level respectively. The core idea of the agent-based system from these two definitions is captured by our definitions. As a result, our definitions are appropriate for the description of the ABSS as well. In addition, the idea of providing a new definition based on automaton theory is to take advantage of Büchi automaton. For example, infinite runs can be captured by finite structures, which is compatible with the continuous invocation of aircraft maintenance.

5.4 Model Checking for the Autonomous System

In this section, we first present our framework for model checking. The meta-model transformation from Büchi automaton to Kripke structure is then proposed. In the following, the model transformation from the ABSS to the NuSMV model is provided. Finally, the specifications of the LTL and CTL are designed.

5.4.1 Model Checking Framework

Fig. 5.5 illustrates a general framework for verifying the behaviour of the ABSS. This framework addresses the difference between global and operational behaviors, which implies the importance of the abstract and detailed design respectively. The idea of the verification is necessary for the design of the ABSS since the agents are built individually and the agent-based models are rather complex. The process involves the global and operational behaviour verification. The former should be verified by the control behaviour. The latter is checked by the general system performance properties. Finally, the verification result can improve systems design by means of generating counterexamples. The counterexample of global behavior can improve the design of the interactions between agents. The operational behaviour counterexample is able to correct the agent behaviour directly.

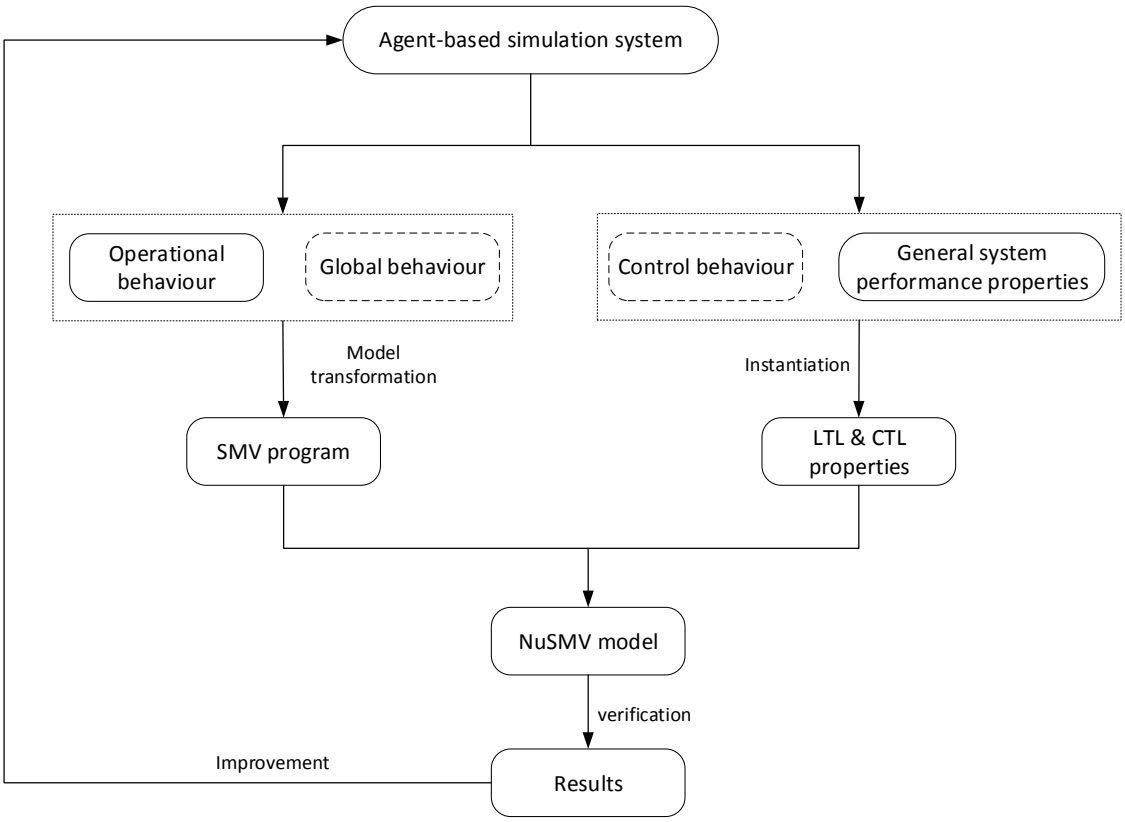

Figure 5.5: The framework on model checking for the ABSS

5.4.2 Meta-model Transformation

To illustrate the feasibility of the transformation from the ABSS to the NuSMV model, the meta-model transformation is accomplished. The meta-model transformation implies the transformation from Büchi automaton to Kripke structure. This transformation process lays the foundations for the next model transformation. The definition of Kripke structure is shown in Definition 5.4.1. The transformation between them has been provided by Bentahar et al. (2013). They discussed the transformation of states and transitions in detail. However, the messages of Büchi automaton has not been involved. The information of messages should be transformed into the labels of Kripke structure.

Definition 5.4.1. (Kripke Structure) A Kripke structure is a 4-tuple $\mathcal{K} =< S, I, \delta, L >$, where:

- S is a finite set of states;
- $I \subseteq S$ is the subset of initial states;
- $\delta \subseteq S \bigtimes S$ is the transition relation;

- $L : S \rightarrow 2^{AP}$ (elements of 2^{AP} are called labels) is a labelling function. AP is a finite set of atomic propositions.

A Büchi automaton is a 5-tuple $\mathfrak{A} =< \Sigma, Q, Q^0, F, \Delta >$, which is generalized by Definition 5.3.1. The transformation process from Büchi automaton to Kripke structure is illustrated as follows:

- $S = Q$: the states of Kripke structure are the same as those of Büchi automaton;
- $I = Q^0$: the initial states of Kripke structure are the same as those of Büchi automaton;
- $\forall s, s' \in S, (s, s') \in \delta$ and $L(s') = a$, iff $\exists s, s' \in Q, a \in \Sigma$ and $(s, a, s') \in \Delta$.

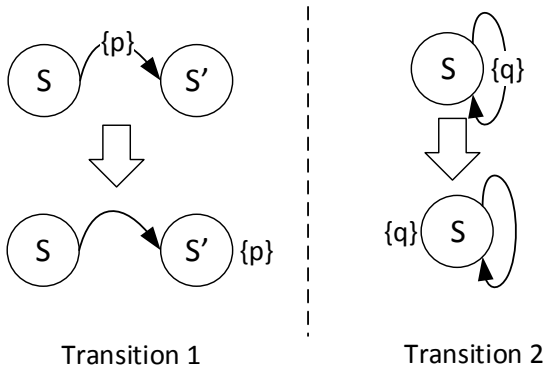

Figure 5.6: The meta-model transformation from Büchi automaton to Kripke structure

Fig. 5.6 provides an example of the transitions in the transformation process. For Transition 1, $s, s' \in S, (s, s') \in \delta$ and $L(s') = \{p\}$, because $\exists s, s' \in Q, \{p\} \in \Sigma$ and $(s, \{p\}, s') \in \Delta$. Transition 2 shows the transition process when s, s' are the same state.

5.4.3 Model Transformation

The model transformation involves global and operational behaviours of the system. The first behaviour is verified by the control behaviour. So, the properties to be checked are extracted from the control behaviour. In order to make the verification clearer, all the states in the state chart of global behaviour are associated with the state names of the control behaviour. As a result, the final state chart of the global behaviour is illustrated in Fig. 5.7 where the names of states are simplified as NA (Not Activated), Re (Received), Ac (Activated), Su (Suspended), Ab (Aborted), Pr (Processed) and Do (Done). The mapping of action names between two behaviours can be seen in Table 5.1. In addition, states of $PlaneScheduled$, $PlanceServiced$, $PlaneArrived$ of the global behaviour correspond to control behaviour $Processed$. The transformation process will be analyzed in the following.

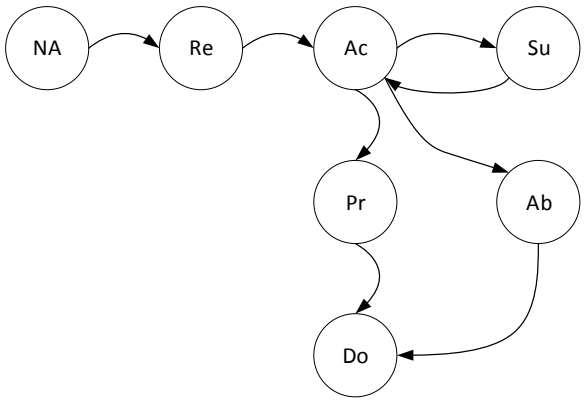

Figure 5.7: The state chart of the global behaviour of the system

It should be noted that the verification of the global behaviour is not involved with labels. Because the verification focuses on the execution sequences of the states in the control behaviour. If labels are added to the transitions of NuSMV model, the properties about the execution sequence will never be satisfied. In general, transition processes are described as *TRANS* constraints. The constraints do not support that one state has more than one following state without labels. While the global behaviour of the system does not have any labels and one state may have more than one following state. Thus, *TRANS* constraints are not appropriate. We have chosen another semantically equivalent *ASSIGN* constraints. The transformation process of the global behaviour of the system is rather easy. The NuSMV code (Fig. 5.8) is given just by capturing the transitions in Fig. 5.7.

```
MODULE main

VAR state:{NA, Re, Ac, Su, Pr, Ab, Do};
ASSIGN
init (state) := NA;
next(state):=
        case
            (state = NA)    : {Re};
            (state = Re)    : {Ac};
            (state = Ac)    : {Su, Pr, Ab};
            (state = Su)     : {Ac};
            (state = Pr)     : {Do};
            (state = Ab)    : {Do};
             TRUE: state;
        esac;
```

Figure 5.8: The NuSMV model of the global behaviour of the system

The other transformation is dedicated to verifying the general system properties of the ABSS. This system has been implemented by simulation tool *Anylogic*. The process principally includes structure and agent transformations. The mapping between the ABSS and the NuSMV model is illustrated in Table 5.2.

Table 5.2: The mapping between the ABSS and NuSMV model

	ABSS	NuSMV model
system structure	agents	processes
	connections	parameters
operational behaviour	initial states	initial states
	transitions	transitions
	messages	actions
	timeout/ rate	Null (action)
	agent arrival	Reach (action)
	condition	condition

The agent-based system structure (Fig. 4.1) is transformed into the NuSMV code shown in Fig. 5.9. Here, we distinguish module *main* in the NuSMV code with the instance of Main agent. The latter is named as **mainn**.

In terms of the agent operational behaviour, since *Anylogic* is used to create the simulation system, the transition conditions of *Anylogic* should be transformed. The triggering type of a transition includes $messages, timeout, rate$, and $condition$. The $condition$ of the ABSS is implied as either the logic combination of agent messages or agent states. Fig. 5.10 shows a complete transformation of Plane agent into the corresponding NuSMV code. This code provides an example of the condition transformation. The condition of the transformation from $ReadyToFly$ to $Scheduled$ in the statechart is that Plane agent receives $Flight.Schedule$. This condition is then transformed into

$(arg1.state = b7 \ \& \ arg2.action = Schedule) : b8.$

```
MODULE main
VAR
   flight   : process Flight(flight, plane, sta);
   plane    : process Plane(plane, flight, sta, ssa, mainn);
   cra      : process CRA(cra, plane);
   sta      : process STA(sta, cra, ssa, sca);
   ssa      : process SSA(ssa, cra, sta);
   sca      : process SCA(sca, sta);
   mainn    : process Main(mainn, plane);
```

Figure 5.9: The transformation for the system architecture

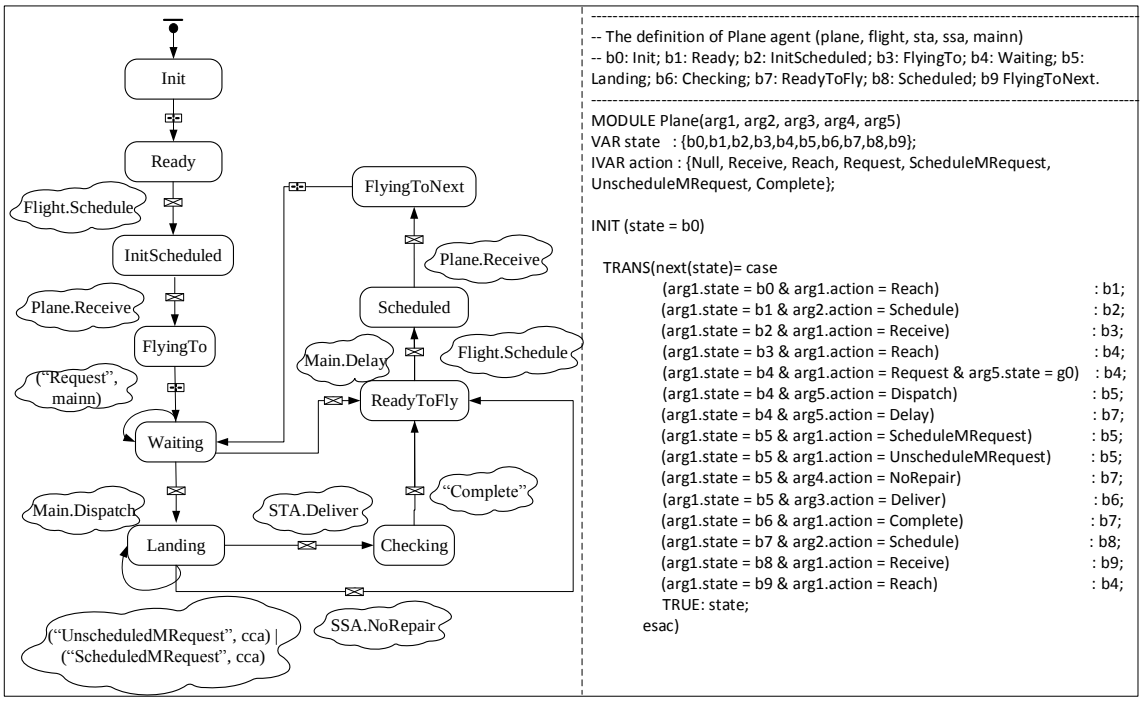

Figure 5.10: The transformation for Plane agent

5.4.4 Properties to Check

NuSMV allows verifying both CTL and LTL formulas (Cavada et al. 2019). CTL formulas specify properties over the computation tree of the FSM (branching-time approach). LTL formulas verify each linear path induced by the FSM (linear-time approach). Here, we employ the logics of the CTL and LTL to express properties. Global and operational behaviour properties are discussed in the following.

The global behaviour properties are extracted from the control behaviour of the ABSS, which are illustrated as follows.

CTL specifications:

- $C1 - \varphi = AG(state = NA -> AX\ state = Re)$;
- $C2 - \varphi = AG(state = Ac -> AX\ (state = Su \mid state = Pr \mid state = Ab))$;
- $C3 - \varphi = AG(state = Ac -> EX\ (state = Ab))$;
- $C4 - \varphi = AG((state = Pr \mid state = Ab) -> AX\ state = Do))$;
- $C5 - \varphi = AG\ (state = NA\ -> EF\ state = Ab)$;
- $C6 - \varphi = AG(state = Su -> EF\ (state = Pr \mid state = Ab))$;

- $C7 - \varphi = AG(state = Su -> AX\ state = Ac)$;
- $C8 - \varphi = AG(state = NA -> AF\ state = Ac)$;

The formulas of $C1$, $C2$ and $C7$ follow the same form $\varphi = AG(state = A -> AX\ state = B)$. This form addresses that the next states of state A are always B. For example, property $C1$ illustrates that state $NotActivated$ is followed by $Received$. Property $C3$ implies that the next state of state $Activated$ is existentially $Aborted$. The properties of $C5$ and $C6$ share form $\varphi = AG(state = A -> EF\ state = B)$, which highlights that state B is reachable from state A. The formula of $C4$ means that the process can reach $Aborted$ state from $NotActivated$. The last formula specifies that the process will eventually in the future reach state $Activated$ from $NotActivated$. Thus, property $C8$ shows that starting from state $NotActivated$, the process will eventually reach state $Activated$.

LTL specifications:

- $L1 - \varphi = G(state = NA -> X\ state = Re)$;
- $L2 - \varphi = G(state = Ac -> X\ F(state = Su \mid state = Pr \mid state = Ab))$;
- $L3 - \varphi = G((state = Pr \mid state = Ab) -> X\ F\ state = Do)$.

Property $L1$ represents the form $\varphi = G(state = A -> state = B)$. This form implies in all possible computations, state A is followed by state B. Thus, this property explains that the next state of state A is always B. The last two properties share the same form $\varphi = G(state = A -> X\ F\ state = B)$. It denotes that one of the future next states of state A will be state B. So, property $L2$ shows that the future next states of $Activated$ will be $Suspended$ or $Processed$ or $Aborted$. Property $L3$ denotes that the system is at either $Processed$ or $Aborted$ state, it will eventually reach $Done$ state.

Similar forms of LTL and CTL formulas have been proposed to verify the behaviour of the composite web services (Bentahar et al. 2013). However, after testing all the possibilities, some properties defined in their work such as properties $L1$, $C1$ and $C2$ are recognized as incorrect properties. Because p & q in the formulas means conditions p and q hold at the same time and it can not express the sequence of states. However, the operator of X, EX, AX implies the satisfactory of the next state. Thus, there is no point in combining & and X, EX, AX. For example, L1- $\varphi = G(!(Su\ \&\ X\ Do))$ always passes. On the other hand, L1'- $\varphi = G(Su\ \&\ X\ Do)$ is impossible to be satisfied. It is because $X\ Do$ is equal to $False$ with no declaration of the current state. Su is treated as $True$ because it is only a simple state declaration. Thus, the counter-example is as simple as this: "State :1.1: state = NA".

Properties from general system properties (Al-Saqqar et al. 2015) like safety, reachability, deadlock freedom, etc. are employed to check the operational behaviour of the system. We will define each property in the following.

5.4.4.1 Safety Property

Safety property is often expressed by formula $AG\,!\,\varphi$ (CTL property), where φ implies something bad. This formula means that the property φ is never satisfied in all possible computations. In the ABSS, bad situations like scheduling signal have been sent to the corresponding plane by Flight agent, whereas the plane has not received it. This situation can be expressed as follows:

CTL - S1 - $\varphi = AG(!(flight.state = Scheduled\ \&\ plane.state = ReadyToFly))$.

5.4.4.2 Liveness Property

Liveness property means something good will be always possible to happen. The formulas of liveness property are shown as follow. For example, the plane is always possible to fly ($CTL-S2$).

CTL - S2 - $\varphi = SPEC\ AG\ EF(plane.state = FlyingToNext)$;

CTL - S3 - $\varphi = SPEC\ AG\ EF(plane.state = Waiting -> plane.state = ReadyToFly)$;

CTL - S4 - $\varphi = SPEC\ AG\ EF(sca.state = Buying)$;

CTL - S5 - $\varphi = SPEC\ AG\ EF(mainn.state = Delaying)$.

5.4.4.3 Reachability Property

Reachability property represents that some particular situation can be reached. The corresponding formulas are shown as follows. For example, in SSA agent, state End can be existentially reached from $StatesCheck$.

CTL - S6 - $\varphi = SPEC\ EF(ssa.state = StatesCheck -> ssa.state = End)$;

CTL - S7 - $\varphi = SPEC\ EF(sca.state = Buying)$.

5.4.4.4 Deadlock Property

Dead-lock property can be checked by using the commands under the interactive mode. The commands are shown as follows:

1. *NuSMV -int ass.smv*;
2. *go*;
3. *check_fsm*.

The first two commands allow the NuSVM to enter the interactive mode. Command 3 checks the dead-lock situation for all the transition relations.

5.5 Verification Analysis and Model Modification

In this section, we first illustrate the verification results from NuSMV model checker. The modification of the agent-based simulation model is then accomplished, conforming to the counterexample of the verification results. Finally, a detailed discussion about our method is provided.

5.5.1 Verification Result

After combining the LTL, CTL specifications and the transformed NuSMV model, a verifiable NuSMV program is obtained. The verification results are separated into two types: global behaviour (Fig. 5.11) and operational behaviour (Figs. 5.12 and 5.14). A counterexample of property $CTL-S1$ is proposed by NuSMV model checker, which is shown in Fig. 5.12. In this figure, the state transitions for Flight agent and Plane agent are concluded in Table 5.3. The meaning of codes like $b0$ can be seen in Fig. 5.10. In order to make it easier to follow the action in a system with a large number of variables, only the values of variables that have changed in the last step are printed in the states of the trace (Cavada et al. 2019). Thus, there is no value of state 1.3 for Flight agent. The transition between states 1.5 and 1.6 implies that Plane agent missed the signal from Flight agent because Plane agent was at state $b4$. However, state $b4$ is not the appropriate state to receive the signal from Flight agent. Thus, some constraints should be imposed on signals sending between these two agents.

Table 5.3: The states transitions between Flight and Plane

	state 1.1	state 1.2	state 1.3	state 1.4	state 1.5	state 1.6
Flight	a0	a1		a4	a0	a1
Plane	b0	b1	b2	b3	b4	b7

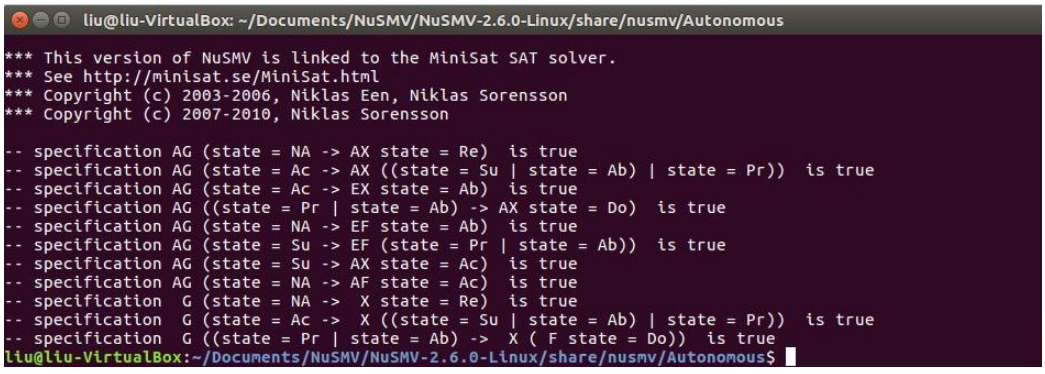

Figure 5.11: The results on checking properties of global behaviour of the system

The condition of “plane.state = Ready | plane.state = ReadyToFly" is added to transition $b0 \rightarrow b1$, which implies the scheduling signal can be sent to Plane agent, iff Plane agent reaches state

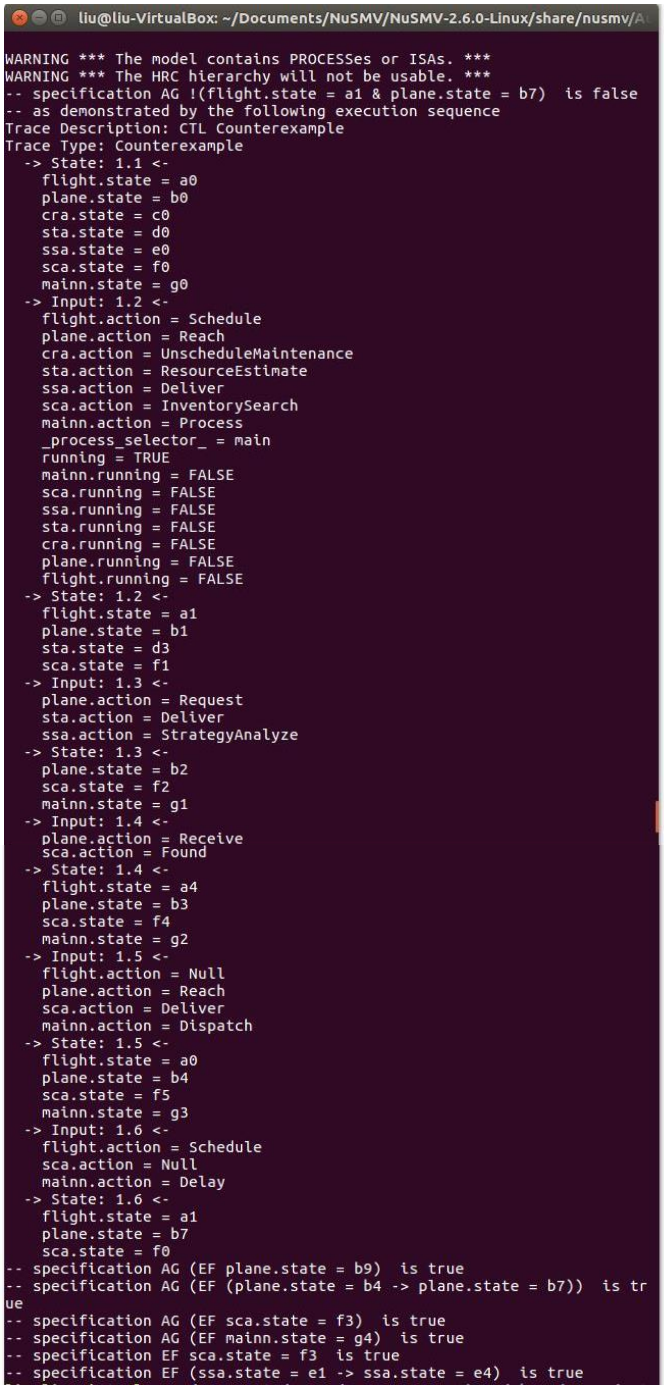

Figure 5.12: The result on checking general system properties (except for deadlock)

"Ready" or "ReadyToFly". Similarly, the condition of "plane.state = ReadyToFly" is added to transition $b2 \rightarrow b1$. The former state $b3$ is deleted. It should be noted that "Ready" means the plane is ready for the first scheduling since then each ready state of the plane is expressed by "ReadyToFly". Specification $CTL-S1$ has passed over NuSMV model checker, which is shown in Fig. 5.13.

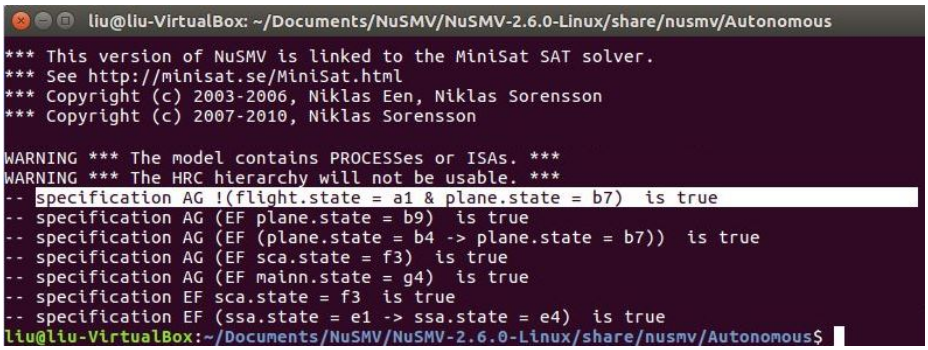

Figure 5.13: The result on the corrected NuSMV model

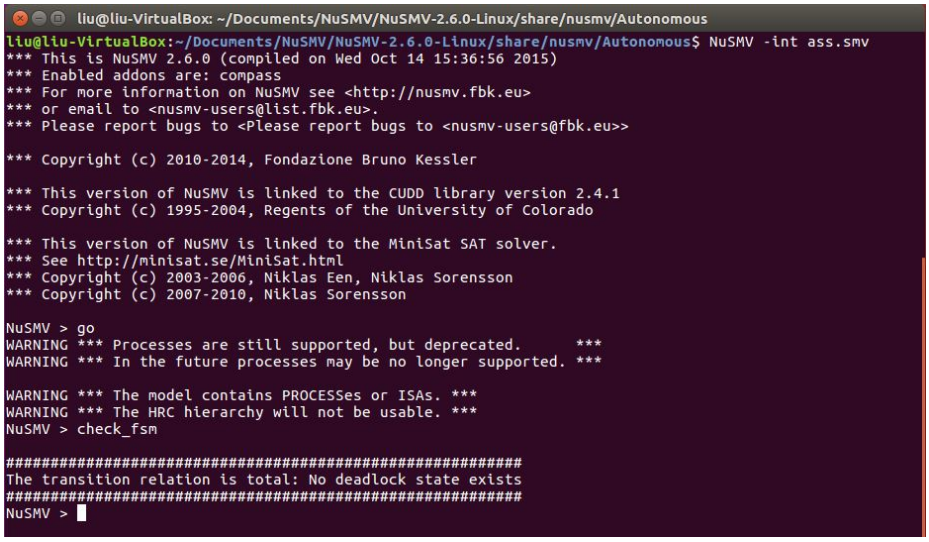

Figure 5.14: The result on checking deadlock property

The other properties are all satisfied by the checker. Deadlock property is checked by using commands and interactive mode of the model checker is necessary. So, this property is verified separately.

Therefore, verification results show that the global behaviour satisfies the constraints of the control behaviour. All the properties on liveness, reachability and deadlock are satisfied. Only property $CTL-S1$ does not pass. The problem is identified as an issue about signals sending between Flight agent and Plane agent.

5.5.2 Simulation Model Modification

Flight agent has been modified with respect to the correction of the NuSMV model. The original process and modified process are shown in Figs. 5.15 and 5.16 respectively. The difference between these two processes is highlighted by red lines. Some basic concepts are introduced, which aims to better understand the process. For example, *downtime* is the difference between *totalRepairTime* and *scheduledInterval*. *scheduledInterval* implies the interval between the time of the arrival and departure for a plane. *totalRepairTime* is only involved all the time used to repair faulty parts, excluding the time of waiting for the existing resources and scheduling. Additionally, *pollingTime* means the time used to check the state of a plane. In terms of service level, *onTimeFlights*, *delayFlights* and *canceledFlights* are defined. One flight is supposed to be cancelled if the delayed time from the departure of a plane is more than 4 hours (Liu et al. 2018).

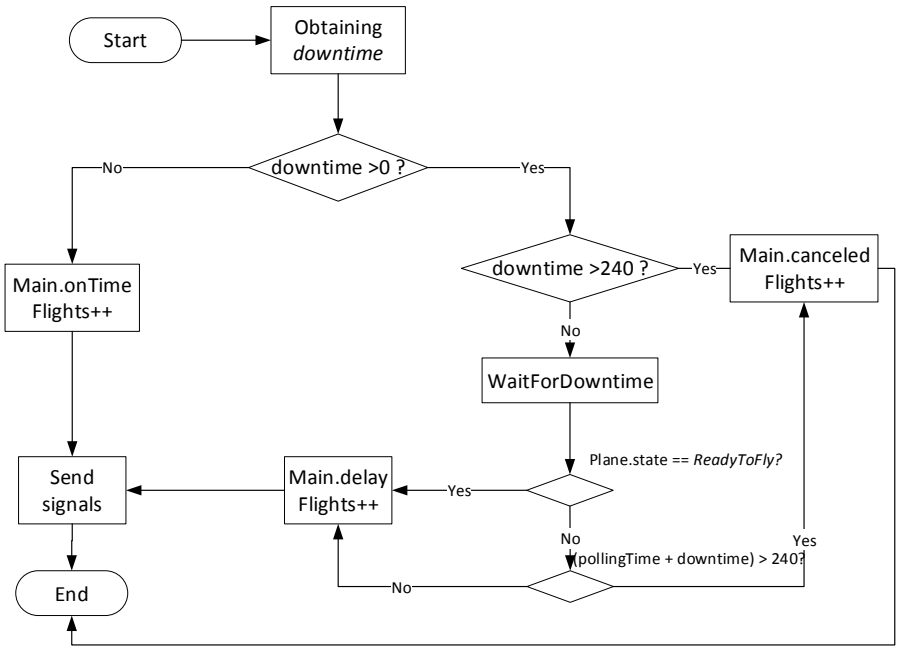

Figure 5.15: The original process for Flight agent

5.6 Case Study

5.6.1 Simulation Configuration

Simulation experiments are implemented by Anylogic PLE 8.2.3[6]. The configuration of the running laptop is 8G memory and 4 processors (i7-6500U CPU). The simulation starts from flight

[6] www.anylogic.com (accessed in May 2019)

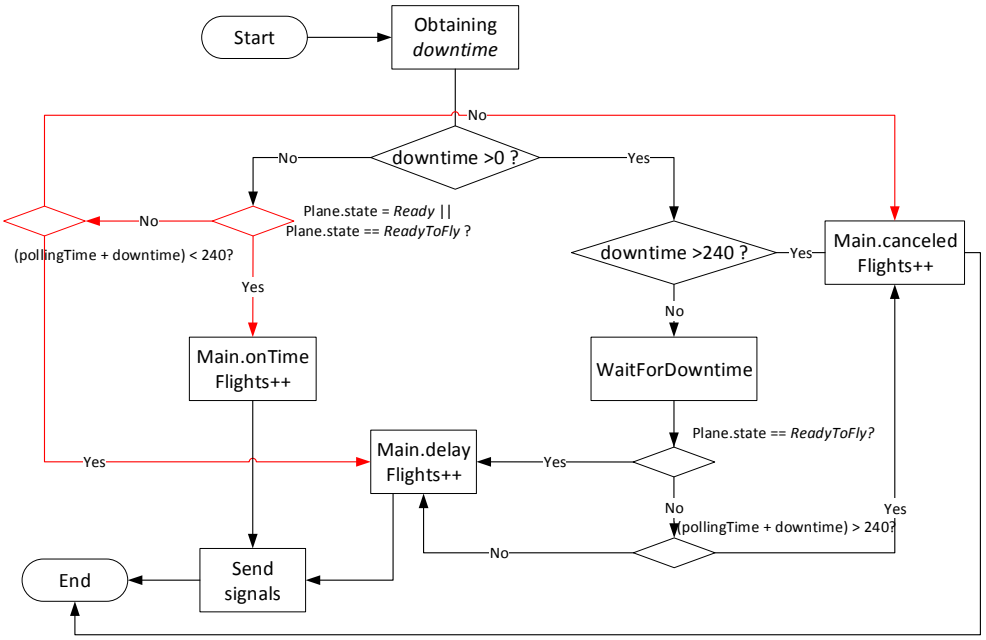

Figure 5.16: The modified process for Flight agent

schedules. The aircraft stay at airports waiting for scheduling signals. Flight legs (A flight leg is a trip of an aircraft, from take-off to landing) are from the website of *AirFrance*[7]. Twelve airports are concerned, including *Paris*, *Lyon*, *London*, *Berlin*, *Madrid*, *Amsterdam*, *Boston*, *Marseille*, *Barcelona*, *Prague*, *Manchester* and *Munich*. There are 50 aircraft distributed in the different airports.

Fig. 5.17 shows the general process of aircraft maintenance for the simulation. At each airport, maintenance sequences and maintenance tasks sequence will be analyzed. In this figure, an example is given for *Paris*, where the maintenance sequence optimisation and maintenance task sequence optimisation are illustrated. Message containers for the maintenance sequence and maintenance tasks sequence are used to collect signals from aircraft for requesting maintenance and for requesting resources respectively. Optimization processes are explained in Sections 4.3.1 and 4.3.5.

Each aircraft performs routes given by flight plans. Different types of aircraft own different equipment. The deterioration of each equipment conforms to its own degradation curve. Each airport is capable of undertaking all kinds of maintenance tasks. Not all resources at airports are the same. The data of flight plans, the available person-power for airports, and the maintenance-related time is provided in Section 5.6.1.2.

[7] www.airfrance.fr (accessed in January 2019)

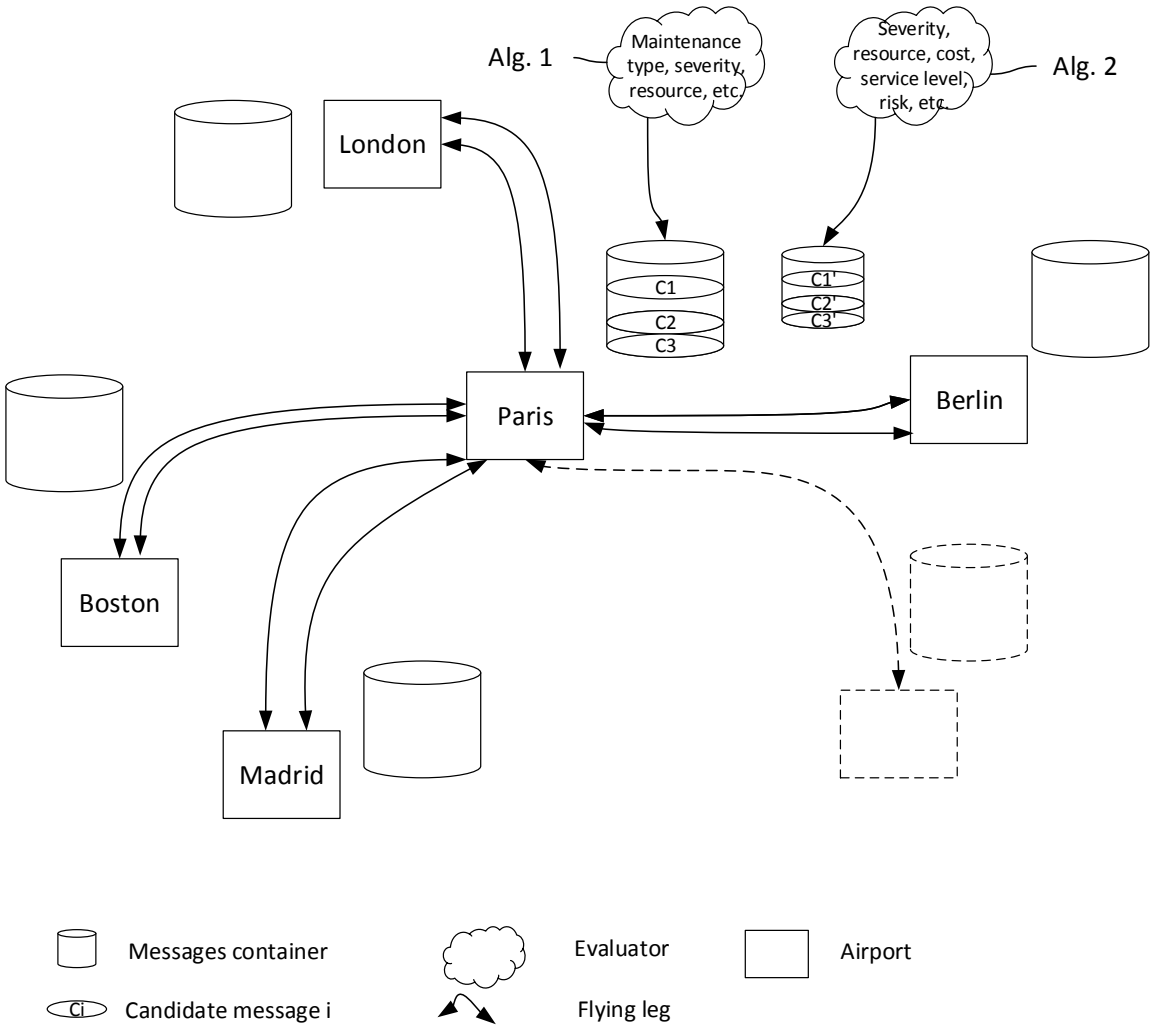

Figure 5.17: The simulation experiment scenario

5.6.1.1 Hypothesis

The major experiment hypotheses consist of:

- fault events can be triggered only when the aircraft enters “Landing" state;
- each faulty part only occupies one technician who is randomly distributed and has matched skills;
- maintenance business processes are included in the combination of PM/CM and with/without uncertain events;
- for *departure* event, if the downtime for departure is over 4 hours, this flight will be cancelled;
- the degradation of components is not considered when they are stored in a warehouse;
- the items of total repair time (shown in Table 4.4) are assigned with random values or are determined by the database.

As for the last hypothesis, the industrial relevance of values is discussed. Assigning these data with values helps the simulation system understand the relevant degree of time-consuming for parameters. We try our best to provide the values with high industrial relevance. For example, the values for the data *inventoryPrepare* (1-3 h) (when the relevant part is out of stock), *repairingTime* (20-120 min) (the faulty parts that can be repaired on the spot) and *repairTime* (5-30 h) (the faulty parts that have to be repaired at the workshop) are of a certain level of industrial relevance. However, in terms of the data *informationDelivery* and *maintenancePlanning*, the values for them are assumed as 2-5 s, which are less of industrial relevance. The data of *faultIdentification*, *exceptionalEventSSA* and *exceptionalEventSTA* are determined, based on the knowledge stored in the database. If the solution has been stored in the database, they will take zero minutes. If not, they will take either 30 min or one hour. Also, the unit prices (Eqs. 4.8 and 4.10) are preassigned as well. The unit price of downtime cost per second is set to be 2.8 dollars (Pohl 2019). The unit price of stock costs is generally supposed to be 10 dollars per day, which is less of industrial relevance.

5.6.1.2 Maintenance-related Data

The maintenance-related data principally involves flight plans, the available person-power for airports and the maintenance-related time. Table 5.4 depicts the flight plan of aircraft 10001 and 10004, where the unit of flight time is minutes. As the total number of flights is 168, we just show the flight plan of aircraft 10001 and 10004.

In terms of person-power, the available of it is shown in Table 5.5, where 24 hours are divided into *P1* , *P2* , *P3*, *P4* , *P5* and *P6*. Since airport *Paris* is the pivot, more person-power is distributed to this airport.

Table 5.4: Flight plan of aircraft 10001 and 10004

flight no.	aircraft no.	airport	destination	flying time /minutes	departure time	arrival time
BA0304	10001	Lyon	Paris	70	2018-1-1 07:20	2018-1-1 08:30
BF0321	10001	Paris	Lyon	70	2018-1-1 09:30	2018-1-1 10:40
BF0333	10001	Lyon	Paris	70	2018-1-1 11:40	2018-1-1 12:50
BF0334	10001	Paris	Lyon	70	2018-1-1 15:30	2018-1-1 16:40
BF0335	10001	Lyon	Paris	70	2018-1-1 19:00	2018-1-1 20:10
BF0336	10001	Paris	Lyon	70	2018-1-1 21:30	2018-1-1 22:40
BA0800	10004	Lyon	Paris	70	2018-1-1 08:20	2018-1-1 09:30
BA0213	10004	Paris	Boston	475	2018-1-1 12:25	2018-1-2 20:20
BA0143	10004	Boston	Paris	475	2018-1-1 21:30	2018-1-2 05:25
BA0632	10004	Paris	Lyon	70	2018-1-2 06:00	2018-1-2 07:10

Table 5.5: The distribution of labors, where P1 = (00:00 - 05:00), P2 = (05:00 - 07:00), P3 = (07:00 - 14:00), P4 = (14:00 - 17:00), P5 = (17:00 - 20:00) and P6 = (20:00 - 00:00)

airport	P1	P2	P3	P4	P5	P6
Paris	8	1	8	8	10	5
Lyon	2	1	3	3	3	2
London	2	1	3	3	3	2
Berlin	2	1	3	3	3	2
Madrid	2	1	3	3	3	2
Amsterdam	2	1	4	4	3	1
Boston	2	1	5	5	5	2
Marseille	1	1	3	3	3	2
Barcelona	1	1	3	3	3	1
Prague	1	1	3	3	3	1
Manchester	1	1	3	3	3	2
Munich	2	1	3	3	3	1

The equipment information is one of the most important information for aircraft maintenance, which is shown in Table 5.6. Each attribute of equipment has an influence on the repair decision. The detailed explanation about repairing strategy was illustrated in Algorithm 2. It should be noted that the difference between *repair time* and *repairing time* was explained in Section 4.3.3.

Table 5.6: Equipment information

equipment type	changeover time/h	repair time/h	repairing time/min	PM frequency /week	price/dollar
motive_generator	5000	20	80	5	5000
APU_generator	5000	20	80	5	5000
fuel_tanks	1000	15	60	5	5000
fuel_pumps	1000	15	60	5	5000
chassis	800	20	80	6	3000
IRU	5000	5	20	6	1000
SG	5000	5	20	5	8000
FCC	3000	5	20	7	3500
MCP	3000	5	20	7	2000
FMC	3000	5	20	9	15000
TCAS	5000	5	20	9	12000
radar	6000	5	20	5	1000
CDU	3000	10	40	5	10000
APU	44000	20	80	5	70000
engine	6500	20	80	12	30000
nose_cone	69000	70	80	10	90000

5.6.2 Case Study on System Performance: Original model VS. Modified model

Comparison experiments have been conducted in order to investigate the impacts of the modified process for Flight agent (Fig. 5.16) on service level. The run length is set to twelve months, which takes around five hours. Fig. 5.18 shows the simulation results. The data about service level is the average of 10 repetitions' data. The warm-up period is set to five days to avoid initialization bias since the simulation system starts with new planes.

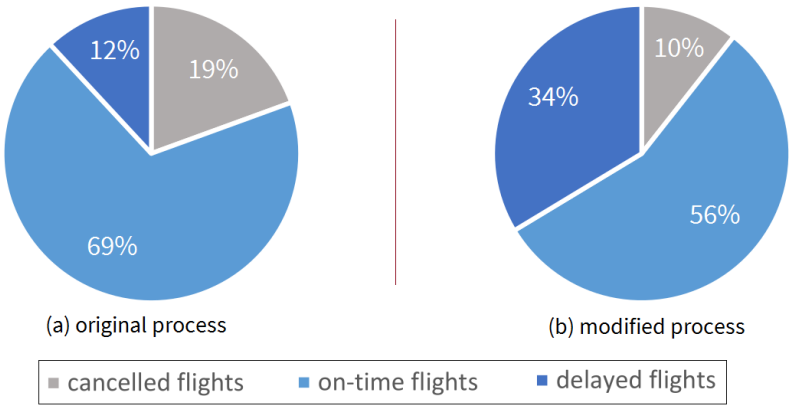

Figure 5.18: The simulation results on the original and modified models of the ABSS

From Fig. 5.18, we can observe that the modified process outperforms the original process on the number of cancelled flights. However, the modified process has failed to surpass the original

process on the numbers of delayed and on-time flights. More precisely, the number of delayed flights has increased from 12% to 34% and the number of on-time flights has decreased from 69% to 56%. The disadvantages of the modified process on delayed and on-time flights cannot illustrate the inefficiency of the modification. In fact, it is quite reasonable to obtain this result. As fewer flights are cancelled in the modified model, it leads to more delayed flights. Delayed flights may result in producing more flights of delay. Therefore, the number of delayed flights has grown. On-time flights could be less for the modified model as well, because delayed flights may cause the "chain reaction" (i.e. one delayed flight maybe causes the next flight delayed). However, if one flight has to be cancelled, it just leads to an increase in the number of cancelled flights but it will start to serve on time in the next time. Thus, less number of on-time flights is appropriate.

Additionally, the efficiency of the modified process can be explained more clearly if the opposite of cancelled flights is considered. 47.37% of cancelled flights have turned to be serviceable flights. Since more flights are serviceable, the resource will become more limited. Therefore, it will definitely influence the numbers of on-time and delayed flights.

To conclude, the modification of Flight agent has assured that Plane agent will not miss signals from Flight agent. It has successfully improved the performance of the ABSS in terms of cancelled flights, which illustrates the feasibility of the correction for the NuSMV model of the simulation system.

5.6.3 Case Study on Maintenance Strategy: FIFO, Optimised and FIFO+Optimized

Simulation experiments are carried out to evaluate the impacts of maintenance strategies of *FIFO*, *Optimized*, and *FIFO+Optimized* (supported by Algorithm 1) on maintenance costs and service level. The run length is set to be three months, which takes around three hours. PM and CM, involving changing parts, purchasing parts, inspecting parts, shifting workers, delayed flights, cancelled flights, etc., will happen within this run length.

Moving average analysis is carried out to set the required number of replications. According to the result (shown in Fig. 5.19), after the $8th$ replication, the moving average lines tend to be more stable. Since the moving average lines fluctuate with a minimum amplitude around the $10th$ replication, the number of replications will be set to be ten to ensure we obtain a better estimate of average. The box plots are used to show the distribution of the results. The impact of different strategies on maintenance costs is illustrated in Fig. 5.20. This figure depicts that three strategies share similar ranges of change on maintenance costs except for the minimum of the strategy of *FIFO+Optimized*. The strategy of *Optimized* does not seem to have any advantages on maintenance costs, regarding all the indicators. Comparing with the strategy of *FIFO*, *FIFO+Optimized* strategy saves the maintenance cost more in general. However, it is possible for *FIFO* strategy to outperform *FIFO+Optimized* strategy in maintenance costs.

In terms of service level, the results about on-time delayed and cancelled flights can be observed

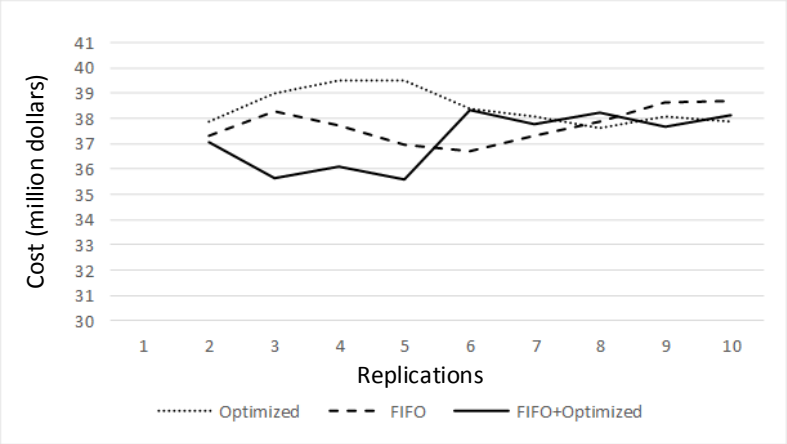

Figure 5.19: The moving averages of three strategies

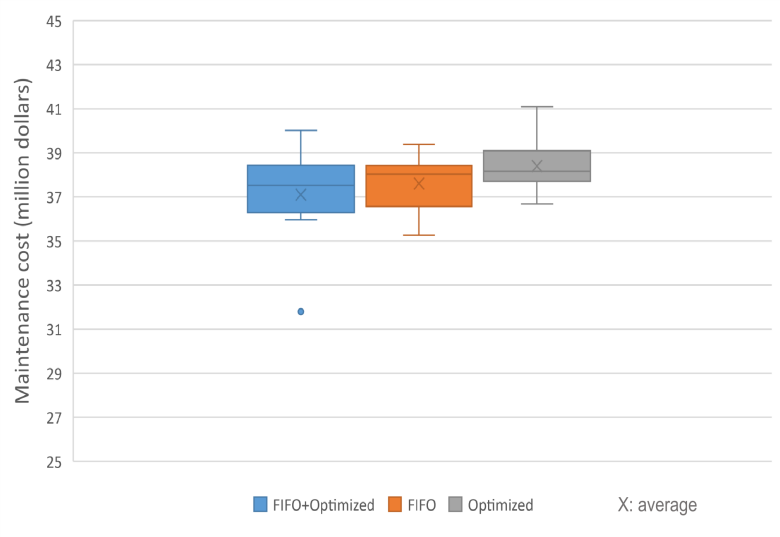

Figure 5.20: The impact of maintenance sequence strategies on maintenance costs

in Fig. 5.21 in detail. Overall, the strategy of *FIFO* outperforms the others. While the strategy of *FIFO+Optimized* has the lowest average of the number of cancelled flights. Comparing with these three strategies, *FIFO* is the most stable strategy in every aspect, which is followed by *Optimized* strategy. The least stable strategy is *FIFO+ Optimized*.

From our results, we can conclude that the best maintenance sequence strategy will vary in the priority of minimizing maintenance costs or maximizing the number of on-time flights. If minimizing maintenance costs is the priority then *FIFO+Optimized* strategy would be the best. This strategy aims to always handle the most urgent maintenance issues in a way. No "starvation

Figure 5.21: The impacts of maintenance sequence strategies on service level

phenomenon" will happen. It always keeps the priority and sequence of maintenance requests

balanced, so this strategy will save the maintenance cost most. If maximizing the number of on-time flights is the priority then *FIFO* strategy would be the best. This strategy handles the requests from aircraft depending on the order of the request arriving time. Therefore, the phenomenon of "the least urgent request cannot obtain resources" can be avoided. The number of on-time flights will be increased. So, this strategy results in the largest number of on-time flights. As a result, the efficiency of maintenance strategies should be investigated from different aspects. Different strategies should be adapted to different aims.

5.7 Discussion

The results of the simulation experiments demonstrate the efficiency of our method. The approach allows us to not only verify global behaviours but check operational behaviours. It focuses on how to verify the system behaviour from different points of view rather than extend the formal logic, which aims to improve the efficiency of model checking.

The original design of Flight agent (Fig. 5.3) seems to be rational. There are two ways to send signals to Plane agent. One is to send signals when there is no delay, the other is to receive the delay signal from STA agent, then continuing checking the state of Plane agent until it is ready, finally sending the scheduling signals. We have not noticed that the delay of one flight may have an influence on the departure of next flights that have no maintenance requests. For example, the downtime of one plane has exceeded the time slot of its next flight. It means this plane will be demanded to serve for the flight after next even if it has not arrived at the airport of the next flight. In this case, the downtime of this plane for the next flight is equal to zero. If sending scheduling signals only depends on delay signals, this plane will miss the signal for the flight after next regarding the scheduling. We can not ensure that one plane will depart on time even if it does not need repairing. Therefore, the original design of Flight agent should be improved. Checking the state of Plane agent is definitely necessary before sending scheduling signals.

Our way of verifying the global behaviour is quite similar to the work of Bentahar et al. (2013). However, we have identified the limits of their work. Firstly, their approach is not suitable to verify the system behaviour where transitions are associated with conditions. The detailed discussion can be seen in the first paragraph of Section 5.2.4. Subsequently, alternating Büchi automaton was chosen to express the control behaviour, since it considered both universal choices and existential choices. The core concept of alternating Büchi automaton is the partial transition function (Vardi 1995). The partial function implies that one state can be transited to two states concurrently with one message in the system. However, the verified system has not been involved with messages. If their system is modified into the system of transitions with conditions. The property of verifying the sequence discussed above will never pass. So, in our opinion, there is no point in choosing alternating Büchi automaton. Finally, the model transformation was provided in their work, but

the messages-related transformation was not discussed. Two fundamental differences between our work and their work are identified. One is that we clarify the limit of the way we verify the global behaviour where the transitions with messages are not considered, the other is that we propose the approach on the verification for the operational behaviour of the system as a complement. Additionally, we provide a complete process for transforming Büchi automaton into Kripke model.

Another advantage of our approach is that it enables us to check the satisfactory of the global behaviour of the ABSS with respect to standards. ABSSs of different domains address different key properties. The difference in properties can be represented by different control behaviours. Thus, a global sense of the satisfactory of the ABSS can be analyzed.

As a result, we propose an approach to verify the behaviours of the ABSS from the system itself point of view. It permits system designers who know systems well but are not experts in analyzing formal logics to better verify the systems of interest.

5.8 Summary

In this chapter, we discussed how to verify the behaviours of the ABSS from the system itself point of view. System behaviours are composed of global and operational behaviours. The division of system behaviours allows us to investigate the satisfactory of systems from both the abstract and the detailed point of view.

The Büchi automaton described system behaviours and the control behaviour, which satisfied the continuous invocation of the plane's service. The meta-transformation between Büchi automaton and Kripke structure (the theory of NuSMV model) was provided, aiming at giving the theoretic support between system behaviours and the corresponding NuSMV model. A mapping between the ABSS and the NuSMV model was then created. The LTL and CTL were used to formulate the specifications, in order to verify the conformity of system behaviours. Simulation experiments were carried out to compare the performances of the original and modified processes, which illustrates the efficiency of the modified process. Besides, the experiments analysing the impact of scheduling strategies on operational support for aircraft maintenance were performed as well. It aids in choosing the best scheduling strategy under specific circumstance.

The proposed approach has attempted to check the global behaviour of the ABSS, which lays the foundation for verifying the system behaviour from the abstract level without concerning any details. It can be recognized as a general way of verifying behaviours of the ABSS implemented by *Anylogic*. This chapter gave also a concrete case study of formal methods in practice and demonstrated the efficiency of formal methods in improving systems design.

Chapter 6

Autonomous System based Fault Diagnoses

A more reliable and robust ABSS is obtained after checking the logics of the system by NuSMV. Evidently, simulation experiments on different scenarios can be performed. A wealth of information is produced in the course of simulating. While neglecting this “precious" maintenance-related information can not let us achieve the maintenance efficiency as it should be. Data mining techniques can be integrated into the simulation system to analyze and predict data. The outcome of predicting data could have a positive impact on correcting the behaviours of the ABSS. This motivates us to concentrate on the combination of simulation with data mining. Fault diagnosis for aircraft involves lots of information. So, this chapter emphasizes how to combine the ABSS and data mining to analyze the data, aiming at improving the efficiency of the fault diagnosis for aircraft.

6.1 Introduction

The advanced technology and complex system design have been broadly used in modern aircraft systems. Fault diagnoses become increasingly difficult because of the intractable faults and the complex connections between systems. Making use of an automatic classification scheme helps remove the need for the detailed analysis of the complex fault information (Mechefske and Mathew 1992). Early fault prediction is a proven technique in achieving high system reliability (Khoshgoftaar and Seliya 2003). Thus, effective FCP can significantly improve the efficiency of fault diagnoses, which finally helps achieve highly efficient maintenance for aircraft. The function demands of the FCP for aircraft maintenance include, but not limited to,

- building knowledge models on faults to store the fault information;
- automating the process of the FCP for providing a fast-response mechanism on aircraft maintenance under the big data environment;
- supporting the analysis of the FCP on the aircraft of new types, in order to prevent them from stopping working as far as possible.

The traditional rules of the FCP for aircraft maintenance are principally from airworthy standards. These rules are refined from thousands of real cases of accidents about aircraft and the design theory (ASD 2019). Many new techniques have been attempted to improve the efficiency about the FCP for aircraft maintenance (Sahin et al. 2007; Jiao et al. 2017; Wan et al. 2018), whereas few of researchers have considered the case that the historic data for fault diagnoses is either not sufficient to be analysed or only valid for specific circumstances. This may raise big issues. For example, if the aircraft of a new type comes out, aircraft manufacturers simply have some limited data generated before delivering them to clients. They may suffer a lack of appropriate ways of fault diagnoses. Indeed, most of the rules still work, but how to investigate the functionalities and interactions of the new components remains unsolved. In addition, acquiring sensitive data like data for aircraft maintenance is exceptionally difficult. Thus, the shortage of data is impeding researchers to study further operational support for aircraft maintenance. Therefore, our study focuses on **how to improve the efficiency of the fault diagnosis for aircraft maintenance without sufficient data**.

The ABSS for aircraft maintenance has been accomplished in Chapter 4, which allows us to build the relationship between equipment failures and reliability data. Thus, in this chapter, the ABSS on the fault diagnosis for aircraft maintenance will be adapted from the ABSS proposed.

Fuzzy-Rough Nearest Neighbour (FRNN) is a promising approach for classifying test objects and predicting their decision values. FRNN is principally based on fuzzy set theory, rough set theory and K Nearest Neighbour (KNN) algorithm. Fuzzy set theory (Zadeh 1965) expressed vague information by utilising a class of objects with a continuum of grades of membership, which was

used for the linguistic representation of patterns, leading to a fuzzy granulation of the feature space. Rough set theory (Pawlak 1982) provided approximations of concepts for the incomplete information, obtaining dependency rules which model informative regions in the granulated feature space. KNN is a well-known technique for the data classification and is one of the simplest approaches for data analysis (Peterson 2009). In ABSS, the reliability information and the fault location information can be considered as the condition and decision attributes respectively. The information represents the vagueness and the incompleteness, because of the random process of generating fault location information. Hence, FRNN is a better fit to predict the relationship between reliability information and fault location information.

The simulation model is built of practices, rules and data in the real world, which can bring out the interactions between involved objects. Comprehensive knowledge can be learned from simulation data. The knowledge learned from simulation data is capable of defining the rules of the fault diagnoses. The rules of the fault diagnoses state the way how we are able to find the probably faulty part. The knowledge means the decision table obtained from simulation data. The table gives the relationship between the reliability-related information and the fault location information. The former is obtained by analysing the simulation data. The latter is gained from the stochastic mechanism provided by simulators. In the ABSS, each fault happening on one part is always accompanied by its relevant reliability-related information. Hence, the reliability-related information is a quite useful clue in finding the probably faulty part. The FRNN approach can be dedicated to learning the knowledge, in order to provide the rules of the fault diagnoses. Therefore, knowledge enables us to define the rules of fault diagnoses.

Moreover, in order to integrate the FRNN approach into the ABSS for the FCP, the major idea of the approach proposed is illustrated in a framework. Subsequently, two models are proposed from different abstract levels, which lay the foundation for implementing the approach. The next step is to technologically accomplish the integration. Finally. simulation experiments are performed, in order to demonstrate the feasibility of the approach proposed.

To the best of our knowledge, it is the first attempt to combine the ABSS with the fuzzy-rough sets theory. The approach proposed provides an alternative to improve the fault diagnosis for the aircraft of new types and to accomplish the automatic self-improvement for aircraft maintenance (Liu et al. 2019a).

6.2 Literature Review

6.2.1 Fault Diagnosis of Aircraft Maintenance

This subsection provides a comprehensive literature review on fault diagnoses for aircraft maintenance. The scope of researches is limited to the domain of aircraft maintenance. A synthesis analysis of reviewed papers is summarised in Table 6.1. Several aspects are proposed to review

the papers, which include proposes simulation models, provides an automated analysis approach, model purposes, and relevance to our research. Moreover, the relevance is evaluated with different levels: low, medium and high.

According to this table, researches seldom simultaneously consider simulation models and fault diagnosis approaches. Fortunately, papers 1 and 3 (Sobie, Freitas, and Nicolai 2018; Wan et al. 2018) got these two approaches together. However, they do not provide an automated analysis method. On the other hand, papers 4 and 8 (Naderi and Khorasani 2018; Bateman, Noura, and Ouladsine 2011) involve simulation models, their models are employed to evaluate the feasibility of approaches. Obviously, no one provides an automated analysis approach on fault diagnoses for aircraft maintenance. Researchers tend to analyse all kinds of fault reports before performing the analysis of fault diagnoses, which causes their approaches partially automated. This is because the reports of natural languages or information redundancy make it hard to easily extract useful information. A wide variety of approaches have been used for fault diagnoses. This also shows that this area is becoming more and more attracting. As for model purposes, most of the researches focus on the fault diagnosis for aircraft engines. This is not only because the engine is the most complex component for aircraft but it generates lots of information. The last aspect is associated with the relevance of our research. Few resources from the literature can be reused as no one combines the ABSS with the FRNN approach. This is the reason why no paper is given a relevance of a high level. Even so, the relevance analysis is still required because others aspects like scopes, advantages, disadvantages etc., can be analysed.

To summarise, the existing works are prone to separating the fault diagnosis from the operating of aircraft, which can cause the analysis for the fault diagnosis with insufficient historic data or the data only valid for specific circumstances delayed or impeded. Another difference is that few researchers are devoted to automating the process of the FCP so that fast responses are possible to be made, in order to deal with real-time issues on operational support for aircraft maintenance. Therefore, we are motivated to build a simulation system with the FRNN approach enabling us to automatically support the fault diagnosis for aircraft maintenance without sufficient historical data.

Table 6.1: An overview of related papers - A (proposes simulation models) and B (provides an automated analysis approach)

Article	A	B	Applied Methods	Model Purposes	Relevance to our research
1.(Sobie, Freitas, and Nicolai 2018)	yes	no	nearest-neighbour dynamics time wrapping	bearing fault classification	medium, similar ideas, no fault prediction, partial automated analysis processes.
2.(Jia et al. 2018)	no	no	complex empirical mode decomposition and random forest theory	fault diagnosis for large wind turbines	low, similar in scope, accurate fault information extraction.
3.(Wan et al. 2018)	yes	no	particle swarm optimisation and back propagation neural network	fault diagnosis for aero-engine fuel regulators	medium, similar ideas, similar forms of representing fault information (fault characteristic table), partial automated analysis processes.
4.(Naderi and Khorasani 2018)	yes	no	data-driven fault detection, isolation and estimation method using Markov parameters	fault detection, isolation, and estimation for aircraft gas turbine engine actuators and sensors	medium, similar in scope, well-suited for real applications, similar ways of evaluation, less restrictive assumption, different ideas, partial automated analysis processes.
5.(Jiao et al. 2017)	no	no	empirical decomposition and probabilistic neural networks	fault diagnosis for airborne fuel pumps	low, similar topics, fast convergence, high efficiency and high performance and recognition for diagnosing typical faults, different ideas, no automated analysis process.
6.(Korvesis 2017)	no	no	neural networks and a regression-based approach	developing data-driven model on predictive maintenance for aircraft	medium, similar ins scope, focusing on fault prediction and investigating data from operational support for aircraft maintenance, different ideas, partially automated analysis processes.
7.(Tayarani-Bathaie, Vanini, and Khorasani 2014)	no	no	dynamic neural networks	fault diagnosis for gas turbine engines	low, similar in scope, concrete measurable variables for studying fault diagnosis schema, different ideas.
8.(Bateman, Noura, and Ouladsine 2011)	yes	no	building Fault Detection Diagnosis (FDD) system based a nonlinear model of the aircraft	fault diagnosis and Fault-Tolerant Control (FTC) for an unmanned aerial vehicle	medium, similar in scope, concrete fault detection diagnosis system, a joint FDD-FTC strategy, different ideas.
9.(Kim, Choi, and Kim 2008)	no	no	fuzzy-tuning interacting multiple model	fault diagnosis for aircraft actuators	low, similar in scope, smoothing the fault detection process, low probability of false fault detection, different ideas.

6.2.2 FRNN-related Theory

6.2.2.1 Fuzzy Set Theory

Fuzzy set theory uses given degrees to define the relationships between objects and their memberships. The definitions of fuzzy sets are manifold. In this paper, we introduce the definition of Richard Jensen's (Jensen and Cornelis 2011), which is shown as follows:

A fuzzy set in X is an $X \rightarrow [0,1]$ mapping, while a fuzzy relation in X is a fuzzy set in $X \times X$. For all y in X, R-forest of y is fuzzy set Ry defined by

$$Ry(x) = R(x,y) \tag{6.1}$$

for all x in X. If R is a reflexive and symmetric fuzzy relation, that is,

$$Ry(x,x) = 1 \tag{6.2}$$

$$R(x,y) = R(y,x) \tag{6.3}$$

holds for all x and y in X, then R is called a fuzzy tolerance relation.

If X is finite, the cardinality of A is calculated by

$$|A| = \sum_{x \in X} A(x) \tag{6.4}$$

6.2.2.2 Rough Set Theory

RST (Rough Set Theory) is a powerful tool for rule induction from incomplete data sets and for concisely extracting information from a domain. Rough set-based data analysis approaches have been widely, successfully employed in fields like medicine, web and text mining, signal and image processing, software engineering, etc. (Pawlak, Polkowski, and Skowron 2008). As RST has been extended and generalized in different ways, in this paper, we use the definition of RST from Jensen and Cornelis (2011) as well, which is explained as follows:

Let(X,Ω) be an information system, where X is a non-empty set of finite objects (the universe) and Ω is a non-empty finite set of attributes such that a: $X \rightarrow V_a$ for every $a \in \Omega$. V_a is the set of values that attribute a may take. With any $B \subseteq \Omega$ there is an associated equivalence relation R_B:

$$R_B = \{(x,y) \in X^2 | \forall a \in B, a(x) = a(y)\} \tag{6.5}$$

If $(x,y) \in R_B$, then x and y are indiscernible by attributes from B. The equivalence classes of the

B-indiscernibility relation are denoted as $[x]_B$. Let $A \subseteq X$. A can be approximated using the information contained within B by constructing the $B-lower$ and $B-upper$ approximations of A:

$$R_B \downarrow A = \{x \in X | [x]_B \subseteq A\} \tag{6.6}$$

$$R_B \uparrow A = \{x \in X | [x]_B \cap A \neq \emptyset\} \tag{6.7}$$

Tuple $\langle R_B \downarrow A, R_B \uparrow A \rangle$ is called a rough set.

6.2.2.3 Fuzzy-rough Set Theory

Fuzzy-rough set theory benefits from the advantages of fuzzy and rough set theories, where both the membership of different degrees, and the lower and upper approximations are considered. A general definition (Radzikowska and Kerre 2002) is illustrated in the following:

$$(R \downarrow A)(x) = \inf_{y \in X} L(R(x,y), A(y)) \tag{6.8}$$

$$(R \uparrow A)(x) = \sup_{y \in X} T(R(x,y), A(y)) \tag{6.9}$$

where, L and T are an implicator and a t-norm respectively. In this paper, T_M is defined by $T_M(x,y) = min(x,y)$, where $x, y \in [0,1]$. L_M is defined by $L_M(x,y) = max(1-x,y)$, where $x, y \in [0,1]$.

Given fuzzy tolerance relation R and fuzzy set A, tuple $\langle R \downarrow A(x), R \uparrow A(x) \rangle$ is called a fuzzy-rough set.

6.2.3 Fault Classification and Prediction Framework

In this section, we propose a framework for fault classification and prediction analysis. Fig. 6.1 provides a global view about how to combine FRNN and the ABSS model, in order to improve the maintenance efficiency and decrease the maintenance cost. It consists of the ability to:

- specify the motivation about combining FRNN and the simulation;
- specify the approach of introducing FRNN into the ABSS model;
- specify FRNN-ABSS model with components and connections.

The FRNN is a novel approach that enables us to further improve the maintenance efficiency from the data point of view, especially when the data shows vagueness and incompleteness. The ABSS is able to simulate the whole process of aircraft maintenance, which can produce large amounts of data as well. The data produced by the simulation model can be learned to support decision-making. Aiming at combing FRNN and the ABSS model, the reliability handbook (Bombardier. 2004) is used as a basis to propose key indicators. These indicators are media to connect

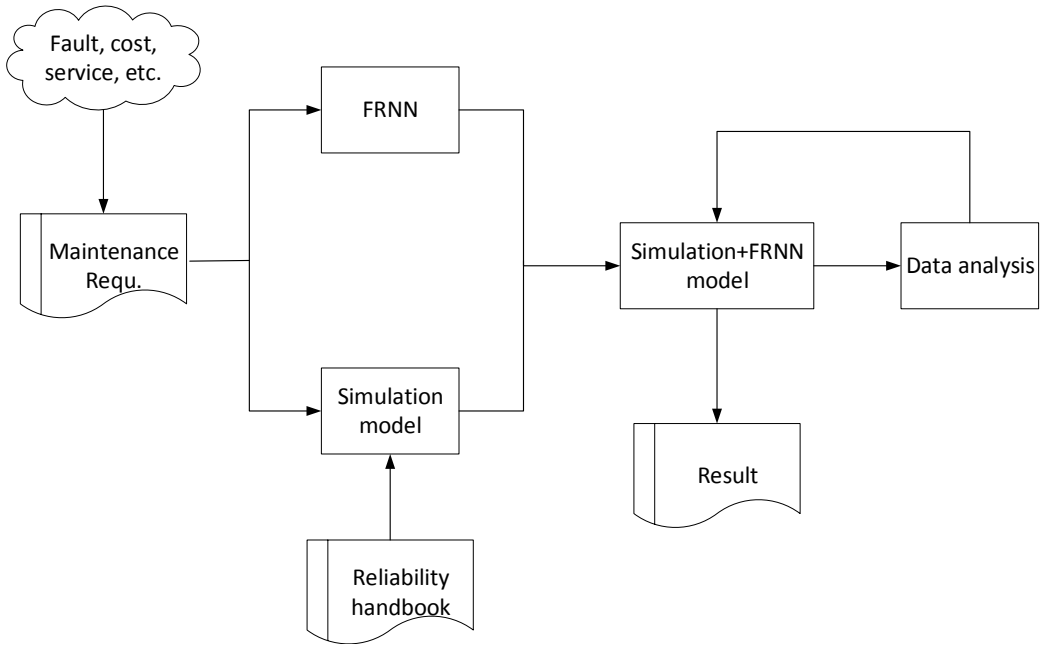

Figure 6.1: The framework for fault classification and prediction

FRNN and the ABSS model. FRNN-ABSS model is then proposed to realize the improvement of maintenance efficiency from the perspective of data. Since this model is capable of automatically getting knowledge from simulation data, the issues of the self-improvement for aircraft maintenance can be performed. Therefore, it is possible for engineers to better define rules of fault diagnoses with the help of the knowledge learned from simulation data.

6.3 FRNN-ABSS Model

In order to better describe FRNN-ABSS model, some important concepts and their relationships are explained first. Fig. 6.2 shows FRNN-ABSS conceptual model. Illustrated by this figure, a number of concepts are concerned, including *flight*, *aircraft*, *equipment*, *requirement*, *service strategy*, *service task*, *supply chain*, *quality* and *FRNN*. In our simulation environment, the concepts of the *flight*, *aircraft* and *equipment* are used to describe the whole process of the operation for aircraft. The *requirement* presents the needs for different maintenance types like scheduled maintenance and unscheduled maintenance, which triggers the maintenance process. The concepts of *service strategy*, *service task*, *supply chain* and *quality* are associated with maintenance processes. The *FRNN* is designed as the module of reliability analysis. Since the manufacturing of aircraft has been improved significantly, most of the equipment is intelligent enough to alarm themselves. The fault location is much easier than before. However, we are not able to know exactly the functionality

of the equipment for the aircraft of new types. We even lack the knowledge of the performance of the combination of equipment. The FRNN approach concentrates on classifying and predicting objects, which enables us to build the relationship between the reliability-related information and equipment failures. Finally, the results from *FRNN* will be synthesised and treated as recommendations sent to *Service Strategy* and *Service Task*. The corresponding FRNN-ABSS model is shown in Fig. 6.3. The details of the ABSS model can be seen in Chapter 4.

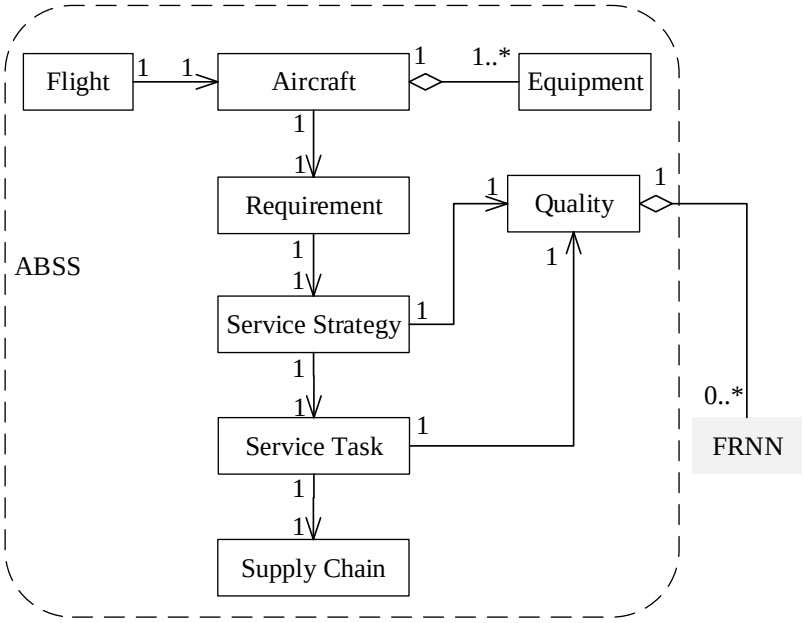

Figure 6.2: FRNN-ABSS conceptual model

6.4 Component Fault Classification and Prediction

In order to accomplish the FCP for aircraft maintenance, three major issues should be considered. The first issue is to understand the relationship between the FRNN approach and the ABSS. The FRNN process is the reliability analysis module in the ABSS (mentioned in Section 3.1). The ABSS directly made a decision about maintenance issues when the FRNN approach was not considered. Now, the ABSS makes a decision after having investigated the prediction results from the FRNN process. Thus, combining the FRNN approach with the ABSS should make sure that the latter is able to provide effective input for the former and the former can deliver the effective output for the latter. The input for the FRNN approach is the decision table information and the real-time reliability information for parts. The input for the ABSS is the specific decision information (which part does not work) regarding parts. The key to combine the FRNN approach with the ABSS is the decision table. The basic elements in this table are proposed in Section 6.4.1. The strategies on

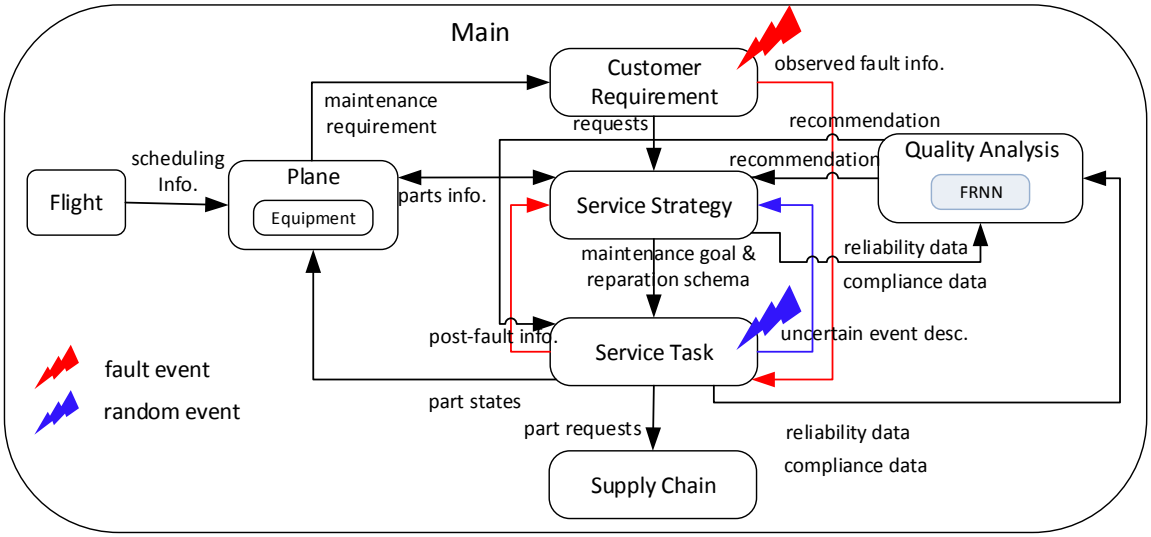

Figure 6.3: FRNN-ABSS model

the fault analysis for engines are the second issue (Section 6.4.2). The last one is about how to technologically fulfil the function of the FCP, which is discussed in Section 6.4.3.

6.4.1 Reliability-related Indicators

The reliability report for aircraft is a collection of historic information about fault diagnoses, which is employed to analyse faults by associating these experiences with the real-time condition of equipment states. It is issued by aircraft manufacturers and airlines at regular time (Bombardier. 2004), which includes the number of faults on parts, the number of removal parts (scheduled removing and unscheduled removing) on equipment, the flying time and the dispatch times, etc. China Civil Aviation stated that the performance of aircraft can be evaluated by the crew report, the delay of flights, the rate of removing parts, the use time of parts, etc (China Civil Aviation 2019). Considering the limit of the simulation data collected, several indicators are chosen, including DIR (Departure Interruption Rate), URR (Unscheduled Removal Rate) and MTBUR (Mean Time Between Unscheduled Repair). DIR reflects the delay level of flights. URR and MTBUR are the traditional statistical indicators for reliability analysis.

The reliability office of Shanghai airline provided the definitions of DIR, URR and MTBUR (Shanghai airline 2019), which are illustrated as follows:

DIR implies the number of delayed flights and cancelled flights (because of technical issues) is divided by the amount of all departures during a specific period;

URR indicates the number of unscheduled repairs divided by the flying hours and then multiplied by the number of parts;

MTBUR denotes the amount of use time of parts divided by the number of unscheduled repairs.

Thus, the indicators of DIR, URR and MTBUR will be treated as the conditional attributes of the decision table.

6.4.2 The Strategies on the Fault Analysis for Engines

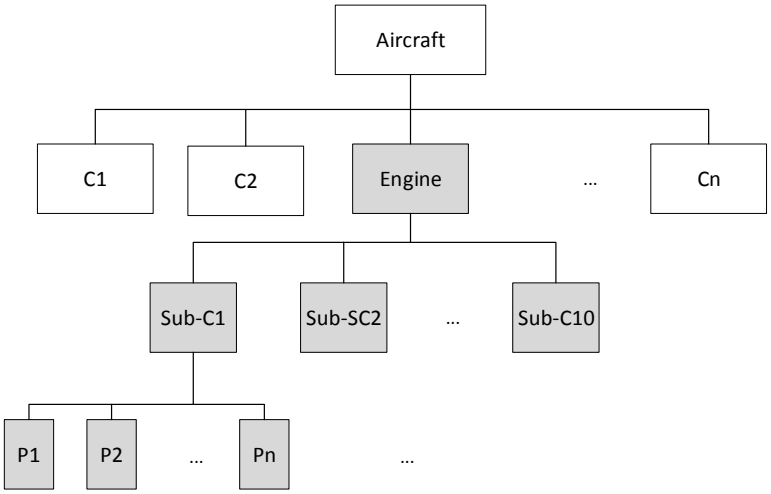

Figure 6.4: The aircraft structure

The structure of aircraft is shown in Fig. 6.4, where the engine component is highlighted. We assume that the warning of the faulty sub-components for the engine can be issued by the controller system of the aircraft. So, the FRNN approach is utilised to predict the fault location in sub-components.

Training sets are the input of the FRNN approach. Two strategies are given for generating training sets, which are shown in Fig. 6.5. *Random* and *Sequential* strategies are considered, which means the inspecting process of parts will be executed in a random or a sequential way. The training sets are generated with the help of Algorithm 4.

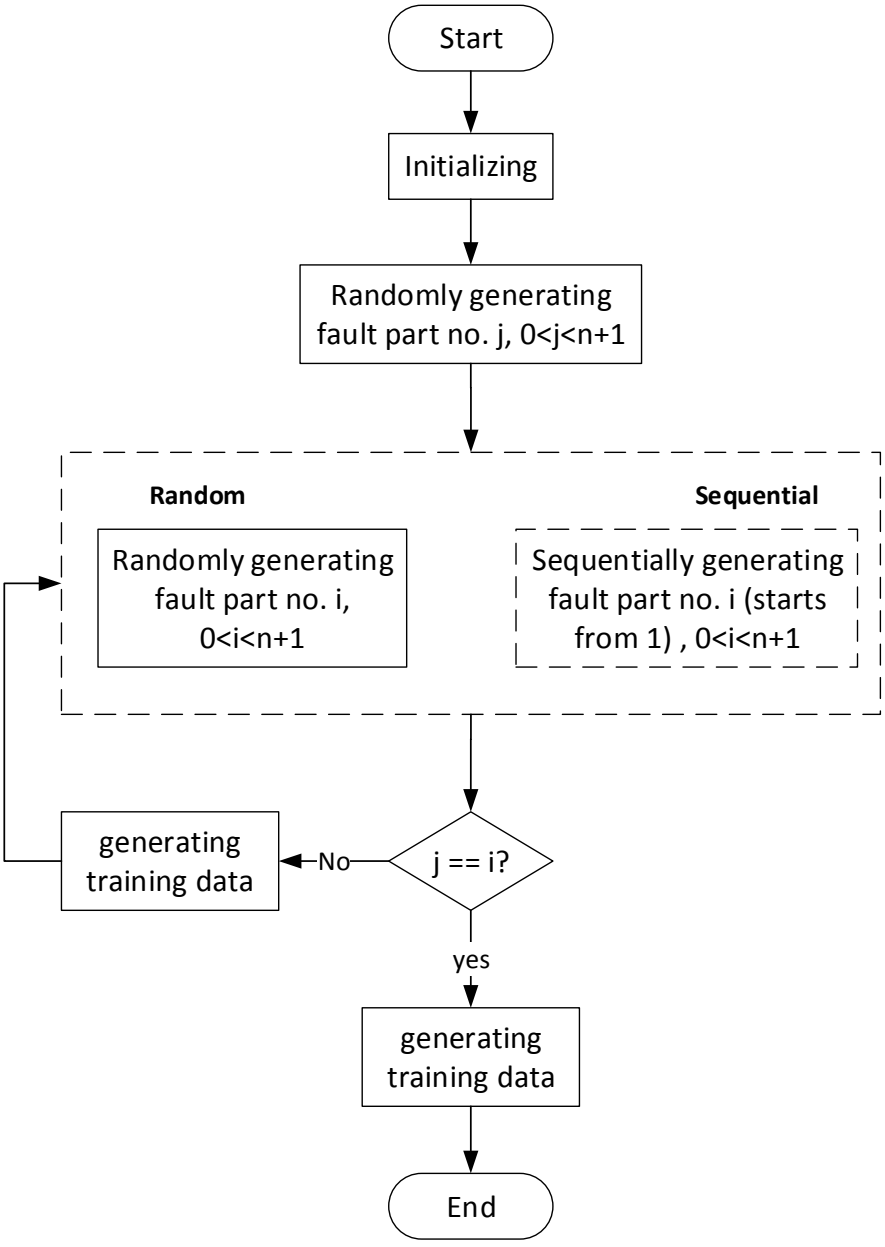

Figure 6.5: Two strategies for generating training sets

6.4.3 Fuzzy-Rough Nearest Neighbour Classification and Prediction

The objective of the FCP for aircraft maintenance is to find the probably faulty part. Five steps are involved, in order to achieve this aim. First of all, the reliability-related information (DIR, URR, MTBUR) about parts belonging to the dysfunctional component should be collected (Section 6.4.3.1). The information collected can be treated as the test data. The second step is to calculate the fuzzy tolerance relation between the test data and the training data (knowledge learned from simulation data) with respect to K nearest neighbours (Section 6.4.3.2). Neighbours are sorted out by calculating the similarities between the instance of the test data and all the instances of the training data. The next step is to calculate the lower and upper approximations of one instance of the test data regarding K nearest neighbours (Section 6.4.3.3). In the following, the averages on the lower and upper approximations of the instance regarding K nearest neighbours are calculated. The decision attribute corresponding to the biggest average will be assigned to the instance (Section 6.4.3.4). The last step is to apply the results predicted by the FRNN approach into the process of the FCP for aircraft maintenance (Section 6.4.3.5). The detailed description of this approach is illustrated in the following.

6.4.3.1 Decision Table Generation

The decision table is the input of the FRNN approach, consisting of conditional attributes and decision attributes. It is the media connecting the ABSS and the FRNN approach. The conditional attributes are determined in Section 6.4.1. The decision attribute is the result of the part inspection, which means it tells whether the inspected part is broken or not.

In order to obtain the decision table, a general process is proposed in algorithm 4. The major idea is to gather the data of use time, flying hours, departure times, delayed flights and cancelled flights to calculate the values of DIR, URR and MTBUR for relevant parts. This process will also be used to generate test data because it is concerned with these indicators as well.

6.4.3.2 Similarity Calculation

The second step is to calculate the fuzzy tolerance relation R between training data and test data. Jensen and Cornelis (2011) has concluded many options to construct R. Finally, they have chosen the way of calculating R as follows:

$$R(x,y) = \min_{a\in\Omega} R_a(x,y) \tag{6.10}$$

$$R_a(x,y) = 1 - \left|\frac{a(x)-a(y)}{a_{max}-a_{min}}\right| \tag{6.11}$$

where, $R_a(x,y)$ is the degree to which objects x and y are similar for attribute a; a_{max} and a_{min} are the maximal and minimal occurring value of that attribute. Since they have illustrated the feasibility

Algorithm 4: Decision table generation

input : table urr(key, useTime, flyingHours), table dir(key, depatureTimes)
// The key implies the string of the aircraft type, aircraft no., sub module no. and fault part no.
output : decision table.

// Collecting raw data for building attributes.
// faultComponent.IsTarget() means the component in which the FRNN approach will be used.

```
if faultComponent.IsTarget() then
    for i = 0 to faultComponent.subComponents.size() do
        String key = getKey();
        String useTime = getCurrentUseTime(part(i));
        String flyingHours = getFlyingHours(part(i));
        writeToExcel(key, useTime, flyingHours);
    end
end
if flight.IsDelayed()|| flight.IsCanceled() then
    if faultComponent.IsTarge() then
        String key = getKey();
        String departureTimes = getTotalDepartureTimes(aircraft(i));
        writeToExcel(key, departureTimes);
    end
end
```

// Generating the decision table: calculating DIR, URR and MTBUR.

$DIR_{part(i)} = count\_tableDir(key)/max\_departureTimes(key)$;

// compoentsNumber(key) indicates the number of components corresponding to aircraft types

$URR_{part(i)} = count\_tableUrr(key)/max\_flyingHours(key) * compoentsNumber(key)$;

$MTBUR_{part(i)} = max\_useTime(key)/count\_tableUrr(key)$;

// DA means the value of decision attributes

writeToExcel(dir, urr, mtbur, DA);

of the FRNN approach for this option, we take this option as well. In our case, Ω includes the conditional attributes of the DIR, URR and MTBUR.

6.4.3.3 Approximation Calculation

The definitions of the lower and upper approximations of a fuzzy set A have been discussed in Section 6.2.2.3. Here, the definitions of approximations have been adapted from the former ones, where the condition is that the K nearest neighbours have been obtained. The definitions, shown as follows, are the major idea about how to calculate the approximation in this paper.

$$(R \downarrow c)(x) = \inf_{y \in X} max(1 - R(x,y), A(y)) \tag{6.12}$$

$$(R \uparrow c)(x) = \sup_{y \in X} min(R(x,y), A(y)) \tag{6.13}$$

where c is the decision class; x is the unclassified object (test data); $A(y)$ is to get the value of c corresponding to training data y.

6.4.3.4 Fuzzy-Rough Nearest Neighbour Prediction

The prediction of decision value of test data can be carried out when the lower and upper approximations have been calculated, The decision class of the maximum mean value of the lower and upper approximations will be chosen as the predictive decision class for test data.

$$c = \max_{c \in C} \frac{(R \downarrow c)(x) + (R \uparrow c)(x)}{2} \tag{6.14}$$

6.4.3.5 The Fault Prediction Process for Aircraft Maintenance

The FRNN prediction process takes real-time data for parts and generates testing sets, which is illustrated in Fig. 6.6. Training sets and testing sets are the inputs of the FRNN approach. The possible set of parts is delivered. The next step is to iterate this set. If the fault part has not been correctly predicted, the rest of the parts will continue to be checked until the right one is chosen.

It should be noted that the above five steps have been implemented as Java code, which allows us to perform the automatic analysis process on the FCP for aircraft maintenance.

6.5 Case Study

6.5.1 Simulation Scenario

The simulation starts from flight schedules. The simulation configuration remains the same as in Section 5.6.1. The engine is the critical component in the aircraft, which contains millions of parts.

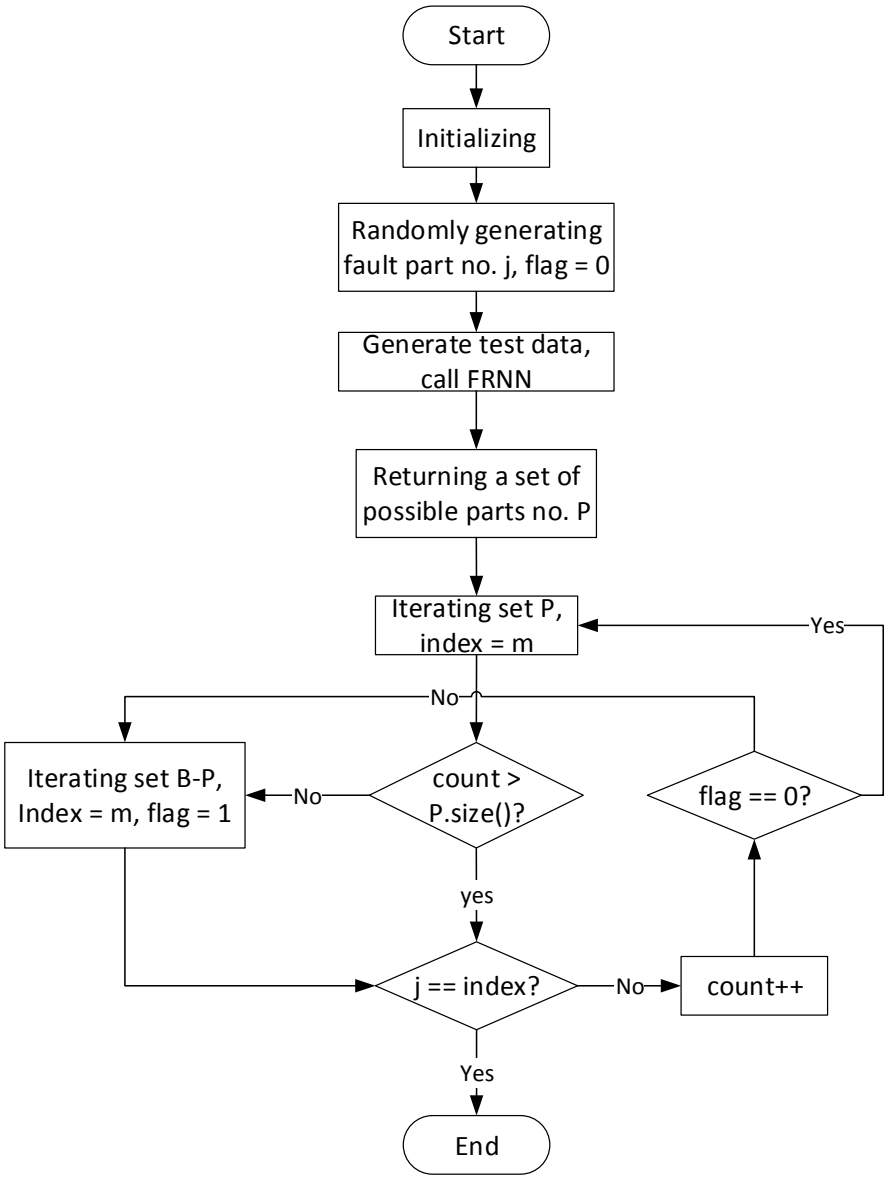

Figure 6.6: The FRNN prediction process

Even though sensors are used to control the major part of the engine, some of the components are still not under control. The fault location of engines can waste some of the precious time. Hence, we choose the engine as the target component where the repairing work will involve the FRNN approach, in order to improve the efficiency of the fault location of engines. In this simulation environment, the engine is made of 10 sub-modules containing 73 parts (NASA 2019) (relatively

high level of description). Furthermore, this simulation system with the FRNN approach is still implemented by Anylogic PLE 8.2.3. The screenshot of this simulation system is shown in Fig. 6.7.

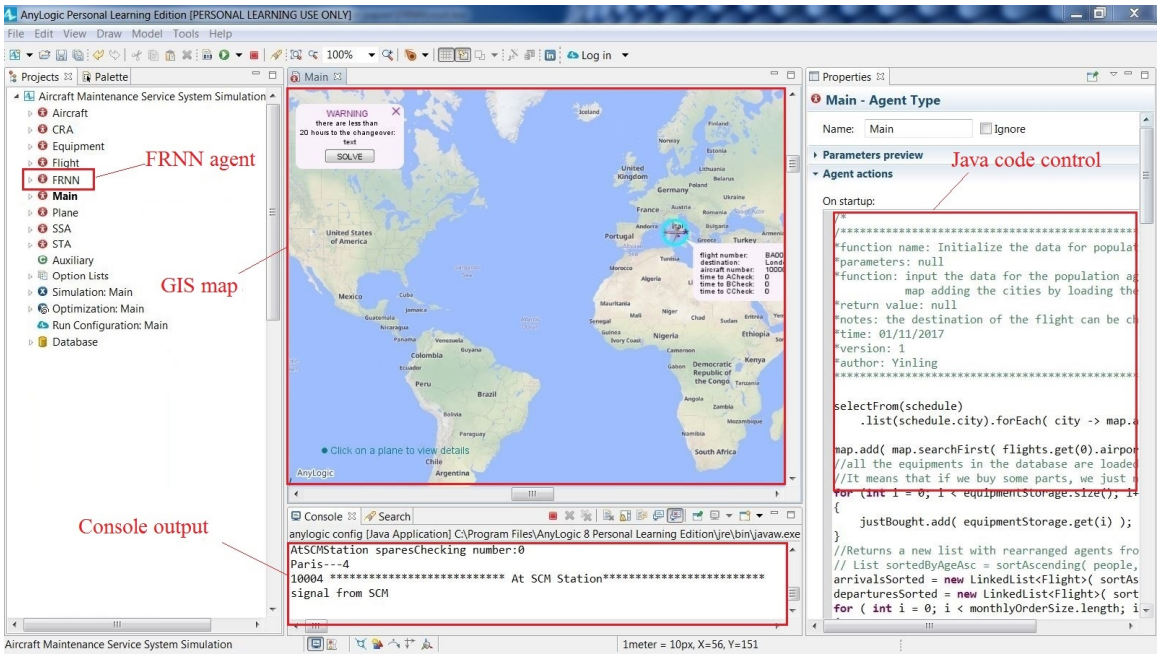

Figure 6.7: The aircraft maintenance service simulation system integrating the FRNN approach

6.5.2 Experiment

Simulation experiments involve two steps. The first step is to explore the impact of K neighbours on the maintenance cost for different strategies. Optimal Ks will then be determined to start the second step which investigates the impact of different strategies on maintenance costs and service level based on the optimal Ks.

The impact of the number of neighbours on the maintenance efficiency is studied by a set of experiments. The simulation starts from 7:20, 1^{th} January, 2017 and stops at 7:20, 1^{th} December 2017, the first 9 months of which are used to generate training sets. The FRNN analysis is performed in the last two months. Here, 15 experiments (K = 1, 2, ..., 15) are conducted for each strategy. The experiment of each K replicates 5 times in order to avoid chanciness. In addition, the experiments without using the FRNN approach are carried out for comparing the performances of maintenance.

The warm-up period is set to five days to avoid initialisation bias since the simulation system starts with a new environment. The results of the impact of K neighbours on maintenance costs

are shown in Figs. 6.8 and 6.9 where black broken and red lines mean the experiments are conducted with/without the FRNN approach respectively. The performance of the FRNN approach on *Random* strategy (Fig. 6.8) is much better than that of *Sequential* strategy (Fig. 6.9). Fig. 6.8 implies that the FRNN approach saves the maintenance cost most when the condition is $K = 15$. Fig. 6.9 shows the choice of K is rather important because more than half of Ks lead to lower efficiency of the FRNN approach. The FRNN approach saves the maintenance cost most when $K = 13$. Therefore, the following analysis on maintenance costs and service level is based on the conditions of $K = 15$ (*Random* strategy) and $K = 13$ (*Sequential* strategy). It should be noted that since our objective is to illustrate the efficiency of this approach and simulation experiments take time (every single simulation experiment takes around 4 hours), we decide to set the maximum of K as 15 instead of continuously increasing the value of K to find the stable performance of the approach.

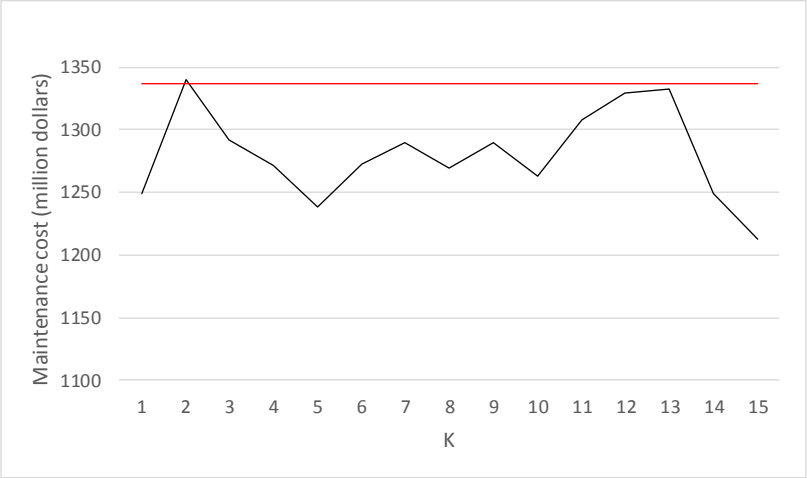

Figure 6.8: The impact of K nearest neighbours on maintenance costs about *Random* strategy

The impact of different strategies (mentioned in Section 6.4.2) on maintenance costs is illustrated in Fig. 6.10. The vertical line represents the maximum, average and minimum of 5 replications' data. "+" in the legend means the simulation experiment is executed without using the FRNN approach. This is also applied to Figs. 6.11 to 6.13. As illustrated in Fig. 6.10, the averages of *Random* and *Sequential* are lower than those of *Random+* and *Sequential+* respectively. More specifically, 9.3% and 2.5% of the maintenance costs have been saved regarding *Random* and *Sequential* strategies respectively. Thus, the efficiency of the FRNN approach can be observed. The FRNN approach achieves a higher precision with *Random* strategy when comparing with the gaps of differences between averages. The margins of maintenance costs for each strategy suggest that the experiments of *Random* strategy remain stabler.

From Figs. 6.11 to 6.13, the result of service level is very coherent with that of maintenance costs regarding *Random* strategy. Nevertheless, the result of *Sequential* strategy represents variability.

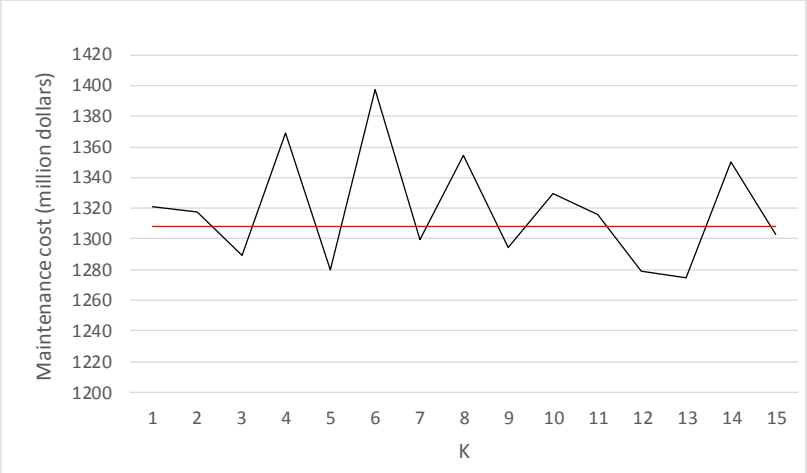

Figure 6.9: The impact of K nearest neighbours on maintenance costs about *Sequential* strategy

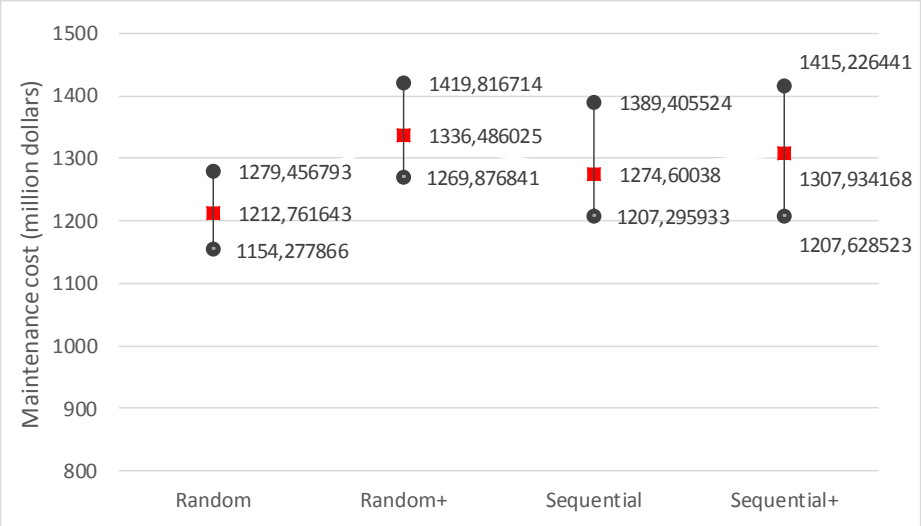

Figure 6.10: The impact of different strategies on maintenance costs

The FRNN approach performs well in terms of on-time flights and delayed flights. As for cancelled flights, the average of *Sequential* is slightly higher than that of *Sequential+*, which depicts the disadvantage of the approach. Even though it is not capable of saving more cancelled flights, its efficiency on *Sequential* strategy still can be observed due to its performance in saving maintenance costs. The downtime costs of maintenance costs imply the difference between total repair times. If the difference of conditions between simulation experiments can be ignored by using averages, the lower maintenance costs mean the shorter total repair time. On the other hand, the impact of different strategies on service level in terms of averages is shown in Fig. 6.14. The result shows

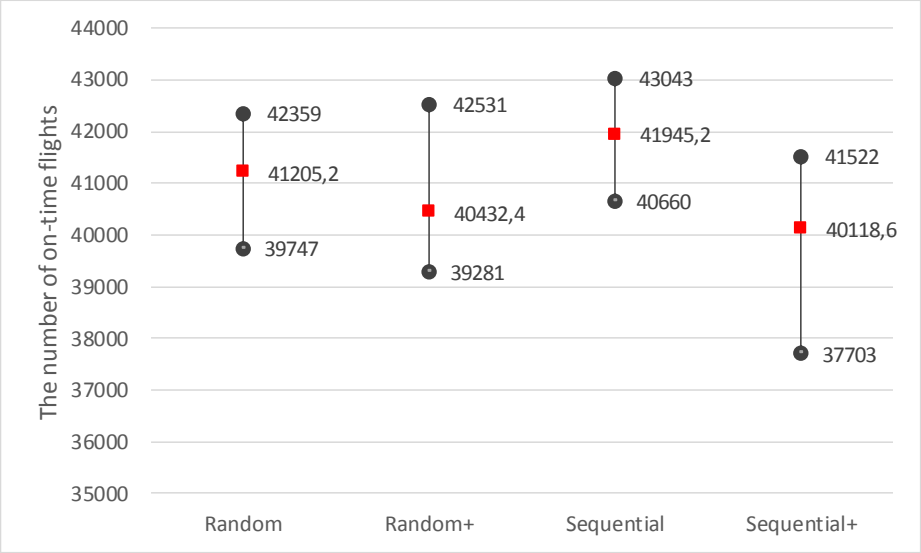

Figure 6.11: The impact of different strategies on on-time flights

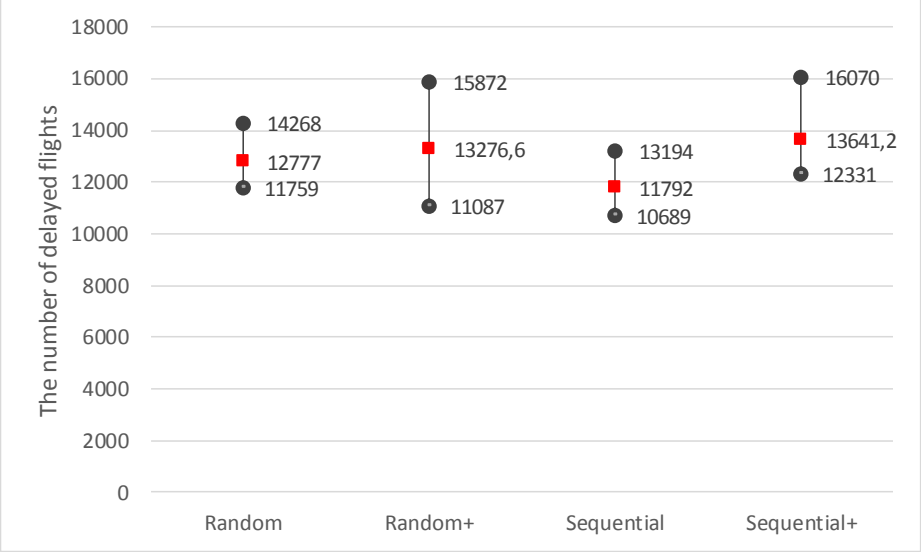

Figure 6.12: The impact of different strategies on delayed flights

that 4.17% and 12.5% of delayed flights have been changed into on-time flights regarding random and sequence strategies respectively. Thus, the FRNN approach also achieves success in predicting faults with respect to *Sequential* strategy.

Service level is another way of examining the efficiency of the FRNN approach. From the above discussion, the analysis of service level helps further investigate the performance of the approach

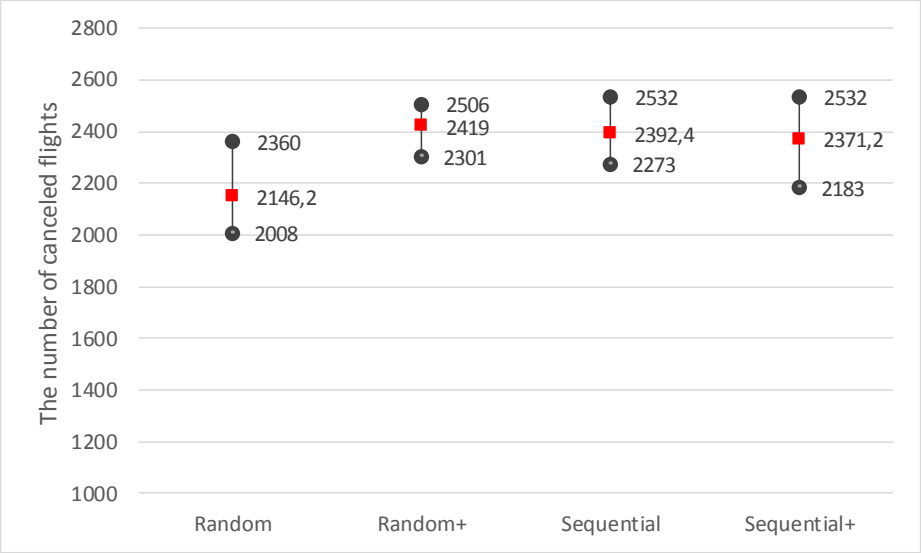

Figure 6.13: The impact of different strategies on cancelled flights

proposed. Therefore, a synthesis of experimental results is necessary to obtain a fair conclusion.

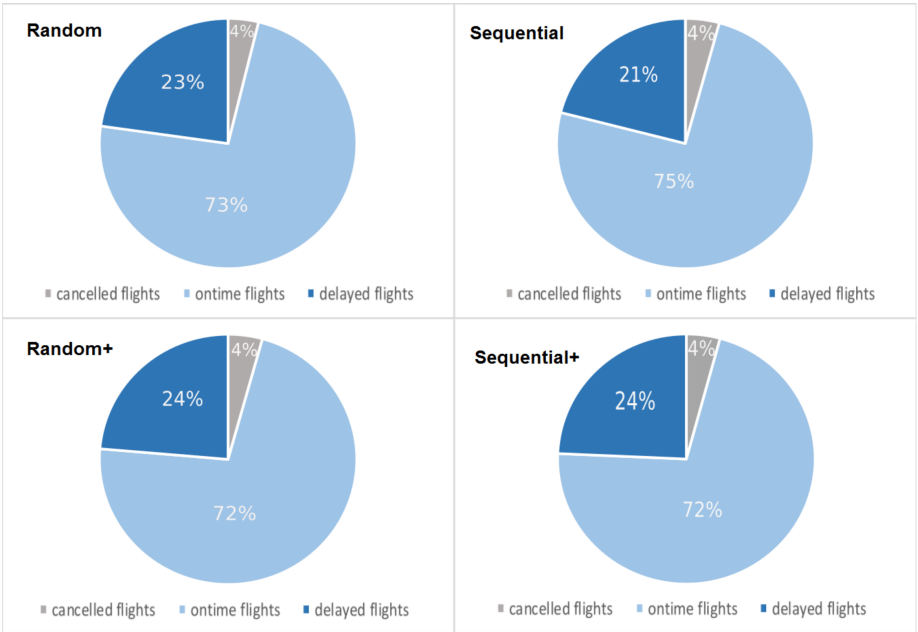

Figure 6.14: The impact of different strategies on service level in terms of average

Above all, we can conclude that the combination of the simulation (ABSS) and the data mining

technique (FRNN) can effectively improve the efficiency of aircraft maintenance. The proposed approach makes it possible to investigate the implying fault diagnosis rules for aircraft of new types. To do this, the degrading curves on components of new aircraft are needed in order to simulate the fault behaviours of components. The simulation system can simulate the life-cycle of aircraft. Thus, a precious data set, including flying hours, inspecting time, repairing time, departure time, delayed flights, cancelled flight, etc., can be collected. In the following, the FRNN approach can be applied to predict the relationship between reliability information and fault location information. Therefore, effective rules of the fault diagnosis can be generated with the virtual experience.

6.6 Discussion

In this chapter, we aim at fulfilling the FCP for aircraft maintenance without sufficient historical data. An idea of combining Data mining techniques and simulation approaches have been proposed. A methodology has been provided to theoretically and technologically realise the integration of the two above approaches. Simulation experiments illustrated the feasibility of the idea.

The approach proposed is capable of automatic carrying out the FCP for aircraft of new types. However, several aspects of the approach still need further improvements. For example, the knowledge learned from simulation data is simple. Apart from that, the strategies we designed for analysing engines are simple as well. The real methods for the fault diagnosis of aircraft maintenance is able to provide answers to these issues but they are hardly accessible due to security and confidentiality. Fortunately, the approach we proposed allows us to make the engine model much more complicated. For instance, additional attributes can be easily included in the component model. Setting connections between components is also possible. Degradation curves can be added to describe the state of the component as well.

6.7 Summary

In this chapter, we discussed the topic about how to accomplish the fault diagnosis for aircraft maintenance without sufficient historical data. The ABSS and the FRNN approach were introduced. The ABSS simulated the maintenance process for aircraft and produced large amounts of reliability-related data. The FRNN approach performed the FCP based on the data collected. The simulator *Anylogic* was utilised to build the simulation system permitting us to not only simulate the maintenance process but also accomplish the function of the FCP because of the permission of programming in Java into simulation models. That is the reason why we are able to technologically achieve the aim of the automation of the whole maintenance process. The simulation experiments of choosing K nearest neighbours and of analysing random and sequence strategies

were conducted to illustrate the feasibility of the approach proposed. Comparing with the existing approaches for aircraft maintenance, our approach is the very novel one, which typically holds the following advantages:

- enabling us to perform the data analysis without sufficient real data;
- supporting the automatic self-improvement on the decision-making process for aircraft maintenance;
- a new attempt for analysing the FCP by combing the information system and data mining.

Even though experiments illustrated the feasibility of the approach proposed, it still suffers from the lack of the completeness of information in terms of combining the ABSS and the FRNN approach. More details on fault diagnoses for aircraft maintenance should be investigated, in order to provide a decision table close to completion.

Chapter 7

Conclusions and Perspectives

This chapter winds up the thesis by summarising everything that has been said before. The significance of findings is highlighted, which is discussed in a more abstract manner. The limitations of the research are also identified. The perspectives are still provided in order to propose what direction will be taken now and what the potential advantages of future research will be like.

7.1 Conclusions

Operational support of aircraft maintenance is suffering from low levels of informatisation. People are accustomed to tackling faults on aircraft through experience. The very high levels of airworthiness and complexity of aircraft are the main reasons. The objective of this study is to provide solutions about efficient operational support of aircraft maintenance under the contemporary society. Generally, the basic idea is to build an informatic platform where each relevant information system on operational support can communicate with each other and share information if necessary. A methodology has been proposed in the way of the framework, in order to accomplish this platform. This methodology provides an alternative way of designing an autonomous system, which is combining systems design and simulation demonstrators instead of simply designing systems with the review of experts. This helps design systems a lot because it may bridge the gap between systems design and real systems. It can also support the evaluation of “the system" in terms of strategies, configurations, etc. when the same manipulation on the real system is definitely prohibited. Based on the simulation system, we still analyse the logic of the system and explore the performance of the system combing data mining techniques. In the following, the conclusions will be discussed in a more specific way.

The simulation system design has taken the real-world issues into consideration. Firstly, ABMS is an appropriate approach to modelling the simulation system. Agents are not only derived from object-oriented ideas but representing autonomy. The object-oriented idea allows the agents to be described in a natural way and be easily scaled up. The autonomy of agents makes it possible to let agents cooperate with each other in an autonomous way. Secondly, the design of aircraft represents realness. Aircraft are composed of components. The remaining useful lifetime for components is always calculated since the simulation experiment starts. Any changes on components are monitored and recorded. So, we calculate the “real indicators" of components such as MTBF, MTTF, MTTR, etc. Besides, the flight schedule information is obtained from websites.

Symbolic model checking on the ABSS gives a concrete case study of formal methods and demonstrates the power of formal methods in improving systems design and providing new insights into the system. Formal methods are becoming increasingly popular. Many researches have emerged in this area. However, they mostly concentrate on extending the expressive power of formal systems. In other words, they are more interested in improving the performance of formal methods instead of solving industrial issues by using formal methods. The less use of formal methods in the industry is usually because industrial cases are often too complicated to be solved by formal methods. Engineers are prone to tackling issues through practical engineering approaches. As a result, more case studies should be provided to illustrate the high efficiency of formal methods. Engineers should be further encouraged to try to use formal methods. Thus, our study on model checking is able to help develop the application of formal methods.

Integrating the FRNN approach into the ABSS supports the data analysis without sufficient real

data. Data mining techniques are available if and only if the training data is enough to some extent. However, sometimes, the data analysis for products of new types is quite necessary as more needs are requested by customers such as how to define strategies for maintaining, how to use them in an appropriate way, etc. While the operational data of new products is limited and the delay of delivering new products can have a negative effect on the competitiveness of products. In this case, integrating data mining techniques into simulation systems could be an excellent way to deal with it.

The framework for developing the autonomous system of aircraft maintenance clarifies the practical value of the work. Firstly, it allows us to gain important insights about the future system in an inexpensive way, especially when the costs, risks or logistics of manipulating the real system of interest are prohibitive. Secondly, it offers a general approach to build an autonomous system that supports reuse and sharing. System designers in other fields are likely to provide a rigorous architecture design of the system and a powerful demonstration of the future system, which illustrates the feasibility of the system design. In the following, verifying the logic of the model and carrying out the performance analysis at the design phase lay the foundation for building the real system. Finally, this framework can also be understood as a methodology for building digital twins. It not only proposes a prototype system of interest but allows us to manipulate the prototype system with part of real data, in order to examine the reaction of the system.

The autonomous maintenance system proposed can be applied to more general fields for tackling maintenance scheduling problems. Three major reasons are contributed towards this attitude. Firstly, the architecture model of the autonomous system is so general that it can be employed in other similar fields. Because it provides us with the fundamental concepts of scheduling, the relationship between concepts, the whole logic of solving maintenance scheduling problems. The concepts of the system are very basic and profound. The importance of concepts varies from field to field. For example, safety-critical domains concentrate more on safety and reliability analysis. While general domains focus more on the cost and service level analysis. Then, the core of the design is information integration. Data is utilized as the input of components as well as the interaction between components. The approach of information integration could be reused to produce more consistent, accurate, and useful information than that provided by any individual data. Finally, to the best of our knowledge, it is the first time that a significant exploration of real-world maintenance data is performed. A solid example of data types is provided. Some efforts should be just made to make it adapted to other domains since most of data types are similar. Therefore, it is possible to apply our system design to other fields.

7.2 Perspectives

Some further improvements for the methodology proposed are still needed, even though the methodology enables us to build the autonomous system and it has its feasibility illustrated. The perspectives will be discussed in terms of systems design, model checking and performance analysis.

A large amount of work on systems design can be done, in order to provide a more precise, significant and useful design for building autonomous systems of aircraft maintenance. The verification of the maintenance process of aircraft maintenance could be first carried out. We tried to give a general process for aircraft maintenance. However, no truly efficient investigation has been done. Opinions on this issue vary from stakeholder to stakeholder. So, it could be a huge amount of work to be accomplished. How to apply the knowledge of standards to direct the maintenance process and decision-making is supposed to be the second one. Piles of information could be explored from standards. We did be inspired by standards and extracted some information from them. However, it is far from enough. In the following, a more complicated aircraft could be built, which makes the system much closer to reality. For example, the relationship between components of aircraft could be considered. Thus, multiple components could be influenced when faults occur. Different strategies for maintaining components could also be proposed. In the end, we suggest that integrating optimisation into simulation would be a good idea since maintaining aircraft is a very complicated issue where different decision combinations are permissible.

In terms of model checking, global behaviours and operational behaviours on the autonomous system have been verified through the approach proposed. However, the verification for the global behaviour can not support transitions between states with messages. This is the limit of NuSMV model checker. Thus, in the future, other model checking techniques like SPIN, PRISM and UPPAAL could be further investigated with the objective of providing a more flexible approach to verifying system behaviours.

Supply chain management on operational support of aircraft maintenance could be further addressed. The flow of goods and service on operations could be better designed, planned, controlled with the intention to create net value, build worldwide logistics and balance supply and demand. For example, the distribution of labours regarding worldwide airports could have a significant influence on maintenance efficiency. How to better design, plan the distribution of labors with respect to worldwide airports could be an interesting issue. Exploring key factors influencing the impact of labour distribution on maintenance efficiency could also be performed. Furthermore, the idea of the research on the labour distribution can be applied to other aspects of supply-chain activities such as the supply of equipment.

Performance analysis is another issue that may need to be further studied. It is suggested that a further study of fault analysis on aircraft could be conducted so that more reliable and reliability-related information could be collected from the operation of aircraft maintenance. Thus, more useful knowledge can be obtained from the simulation system. In addition, the FRNN approach

could be applied to other processes of aircraft maintenance since it is a general approach to perform classification and prediction. Besides, other data mining techniques may be utilised for comparing the performance of techniques. Therefore, a more appropriate approach for data analysis can be chosen.

Besides, a comprehensive literature review of operational support of aircraft maintenance could be also an interesting topic, in order to reinforce our understanding of this domain and provide guidelines to relevant researchers. Chapter 2 has provided an overview of maintenance business involved in our research. Clearly, different aspects of maintenance businesses can be further extended and reviewed. The gaps in current research could be identified to provide propositions for further investigation.

Appendix A

Statecharts of Agents

A.1 Basic Agents

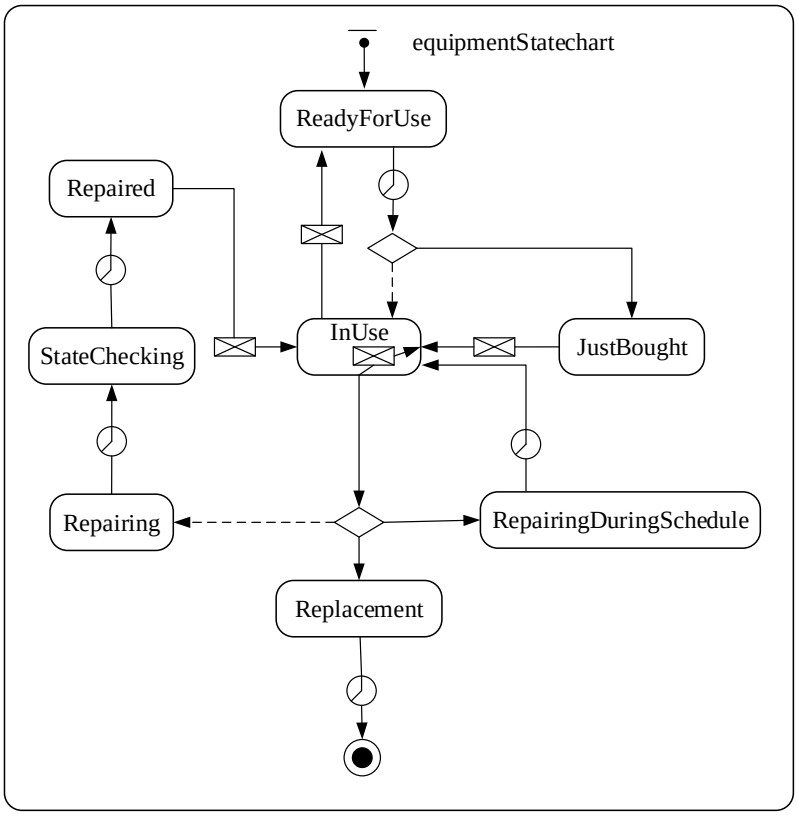

Figure A.1: The statechart of Equipment Agent

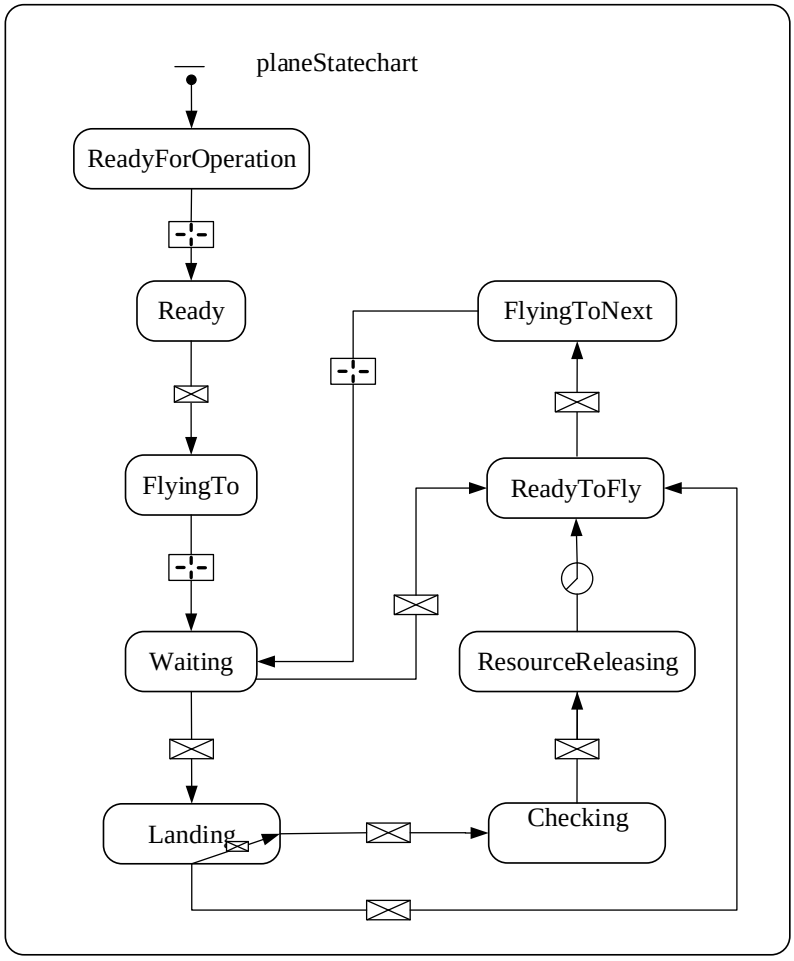

Figure A.2: The statechart of Plane Agent

A.2 Maintenance Agents

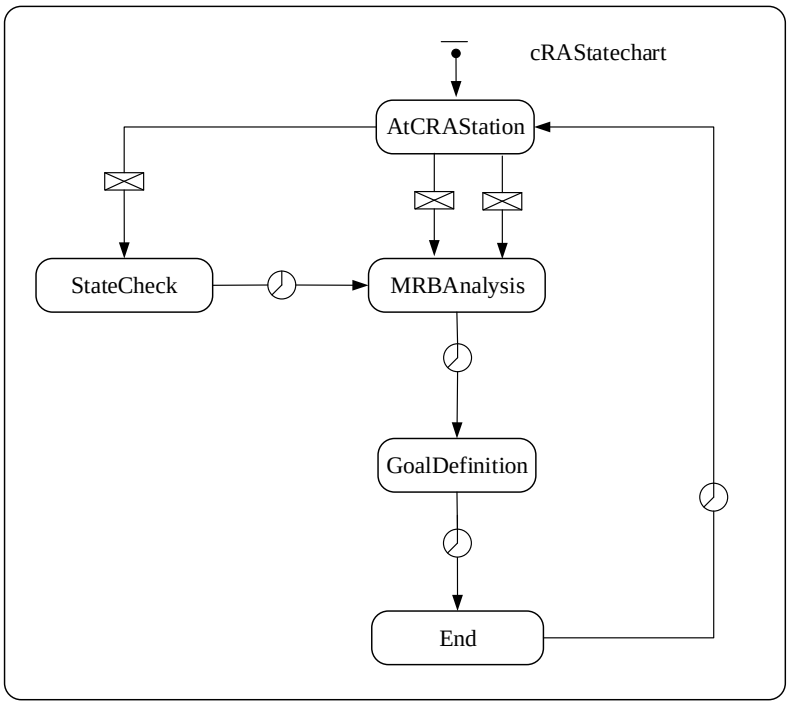

Figure A.3: The statechart of CRA

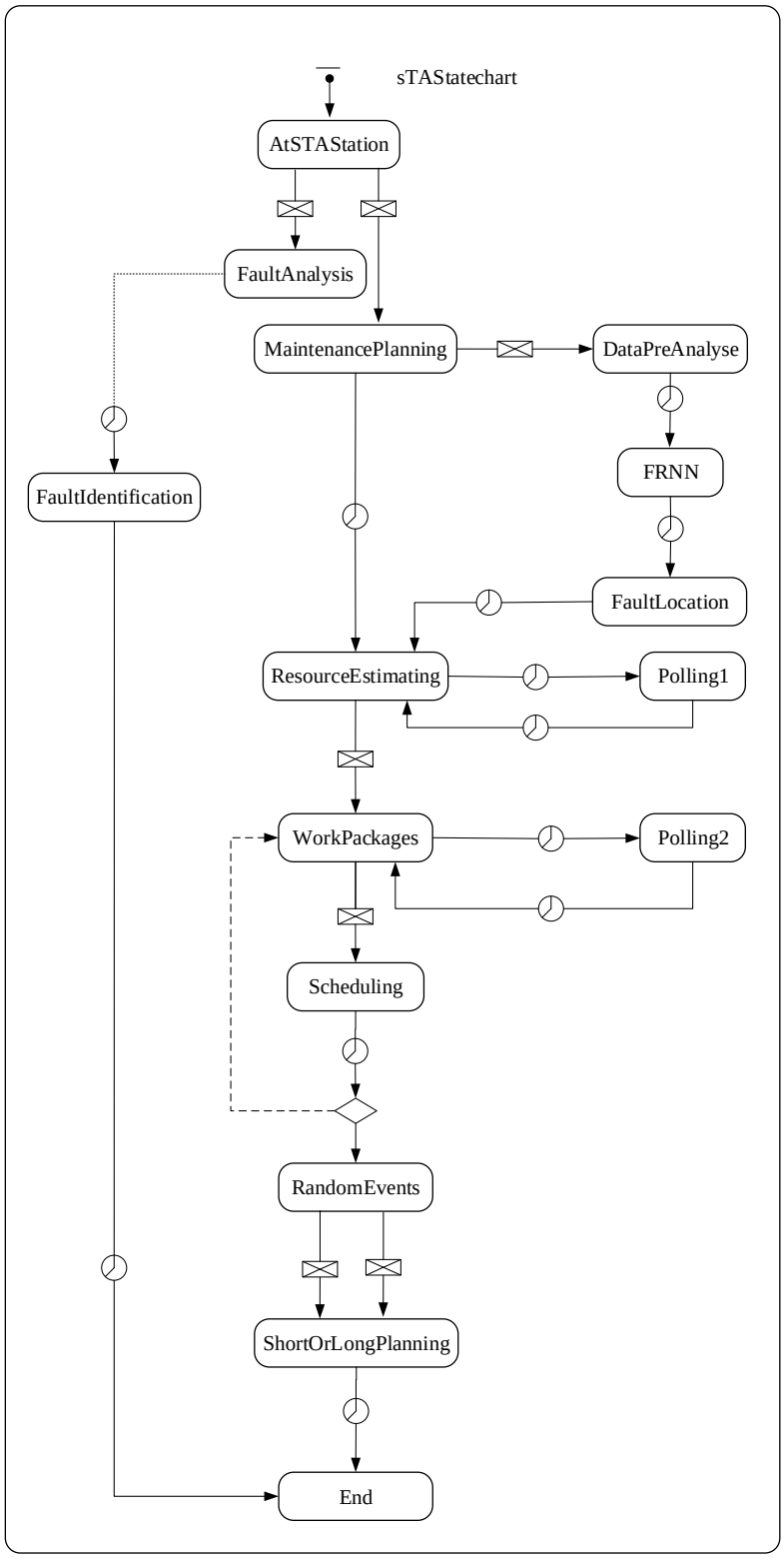

Figure A.4: The statechart of STA

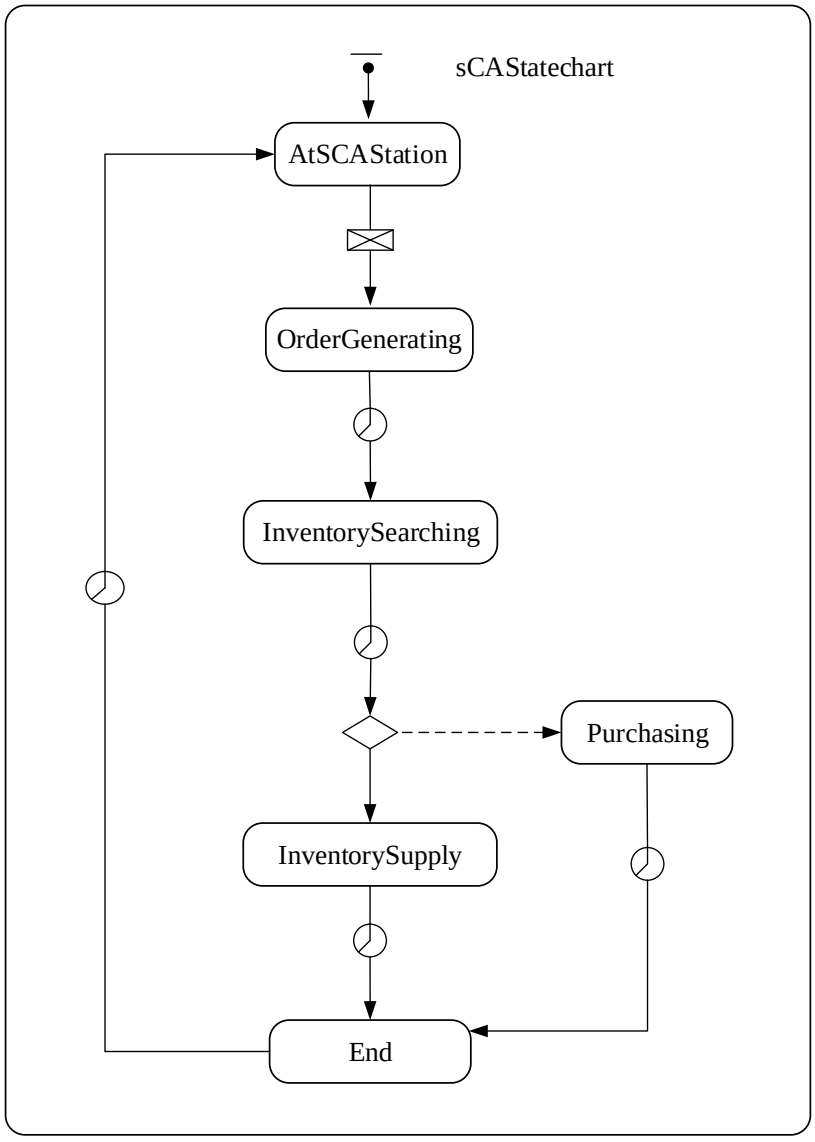

Figure A.5: The statechart of SCA

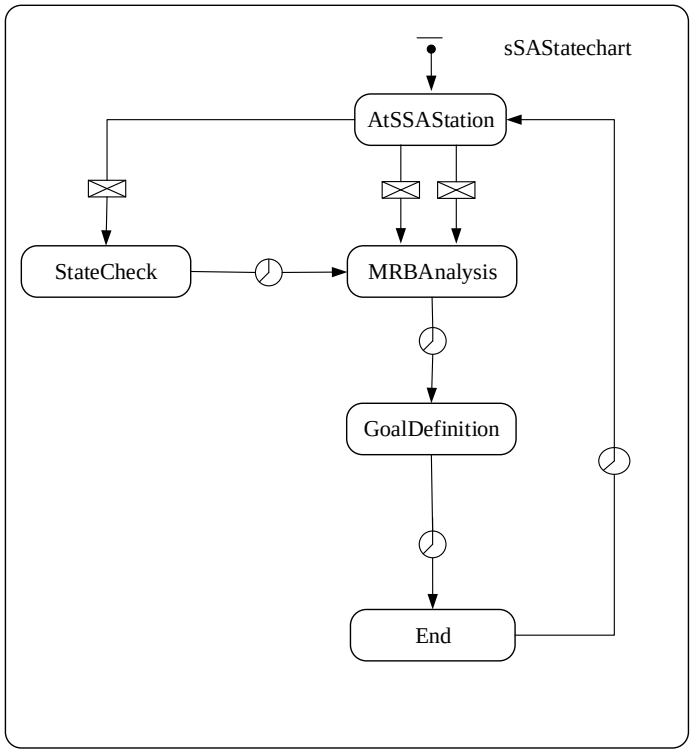

Figure A.6: The statechart of SSA

Appendix B

The Code of NuSMV Model

B.1 Autonomous System

```
-----------------------------------------
--    Autonomous System Simulation Platform (agent-based simulation system)
--    Date: 2 February 2019
--    Model checking

-----------------------------------------
MODULE main

VAR
  flight    : process Flight(flight, plane, stm);
  plane     : process Plane(plane, flight, stm, ssm, mainn);
  crm       : process CRM(crm, plane);
  stm       : process STM(stm, crm, ssm, scm);
  ssm       : process SSM(ssm, crm, stm);
  mainn     : process Main(mainn, plane);


-------------------------------------------------------------------------------
-- Specification
--SPEC AG (plane.state = b5 -> AX plane.state != b2); -- !!!!!!!!!!!!! Intuitive!

--SPEC AG(!((flight.state = a0 ) & (plane.state = b2 )));
SPEC AG(!(flight.state = a1 & plane.state = b7));
--SPEC AG(!(flight.state = a0 & plane.state = b8));
--SPEC AG(!(flight.state = a3 & plane.state = b8));
--SPEC AG(!(flight.state = a0 & plane.state = b8) | !(flight.state = a3 & plane.state = b8));
--SPEC AG !((flight.state = a0 & plane.state = b8) & (flight.state = a3 & plane.state = b8));
```

```
-------------------------------------------------------------------------------

-----------------------------------------------
-- The definition of Flight Agent (flight, plane, stm)
-- a0: Init; a1: Scheduling; a2: Checking; a3: Ready; a4: End.
-----------------------------------------------
MODULE Flight(arg1, arg2, arg3)
VAR state   : {a0,a1,a2,a3,a4};
IVAR action : {Schedule, Null};


INIT (state = a0)
   TRANS(next(state)= case
        (arg1.state = a0 & arg1.action = Schedule)        : a1;
        (arg1.state = a1 & arg2.action = Receive)         : a4;
        (arg1.state = a0 & arg3.action = Delay)           : a2;
        (arg1.state = a2 & arg1.action = Null)            : a3;
        (arg1.state = a3 & arg1.action = Schedule)        : a1;
        (arg1.state = a4 & arg1.action = Null)            : a0;
    TRUE: state;
   esac)

-----------------------------------------------
-- The definition of Plane Agent (plane, flight, stm, ssm, mainn)
-- b0: Init; b1: Ready; b2: InitScheduled; b3: FlyingTo; b4: Waiting; b5: Landing;
-- b6: Checking; b7: ReadyToFly; b8: Scheduled; b9 FlyingToNext.
-----------------------------------------------
MODULE Plane(arg1, arg2, arg3, arg4, arg5)
VAR state   : {b0,b1,b2,b3,b4,b5,b6,b7,b8,b9};
IVAR action : {Null, Receive, Reach, Request, ScheduleMRequest, UnscheduleMRequest, Complete};


INIT (state = b0)

   TRANS(next(state)= case
        (arg1.state = b0 & arg1.action = Reach)                          : b1;
        (arg1.state = b1 & arg2.action = Schedule)                       : b2;
        (arg1.state = b2 & arg1.action = Receive)                        : b3;
        (arg1.state = b3 & arg1.action = Reach)                          : b4;
        (arg1.state = b4 & arg1.action = Request & arg5.state = g0)      : b4;
        (arg1.state = b4 & arg5.action = Dispatch)                       : b5;
        (arg1.state = b4 & arg5.action = Delay)                          : b7;
        (arg1.state = b5 & arg1.action = ScheduleMRequest)               : b5;
        (arg1.state = b5 & arg1.action = UnscheduleMRequest)             : b5;
        (arg1.state = b5 & arg4.action = NoRepair)                       : b7;
        (arg1.state = b5 & arg3.action = Deliver)                        : b6;
        (arg1.state = b6 & arg1.action = Complete)                       : b7;
```

```
        (arg1.state = b7 & arg2.action = Schedule)                          : b8;
        (arg1.state = b8 & arg1.action = Receive)                           : b9;
        (arg1.state = b9 & arg1.action = Reach)                             : b4;
                TRUE: state;
    esac)


-------------------------------------------------
-- The definition of CRM Agent (crm, plane)
-------------------------------------------------
MODULE CRM(arg1, arg2)
VAR state   : {c0,c1,c2};
IVAR action : {ScheduleMaintenance, UnscheduleMaintenance, Null};

INIT (state = c0)
   TRANS(next(state)= case
        (arg1.state = c0 & arg2.action = UnscheduleMRequest)    : c1;
        (arg1.state = c0 & arg2.action = ScheduleMRequest)      : c2;
        (arg1.state = c1 & arg1.action = UnscheduleMaintenance) : c1;
        (arg1.state = c2 & arg1.action = ScheduleMaintenance)   : c2;
        (arg1.state = c1 & arg1.action = Null)                  : c0;
        (arg1.state = c2 & arg1.action = Null)                  : c0;
       TRUE: state;
    esac)

-------------------------------------------------
-- The definition of STM Agent (stm, crm, ssm, scm)
-------------------------------------------------
MODULE STM(arg1, arg2, arg3, arg4)
VAR state   : {d0,d1,d2,d3,d4,d5,d6};
IVAR action : {ResourceEstimate, PossiblePartsDeliver, Delay, Deliver, Null};


INIT (state = d0)

   TRANS(next(state)= case
        (arg1.state = d0 & arg3.action = Deliver)               : d3;
        (arg1.state = d0 & arg2.action = UnscheduleMaintenance) : d1;
        (arg1.state = d1 & arg1.action = PossiblePartsDeliver)  : d2;
        (arg1.state = d2 & arg1.action = Null)                  : d0;
        (arg1.state = d3 & arg1.action = ResourceEstimate)      : d4;
        (arg1.state = d4 & arg4.action = Deliver)               : d5;
        (arg1.state = d5 & arg1.action = Deliver)               : d6;
        (arg1.state = d5 & arg1.action = Delay)                 : d6;
        (arg1.state = d6 & arg1.action = Null)                  : d0;
         TRUE: state;
    esac)
```

```
-----------------------------------------------
-- The definition of SSM Agent (ssm, crm, stm)
-----------------------------------------------
MODULE SSM(arg1, arg2, arg3)
VAR state   : {e0,e1,e2,e3,e4};
IVAR action : {StrategyAnalyze, GoalDefine, NoRepair, Deliver, Null};


INIT (state = e0)

   TRANS(next(state)= case
        (arg1.state = e0 & arg2.action = ScheduleMaintenance)  : e1;
        (arg1.state = e0 & arg3.action = PossiblePartsDeliver) : e2;
        (arg1.state = e1 & arg1.action = StrategyAnalyze)      : e2;
        (arg1.state = e1 & arg1.action = NoRepair)             : e4;
        (arg1.state = e2 & arg1.action = GoalDefine)           : e3;
        (arg1.state = e3 & arg1.action = Deliver)              : e4;
        (arg2.state = e4 & arg1.action = Null)                 : e0;
        TRUE: state;
    esac)

-----------------------------------------------
-- The definition of SCM Agent (scm, stm)
-----------------------------------------------
MODULE SCM(arg1, arg2)
VAR state   : {f0,f1,f2,f3,f4,f5};
IVAR action : {InventorySearch, Found, NotFound, Buy, Deliver, Null};

INIT (state = f0)
   TRANS(next(state)= case
        (arg1.state = f0 & arg2.action = ResourceEstimate) : f1;
        (arg1.state = f1 & arg1.action = InventorySearch)  : f2;
        (arg1.state = f2 & arg1.action = Found)            : f4;
        (arg1.state = f2 & arg1.action = NotFound)         : f3;
        (arg1.state = f3 & arg1.action = Buy)              : f4;
        (arg1.state = f4 & arg1.action = Deliver)          : f5;
        (arg1.state = f5 & arg1.action = Null)             : f0;
        TRUE: state;
    esac)

-----------------------------------------------
-- The definition of Main Agent (mainn, plane)
-- g0: Init; g1: Receiving; g2: Processing; g3: Dispatching; g4: Delaying.
-----------------------------------------------
MODULE Main(arg1, arg2)
VAR state   : {g0, g1, g2, g3, g4};
```

```
IVAR action : {Process, Dispatch, Delay, Null};

INIT (state = g0)

   TRANS(next(state)= case
        (arg1.state = g0 & arg2.action = Request)  : g1;
        (arg1.state = g1 & arg1.action = Process)  : g2;
        (arg1.state = g2 & arg1.action = Dispatch) : g3;
        (arg1.state = g2 & arg1.action = Delay)    : g4;
        (arg1.state = g3 & arg1.action = Null)     : g0;
        (arg1.state = g4 & arg1.action = Null)     : g0;
        TRUE: state;
    esac)

FAIRNESS
   running
```

Bibliography

Abdelghany, Ahmed, Khaled Abdelghany, and Farshid Azadian. 2017. "Airline flight schedule planning under competition." *Computers & Operations Research* 87: 20–39.

Ackert, SP. 2010. "Basics of aircraft maintenance programs for financiers." http://www.aircraftmonitor.com/uploads/1/5/9/9/15993320/basics_of_aircraft_maintenance_programs_for_financiers___v1.pdf.

Al-Saqqar, Faisal, Jamal Bentahar, Khalid Sultan, Wei Wan, and Ehsan Khosrowshahi Asl. 2015. "Model checking temporal knowledge and commitments in multi-agent systems using reduction." *Simulation Modelling Practice and Theory* 51: 45–68.

Ali, Shallaw Mohammed, Martina Doolan, Paul Wernick, and Ed Wakelam. 2018. "Developing an agent-based simulation model of software evolution." *Information and Software Technology* 96: 126–140.

Allen, John M. 2012. "Air Carrier Maintenance Programs." *U.S. Department of Transportation* .

Alrabghi, Abdullah, and Ashutosh Tiwari. 2015. "State of the art in simulation-based optimisation for maintenance systems." *Computers & Industrial Engineering* 82: 167–182.

Alrabghi, Abdullah, and Ashutosh Tiwari. 2016. "A novel approach for modelling complex maintenance systems using discrete event simulation." *Reliability Engineering & System Safety* 154: 160–170.

ASD. 2019. "International specification for developing and continuously improving preventive maintenance." Accessed 2019-08-10. http://www.s4000p.org/docs/S4000P_Issue_2.0.pdf.

Atdelzater, TF, Ella M Atkins, and Kang G Shin. 2000. "QoS negotiation in real-time systems and its application to automated flight control." *IEEE Transactions on Computers* 49 (11): 1170–1183.

Automata. 2019. "Automata Theory." Accessed 2019-08-10. https://en.wikipedia.org/wiki/Automata_theory.

Bateman, Francois, Hassan Noura, and Mustapha Ouladsine. 2011. "Fault diagnosis and fault-tolerant control strategy for the aerosonde UAV." *IEEE Transactions on Aerospace and Electronic Systems* 47 (3): 2119–2137.

Beliën, Jeroen, Brecht Cardoen, and Erik Demeulemeester. 2012. "Improving workforce scheduling of aircraft line maintenance at Sabena Technics." *Interfaces* 42 (4): 352–364.

Bentahar, Jamal, Hamdi Yahyaoui, Melissa Kova, and Zakaria Maamar. 2013. "Symbolic model checking composite Web services using operational and control behaviors." *Expert Systems with Applications* 40 (2): 508–522.

Blanchard, Benjamin S. 2004. *System engineering management*. John Wiley & Sons.

Boeing. 2019. "Boeing Global Service." http://www.boeing.com/resources/boeingdotcom/services/bgs_infograph_new.pdf.

Bombardier. 2004. "CSP A2113 CRJ 100 P200." *Quarterly P Monthly FRACAS report[Z]* 88: 135–136.

Boniol, Frédéric, and Virginie Wiels. 2014. "The landing gear system case study." In *International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z*, 1–18. Springer.

Booch, Grady. 2005. *The unified modeling language user guide*. Pearson Education India.

Bordini, Rafael H, Michael Fisher, Willem Visser, and Michael Wooldridge. 2006. "Verifying multi-agent programs by model checking." *Autonomous agents and multi-agent systems* 12 (2): 239–256.

Cavada, Roberto, Alessandro Cimatti, Gavin Keighren, Emanuele Olivetti, Marco Pistor, and Marco Roveri. 2019. "NuSMV 2.5 Tutorial." Accessed 2019-03-1. http://nusmv.fbk.eu/NuSMV/tutorial/v25/tutorial.pdf.

Chang, Qing, Jun Ni, Pulak Bandyopadhyay, Stephan Biller, and Guoxian Xiao. 2007. "Maintenance staffing management." *Journal of Intelligent Manufacturing* 18 (3): 351–360.

China Civil Aviation. 2019. "Reliability report for civil arcraft." https://wenku.baidu.com/view/3237fcaae43a580216fc700abb68a98271feac07.html.

Conn, P., K. Jones, and P. White. 2008. "ATA e-Business Specification Overview." http://www.ataebiz.org/forum/2008_ata_e-biz_forum_agenda/eBizSpec0verview.pdf.

Conroy, M., P. Gill, B. Hill, B. Ibach, C. Jones, D. Ungar, and K. Bengtsson. 2014. "Technical Data Interoperability (TDI) Pathfinder Via Emerging Standards." https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20160000235.pdf.

Cook, Andrew J, Graham Tanner, and Stephen Anderson. 2004. *Evaluating the true cost to airlines of one minute of airborne or ground delay*. Technical Report. Eurocontrol.

De Bruecker, Philippe, Jeroen Beliën, Jorne Van den Bergh, and Erik Demeulemeester. 2017. "A three-stage mixed integer programming approach for optimizing the skill mix and training schedules for aircraft maintenance." *European Journal of Operational Research* .

De Bruecker, Philippe, Jeroen Beliën, Jorne Van den Bergh, and Erik Demeulemeester. 2018. "A three-stage mixed integer programming approach for optimizing the skill mix and training schedules for aircraft maintenance." *European Journal of Operational Research* 267 (2): 439–452.

Diaz, J., J.I. Huertas, and F. Trigos. 2014. "Aircraft maintenance, routing, and crew scheduling planning for airlines with a single fleet and a single maintenance and crew base." *Computers & Industrial Engineering* 75: 68–78.

Dinis, Duarte, Ana Barbosa-Póvoa, and Ângelo Palos Teixeira. 2019. "Valuing data in aircraft maintenance through big data analytics: A probabilistic approach for capacity planning using Bayesian networks." *Computers & Industrial Engineering* 128: 920–936.

Doi, Tsubasa, Tatsushi Nishi, and Stefan Voß. 2017. "Two-level decomposition-based matheuristic for airline crew rostering problems with fair working time." *European Journal of Operational Research* .

Duffuaa, S. O., M. Ben-Daya, K. S. Al-Sultan, and A. A. Andidjani. 2001. "A generic conceptual simulation model for maintenance systems." *Journal of Quality in Maintenance Engineering* 7 (3): 207–219.

Dupuy, Michael J, Daniel E Wesely, and Cody S Jenkins. 2011. "Airline fleet maintenance: Trade-off analysis of alternate aircraft maintenance approaches." In *Systems and Information Engineering Design Symposium (SIEDS), 2011 IEEE*, 29–34. IEEE.

Durazo-Cardenas, Isidro, Andrew Starr, Christopher J Turner, Ashutosh Tiwari, Leigh Kirkwood, Maurizio Bevilacqua, Antonios Tsourdos, et al. 2018. "An autonomous system for maintenance scheduling data-rich complex infrastructure: Fusing the railways' condition, planning and cost." *Transportation Research Part C: Emerging Technologies* 89: 234–253.

El Menshawy, Mohamed, Jamal Bentahar, Warda El Kholy, and Amine Laarej. 2018. "Model checking real-time conditional commitment logic using transformation." *Journal of Systems and Software* 138: 189–205.

Fagin, Ronald, Joseph Y Halpern, Yoram Moses, and Moshe Vardi. 2004. *Reasoning about knowledge*. MIT press.

Ferguson, John, Abdul Qadar Kara, Karla Hoffman, and Lance Sherry. 2013. "Estimating domestic US airline cost of delay based on European model." *Transportation Research Part C: Emerging Technologies* 33: 311–323.

Gary, Linnéusson, Ng HC Amos, Aslam Tehseen, et al. 2018. "Towards strategic development of maintenance and its effects on production performance by using system dynamics in the automotive industry." *International Journal of Production Economics* 200 (C): 151–169.

Helbing, Dirk. 2012. "Agent-based modeling." In *Social self-organization*, 25–70. Springer.

Heppenstall, Alison J, Andrew T Crooks, Linda M See, and Michael Batty. 2011. *Agent-based models of geographical systems*. Springer Science & Business Media.

ISO. 2011. "Systems and software engineering–architecture description." ISO/IEC/IEEE 42010.

Jensen, Richard, and Chris Cornelis. 2011. "Fuzzy-rough nearest neighbour classification and prediction." *Theoretical Computer Science* 412 (42): 5871 – 5884.

Jia, Rong, Fuqi Ma, Jian Dang, Guangyi Liu, and Huizhi Zhang. 2018. "Research on Multidomain Fault Diagnosis of Large Wind Turbines under Complex Environment." *Complexity* 2018.

Jiao, Xiaoxuan, Bo Jing, Yifeng Huang, Juan Li, and Guangyue Xu. 2017. "Research on fault diagnosis of airborne fuel pump based on EMD and probabilistic neural networks." *Microelectronics Reliability* 75: 296–308.

Keshanchi, Bahman, Alireza Souri, and Nima Jafari Navimipour. 2017. "An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing." *Journal of Systems and Software* 124: 1–21.

Keysan, Gizem, George L Nemhauser, and Martin WP Savelsbergh. 2010. "Tactical and operational planning of scheduled maintenance for per-seat, on-demand air transportation." *Transportation Science* 44 (3): 291–306.

Khoshgoftaar, Taghi M, and Naeem Seliya. 2003. "Fault prediction modeling for software quality estimation: Comparing commonly used techniques." *Empirical Software Engineering* 8 (3): 255–283.

Khurshid, Sarfraz, Corina S Păsăreanu, and Willem Visser. 2003. "Generalized symbolic execution for model checking and testing." In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 553–568. Springer.

Kilpi, Jani, Juuso Töyli, and Ari Vepsäläinen. 2009. "Cooperative strategies for the availability service of repairable aircraft components." *International Journal of Production Economics* 117 (2): 360–370.

Kim, Seungkeun, Jiyoung Choi, and Youdan Kim. 2008. "Fault detection and diagnosis of aircraft actuators using fuzzy-tuning IMM filter." *IEEE Transactions on Aerospace and Electronic Systems* 44 (3): 940–952.

Kinnison, Harry A, and Tariq Siddiqui. 2012. *Aviation maintenance management*. McGraw-Hill Professional.

Korvesis, Panagiotis. 2017. "Machine Learning for Predictive Maintenance in Aviation." PhD diss., Université Paris-Saclay.

Kozanidis, G., G. Gavranis, and G. Liberopoulos. 2014. "Heuristics for flight and maintenance planning of mission aircraft." *Annals of Operations Research* 221 (1): 211–238.

Kullstam, Per A. 1981. "Availability, MTBF and MTTR for repairable M out of N system." *IEEE Transactions on Reliability* 30 (4): 393–394.

Kumar, U Dinesh, John Crocker, Jezdimir Knezevic, and Mohamed El-Haram. 2012. *Reliability, maintenance and logistic support:-A life cycle approach*. Springer Science & Business Media.

Lafonsee, J. 2015. "ATA iSpec 2200: The Lifecycle of the Spec." `http://www.ataebiz.org/forum/2015_ATA_e-BusinessForum/4_iSpec2200_LaFontsee.pdf`.

Li, Ping, Wenbin Wang, and Rui Peng. 2016. "Age-based replacement policy with consideration of production wait time." *IEEE Transactions on Reliability* 65 (1): 235–247.

Liu, Bin, Jun Wu, and Min Xie. 2015. "Cost analysis for multi-component system with failure interaction under renewing free-replacement warranty." *European Journal of Operational Research* 243 (3): 874–882.

Liu, Yinling, Tao Wang, Haiqing. Zhang, and Vincent Cheutet. 2017. "Towards Standards Analysis and Application in Process of Aircraft Maintenance Repair Overhaul." In *Proceedings of the IFAC World Congress*, Toulouse - France.

Liu, Yinling, Tao Wang, Haiqing Zhang, and Vincent Cheutet. 2018. "Information Systems Simulation for Performance Evaluation-Application in Aircraft Maintenance." In *IFIP International Conference on Product Lifecycle Management*, 789–799. Springer.

Liu, Yinling, Tao Wang, Haiqing Zhang, and Vincent Cheutet. 2019a. "Simulation based fuzzy rough nearest neighbour fault classification and prediction for aircraft maintenance." *Journal of Simulation* Accepted.

Liu, Yinling, Tao Wang, Haiqing Zhang, Vincent Cheutet, and Guohua Shen. 2019b. "The design and simulation of an autonomous system for aircraft maintenance scheduling." *Computers & Industrial Engineering* 106041.

Liu, Yutu, Anne H Ngu, and Liang Z Zeng. 2004. "QoS computation and policing in dynamic web service selection." In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, 66–73. ACM.

Llinas, James, and David L Hall. 1998. "An introduction to multi-sensor data fusion." In *Circuits and Systems, 1998. ISCAS'98. Proceedings of the 1998 IEEE International Symposium on*, Vol. 6, 537–540. IEEE.

Maamar, Zakaria, Quan Z Sheng, Hamdi Yahyaoui, Jamal Bentahar, and Khouloud Boukadi. 2009. "A new approach to model web services' behaviors based on synchronization." In *Advanced Information Networking and Applications Workshops, 2009. WAINA'09. International Conference on*, 43–49. IEEE.

Macal, Charles M, and Michael J North. 2005. "Tutorial on agent-based modeling and simulation." In *Simulation conference, 2005 proceedings of the winter*, 14–pp. IEEE.

Macal, Charles M, and Michael J North. 2010. "Tutorial on agent-based modelling and simulation." *Journal of simulation* 4 (3): 151–162.

MacKenzie, A., J.O. Miller, and R.R. Hill. 2012. "Application of agent based modelling to aircraft maintenance manning and sortie generation." *Simulation Modelling Practice and Theory* 20 (1): 89–98.

Magee, Christopher, and Olivier de Weck. 2004. *Complex system classification*. Technical Report. International Council On Systems Engineering (INCOSE).

Mechefske, CK, and J Mathew. 1992. "Fault detection and diagnosis in low speed rolling element bearings Part II: The use of nearest neighbour classification." *Mechanical Systems and Signal Processing* 6 (4): 309–316.

Meski, Artur, Wojciech Penczek, Maciej Szreter, Bożena Woźna-Szcześniak, and Andrzej Zbrzezny. 2014. "BDD-versus SAT-based bounded model checking for the existential fragment of linear temporal logic with knowledge: algorithms and their performance." *Autonomous Agents and Multi-Agent Systems* 28 (4): 558–604.

Michaels, Kevin. 2010. "MRO Industry Outlook." `https://www.aeromontreal.ca/download/fca8adddfff353/01-MRO+Industry+Outlook_Kevin+Michaels.pdf`.

Mobley, R Keith. 2002. *An introduction to predictive maintenance*. Butterworth-Heinemann.

Monostori, László, József Váncza, and Soundar RT Kumara. 2006. "Agent-based systems for manufacturing." *CIRP Annals-Manufacturing Technology* 55 (2): 697–720.

Moyaux, Thierry, Brahim Chaib-Draa, and Sophie D'Amours. 2006. "Supply chain management and multiagent systems: an overview." *Multiagent based supply chain management* 1–27.

Naderi, Esmaeil, and Khashayar Khorasani. 2018. "Data-driven fault detection, isolation and estimation of aircraft gas turbine engine actuator and sensors." *Mechanical Systems and Signal Processing* 100: 415–438.

NASA, National Aeronautics Space Administration. 2019. "Wright 1903 Engine Parts." `https://www.grc.nasa.gov/www/k-12/airplane/engparts.html`.

Navimipour, Nima Jafari, Ahmad Habibizad Navin, Amir Masoud Rahmani, and Mehdi Hosseinzadeh. 2015. "Behavioral modeling and automated verification of a Cloud-based framework to share the knowledge and skills of human resources." *Computers in Industry* 68: 65–77.

Papakostas, Nikolaos, P Papachatzakis, Vangelis Xanthakis, Dimitris Mourtzis, and George Chryssolouris. 2010. "An approach to operational aircraft maintenance planning." *Decision Support Systems* 48 (4): 604–612.

Park, Yongha, and Morton E O'Kelly. 2018. "Examination of cost-efficient aircraft fleets using empirical operation data in US aviation markets." *Journal of Air Transport Management* 69: 224–234.

Pawlak, Zdzisław. 1982. "Rough sets." *International journal of computer & information sciences* 11 (5): 341–356.

Pawlak, Zdzislaw, Lech Polkowski, and Andrzej Skowron. 2008. "Rough Set Theory." .

Peterson, Leif E. 2009. "K-nearest neighbor." *Scholarpedia* 4 (2): 1883.

Pnueli, Amir. 1977. "The temporal logic of programs." In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, 46–57. IEEE.

Pohl, Thomas. 2019. "Cost per hour of downtime per aircraft is 10,000 USD more." Accessed 2019-01-10. `https://blogs.sap.com/2013/05/02/cost-per-hour-of-downtime-per-aircraft-is-10000-usd-more`.

Quan, Gang, Garrison W Greenwood, Donglin Liu, and Sharon Hu. 2007. "Searching for multiobjective preventive maintenance schedules: Combining preferences with evolutionary algorithms." *European Journal of Operational Research* 177 (3): 1969–1984.

Radzikowska, Anna Maria, and Etienne E Kerre. 2002. "A comparative study of fuzzy rough sets." *Fuzzy sets and systems* 126 (2): 137–155.

Raimondi, Franco. 2006. "Model checking multi-agent systems." PhD diss., University of London.

Randall, Wesley S, Terrance L Pohlen, and Joe B Hanna. 2010. "Evolving a theory of performance-based logistics using insights from service dominant logic." *Journal of Business Logistics* 31 (2): 35–61.

Royce, Winston W. 1987. "Managing the development of large software systems: concepts and techniques." In *Proceedings of the 9th international conference on Software Engineering*, 328–338. IEEE Computer Society Press.

Safaei, N., D. Banjevic, and A.D. Jardine. 2011. "Workforce-constrained maintenance scheduling for military aircraft fleet: a case study." *Annals of Operations Research* 186 (1): 295–316.

Sahin, Ferat, M Çetin Yavuz, Ziya Arnavut, and Önder Uluyol. 2007. "Fault diagnosis for airplane engines using Bayesian networks and distributed particle swarm optimization." *Parallel Computing* 33 (2): 124–143.

Sahnoun, M'hammed, David Baudry, Navonil Mustafee, Anne Louis, Philip Andi Smart, Phil Godsiff, and Belahcene Mazari. 2015. "Modelling and simulation of operation and maintenance strategy for offshore wind farms based on multi-agent system." *Journal of Intelligent Manufacturing* 1–17.

Saltoğlu, Remzi, Nazmia Humaira, and Gökhan İnalhan. 2016. "Aircraft scheduled airframe maintenance and downtime integrated cost model." *Advances in Operations Research* 2016.

Samaranayake, Premaratne, and Senevi Kiridena. 2012. "Aircraft maintenance planning and scheduling: an integrated framework." *Journal of Quality in Maintenance Engineering* 18 (4): 432–453.

Shanghai airline. 2019. "Reliability mangement." `https://wenku.baidu.com/view/c78de4e9102de2bd960588bb.html`.

Sheng, Quan Z, Zakaria Maamar, Hamdi Yahyaoui, Jamal Bentahar, and Khouloud Boukadi. 2010. "Separating operational and control behaviors: A new approach to Web services modeling." *IEEE Internet Computing* 14 (3): 68–76.

Shenneld, A., P. Fleming, and J. Allan. 2010. "Optimisation of maintenance scheduling strategies on the grid." *Annals of Operations Research* 180 (1): 213–231.

Sheu, Shey-Huei, Hsin-Nan Tsai, Fu-Kwun Wang, and Zhe George Zhang. 2015. "An extended optimal replacement model for a deteriorating system with inspections." *Reliability Engineering & System Safety* 139: 33–49.

Siebers, Peer-Olaf, Charles M Macal, Jeremy Garnett, David Buxton, and Michael Pidd. 2010. "Discrete-event simulation is dead, long live agent-based simulation!" *Journal of Simulation* 4 (3): 204–210.

Skersys, Tomas, Paulius Danenas, and Rimantas Butleris. 2018. "Extracting SBVR business vocabularies and business rules from UML use case diagrams." *Journal of Systems and Software* 141: 111–130.

Sobie, Cameron, Carina Freitas, and Mike Nicolai. 2018. "Simulation-driven machine learning: Bearing fault classification." *Mechanical Systems and Signal Processing* 99: 403–419.

Song, Wen, Hui Xi, Donghun Kang, and Jie Zhang. 2018. "An agent-based simulation system for multi-project scheduling under uncertainty." *Simulation Modelling Practice and Theory* 86: 187–203.

Souri, Alireza, and Nima Jafari Navimipour. 2014. "Behavioral modeling and formal verification of a resource discovery approach in Grid computing." *Expert Systems with Applications* 41 (8): 3831–3849.

TAT Group. 2010. "Sabena Technics - Airframe services." Http://www.sabenatechnics.com, accessed October 18, 2017.

Tayarani-Bathaie, S Sina, ZN Sadough Vanini, and Khashayar Khorasani. 2014. "Dynamic neural network-based fault diagnosis of gas turbine engines." *Neurocomputing* 125: 153–165.

Tinga, T. 2013. *Principles of Loads and Failure Mechanisms*. Springer-Verlag.

Tinga, Tiedo. 2010. "Application of physical failure models to enable usage and load based maintenance." *Reliability Engineering & System Safety* 95 (10): 1061–1075.

Tsagkas, Vassilis, Dimitris Nathanael, and Nicolas Marmaras. 2014. "A pragmatic mapping of factors behind deviating acts in aircraft maintenance." *Reliability Engineering & System Safety* 130: 106–114.

TSBC. 2019. "Statistical Summary – Aviation Occurrences 2016." Accessed 2019-01-10. `http://www.tsb.gc.ca/eng/stats/aviation/2016/ssea-ssao-2016.asp`.

Van den Bergh, Jorne, Philippe De Bruecker, Jeroen Beliën, Liesje De Boeck, and Erik Demeulemeester. 2013. "A three-stage approach for aircraft line maintenance personnel rostering using MIP, discrete event simulation and DEA." *Expert Systems with Applications* 40 (7): 2659–2668.

Vardi, Moshe Y. 1995. "Alternating automata and program verification." In *Computer Science Today*, 471–485. Springer.

Verhagen, Wim JC, and Lennaert WM De Boer. 2018. "Predictive Maintenance for Aircraft Components using Proportional Hazard Models." *Journal of Industrial Information Integration* .

Viles, E, D Puente, MJ Alvarez, and F Alonso. 2007. "Improving the corrective maintenance of an electronic system for trains." *Journal of Quality in Maintenance Engineering* 13 (1): 75–87.

Visser, Willem, Klaus Havelund, Guillaume Brat, SeungJoon Park, and Flavio Lerda. 2003. "Model checking programs." *Automated software engineering* 10 (2): 203–232.

Waltz, E, and DL Hall. 2001. "Requirements derivation for data fusion systems." *Handbook of Multisensor Data Fusion* 15.

Wan, Shan, Dongbo Li, James Gao, Rajkumar Roy, and Fei He. 2018. "A collaborative machine tool maintenance planning system based on content management technologies." *The International Journal of Advanced Manufacturing Technology* 94 (5-8): 1639–1653.

Wang, Hongzhou. 2002. "A survey of maintenance policies of deteriorating systems." *European journal of operational research* 139 (3): 469–489.

Ward, Marie, Nick McDonald, Rabea Morrison, Des Gaynor, and Tony Nugent. 2010. "A performance improvement case study in aircraft maintenance and its implications for hazard identification." *Ergonomics* 53 (2): 247–267.

Yahyaoui, Hamdi, Zakaria Maamar, and Khouloud Boukadi. 2010. "A framework to coordinate web services in composition scenarios." *International Journal of Web and Grid Services* 6 (2): 95–123.

Yan, Shangyao, Ching-Hui Tang, and Tseng-Chih Fu. 2008. "An airline scheduling model and solution algorithms under stochastic demands." *European Journal of Operational Research* 190 (1): 22–39.

Yang, Zimin Max, Dragan Djurdjanovic, and Jun Ni. 2008. "Maintenance scheduling in manufacturing systems based on predicted machine degradation." *Journal of intelligent manufacturing* 19 (1): 87–98.

Zadeh, L.A. 1965. "Fuzzy sets." *Information and Control* 8: 338–353.

Zhang, Mimi, Zhisheng Ye, and Min Xie. 2014. "A condition-based maintenance strategy for heterogeneous populations." *Computers & Industrial Engineering* 77: 103–114.

Zhang, Pengcheng, Henry Muccini, and Bixin Li. 2010. "A classification and comparison of model checking software architecture techniques." *Journal of Systems and Software* 83 (5): 723–744.

Zhang, Zhinan, Gang Liu, Zhichao Jiang, and Yong Chen. 2015. "A cloud-based framework for lean maintenance, repair, and overhaul of complex equipment." *Journal of Manufacturing Science and Engineering* 137 (4): 040908.