Aufgabenstellung

zur

Abschlussarbeit
im Studiengang Master Automotive Software Engineering

für

Herrn Girish Patil
geb. am 21. Oktober 1990 in Raichur

zum Thema

Design of a generic runtime monitor approach using formal specifications to enhance UAV situational awareness

Betreuer/ Prüfer: Prof. Dr. Wolfram Hardt
Auszugedatum: 13.08.2015
Abgabedatum: 20.01.2016
Tag der Abgabe:

Unterschrift: Prof. Dr. F. Hamker
Vorsitzender des Prüfungsausschusses
"Design of a Generic Runtime Monitor Approach using Formal Specifications to Enhance UAV Situational Awareness"

Master Thesis
"Final Report"

Fakultät Informatik
Masters in Automotive Software Engineering

At the

Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)
German Aerospace Centre, Institute of Flight Systems
Braunschweig, Germany

Author:
Girish Patil
ppatilgirish@gmail.com
Mat Nr. 328702

Supervisors:
Dipl.-Inform. Christoph Torens
Christoph.Torens@dlr.de
Prof. Dr. Wolfram Hardt
hardt@cs.tu-chemnitz.de
Dr. Ariane Heller
ariane.heller@informatik.tu-chemnitz.de

January 19, 2016
Acknowledgement

At First and Foremost i would like to thank from the bottom of my heart to Prof. Dr. Wolfram Hardt Leader of Professorship Computer Engineering, Technische Universität Chemnitz for his constant support throughout my master course. I can never forget his guidance and various opportunities that he provided me during my master course. I have been so lucky to complete my master course under his leadership. He is very friendly with students and its always awesome to be around with him. I have quite lot of awesome memories with professor during conferences at Laubusch. Thank you so much Professor for all the support, motivation and opportunities.

Dr. Ariane Heller faculty, Computer Engineering, Technische Universität Chemnitz has supported me a lot throughout my master course and has always helped me through tough times. She has encouraged me a lot to do my work perfectly. I am really lucky to have you as my supervisor. Any kind of academic difficulty which i came across during my course the first person whom i always approached for help was her. Thank you madam for your immense support and encouragement.

Dipl.-Informatik Christoph Torens Research Scientist, Deutsches Zentrum für Luft und Raumfahrt has guided throughout my internship and thesis work at DLR. Its been so wonderful to work under him. He is a wonderful person at heart and he is one such person who never lets his students down. He pumps the energy in students to do the work in clean and perfect way. I have been so lucky to work under him and i have learnt many things not only for professional life but also for personal life. He always lends his helping hands for students irrespective of his work time and day. I once again thank him for all that he has shaped me professionally. He shall always be very special to me

M,Sc Florian Adolf Research Scientist, Deutsches Zentrum Für Luft und Raumfahrt has been a brainstormer during my internship work and as well as for thesis work. He is such a person who provides a clear path ahead in the work. He makes it simpler and achievable by his motivational thoughts. Its always fun to be around with him and he always makes people around him so comfortable. I am lucky to get his guidance. Amidst his busy schedules he used to give time for students and that has helped me a lot.

I would be really thankful to Prof. Dr. Uranchimeg Tudevdagva for various opportunities and support.
Abstract

Software is the crux of many commercial, industrial and military systems. The software systems need to be very reliable especially in case of safety critical systems. Unmanned Aerial Vehicle (UAV) and manned aircraft are safety critical systems and hence failures related to software or software-hardware interaction leads to huge problems. The software systems need to be certified before they are deployed. Even after being certified several accidents and incidents have occurred and are occurring. The software errors can occur during any phase of software development. The reliability of the software is enhanced using the verification process. Runtime monitoring has various advantages over testing and model checking. Hence this thesis work explores runtime monitoring of UAV. The runtime monitoring shall verify the run of the current system state. The runtime monitoring shall monitor the health of the UAV and shall report to the operator about its status. The software faults and errors if not prevented shall lead to software failure. UAV lacks the situational awareness due to absence of pilot onboard. This motivated to use runtime monitor to enhance the situation awareness. The runtime monitor shall detect the software errors and avoid failures. This monitor shall also enhance the situational awareness of the remote operator. The runtime monitor that enhance situation awareness shall not only be applicable to specific UAV but this shall be applicable to all the UAV’s. Hence this work provides an independent Generic Runtime Monitor (GRM) to enhance the situation awareness. The runtime monitor has various methods but using formal specifications in specific using Linear Temporal Logic (LTL) to generate monitor is considered in this work. Runtime monitoring makes UAV more safe and at the same time reduces the costs as it verifies only the current run of the system state by providing a detection of critical errors. The situation awareness includes functional and environmental states that remote pilot shall not be aware of. The architecture plays vital role for the system design. GRM architecture is one such architecture which chalks out the overall independent system design for the runtime monitoring of the UAV system. This architecture is an extensible one. The generic requirements were elicited from different sources such as Aircraft Incidents and Accidents, Boeing Aero Magazine, Autonomous Rotorcraft Testbed for Intelligent Systems (ARTIS) requirements, generic Autonomy Levels for Unmanned Rotorcraft Systems (ALFURS) framework etc. The situation awareness can be categorized into three levels namely perception, comprehension and projection. The requirements were elicited for all the three levels of situation awareness. These requirements further formalized using temporal logics. The formalized requirements further translated into state automaton automatically.

Keywords Runtime Monitoring, Guidance System, Navigation System, Situational Awareness, Perception, Comprehension, Projection.
## Contents

| Acknowledgment | ................................................. | ii |
| Abstract       | ................................................. | iii |
| List of Figures | ................................................. | viii |
| List of Tables  | ................................................. | ix |

### 1 Introduction

| 1.1 Motivation                                           | ................................................. | 1 |
| 1.2 Problem Statement and Research Objective            | ................................................. | 3 |
| 1.3 Definition of Terms                                  | ................................................. | 6 |
| 1.4 Structure of Thesis                                 | ................................................. | 7 |
| 1.5 Summary                                             | ................................................. | 8 |

### 2 Elementary

| 2.1 Runtime Monitoring                                  | ................................................. | 9 |
| 2.1.1 Runtime Monitoring Features and Purposes          | ................................................. | 10 |
| 2.1.2 Runtime Monitoring Applications and Constraints   | ................................................. | 10 |
| 2.1.3 Runtime Monitoring vs Formal Methods/ Other Techniques | ..................................... | 11 |
| 2.1.4 Monitor                                           | ................................................. | 13 |
| 2.2 Situational Awareness                               | ................................................. | 14 |
| 2.2.1 Levels of Situational Awareness                   | ................................................. | 15 |
| 2.2.2 Traits of situational Awareness                   | ................................................. | 18 |
| 2.3 Requirements Elicitation and formalized Requirements | ............................................. | 21 |
| 2.3.1 Requirements Formalisation                        | ................................................. | 23 |
| 2.3.2 State Automata                                    | ................................................. | 25 |
| 2.4 Software Architecture                               | ................................................. | 26 |
| 2.4.1 Aspects of Architecture                           | ................................................. | 27 |
| 2.4.2 Architectural Design Role in Software Engineering  | ............................................. | 28 |
| 2.5 Summary                                             | ................................................. | 28 |

### 3 State of Art

| ................................................. | ................................................. | 30 |

### 4 Methodology

| 4.1 Runtime Monitoring Methods                          | ................................................. | 34 |
| 4.1.1 Design by Contract                               | ................................................. | 34 |
9.1.2 Taxonomy of Inputs based on GNC System................................. 81
9.2 Monitor Taxonomy .................................................................. 81
9.3 Summary .............................................................................. 82

10 Results ................................................................................... 86

11 Conclusions and Future Scope ............................................... 88
List of Figures

2.1 Model Checking .................................................. 12
2.2 Runtime Monitoring ............................................. 13
2.3 Testing .......................................................... 14
2.4 Runtime Monitoring Flow Chart [20] .......................... 15
2.5 Runtime Reflection [29] ......................................... 16
2.6 Geographical SA .................................................. 18
2.7 Spatial/Temporal SA .............................................. 20
2.8 System SA ........................................................ 20
2.9 Environmental SA ................................................ 21
2.10 Tactical SA ...................................................... 22
2.11 Decision Making Process [8] ................................... 22
2.12 Next X p .......................................................... 23
2.13 Final F p .......................................................... 23
2.14 Global G p ........................................................ 23
4.1 Design by Contract ................................................. 35
4.2 Runtime Verification ............................................. 36
4.3 Monitor Oriented Programming ................................. 36
4.4 GRM Architecture Design ...................................... 38
4.5 System Activities [48] ............................................ 40
4.6 Legal Obligation [48] .............................................. 40
4.7 Fine Attuniation [48] .............................................. 41
4.8 Logical and Temporal Conditions [48] ......................... 41
6.1 GRM Architecture ................................................ 51
6.2 Onboard Processing ............................................. 56
6.3 Offboard Processing .............................................. 57
7.1 GRM Requirement Sources .................................... 60
8.1 Monitor Generation Process ................................... 71
8.2 Methods and Tools used to translate LTL to State Automaton ................................................. 72
8.3 LTL to State Automaton using SPOT tool, A state-based de-generalised Büchi automaton

8.4 LTL to State Automaton using SPOT tool, A transition-based Büchi automaton

8.5 LTL to State Automaton using SPOT tool, A Spin never claim

8.6 Digraph Automaton

8.7 State Automaton

9.1 Input Taxonomy Based On Aeronautical Knowledge

9.2 Input Taxonomy Based On GNC System

9.3 Taxonomy of Health Monitor related terms
List of Tables

1.1 Australian Transportation Safety Board report [52] ................................ 4
1.2 Situational Awareness Errors [31] ....................................................... 5

2.1 Situational awareness level 1 Errors [31] ............................................. 17
2.2 Situational awareness level 2 Errors [31] ............................................. 18
2.3 Situational awareness level 3 Errors [31] ............................................. 19

7.1 GRM Requirements for all 3 SA Levels ............................................... 65
7.2 GRM Requirements for all 3 SA Levels ............................................... 66
7.3 GRM Requirements for all 3 SA Levels ............................................... 67
7.4 GRM Requirements Aspect Priority ..................................................... 68
7.5 GRM Requirements for all 3 Aspects ................................................... 69

8.1 Boolean Operators for Temporal Formulas ......................................... 73
8.2 Syntax for Temporal Formulas ........................................................... 74
Chapter 1

Introduction

1.1 Motivation

Software systems are integral part of day to day human life. The software failure in safety critical systems not only leads to loss of capital but also leads to loss of lives. Software failure is termed when software does not perform the intended function. Software faults could occur due to inappropriate planning or monitoring of system or execution. UAV and manned aircraft are safety critical systems and hence failures related to software or software-hardware interaction leads to huge problems. Considering one of the real incident of passenger aircraft that is On 4/11/2010 a Qantas Flight 32, Airbus A380 passenger flight with 440 passengers on board, made a hard time for pilots to analyse and sort out 54 diagnostic messages after the engine explosion. It took 50 minutes for pilots to work on all the warning messages [4]. In case of any software related problems the diagnostic system shall be capable of detecting errors within stipulated time which is critical. In day to day life, technology has made our lives simpler from mobile phones to complex systems such as aircrafts. The software failure in mobile phones might not lead to great disasters but if failure occurs in aircrafts it leads to greater loss. These systems needs to be trusted and trust can be reassured by verifying the developed system. The three main types of system verification techniques are theorem proving, model checking and testing. The model checking and the theorem proving are the formal methods and as the complexity of the system grows it becomes difficult for verification. Testing scales better with the increase in the complexity. The draw back is that testing is not completely reliable for highly critical systems [11]. Formal verification or testing is not capable enough to demonstrate reliability of safety critical systems [10]. The pitfalls of the traditional methods need to be avoided. Runtime Monitoring avoids problems such as state explosion problem and system model abstractions. The runtime monitoring is cost effective as it monitors only the current run of the system state. This runtime monitor of UAV alerts the UAV operator to take necessary action before any error leads to worse condition. This paper provides a detailed overview of generic runtime monitoring of UAV.
The software faults if not detected and reacted it leads to failures. The software faults or failures leads to lack of situation awareness. If these software faults are detected as soon as possible and appropriate action is taken then failures does not occur. Runtime monitoring indulges in diagnosis of system as soon as possible and monitors only the run of the system state. Considering some examples where the software failures has led to loss of situational awareness in real incidents. On 2/10/1996 Aeroperu Flight 603 faced warning messages from onboard computer and out of those warning messages some of them were valid and rest were not valid ones. The pilot was unable to monitor aircraft airspeed and this led to rapid loss of altitude. The flight altitude constantly lowered and this led to aircrafts wingtip strike the water and aircraft crashed into water. All the 70 passengers including the crew members lost their lives. The warning messages due to failure of digital onboard system distracted the pilots and this led to loss of situation awareness, the pilots did not notice the radar altimeter. There is need for the software to be monitored runtime enhancing the situation awareness. If the radar altimeter was runtime monitored in the above incident to enhance the situation awareness then an alarm message shall have alerted the pilots as and when minimum altimeter had reached.

UAV is an aircraft without pilot on board. This absence of pilot onboard opens door for many possibilities of mission to be accomplished where human life would be at stake. There are certain challenges to be tackled due to absence of pilot onboard. One such challenge is lack of situation awareness. Situation awareness is to stay advanced of airplane. Perceiving the environmental elements where the airplane is flying, to understand these elements and to project their status in future is situation awareness. Pilot onboard the aircraft has better situation awareness due to possibility of real instinct with the environment around him. Good situational awareness is always necessary for safer flight operations be it manned aircraft or UAV. UAV situational awareness is bit of challenging to accomplish than compared to aircraft due to the reason that the operator shall rely on the sensor output. There is no possibility of natural instinct of UAV operator due to absence onboard flight. There is a loss of situational awareness due to the operator being placed outside the control loop.

Loss of situational awareness is been leading factor for accidents according to military aviation mishaps [7]. According to Endsley [55] larger percentage of human errors leading to accidents is attributed to loss of situational awareness. National Aeronautics and Space Administration’s Aviation Safety Program has provided seven targeted task areas were the human error can be reduced. Out of those seven tasks situational awareness is also one such task which needs to be tackled [14]. According to Australian Transportation Safety Board (ATSB) [21] the manned aircraft indicated that larger percent of all the incidents and accidents mainly due to loss of situational awareness. The table [1.1] depicts the percentage of casual factors involved in landing and approach accidents of manned aircraft. The table [1.1] is considered from AIRBUS technical report [52]. The row highlighted indicates that the 52 % of incidents have occurred due to loss of vertical and horizontal situation awareness. According to analysis conducted
by Endsley and Jones based on NASA’s Aviation Safety Reporting System (ASRS) Table 1.2 shows clearly SA Levels along with percentage of error[7].

Considering real incidents that occurred in unmanned aerial vehicles and passenger flights due to loss of situational awareness.

On 15/08/2011 at eastern Afghanistan RQ-7B Shadow UAV collided with the C-130 Hercules carrying soldiers and fortunately none were hurt. Due to loss of situation awareness UAV operator lost track of his aircraft [54]

On 19/11/2015 at Jaisalmer, western India Nishant UAV collided with high tension overhead wires. The operator lost control after the takeoff which led to this incident. [2]

On 28/12/1978 at Portland, Oregon United AL ran out of fuel while the pilot was busy analyzing the landing gear problem. This incident happened due to lack of situational awareness as crew concentrated only on the issue to solve the landing gear problem but unfortunately they didn’t concentrate on the fuel starvation.[47]

On 1/15/2009 at New York, US Airways ditched in the Hudson River due to loss of both engines after collision with Canadian Geese. There are many other incidents and accidents similar to these were loss of situational awareness leads to accidents.[47]

It is really necessary for the UAV operator to have a good situation awareness be it the internal conditions or the external environmental conditions. These real incidents and accidents convey the need for enhancement of situational awareness which is an important aspect for decision making process.

1.2 Problem Statement and Research Objective

DLR, German Aerospace Center, Institute of Flight System, Department Unmanned Aircraft, Braunschweig [37] research is focused on fleet of UAV such as Mini, Midi and Super-ARTIS. These UAV’s are categorized based on the size and altitude, the ARTIS fleet vehicles have vertical take off and landing systems. Apart from the ARTIS fleet research is also on fixed wing aircraft such as Prometheus and Micro Air Lab. Currently formal methods are used for validation and verification of the software system. Department Unmanned Aircraft, DLR actively researches on usage of formal methods such as model checking for verification and validation. The formal methods has its own advantages and disadvantages. One such disadvantage is that application of formal methods for large systems would be difficult and complex. This led to explore runtime monitoring of system which takes into account the current run of the state rather the complete system as incase of model checking.
Casual Factors in Approach and Landing Accidents in Manned Aircraft

<table>
<thead>
<tr>
<th>Factor</th>
<th>% of Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inadequate Decision Making</td>
<td>74</td>
</tr>
<tr>
<td>Omission of action or inappropriate action</td>
<td>72</td>
</tr>
<tr>
<td>Non-adherence to criteria for stabilised approach</td>
<td>66</td>
</tr>
<tr>
<td>Inadequate crew coordination, cross-check and back-up</td>
<td>63</td>
</tr>
<tr>
<td>Insufficient horizontal or vertical Situational Awareness</td>
<td>52</td>
</tr>
<tr>
<td>Inadequate or insufficient understanding of prevailing conditions</td>
<td>48</td>
</tr>
<tr>
<td>Slow or delayed action</td>
<td>45</td>
</tr>
<tr>
<td>Flight handling difficulties</td>
<td>45</td>
</tr>
<tr>
<td>Deliberate non-adherence to procedures</td>
<td>40</td>
</tr>
<tr>
<td>Inadequate training</td>
<td>37</td>
</tr>
<tr>
<td>Incorrect or incomplete pilot or controller communication</td>
<td>33</td>
</tr>
<tr>
<td>Interaction with automation</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 1.1: Australian Transportation Safety Board report [52]
The runtime monitoring is a lightweight verification technique. The important aspect of runtime monitoring is its behaviour of monitoring at runtime which provides an opportunity to act when the incorrect behaviour of the software system is identified. The lack of UAV situational awareness leads to accidents and enhancement is necessary. Therefore this thesis work combines both tasks that is runtime monitoring of UAV along with enhancement in situational awareness of UAV. This thesis proposed generic runtime monitor methodology to enhance the situational awareness of the UAV using the formal specifications. The larger goal or further work which is not a part of this thesis is not only detect the errors during runtime but at the same time correct the detected errors so that UAV operates back as normal without any technical comprise.

Each phase of the project comes across many problems that need to be answered during the research. This work addresses the problems starting from the requirements elicitation. The first and foremost part of the software life cycle is requirements elicitation. The requirements need to be generic that is it has to be applicable to all the UAV’s and these requirements shall be used for runtime monitoring. The constraints behind the elicitation of the generic requirements needs to be researched. This work integrates the runtime monitoring and enhancement of situational awareness. To do this the elicited generic requirements shall not only be applicable to runtime monitor but shall also enhance situation awareness. The situational awareness has three levels such as perception, comprehension and projection. The possibility of extraction of generic requirements for all the three levels of situation awareness need to be researched. Among the elicited set of generic requirements which of them can be runtime monitored is question that needs to be answered.

The requirements elicited were in textual format and hence there are possibility of human error. To avoid the human error the requirements need to be formalized.
The generic requirements need to be formalized to create a formal generic monitor. The question to be answered is whether this generic formal monitor can be regulatory compliance monitor. The classification of all these generic properties into a common taxonomy needs to be researched. If a common taxonomy is created then a standard template can be obtained. The retrofitting of runtime monitor has to be designed and to check the possibilities of accomplishment.

The unmanned aircraft are small aircraft compared to manned aircraft. They have small computation power and due to this possibilities for mapping of the onboard monitoring properties and off board monitoring properties shall be made adaptive. If this can be made adaptive then generic runtime monitor design shall be cost effective. This also enables to verify only the property that is vital onboard and hence needs less computation power.

The research objective of this thesis work is to design the generic runtime monitor. This generic runtime monitor needs to have an architecture which shall be capable to be modified further. The generic runtime monitor shall be generated with the help of formal specifications specifically linear temporal logics. The requirements considered shall enhance the situational awareness and shall be from all three different levels. The generic runtime monitor shall detect the errors as soon as possible. This shall prevent the error becoming a failure. This GRM shall provide information to operator and alarm whenever the violation in the system behaviour is detected. The taxonomy of inputs and different terms related to health monitor shall be created. A taxonomy of inputs provides overview to know better about categories of the respective inputs. A taxonomy of health monitor related terms distinguishes between the meaning of different terms and their corresponding category.

1.3 Definition of Terms

This section introduces some of the terms related to health monitoring by defining them. These terms are further co-related and compared in the chapter 3.

**Health Monitor**
It monitors hardware, application faults and failures. Its vital task is to isolate faults and prevent failure propagation [50]. The main aim of the health monitoring system is to detect and diagnose the initiation of any defect to analyse its effect and to trigger the maintenance workflow.

**Software Health Management**
Software Health Management address the critical need to detect, diagnose, predict and mitigate adverse events due to software faults and failures [23].

**Runtime Verification**
It deals with the study, development and application of those verification techniques
that allow checking whether a run of a system under scrutiny satisfies or violates a given correctness property [12].

**Monitor**

It is a device that reads a finite trace and yields a certain verdict [29]. The system health management techniques can be classified according to the life cycle phase in which they are applied [SHM], the run-time techniques are applied during the post-deployment phase.

**System Monitoring**

It is defined as a process within a distributed system for collecting and storing data. This is a fundamental principle supporting application performance management (Monitoring and Management of performance and availability of software applications) [50].

**Contingency Management**

The problem of recognising, assessing and responding to unanticipated events or conditions that impact plan execution [22]. The contingency planning system shall identify specific events that may occur in future and it shall plan for the alternate routes in order to respond to those upcoming events. The contingency management enables unmanned systems to become the autonomous systems. The prime elements are:

- Multi-Level Assessment: At multiple levels the monitoring, assessment and response occurs.
- Plan-Based Assessment: Based on the evaluation of the dependencies and constraints on the execution of the plan the monitoring is triggered.
- Capability-Based Assessment: Assessment of the vehicle mission capabilities based on the subsystem and environmental status.
- Predictive Assessment: Assessment and monitoring of future events.
- Team-based assessment: Assessment of the group of vehicles.

1.4 **Structure of Thesis**

The initial two chapters are introduction chapters. Chapter 1 provides an overall introduction to thesis work. The motivation behind choosing this work and the problems to be tackled in order to achieve the goals are explained in the first chapter. The various health monitoring terms are defined. The chapter 2 gives detailed overview about situation awareness, runtime monitoring, software architecture and state automaton. Runtime monitoring compared with the testing and the formal methods. The runtime monitoring advantages over the testing and the model checking is discussed. The applications of runtime monitoring is considered. The various features and also purposes of runtime monitoring are described. It also provides an introduction to the situational awareness and levels of situational awareness. The situational
awareness loss leads to accidents and percentage of error is discussed with respect to the individual levels. Further software architecture basics are delineated. This chapter also discusses the need for architecture and the vital trails of architect. Chapter 3 delineates the related work and also explains what is different in this thesis from the existing works. The health monitoring terms are analysed and comparison between them is discussed. The chapterput forth various methodology of runtime monitoring. There are four methodology of runtime monitoring and all the methodologies are discussed. The requirement elicitation methodology such as analytical approach followed by natural language method. Formalisation of the requirements are discussed.

The chapter discuss the contributions towards this thesis. The brief overview of the thesis contributions are explained. Architecture plays a vital role in system design phase. In the chapter Generic Runtime Monitor Architecture is discussed in depth along with detailing of the monitor along with process outline. In Chapter GRM requirements are discussed and the requirements are stated as some of the examples. The prototyping of the generic runtime monitor is explained in the chapter In Chapter explains about the taxonomy of the GRM inputs and also about the health monitor related terms. The results of the overall thesis work is summarised in chapter Last chapter that is is the conclusion and the work is over looked once.

1.5 Summary

This sections introduces to many of the health related terms and their definitions. The software errors are increasing in day to day software applications. The verification of the system is been a vital instrument to avoid errors. The software errors in safety critical systems not only leads to loss of property but also loss of life. The aircraft incidents and accidents are considered. The verification techniques such as model checking and testing have limitation in the applicability as the complexity of the system increases. The runtime monitoring overcomes all the drawbacks of both testing and model checking. The runtime monitoring verifies the run of the system. The runtime monitoring detects the software errors as soon as possible. The software faults leads to failures. If the software faults are detected earlier then failure does not occur. The UAV lacks the situational awareness due to the absence of the pilot on board. The accidents of UAV due to situation awareness is high. This motivated to enhance the situational awareness of the remote pilot operator using runtime monitoring. This chapter discuss the various research problems and the objective of this work. Finally the structure of the thesis report is described.
Chapter 2

Elementary

Elementary/Fundamentals this chapter provides the introduction to many of the important concepts. These concepts which are discussed in this unit are considered in further chapters. The understanding of these concepts helps to comprehend the thesis work. Runtime monitoring is the light weight verification technique. Runtime monitoring shall detect the errors as soon as possible. Runtime monitoring is further explained in detailed in comparison with model checking and testing. Further the runtime monitoring features, purposes, applications and constraints are discussed. Staying advanced to airplane can be simply termed as situation awareness. Situation awareness has been categorized into three levels namely perception, comprehension and projection. Situation awareness along with all these three levels are explained in this chapter. Design of any software systems need architecture. The software architecture and its need is further explained. The state automaton basics are discussed.

2.1 Runtime Monitoring

The software systems being a part of embedded application domains which involves lives of the people is crucial. The software deployed needs to be secure, correct and reliable. For example, the software within the UAV’s Anti-icing system needs to alert the UAV operator as soon as the accretion of icing starts on the non-protected surfaces, else it leads to problems. Current days safety critical systems needs to have certification for the system which requires proof of the software properties in terms of the documented verification process. Verification has to check whether each property acts as it was supposed to act. There are formal verification techniques such as theorem proving, model checking etc. Apart from formal methods, testing is also used for verification. Runtime Verification is lightweight verification technique [29]. Further subsection 2.1.3 delineates comparison of runtime monitoring against model checking and testing. Runtime Monitoring approach extracts the important information from the system which is running and this information is used to detect the faults. Further
possibly to react to observed behaviours which violates properties. The properties captured are formal specifications. These formal specifications are basically expressed such as finite state machine, linear temporal logics, regular expressions etc. The complexity involved by usage of formal methods shall be avoided by runtime monitoring.

2.1.1 Runtime Monitoring Features and Purposes

The key features of runtime monitoring are:

- It is used to easily check the actual execution of the system in order to make sure that the implementation actually meets the correctness properties.
- Some information really needs to be checked at runtime only and in such cases the runtime verification is conveniently checked.
- Application behavior might be depended on the environment of the target system and if in case the precise knowledge about the environment does not exist then it is hard to predict the behavior and it is also hard to analyze before execution so here application needs to be verified at the runtime.
- In safety critical systems it is important to monitor the behavior or properties so that nothing goes wrong at any time

The various purposes of the runtime monitoring are:

- Debugging
- Testing
- Security
- Safety Monitoring
- Verification
- Validation
- Fault Protection
- Profiling
- Behaviour Modification

2.1.2 Runtime Monitoring Applications and Constraints

The applications of runtime monitoring are:

- Safety/ Mission Critical Systems
- Enterprise
• System Software
• Autonomous Systems
• Reactive Control Systems
• Health Management System
• Diagnosis System
• System Security and Privacy

The vital constraints to be concentrated while using run-time monitoring in critical real time software are:

• Functionality: The runtime monitoring shall not change the target’s behavior unless and until the target has violated the specification.

• Certifiability: The runtime monitoring must not make re-certification of the target onerous.

• Timing: The runtime monitoring must not interfere with the target’s timing.

• SWaP: The runtime monitoring must not exhaust size, weight and power (SWaP) tolerances [46].

2.1.3 Runtime Monitoring vs Formal Methods/ Other Techniques

The formal verification techniques provides such a mathematical proof such that the design can never violate the specification. The formal techniques include Model Checking, Theorem Proving, Bounded Model Checking, Model Based Testing. The formal verification is different form the dynamic verification. Formal verification techniques for example model checking basically checks if all the executions of the system conform to the specifications mentioned. Dynamic verification checks whether a particular verification of the system conforms to the specification.

Runtime Verification Vs Model Checking

This section brings the difference between the runtime monitoring and model checking. Even though similarities does exist between both the techniques but there are differences between them. In model checking the system under scrutiny need to be examined for all its executions against their correctness property to check whether they satisfy the properties or not. It faces the language inclusion problem. Whereas with the runtime monitoring the system under scrutiny faces the word problem. According to [12] word problem is considered to be lower complexity than compared to the language problem. Runtime Monitoring with the LTL considers the finite traces. The model checking with LTL considers the infinite traces [12]. Considering the figure 2.1 which expresses that all the system executions must be checked in the model checking.
In figure 2.1 each and every node represents the state of a system. In model checking all the system states shall be checked before actually running the system. In case of runtime monitoring there is no need to have a model in hand prior to execution of the system. The runtime monitoring deals only with the observed executions as it is shown in figure 2.2. In figure 2.2 each and every node represents the state of the system. All the states are not checked here as it was done in case of model checking. Only the states that are currently executing are verified. Runtime monitoring is applicable for the black box systems because there is no prior need for model. Model checking has disadvantage as it suffers from the state explosion problem. It is very easy to build a model for a system which is not so complex but it gets really difficult when system is complex. The fact of analyzing all the executions of such a complex system becomes huge.

**Figure 2.1: Model Checking**

**Runtime Monitoring Vs Testing**
Runtime Monitoring and testing have similarities in common. The reason is that runtime monitoring does not take into account all the executions of the system but only single or finite subset and hence does testing. In runtime monitoring the monitor is either attached to system or it is in inbuilt in system which checks the properties. Monitor is generated with formal specifications. Only the input sequences are considered during testing but it is not the same with runtime monitoring [12]. Testing checks some of the states and it usually has coverage problem. As shown in the figure 2.3 the nodes represent the states and the vertical lines shows the transaction form one state to the other state. In case of testing most of the states won’t be even checked [5].
Figure 2.2: Runtime Monitoring

2.1.4 Monitor

The runtime monitoring traditional approach is shown in figure 2.4. The system or subsystem need to be initialised and once the system is initialised it requires instrumentation to send state variables to monitor. System then basically waits for any of the updates to happen. If the system state is not changed then system waits until any changes to happen. When the system state changes then safety property violation is checked. If the safety property is violated then alert of property violation shall be sent and later shall be reset. If there is no safety property violation then system waits for updates.

**Online Verification Vs Offline Verification**

The execution trace of the system can be verified during online or offline. During the online verification of the system the target system shall be run synchronously in parallel to the monitoring system. Online verification has an advantage that if property violation is detected during the runtime then the bad behaviour of the system can be corrected. In case of offline verification the monitoring is done for the recorded events, log of the system is monitored. This does not provide a chance to correct the bad behaviour while flying. The online and offline verification mainly depends on the purpose of the usage. If the purpose is to detect the errors and to correct the bad state of the system on board aircraft then online verification can be used. If the purpose is to just find errors in the target system then either offline verification or online verification can be used.

**Runtime Reflection**

The runtime reflection mitigates the error if the safety property violation exists. The figure 2.5 shows the architecture of runtime reflection. The first layer is logging which
records all the events. It not only records all the events but it also provides the recorded events in a suitable format for further monitoring layer. Monitor layer detects the faults in the system. If the safety property is violated then alarm signal is generated. The diagnosis layers provides the explanation for the failure or property violation. The system is then reconfigured to solve the problem through mitigation layer.

### 2.2 Situational Awareness

The pilot not being on board the UAV has many advantages but at the same time there are many disadvantages also. Critical missions where human life shall be at stake can easily be accomplished with UAV’s. The major disadvantage is that there is lack of situational awareness due to the absence of the pilot on-board. Situational awareness is one of the cognitive capabilities of the human being. The absence of onboard pilot creates lack of situation awareness and this motivated for the enhancement of situation awareness. Additionally there are lot of aircraft accidents have happened till date due to lack of situational awareness. According to Australian Transportation Safety Board (ATSB) the manned aircraft indicated that larger percent of all the incidents and accidents mainly due to loss of situational awareness [52]. The table 1.1 depicts the percentage of casual factors involved in landing and approach accidents of manned aircraft. Situational awareness drives for the decision making and helps to take appropriate actions. Further [1] provides an overview of human situational awareness against the UAV situational awareness.
2.2.1 Levels of Situational Awareness

Situational Awareness can be better understood with the help of three levels. The three levels of situational awareness are:

**Perception of environmental elements**

The perception means being aware of things through senses. Perception of the environmental elements is the lowest level of situational awareness. The UAV operator needs to perceive many relevant elements in the environment. The environmental elements that are important during flying such as airspeed, position, altitude, temperature, wind etc. The operator needs to accurately perceive all the environmental elements. In the manned aircraft perception is to be aware of elements such as relevant system data, flight data, position of the other aircraft in the cockpit. The elements are analysed individually. Considering some of the examples of SA Level 1 [9]

- Vehicle Status such as “Speed of the Vehicle”, “Location of the Vehicle”, “Fuel Level”, “Heading of the Vehicle”.

---

Figure 2.4: Runtime Monitoring Flow Chart [26]
Weather Conditions such as “Temperature”, “Wind”, “Pressure”, “Thunderstorm”.

Obstacle/Object Status such as “Location of Obstacle”, “Distance to Obstacle”.

Vehicle Orientation such as “Pitch of Vehicle on Terrain”, “Roll of the Vehicle on Terrain”.

The perception is being aware of all the environmental elements through pilot sense organs such as mouth, eye, tongue, ears and nose. Due to non availability of pilot on vehicle, the sensors on-board have to play a vital role in assisting the UAV operator who is remotely located. The perception of the environmental elements acts as the input for the next levels of situational awareness, SA Level 2 and SA Level 3. The table 2.1 considered from the [7] shows the situational awareness level 1 error percentage in detail.

Comprehension of current situation

The second level of the situational awareness is the comprehension of the current situation. The first level of situational awareness is to be aware about the environmental elements individually. The second level is understanding of those individual elements and their significance based on the mission goals. Pilot/Operator need to be able to integrate those level 1 data elements and understand them in the view of ones goals. The examples of situation awareness level 2 are:

- Impact of Weather and Terrain
Situational Awareness Level 1 Errors

<table>
<thead>
<tr>
<th>Situational Awareness Level 1</th>
<th>% of Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data not available</td>
<td>11.6</td>
</tr>
<tr>
<td>Data hard to detect</td>
<td>11.6</td>
</tr>
<tr>
<td>Failure to monitor/observe data</td>
<td>37.2</td>
</tr>
<tr>
<td>Misinterpretation of data</td>
<td>8.7</td>
</tr>
<tr>
<td>Memory Loss</td>
<td>11.1</td>
</tr>
</tbody>
</table>

Table 2.1: Situational awareness level 1 Errors [31]

- Deviations from the Pre Planned Operation

The comprehension of the current situation is based on the combination of observations from real world with knowledge and past experience. The experienced pilot has better edge over the novice pilot. Integrating data elements without much experience shall be difficult for novice pilots. Novice pilot will be capable enough to analyse the environmental elements as same as the experienced pilot in SA level 1. The novice pilot lags the experiences for the memory recall to understand the situation well. The table 2.2 considered from the [7] shows the situational awareness level 2 error percentage in detail.

Projection of future status
After understanding the overall situation the next up is to think ahead in order to avoid the circumstances from happening. The third and the highest level of situational awareness is projection of future status. To achieve this level of situational awareness it is very vital to carefully gather the data and be aware of it and to understand the overall situation correctly so that the projection towards achieving goals are successful. Considering some of the examples:

- Projected location
- Projected collision distance
- Projected destination of the vehicle
### Table 2.2: Situational awareness level 2 Errors

<table>
<thead>
<tr>
<th>Situational Awareness Level 2</th>
<th>% of Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incomplete mental model</td>
<td>3.5</td>
</tr>
<tr>
<td>Incorrect mental model</td>
<td>6.4</td>
</tr>
<tr>
<td>Over-reliance on default values</td>
<td>4.7</td>
</tr>
</tbody>
</table>

- Projected locations without communications
- Projected time to task completion
- Projected next task

The table 2.3 considered from [7] shows the situational awareness level 3 error percentage in detail.

#### 2.2.2 Traits of situational Awareness

The traits of situational awareness considering from [8]

- **Geographical SA**: The geographical situation awareness includes runway assignments, location of own aircraft and in comparison to the other aircraft, the corresponding terrain features, path to desired location etc. The figure 2.6 shows geographical situational awareness.

- **Spatial/Temporal SA**: The attitude of an aircraft, the heading direction, altitude, deviation from the flight plan, velocity all these come under the spatial or temporal situation awareness. The figure 2.7 shows all that comes under spatial/temporal situational awareness.

- **System SA**: The system situation awareness includes system status, flight modes, system performance, air traffic control communication, time and distance available on fuel etc. The figure 2.8 shows all that comes under System situational awareness.

- **Environmental SA**: The environmental awareness as the name itself suggests that all the environmental elements/parameters come in this category. The parameters are environment temperature, winds, icing, fog etc. The figure 2.9 depicts related elements of environmental situational awareness.
### Situational Awareness Level 3 Errors

<table>
<thead>
<tr>
<th>Error</th>
<th>% of Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incomplete/poor mental model</td>
<td>0.4</td>
</tr>
<tr>
<td>Over-projection of current trends</td>
<td>1.1</td>
</tr>
<tr>
<td>Other</td>
<td>1.9</td>
</tr>
</tbody>
</table>

Table 2.3: Situational awareness level 3 Errors

Figure 2.6: Geographical SA
Figure 2.7: Spatial/Temporal SA

Figure 2.8: System SA
Tactical SA: The tactical situation awareness includes aircraft directions, tactical type, mission status, mission timing, projected threat intentions etc. The figure 2.10 depicts related elements of tactical situational awareness.

The situational awareness is the driving force for the decision making process as shown in figure 2.11. Further based on these decisions certain actions need to be performed. All situational awareness levels need to be evaluated correctly. Being aware of all the environmental elements. Understanding the current situation by integrating all the elements and then to project the future status. All levels need to be perfectly analysed in order to take decisions which further leads to performing relevant actions.

2.3 Requirements Elicitation and formalized Requirements

The requirement engineering process includes the four major steps namely requirement elicitation, requirement analysis, requirement documentation and validation. Requirements elicitation process includes activities that enable comprehending of results and this motivates building up interested system [13]. The need for the requirement has to be judged by thinking on the ‘happening of the consequence’, if not included. The requirement statement shall be verifiable either by examination, analyses, test or demonstration [30]. Requirement in order to be attainable it must be feasible technically and should fit within the budget and schedule. The requirement if cannot be affordable shall not be worth to consider in writing. The commonly made mistakes are:

- Bad Assumptions: This occurs mainly due to two reasons. Firstly because the
Figure 2.10: Tactical SA

Figure 2.11: Decision Making Process

CHAPTER 2. ELEMENTARY
requirement author does not have access to enough information and secondly the information itself does not exist.

- Usage of inappropriate terms: In a specification certain terms has to be avoided and some has to be used in the specific manner. The terms 'are', 'is' and 'was' have no prominence in the requirement. The words to be avoided while writing the requirements are 'etcetera' and 'but not limited to', these terms are ambiguous.

- Over-Specifying: It leads to cost overrun on their respective programs. Stating something that is unnecessary or stating stringent requirements basically over-specifying the requirements.

- Unverifiable: Usage of ambiguous terms makes a pathway for the unverifiable requirements. Each and every requirement has to be verified. The writing of the unverifiable requirements was avoided by restricting the usage of the terms such as maximize, minimize, rapid, user-friendly, easy, sufficient and quick.

- Missing: Requirements shall be missed when focused on only one part of the system. To avoid missing of the requirements an outline for the specification was short down.

2.3.1 Requirements Formalisation

The formal specification translates the non-mathematical description into a formal specification. The informal specification of the requirement lacks organization. The ambiguity remains certain in the informal specification of the requirements. These informal requirements lack the completeness and they create huge misunderstanding among the readers. The only solution for all the above mentioned problems is to represent them using the mathematical way and that can be done using the formal specification. The formal specification not only eliminates the imprecision but it also removes the ambiguity. The formally specified requirements are used for model checking. There is growing need of usage of formal specification. In our work the requirements were specified using the template based method that is the semi formalized specification method. The requirements can be high level or it can be a low level one. The high level requirement delineates the intended software functionalities, 'WHAT' the software shall do. The low level requirement delineates about 'HOW' the software shall execute the delegated functionality.

Temporal Logics

Dining Philosophers Problem: The resource sharing is the common problem in many of the systems. Considering philosophers seated around a table with food in the center. There are only 5 forks lying between the each pair of philosophers. Philosopher needs a pair of forks to eat one to their left and the other to their right. The
philosophers are allowed to pick up the forks simultaneously or in any order they like.

The formal specification helps to capture the essence of the above problem because the formal specification has well defined semantics. The temporal logic is classified based on the particular perception of time.

The specification can be represented using the logical framework. The logic becomes the necessity part to develop language that can model the situations and that makes the way to reason about them formally. The propositional logic can express the specification such a way that the logical structure can be brought out.

The temporal logic helps to represent the requirements in the mathematical way. Using the temporal logic the system properties can be expressed formally. The temporal logics delineate the temporal relations between the events occurring over time. The temporal logic notation is often simpler and clearer. Model checking is one of the methods to verify the temporal properties of the system. Temporal logic allows for formal specification of properties such as safety (nothing bad will happen), liveness (something good will happen) and fairness (independent processes will progress) [43].

Linear Temporal Logic LTL

The linear temporal logic describes the events only along the single computation path. The linear temporal logic has got two operators namely logical operators and temporal operators.

Logical Operators:

- \&: Logical AND
- |: Logical OR
- !: Logical NOT

Temporal Operators:

- X: It refers to next state
- G: Globally
- F: Finally
- U: Until
- Y: Previous state
- S: Since
The temporal operators such as $X$, $G$, $F$ and $U$ basically represent the future and the operators such as $Y$ and $S$ represents the past. The figure 2.12 indicates the Next state. The circles in the figure refer to states. The figure 2.13 indicates the final state. The figure 2.14 indicates globally for all the states.

2.3.2 State Automata

Finite Automaton is a graphical representation which consists of finite number of vertices called states. The edges in case of the finite state automaton are called as transitions. States are represented in circles. Circle inside a circle represents final state of an automaton. In each and every finite automaton there shall be an initial
state and final state. States and transitions are the vital components of the finite state machines. A finite state automaton is also called as finite state machine. This machine can only be in one particular state at a time. The state it is at a particular time is called as current state. Triggering event or condition when initiated or triggered can change a state from one state to another. This change of state from one to another is called as transition. Considering some of the examples of the state machine being applied in day to day applications are vending machines, elevators and traffic lights. State indicates the current status of the system which indeed is waiting to execute a transition. Further transition can be defined as the set of actions that need to be executed/run if in case the condition/transition is fulfilled.

Büchi automaton extends a finite automaton to infinite inputs. Infinite words on automaton was introduced by büchi automaton. Considering a specification which is bounded, eventuality is bounded in the example then we can opt finite words on automaton. When our specification is not bounded or it not comprehensible how one can specify a property of that kind as finite behaviour. This infinite behaviour can be easily expressed using the linear temporal logic. But even though linear temporal logics can express the infinite behaviour but it is not that expressive. Hence this drives towards büchi automaton. Büchi automaton are more expressive than LTL.

Transition Based Generalised Büchi automaton consists of labels on the transitions. Importantly generalised acceptance condition on transition too. Run is said to be accepting if transitions are labelled by each acceptance condition infinitely often. The acceptance conditions on transitions. State Based De-generalised Büchi Automaton is quite popularly used in many of the main stream algorithms. Acceptance condition is on states.

2.4 Software Architecture

Architecture is defined by the recommended practice as the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution (IEEE 2000). It can simply be said that architecture is the macroscopic organization of the system and how it can influence the overall behaviour of the system. Every system has an architecture and it is not like there is an software system without an architecture. Considering an example of UAV GNC systems, in order to understand it we should know about how the individual parts within the systems work together, by what means are they connected to each other and how would they interact with the environment around them, the other word for this is called as architecture.

The structure of the system shall be defined by the architecture of that system. The requirements that we consider need to be met by the architecture design. An architecture specifies the component communication. Architectural patterns provides the structures for communication between the components. Architectural patterns
makes it simple for all to understand and follow. The architectural patterns not only provides the communication structure but it also provides the structure for the components.

The architecture designs must be decided at the early stages of the development of a project because it would be really expensive as well as sometimes might not be possible to make any changes at later stages of the lifecycle. After the completion of the architectural development the later would be coding. The necessary confidence for an architect to go further with his architectural design can only be assured with the help of the prototyping of design.

Need for Architecture
The need for an architecture can be understood when we try to consider the things that’s done without considering the architecture. So it would be really good to know the consequences when the architecture is not being considered which clearly justifies the need for an architecture [18]. The following points mentions about the consequences without architecture such as

- Developers optimize locally and the main important aspect of considering the big picture is totally missed. In this case framework chosen would be lousy
- It leads to the poor communication among the team members
- Shallow analysis of the design options. It needs high efforts. There would be very little attention given towards the rationale.
- Architectural patterns would be ignored or there are chances that the patterns might be chosen incorrectly.

2.4.1 Aspects of Architecture
The architecture has different aspects. The important aspects are:

- At first and foremost the architecture delineates the purpose of the system and the usage of this system to particular community.
- The relationship of the system with the other subsystem and also with the environment around it can easily be understood with the help of an architecture of the system.
- The requirements are prominent as they shape the overall architecture
- The main building blocks of the architecture are the subsystems and it also delineates the way to combine all these subsystems for the proper functioning.
- The interfaces between the subsystems are clearly indicated in the architecture.
- The architecture delineates about all the subsystems [39]
2.4.2 Architectural Design Role in Software Engineering

Effective software engineering requires the architectural software design [32]. The important points which justifies the statement are

- Relationship among the high level systems can be clearly understood based on the recognition of the common paradigms which provides the modularity aspect that the system can further be upgraded.

- Software system design success depends on the architectural design of the system, architectural design is crucial and hence the mistake within the architectural design can lead to bad results

- Architectural design provides the big picture of the system at its entirety

- Architectural design helps to analyse the complex system properties

2.5 Summary

Runtime Monitoring is the lightweight verification technique. The runtime monitoring is compared with testing and formal methods such as model checking. The runtime monitoring tackles some of the problems faced by testing and model checking. As the system complexity increases usage of formal methods such as model checking would be not feasible as it faces the state explosion problem. The runtime monitor has a major advantage of its nature of monitoring at runtime. Due to this reason runtime monitoring does not face state explosion problem. The runtime monitoring can be used to monitor online as well as for offline monitoring. The runtime monitoring constraints such as functionality, certifiability, timing and SWaP are important factors. The traditional methodology of the runtime monitoring flow chart is delineated.

The situational awareness is about just getting to know around us. The situational awareness in case of aviation system is to have the environmental awareness along with the internal behaviour of the aircraft. Situation Awareness (Staying advanced of aircraft) is the driving force for the decision making process. Best trained and most experienced pilots can make wrong decisions if they have incomplete or inaccurate situation awareness. Australian Transportation Safety Board (ATSB) [21] indicates larger percentage of accident reports due to loss of situational awareness. In case of UAV there is further more need for the enhanced situational awareness due to the lack of pilot on-board. This lack of awareness creates more problems during UAV operation. Runtime monitoring such requirements provides a greater support to the operator to fly UAV. The situational awareness enhancement is very crucial as the table 2.1, table 2.2 and table 2.3 provides the error percentage of SA level 1, 2 and 3 respectively.

The architecture plays a prominent role in system design. The need for the architecture, aspects of architecture are introduced. Effective software engineering requires
architecture to be part of it. The architectural design role in software engineering is also discussed. Further the requirements elicitation is another important step towards design process. Writing good requirements is discussed in this chapter. The requirements needs to have common structure. The formalisation of requirements helps to generate the monitor hence the introduction of the formalisation of the requirements in explained above. The temporal logics is considered in this work for the formalisation. The basics of the linear temporal logics is discussed.
Chapter 3

State of Art

This thesis work initially indulges in the analysis of the related work on the different health monitoring terms which are related to each other. The difference in their meaning is discussed. The different terms analysed are health monitoring, system health monitoring, structural health monitoring, system health management and runtime monitoring. Later part of the related work describes the need for runtime monitoring and situational awareness. Health monitoring systems are taken into account on structural level as well as on systems level. Structural health monitoring takes into account of the structural integrity by online monitoring of the damage. System health monitoring looks on to the degradation of the performance and also on the functional aspects.

Considering related work on health monitoring of engine system, Bai et al. [20] proposed the wireless sensor network for the monitoring the health of the aircraft engines. This work describes about the architecture of the wireless sensor network which reduces the usage of huge wiring and wiring harnesses. Tumer et al. [35] work presents the survey of the engine health monitoring systems for commercial aircraft. Engine health Monitoring system poses challenges mainly due to the abundance and ambiguity of the data to be interpreted and also due to the large number of false alarms. Their intended survey was to identify the major need in the area of the health monitoring and also resolve the existing problems in current practices. This work explains about the parameters taken into account for engine monitoring systems, data collection and also on model based diagnosis. Health monitoring in case of helicopters, Larder [27] work describes the benefits of the Health and Usage Monitoring and Flight Data Recording. The Health and Usage Monitoring system on helicopter was installed to improvise the safety as the main rationale and this led to secondary benefits such as making helicopters more acceptable to the public as a means of transport and also helping to keep down the insurance costs. This work also delineates the successful fault detections using health and usage monitoring system. It mentions about the achieved benefits and also about the benefits to be achieved in future.
Considering state of art for structural health monitoring of aircraft. Gerardi [36] work explains about approach for insuring the aircraft structural integrity and also about the future of the health monitoring aircraft. Kressel et al [25] work delineates structural health monitoring system for unmanned aerial vehicles composite structure. The system is based on embedded optical fiber Bragg sensors interrogated in real time during flight. The health monitoring system and health management system are related terms but they differ from one another. The health monitoring system involves in detecting and diagnosis of faults in the system but doesn’t not indulge in mitigation of the system faults.

Integrated Health Management System (IVHM) [45] indulges in detection, diagnosis and mitigation of faults and failures during flight. This IVHM concentrates on hardware which includes aspects such as structural integrity to mechanical system to built-in electronics. IVHM basically diagnose the degrading performance of materials and hardware but software needs to reliable. National Aeronautics and Space Administration has realized that traditional IVHM approaches fail to count for software failures. Divakaran et al [53] work describes about the integrated vehicle health management of a transport aircraft landing gear system. It explains about the landing gear system, its failure modes and detection mechanism and integrated vehicle health management system architecture. Software Health Management bears many similarities to that health management of physical system. Aircraft software is inherently coupled with the physical systems and many faults are basically triggered due to the interactions with the physical phenomena. Schumann et al [23] work expounds that health management software itself is a complex software which monitors certain parameters based upon the sensor signals and other dynamic information. It is very important that health management software has to work correctly and reliably. False alarms creates nuisance and missed alarms can produce serious problems. Health management software needs to undergo validation and verification. Goldberg et al [19] describes the model based approach in which the behavior of the software is monitored against a model of its expected behavior and explains about the process of incorporating this into the ARINC 653 health management architecture.

The reliability of the critical software is evident either by ‘certification’, ‘testing’ or ‘formal verification’. In case of civilian aircraft the Federal Aviation Authority demands high assurance software development process through the standards such as DO-178B [24]. These Standards rely upon software testing. Testing cannot identify the errors in depth and hence it cannot ensure reliability for the ultra-critical real time software [46]. Certification ensures that the good practice is being followed in the software development but it does not guarantee the correctness of the end artifact. Boeing 777 Malaysian Airline software problem went undetected during DO-178B certification [10]. Formal Verification boosts the confidence in software correctness but it is difficult to completely verify the complex implementation nevertheless the formal verification can be applied to the abstraction but there shall be risk that the abstraction might not hold.
All the three evidences for the reliability of the critical software have their own disadvantages which leads this thesis work to explore another way. The other way is runtime motoring of system. The runtime monitoring checks the run of the current state of the system. The comparison of runtime monitoring along with testing and formal methods such as model checking is provided by Martin Leucker et al [29]. NASA [10] proposed the runtime checking for distributed real time systems to enhance the reliability of the safety critical systems. Rushby delineates that runtime monitoring can be considered as a form of runtime certification [49]. Beth [71] provides tutorial for online monitoring, delineates that online monitoring can complement formal techniques to increase application dependability. The monitors discussed are independent and non-independent monitors. Lee Pike et al [33] implements fault tolerance mechanisms using runtime verification techniques for hard real time systems using copilot language and compiler. Temporal logic based task planning and execution monitoring framework for rotor based unmanned aircraft system is developed by Doherty et al [15].

The execution traces are checked against the formal specifications in terms of monitors explained by Falcone et al [17]. The component to typically verify whether execution meets the correctness property is performed by monitor. Sonali Dutta et al [51] proposed the CHIMP tool for assertion based dynamic verification of system C models. Chimp tool generates the monitor automatically from temporal assertions, automatic instrumentation of the model-under-verification.

There is also another evident that this thesis work would like to tap and explore is the runtime monitoring for enhancing situational awareness. The software errors in the aircraft leads to the lack of situation awareness. The lack of situation awareness leads to accidents. Considering a real incident that drove the software failure to loss of situation awareness. On 6/02/1996 Birgenair Flight 301 [57] crashed into atlantic ocean and all passengers were killed. The pitot tube was blocked and this provided faulty readings to pilots. The pilots were confused analyzing airspeed and they did not recognise the stick-shaker stall alert. The failure of pitot tube led to loss of situation awareness of pilots, they did not recognise the stick-shaker stall alert.

The lack of situational awareness leads to accidents in aviation. Endsley [8] delineates about the situational awareness in aviation systems along with different levels of SA. Endsley [30] provides the information requirements for commercial airline pilots.

The Adaptive Health Monitoring Systems (AHMS) for autonomous aircraft by [44] incorporates good situational awareness. This AHMS monitors flight behaviour / flight outcome of the aircraft considering both the environmental conditions as well as the internal behaviour of the system. AHMS monitors the behaviour and indicated during erroneous flight behaviour and this further change in health value point out cause for the error and subsequent compensation indicates to tackle the detected error. Elicitation of safety trigger conditions is proposed by Amina Mekki Mokhtar [11].

This thesis work provides an generic (applicable to all UAV’s) independent run-
time monitor approach that can enhance the situational awareness using the formal specifications. The formal specification using temporal logics such as linear temporal logic. The generic runtime monitor architecture and taxonomy of inputs and health monitoring related terms are discussed.
Chapter 4

Methodology

The software faults if left uncorrected then it leads to failures. These software faults can be avoided by verification of the system. The model checking and testing can be used. The model checking faces the state explosion problem and testing capability for complex system verification is limited. Runtime monitoring overcomes the problems of both testing as well as model checking. This thesis work deals to identify the software errors as soon as possible by runtime monitoring the system. Along with runtime monitoring the situational awareness of the operator shall also be enhanced.

4.1 Runtime Monitoring Methods

Runtime Monitoring approach extracts the important information from the system which is running. This information is used to detect the faults and possibly to react to observed behaviours which violates properties. Runtime monitoring of system can be done through different methods. Initially this chapter describes the different methods of runtime monitoring and later shall discuss the method opted for this thesis work.

4.1.1 Design by Contract

The design by contract is a manual process where the checks are inserted. As this process is manual it needs lot of efforts to insert the assertions. The design by contact is one type of runtime monitoring where pre conditions and post conditions are important and they act as monitors. The pre conditions and post conditions are added at places in the code where we want to monitor particular property in the system. There is no code instrumentation necessary in this technique because the assertions are coupled to the system. The offline monitoring through design by contract is not possible. Due to the assertions inserted is a manual work hence there are possibilities for human error. Usage of this technique in the safety critical system would not be helpful due to possibilities of human error. The figure 4.1 shows the pre conditions and post
4.1.2 Runtime Verification

Runtime Verification is different approach than compared to the design by contract. In case of this technique the formal specifications are verified against the software to be monitored. The properties are represented in the formal notation. The properties formalized later shall be instrumented into the target system automatically. The runtime verification checks the safety violation and properties that are violated. If the property is violated then system behaviour shall be erroneous. In order to correct the behaviour of the system this technique proposes code which consists of the monitoring code along with the code that is triggered when the safety violation occurs as shown in figure 4.2. Compared to the design by contract the runtime verification is applicable to the safety critical system as there is not any manual insertion of assertions and hence avoids the human error. Due to possibility of correcting the bad behaviour this technique find more applicable in safety critical systems. This technique can be applied for offline as well as online monitoring.

4.1.3 Monitor Oriented Programming

This technique goes beyond the runtime verification technique as error handling mechanism is integral part of the system design as shown in the figure 4.3. Monitoring is integrated into the system. Summarising the three runtime monitoring methods discussed above, design by contract method is manual process and hence there are lot of chances of human error. This design by contract method can be applicable for smaller
system and not applicable for safety critical systems. The next method is runtime verification. The runtime verification indulges usage of formal specifications. These formal specifications are nothing but the formal notation of the textual requirement. These formal specifications are verified against system to be monitored. These properties are instrumented with target system. This runtime verification can be used for critical system verification. The monitor oriented programming can handle the errors which cannot be handled by runtime verification method and this monitor is integral part of the system to be verified.

In this thesis work we would like opt runtime verification as we would like to have an independent monitor. This independent monitor shall not be an integral part of system to be monitored. The instrumentation process involved in the runtime verification shall not taken into consideration in our work. The instrumented code along with the system code shall make the certification process bit difficult. Hence the automatic generation of monitor using the formal specifications is considered during this thesis work.

4.2 Architectural Process

Architectural design plays a pivotal role to realize the complex system. The architecture is the starting step for the design of any system in general. The architecture design makes the further process much simpler and more clearer.

The process considered in this work to design the GRM architecture is shown in figure 4.4. This process basically involves three steps and all these three steps helps...
us to know the overall activities that has taken place during the design of the GRM architecture. The three steps are:

- **GRM Architecture Requirements:**
  The GRM requirements elicited shall monitor the UAV by enhancing situational awareness. The requirements were elicited based on the motive to reduce aviation incidents and accidents happening due to loss of situational awareness. The requirements that drive this architecture were elicited through the Template Based Methodology. The GRM requirements are explained in detail in the Chapter.

- **GRM Architecture Design / Structure of GRM Architecture:**
  This involves structuring of the GRM architecture. Along with the structuring of the GRM Architecture it also provides the details about the component/subsystem responsibilities in the GRM architecture.

- **Validation:**
  The validation is just like testing the architecture in simple terms. The GRM architecture after designed fully need to be tested against the requirements elicited. Thinking about how to represent the future possible requirements helps to edit the design of the GRM architecture if in case the requirements does not map with the architecture.
The above process is iterative optimizeone and this was iterated during this thesis work. After validating the requirements it helps to know about the necessary modifications that was necessary to make sure that the requirements are mapped to the architecture. This helped to enhance the design and requirements until requirements mapped to the GRM architecture.

The requirements elicitation is the first process under the design of the generic runtime monitoring architecture. The requirements can be elicited using various methods. Considering the various methods of requirements elicitation in further section.

4.3 Requirement Elicitation Methods

The requirements can be elicited based on various methods. This section provides an analysis of various methods and it also discusses the method which is opted in this thesis work.

4.3.1 Analytical Approach

The analytical approach starting point is by asking the stakeholders about their requirements. These descriptions provided by the stakeholders are documented. The requirement specifications are formed on a sentence-by-sentence basis and asking for all the information that could be missing. The main rationale behind the analytical approach not being applied to this work is that requirements structure are different and also the usage of terms is quite different. The description of the requirement differs from person to person.
4.3.2 Natural Language Method

The natural language method basically provides the better approach than compared to the analytical approach. In natural language method the requirements are written more precisely and these requirements are clearer. The reasons that the requirements are clearer basically due to investigation on the linguistic effects are included and the knowledge gained is being integrated into the statements [1]. The disadvantage of this approach is that the requirements specified by the different stakeholders do not certainly remain to be the same. This approach cannot provide a common and uniform understanding of the system to be developed for all the participants due to this reason the natural language method was not used in our work. The main quest was to find the approach which provides the common and uniform understanding of the system to all the participants for which the system to be developed.

4.3.3 Template Based Method

The template based method avoids the errors in the requirement specification from the start of the specification process. The requirement specification in this approach shall be based on the predefined rules. Each and every requirement in template based method shall have uniform structure. The clear predefined rules for the requirement specification shall guide to a qualitatively high-grade requirement. In this approach syntactic approach of requirement template is provided, as it is just the syntax of the requirement, which is being established, and not the semantics. The template based approach is defined as the blueprint which delineates the syntactic structure of a requirement. The requirements with the incomplete and uncertain information are avoided. Each requirement is assembled in accordance with a plan or a template.

The methodology of the template-based requirement [15] is delineated in following steps

**Step 1: Process Determination:** The functionality of the requirement is referred to a process. The process delineates the operations or activities and shall be defined by the verbs. This verb has the importance for the requirement and this verb can also be termed as process word. The first step in template based methodology is to determine the process.

**Step 2: System Activity:** The template based requirements are always expressed in active voice which always highlights the concerned subject. The system activity is strongly linked with the process word. System activities can be classified into 3 types

- Independent System Activity
- User Interaction
- Interface Requirement

All three different system activities have their own template. Based on the system activity that is required for the system either of the three templates are used for the
CHAPTER 4. METHODOLOGY
Figure 4.7: Fine Attuniation [48]

Figure 4.8: Logical and Temporal Conditions [48]
requirement specification. These three system activities can be basically represented with the block diagram as shown in figure 4.5. The requirement specification relies on the system activity and the requirement can be specified easily using these system activity templates those are not same as the other. The template of each system activity is further explained in detail.

- **Independent System Activity**: The independent system, term itself delineates that system is not depend on the user. In this case the system executes the activity independently.

- **User Interaction**: The requirement in which system involves the interaction with the user to be specified then this template is more relevant. The ability to the user is provided by the system for certain functionality can be easily specified using this template.

- **Interface Requirement**: System executes an activity based on the third party then such a requirement can be specified under this activity. In this case the user does not provide the necessary information but the third party is providing it. The information can be acyclical and can be unpredictable. The requirement of this kind cannot be specified using independent system activity nor using the user interaction template. The following phrase is suitable to specify such a requirement.

**Step 3 : Legal Obligation**: To specify a requirement, system activity is important but that does not complete a core of a requirement. The legal relevance is obligatory part of the requirement specification. The requirement are distinguished between the ‘legally binding’, ‘strongly binding’ and ‘future requirements’ using the modal verbs ‘shall’, ‘should’ and ‘will’ respectively. The pictorial representation of the legal relevance is shown in figure 4.6.

In this work legally binding is used in all the requirements specification. The requirement number, version number and specification date all are part of the documentation process. The main rationale behind it is that these requirements shall sometimes be upgraded further and mentioning the version and date helps to know about the changes being made.

**Step 4 : Fine Tuning**: The requirement is not still complete it just constitutes the core of the requirement but not a complete requirement. The part that is missing in the requirement is the complementary information of the process word as it is also important aspect. The objects or the complements that are missing shall be determined and should be included while specifying the requirement. The pictorial representation of this is shown in figure 4.7.
Step 5: Logical and Temporal Conditions: Typically desired functionality in the requirement is not constant but executed during certain conditions and these conditions might be either logical or temporal. To signify the difference between the temporal and logical conditions the temporal conjunction ‘when’ is used for temporal conditions and the conditional conjunction ‘if’ for the logical condition as shown in the figure [53].

Summarising all the three methods for requirement elicitation. The analytical approach of requirements elicitation has major disadvantage of non uniform structure of representation. There is no standard rule for the elicitation. The next method discussed above is the natural language method. The natural language method does not provide the uniform understanding of the system requirements. The template based methodology is the semi formalized method of representing requirements. The template based methodology as discussed above follows certain rules to write the requirements. This provides the uniform structure for all the requirements. Template based method was used in this thesis work.

4.4 Requirement Formalization Methods

After the requirements elicitation the next process is to formalize the requirements. Formalisation of the requirements can be done using many methods. The different methods that can be applied to formalize the requirements shall be discussed. The method that is used in this this work and the reason behind its usage is described.

The propositional logic is based on the propositions that express either being true or false. The propositional logic can not express assertions of all types [59]. The predicate logic came up as a powerful logic than compared to the propositional logic. Predicate logic is the generalisation of the propositional variable. The vital part of both these two logics namely predicate logic and propositional logic represent the specifications whose truth value is constant in time. So the specifications whose truth value changes with respect time need to be addressed. This needs the powerful logic namely temporal logic. The temporal logic helps to represent the requirements in the mathematical way. Using the temporal logic the system properties can be expressed formally. The temporal logics delineates the temporal relations between the events occurring over time. The temporal logic notation is often simpler and clearer. Model checking is one of the methods to verify the temporal properties of the system. Hence temporal logic is used in this thesis work.

4.5 Summary

The runtime verification extracts the information from the system that is running. Summarising the runtime monitoring methods, design by contract method can be
used for smaller systems. The manual insertion of the assertions are error prone hence this method does not find its application in safety critical system. Runtime verification use the formal specifications to instrument into target system automatically. The event monitor extracts all the events from the target system and it provides to the monitor. The other runtime monitoring method is monitor oriented programming, this technique is capable of handling errors as the error handling mechanism is integrated into the system. The property monitor is integrated into the target system in monitor oriented programming method. The runtime verification method without the aspected oriented instrumentation is considered in this thesis work. The architectural process plays a vital role in the development of a system. The three main processes are architecture requirements, structure/design of the architecture and validation. These processes are iterative. The requirement can be elicited using various methods. The analytical approach and the natural language method does not provide a uniform structure for the requirement elicitation. The template based method has a uniform structure and requirements elicited shall be semi-formalized ones. It provides a common and uniform understanding of the system to be developed for all the participants. This thesis work choose template based method for requirement elicitation. Further the requirement formalisation can be done using various methods such as propositional logic, predicate logic, temporal logic. The predicate logic and propositional logic represents the specifications whose truth value is constant in time. So the specifications whose truth value changes with respect time need to be addressed. Temporal logic can address specification whose truth value changes with respect to time. Temporal logic represents the requirements in mathematical way. Temporal logic is chosen in this thesis work to formalize the requirements.
Chapter 5

Contributions

The contribution towards the master thesis is briefly summarised in this chapter. The detailed overview is discussed in the chapters 6, 7, 8 and 9.

5.1 Generic Runtime Monitoring Architecture Design

Generic runtime monitor is an independent runtime monitor enhancing situational awareness of UAV. This generic runtime monitoring architecture shall be applicable to all the UAV’s. The generic runtime monitor diagnosis the UAV and detects the error as soon as possible. The runtime monitoring is cost effective as the system verifies only the run of the system. This generic runtime monitor shall be able to monitor the UAV and provide regular status update to the operator. The designed GRM shall not only monitor health of the UAV but it shall also enhance the situation awareness. This architecture provides a detailed overview of the monitoring unit. Monitor data can be processed both on-board as well as off-board both the aspects are discussed in chapter 6.

5.2 Requirements Enhancing Situational Awareness for UAV Operator

The thesis objective is to enhance the situational awareness using runtime monitoring. The requirements elicited enhances the SA. There are three levels of SA namely perception, comprehension and projection. The requirements can be elicited using various methods but template based method is used in this thesis work. The template based method provides a standard structure of the requirements elicitation. The requirement specification shall be based on predefined set of rules. The requirements with incomplete or uncertain information are avoided. Hence the template based method
was used in this thesis. The requirements for all the three levels of situation awareness were elicited. These requirements were categorized into many categories which is discussed in detail in the chapter [7]

5.3 Independent Runtime Monitoring Generation using State Machine

The GRM requirements elicited were formalized using the temporal logics specifically using linear temporal logics. These requirement shall be translated into monitors automatically using the tools. The process after the elicitation of the requirements that runtime monitor UAV alongside enhancing situation awareness is formalisation of the requirements. The requirements can be formalized using various methods but in this thesis work temporal logics is used. The requirements are formalized using the linear temporal logics. These formalized requirements can be converted into state automaton directly. The tools used in this thesis for conversion of LTL to state automaton are Spot and BRICS automaton. The Spot tool converts the LTL formula into a transition based generalised büchi automaton. An online translator exists for Spot tool. LTL formula can also be transformed into a state-based de-generalised büchi automaton using Spot tool. These automatons are further discussed in detail in the chapter [8] In case of BRICS automaton tool LTL formulas cannot be converted into state automatons directly. LTL logic must be coded into JAVA. The automaton obtained after run shall be used to obtain the graphical state automaton using the Graphviz tool.

5.4 Taxonomy of Inputs and Health Monitor

Taxonomy of health monitor related terms was created along with the taxonomy of GRM inputs as the part of thesis work. The health monitoring related terms are quite a lot and taxonomy plays vital role to analyse and comprehend them. The GRM input taxonomy helps to categorized the inputs. These categorized inputs shall further be mapped with the architecture. The taxonomy of inputs plays vital role in the design of the GRM architecture. The health monitoring related terms taxonomy provides distinction between the meaning of the words along with their actions. The taxonomy of both the inputs and monitor is discussed in detail in the chapter [9]

5.5 Summary

The Generic runtime monitoring architecture is an independent monitor which is applicable to all the UAV’s. This is a layered architecture. The architecture is designed considering the health monitoring of UAV and enhancement of situation awareness at runtime. The requirements elicited shall also be capable to enhancing the situation awareness alongside runtime monitoring of UAV internally. The situation awareness
categorized into three levels. The requirements were elicited for all the three levels of SA. These requirements further formalized. The formalized requirements using the temporal logic shall be converted into state automaton automatically. The tools such as Spot and BRICS automaton were considered to translate LTL into state automaton.
Chapter 6

Generic Runtime Monitor Architecture

GRM architecture refers to the high level structure of an independent runtime monitoring system, it delineates the need for the enhancement of the situational awareness of the UAV operator. Generic Runtime Monitoring architecture shall be applicable to all the UAV’s and for all monitoring tasks.

Architecture provides us the overall GRM design. The overall design includes

- Subsystems
- Interconnections
- Collaborations

GRM architecture was designed to improve the situational awareness of the UAV. Architecture plays a vital role to understand the complex system, before we actually design the subsystems we need to know the interconnections of the subsystems and collaborations between them and this is where architecture makes it clear. The architectural design concept is crucial and important in many disciplines. Architectural design helps us to organise the development of the GRM before it is being built and it becomes easy to go further in the developmental process once the architectural approach is considered in the beginning. The existing components are identified and they are reused. The architecture helps us to identify the overall dependencies among different subsystems.

Design of the architecture marks the initial start for the development of the system. The architecture depicts the generic runtime monitor representation which justifies the reasoning about the behaviour and also the relationship between the different layers undergone during the generation of the monitor. The architecture is the face of the system to be developed and contains all the descriptions needed to develop the system.
The architecture framework for the black box verification does not need the details of the system implementation. The system implementation based on certain design parameters is not that important during offline/log/black box verification but the interfaces defined between the subsystems are important for verification. Verification is not dependent on any specific implementation architecture. The abstract level errors within the interfaces can be localized. The black box verification is more helpful for the system maintenance after the flight and hence reflecting to errors during on-board is not possible. The detection of the errors in case of black box verification is not so faster than compared to white box verification [28].

The architecture framework for the white box verification needs in depth knowledge about the system design and implementation. The errors can be detected sooner in white box verification than compared to black box verification. The identified errors can not only be localized but they can also be runtime reflected and try to avoid the failure. The runtime verification during on-board can identify implementation errors and design errors of a specific subsystem or individual functions.

GRM architecture is the extensible one and this helps to use the same monitoring architecture for further larger system design.

### 6.1 Architectural Principles

Architectural principles taken into account are

- Abstraction
- Encapsulation
- Structuring

The proper abstractions at the architectural levels were identified and the internal information or internal detailing is defined as encapsulation. In simple terms abstraction and encapsulation can be delineated as separating of what and how the GRM architecture works respectively. Abstraction of the system makes it simpler for all the others to follow and understand the system. By doing this the details of the system are not concentrated by which the focus basically remains on the salient architectural issues [35]. Separating the process what module does and how the module is done. Single point of references for subsystems is really important. The dependencies can be easily managed through single point of reference. If some changes are going through the subsystems then we can easily identify them through the single point of references. Structuring or Modularity is another important thing which comes after the identification of abstraction. The identified abstraction is structured in modularity.
After getting to know about the principles of the architecture the next thing curious to be known is when to create an architecture and the answer to it would be that architectures are created at the early stages of development of the system. Architecture design is the first and foremost thing to be done. GRM architecture is the layered pattern. In the layered architecture the top most layer has the higher abstraction than compare to the low layer. The top layer access the lower layer.

6.2 Graphical Illustration of GRM

The architecture graphically illustrated in figure 6.1 the different nodes or rectangular boxes represents the subsystems which can also be called as components. The arcs in-between the subsystems or components are called as connectors. These connectors represents the communication between the subsystems.

6.3 Architectural Layer Detailing

The best means for describing the architecture is through architectural decomposition. The above Figure delineates about the architectural decomposition of the Guidance, Navigation and Control Systems. The individual subsystems are further decomposed into further detailed structure within. The architecture is further detailed into 3 layers. The top most layer Guidance, Navigation and Control layer is further detailed into two more layers. To further getting into details about the individual subsystem would help us to know about particular system in detail. The more important subsystem in our work is the navigation system as our requirements that enhance situational awareness belong to the navigation system.

The first thing that needs to be taken into account while designing an architecture would be to think about the problem to be overcome. The expectation of the system that need to be designed need to be taken into consideration and the reason behind the system and the ways it benefits. At the end of the day the designed architecture must be able to meet all the criteria.

The flow of the work shall be presented here:

• At first the entire system will be at focus and when getting into deeper analysis of which subsystem are we trying to concentrate on, then it would be navigation subsystem as the situational awareness is the part of the navigation system but that doesn’t mean that there weren’t requirements other than situational awareness ones, there are also system requirements which were considered to runtime monitor.
Figure 6.1: GRM Architecture
• The second process was to extract the requirements from the navigation subsystem, the detailed requirements were taken into account along with interactions between the subsystems such as guidance system, control system, ground control station and with the environment.

• Then next up was to consider about which pattern does GRM architecture fall into.

The architecture design would not be done at first shot it would need an iteration to finalise it making sure all the requirements are taken into account.

6.4 Generic Runtime Monitoring Architecture Documentation and Tools

The architecture work documenting is as important as the other things the sketches from the initial stages to the final stage would teach a lot and it is necessary to document it properly. Making notes was quite helpful during the overall work so that it helps to follow up and make changes whenever it was necessary. The things which were not helpful for our work were justified with reason and making note of it was helpful to justify the alternative one was good.

The tool to design would be next up to discuss and it depends upon the expectations. The main important part would be to make sure that tool used for the architecture need to be understandable and followable by all. In other words when the architecture design shown to others it need to be understood. There are many tools solely for the design of architecture. The commercial tools shall be expensive but they shall always have many more features. GRM architecture design was created using the Microsoft Office Visio.

6.5 System Detailing

Considering the definitions of important terms such as navigation system and guidance system before getting into details. These system definitions are taken from ALFURS [42].

**Navigation System** In the broad sense, navigation is the process of monitoring and controlling the movement of a craft or vehicle from one place to another. For RUAS, navigation can be defined as the process of data acquisition, data analysis, and extraction and inference of information about the vehicle’s states and its surrounding environment with the objective of accomplishing assigned missions successfully and safely. This information can be metric, such as distances, topological, such as landmarks, or any other attributes that are useful for mission achievement. The main autonomy-enabling functions of a navigation system, from lower to higher level, are as
follows:

- Sensing: A sensing system involves one or a group of devices (sensors) that respond to a specific physical phenomenon or stimulus and generate signals that reflect some features of or information about an object or a physical phenomenon. Sensors such as gyroscopes, accelerometers, magnetometers, static and dynamic pressure sensors, cameras, and LIDARs are commonly used onboard UAS to provide raw measurements for state estimation and perception algorithms.

- State Estimation: This concerns mainly the processing of raw sensor measurements to estimate variables that are related to the vehicle’s state, particularly those related to its pose and motion, such as attitude, position, and velocity. These estimates can be absolute or relative. Localization is a particular case of state estimation that is limited to position estimation relative to some map or other locations.

- Perception: RUAS perception is the ability to use inputs from sensors to build an internal model of the environment within which the vehicle is operating, and to assign entities, events, and situations perceived in the environment to classes. The classification (or recognition) process involves comparing what is observed with the RUAS’s a priori knowledge (Huang, 2008). Perception can be further divided into various functions on different levels such as mapping, obstacle and target detection, and object recognition.

- Situation Awareness (SA): The perception of elements in the environment within a desirable volume of time and space, the comprehension of their meaning, and the projection of their status in the near future.

**Guidance System** A guidance system can be defined as the “driver” of a RUAS that exercises planning and decision-making functions to achieve assigned missions or goals. The role of a guidance system for RUAS is to replace the cognitive processes of a human pilot and operator. It takes inputs from the navigation system and uses targeting information (mission goals) to make appropriate decisions at its high level and to generate reference trajectories and commands for the AFCS at its low level. GS decisions can also spark requests to the navigation system for new information. A guidance system comprises various autonomy-enabling functions including trajectory generation, path planning, mission planning, and reasoning and high-level decision making.

- Trajectory Generation: A trajectory generator has the role of computing different motion functions (reference position, reference heading, etc.) that are physically possible, satisfy RUAS dynamics and constraints, and can be directly used as
reference trajectories for the flight controller. Reference trajectories can be pre-programmed, uploaded, or generated in realtime onboard the RUAS (dynamic trajectory generation) according to the outputs of higher-level guidance modules.

- Path planning: The process of using accumulated navigation data and a priori information to allow the RUAS to find the best and safest way to reach a goal position/configuration or to accomplish a specific task. Dynamic path planning refers to onboard, real-time path planning.

- Mission Planning: The process of generating tactical goals, a route (general or specific), a commanding structure, coordination, and timing for a RUAS or a team of unmanned systems (Huang, 2008). The mission plans can be generated either in advance or in real time. They can be generated by operators or by onboard software systems in either centralized or distributed ways. The term dynamic mission planning can also be used to refer to onboard, real-time mission planning.

Guidance, Navigation and Control System
Guidance system of UAV provides the control over the movement of the UAV. Guidance system calculates the changes of position, velocity, attitude of UAV in order to follow the reference trajectory. The navigation system calculated the current position of the UAV with the help of various sensors. The navigation system provides input to the guidance system. The environmental conditions such as wind, temperature and so on all are fed to guidance system. The control system consists of the actuators and these help to manipulate the UAV flight path and its orientation without direct human control. Navigation system includes the Sensing, State Estimation and Situational Awareness under it. The situational awareness further it is divided into three levels namely perception, comprehension and projection.

Autopilot System
The hands-on control of UAV by operator during the surveillance task would be too boring in order to avoid that autopilot system will be really helpful which controls the trajectory of UAV without necessity of constant hands-on control by human operator. This does not mean that autopilot could completely replace UAV operator but it helps the operator to concentrate on other aspects.

Payload Management
The payload is really important in order to accomplish mission, thence management of the payload is necessary. It might be just camera or any other payload but it needs to fit into UAV specifications. The size and weight of the payload need to be taken into account.
Ground Control Station

The Ground control station where the operator controls UAV and the telemetry data from UAV is received at debug computer or 3rd computer and the operator communicates with UAV from ground control station.

The GRM architecture shall make the UAV more safe by detecting the errors as soon as possible and in the future shall be extendable such that the possible mitigations of the errors is possible, at the same time the cost of integration shall be low. The GRM architecture shall provide enhanced situational awareness to the UAV operator who controls the UAV from GCS by runtime monitoring GNC system. The data shall be processed on-board. The interfaces considered here is the direct interfaces between the systems.

6.6 Monitor Detailing

On-Board Processing

The monitor data can be processed either onboard or off board and both have their own advantages and disadvantages. The GRM architecture onboard processing is taken into account. The figure [6.2] shows the data processing onboard the UAV. The data is collected from payload management system, navigation system, guidance system, autopilot system and flight control systems. Each of these systems have further subsystems and the data from all these systems and subsystems is collected through direct interface. Once the data is acquired from all the system and subsystems. This data is sent to pre-processing stage. In pre processing stage the noise which shall be introduced while collecting the data from various systems and subsystems shall be eliminated. Further the data shall be sent to relevant data extraction stage. In the relevant data extraction stage only the relevant data and by doing this a lot of data shall not be taken into account. Once only the relevant data is extracted for monitoring rest of the data shall not be considered for further stages. This reduces the huge data which wasn’t necessary to be part of monitoring unit. The relevant data is sent to the monitoring unit. This monitoring unit shall monitor the key properties and shall forecast the anomaly. The monitoring unit shall evaluate the health state of the particular state where the anomaly is detected. The fault needs to be localized so that further actions can be taken. The fault localization identifies form which system the fault has occurred. Fault needs to be localized accurately else it leads to difficulties in identifying which system is misbehaving. This localized fault needs to be communicated to the debug computer or third computer at the ground control station.

Off-Board Processing

In case of off board processing as shown in the figure [6.3] the data form all the systems and their respective subsystems is collected and as well as sent to the ground control station for further processing. The entire data without removing noise shall be sent to ground control station. In the ground control station all the data is collected and further pre processing is done to remove noise and in case of data transfer from the
Figure 6.2: Onboard Processing
UAV to ground control station much of the data might be lost. After pre processing, relevant data is extracted by reducing the whole irrelevant data and then sent to monitoring for anomaly detection. The fault shall be identified and reported to debug computer.

6.7 Summary

The architecture plays the important role to understand the complex systems. The architecture helps to organise the overall development of GRM. Generic runtime monitoring architecture shall be applicable to all the UAV’s and for all the monitoring tasks. The architectural principles such as abstraction and encapsulation can be delineated as separating of ‘what’ and ‘how’ the architecture. The GRM architecture is the extensible one and this helps to use the same monitoring architecture for further larger system design. The design of the architecture needs iteration to make sure all the requirements are taken into account. Guidance, Navigation and Control systems
are defined and explained in context of generic runtime monitoring. The monitor can be processed onboard as well as offboard.
Chapter 7

Generic Runtime Monitor Requirements

Generic Runtime Monitoring requirements shall be applicable for all the UAV’s. The requirements that are capable of runtime monitor UAV and enhancing the situation awareness are elicited. The requirements are extracts from various sources such as ARTIS requirements. National Transportation Safety Board, Boeing Aero magazine and so on. The generic runtime monitor requirements are classified into various categories such as inputs, transitions, ALFURS autonomy level and so on. These various categories are explained further.

7.1 Generic Runtime Monitor Requirements Elicitation

The requirements in the simple terms can be called as the textual statements which describe the inclusion of things to solve the problem. This section brings up the sources considered for the elicitation of GRM requirements. The process after getting to know about the problem that we would like to tackle is the elicitation of the requirements. In this work lack of situational awareness due to absence of pilot onboard in UAV is tackled. This problem of lack of situational awareness has to be translated into the detailed requirements which could overcome the problem of situational awareness and also at the same time runtime monitor UAV. This GRM monitor shall be generically (applicable to all UAV’s) able to monitor specific UAV requirements to save the aircraft from the critical situation. The level of details in the requirements is directly proportional to the complexity of the system. The situational awareness has 3 levels namely perception, comprehension and projection. The main sources of the requirements are explained in the subsequent sub-section 7.1.1.
7.1.1 Generic Runtime Monitor Requirement Sources

Good set of requirements is really important to have good architecture. The figure 7.1 shows the sources of the requirements elicitation. The requirements were elicited from different sources considering the wide overview of the problems being occurring due to the lack of situational awareness. According to the Australian Transport Safety Board (ATSB) [21] statistics as mentioned in the section 1.1 show larger percentage of accidents do happen due to lack of SA. So during the elicitation of GRM requirements such real incidents and accidents that happened due to lack of SA were considered along with the non-health status requirements. Many incidents such as United Airlines ran out of fuel and US Airways collision with birds as mentioned in the section 1.1 are taken into account to come up the requirements such that these type of events alert operator much prior.

![Figure 7.1: GRM Requirement Sources](image)

7.1.2 Typical GRM Requirements along with sources

Considering some of the GRM requirements from the thesis work:

Requirement: Req 5, SA Level 1: “Icing Accretion shall not be greater than maximum accretion”.

The ice accretion on the non protected parts of the aircraft is really important and there are incidents were ice accretion has led to the disaster. This GRM monitors the accretion of the ice on the aircraft parts and when ice accretion is reached maximum to a point such that further accretion shall lead to failure at that point it shall alert the operator. This requirement shall also be applicable to manned aircraft.

Source: On June 1, 2009 an Airbus A330-200, Air France, crashed into sea and all the 228 occupants on board were no more. This incident took place due to loss of aircraft
control due to inappropriate responses from the flight crew due to the wrong airspeed indication and this was mainly due to failure of the pitot heads due to the icing. To avoid such incidents runtime monitoring of ice accretion on the non protected surfaces of the aircraft is necessary. Hence this requirement was part of GRM requirements [3].

Requirement: Req 7, SA Level 1: “Bird Strike Radar indications shall not be greater than maximum allowed probability of bird strike”.

The above requirement states that GRM shall monitor the maximum probability of the bird strike radar and if the probability of the bird strike radar is say X which is greater than maximum allowed probability then GRM shall raise an alarm. This GRM requirement is applicable for UAV and as well as for the manned aircraft. Bird ingestion in the engine leads to engine failure and many such incidents have occurred in manned aircraft.

Source: Considering the recent incidents of bird strike, On 6th June 2010, Boeing B734, Amsterdam Netherlands, Passenger Flight, this flight encountered flock of geese and due to the bird ingestion on the left engine MAYDAY was declared. 162 passengers were on board and none were injured but extensive damage was found to be caused in the internal and external to the left engine [3]. This indicates that much more of efforts need to be put to reduce such kind of bird strikes and hence this requirement was chosen as GRM requirement were this probability of collision with UAV is runtime monitored onboard.

Requirement: Req 10, SA level 1: “Fuel quantity shall not be lower than minimum fuel ”.

This requirement was considered mainly due to the reason that if the pilot on board if he lacks situational awareness by any unconditional events then GRM shall alert him about the fuel quantity if it is going low. The reasons for fuel lowering might be due to environmental conditions which might lead to take an alternate path. The other reason might be due to lack of pilots knowledge and his concentration on other things.

Source: On 28/12/1978 at Portland, Oregon United Airlines pilot was busy analyzing the landing gear problem and didn’t concentrate on any other factors this point of time fuel ran out. This incident happened due to lack of situational awareness as crew concentrated only on the issue to solve the landing gear problem but unfortunately they didn’t concentrate on the fuel starvation[17]. GRM plays vital role to avoid such incidents further.

7.2 Analyzing GRM Requirements

The requirement analysis is done after the elicitation of the GRM requirements. The analysis process involves analyzing the requirements by determining whether requirements are clear, consistent, complete, unambiguous and more precise. GRM require-
ments were analysed and iterated during the thesis work. The reason to create the architecture is to make sure that all the requirements are met and are successfully implemented. The list of important analyzing aspects performed on GRM requirements are:

- The incomplete specification shall be avoided by reviewing.
- The conflicting requirements shall be avoided which lead to misunderstanding
- The hidden requirements shall be identified and need to be specified

**GRM Requirement Before Analysis:**

Req 2.1, SA Level 2: “GRM shall monitor environment temperature and runway length during takeoff and if the current temperature is greater than threshold value and runway length is smaller than threshold value then GRM shall raise an alarm”.

Req 11.1, SA Level 2: “GRM shall monitor the UAV position and ADS-B and if the current position of the UAV is not updated after certain threshold time and if the ADS-B is in the standby state then GRM shall project the other aircraft position and if it is less than threshold value then GRM shall raise an alarm”.

The above requirements tends to be much elaborative with unclear sentence formation and this can mislead. Considering the same requirements after the analysis.

**GRM Requirement After Analysis:**

Req 2.1, SA Level 2: “Environment temperature shall not be greater than maximum temperature and runway length shall not be smaller than minimum runway length ”.

Req 11.1, SA Level 3: “Current position of the UAV shall not take more than maximum time to update and ADS-B shall not be in standby state and projected other aircraft position shall not be less than reference value ”.

These requirements after iteration and analysis is much more precise and easy to understand. The precision and clarity is important in the requirements else the requirements shall not be followed by others. The requirements shall be understandable to all the technical as well as non technical team.

### 7.3 Generic Runtime Monitor Requirement Specification

The requirement methodology as discussed in [4.3] are different from one other and each one of it has its own advantage and disadvantage based on the application need. The GRM requirements are elicited using the Template Based Method. The GRM requirements after elicitation shall be formalized further for monitor generation. Hence the
methodology should support the representation. The template methodology itself is a semi formalized methodology. The representation of the formalized GRM requirements shall be a part of template based methodology as an additional block. Microsoft Excel was used to elicit the textual GRM requirements. The excel was much opted due to easy representable categories as the GRM requirements are further categorized into 8 categories discussed in section 7.3. Considering some of the examples of GRM requirements elicited based on template based methodology are:

Req 2: “GRM shall monitor the environment temperature and if the temperature is greater than threshold value then GRM shall raise an alarm”.

Req 4: “GRM shall monitor the absolute altitude and if the absolute altitude is lesser than threshold value then GRM shall raise an alarm”.

7.3.1 Generic Runtime Monitor Requirement Formalization

The requirements even after analyzing there are chance that some requirements might not be so precise. Informal requirements lack the completeness and they create huge misunderstanding. This can be avoided by formalising the requirements. The formal specifications such as temporal logics (LTL) were used to formalize the GRM requirements. These formalized requirements further shall be automatically translated to monitoring code. Considering some of the examples.

- Req 1: “Indicated altitude shall not be greater than maximum indicated altitude”. Formalized requirement using LTL: G(Indalt > Maxalt)->X (alarm)
- Req 21: “Indicated airspeed shall not be greater than maximum airspeed”. Formalized Requirement using LTL: G(Indair > Maxair)->X (alarm)
- Req 9: “Turn rate shall not be greater than maximum angle”. Formalized Requirement using LTL: G(Turratcategorizee > Maxang)->X (alarm)
- Req 37: “Time taken to generate mission plan shall not be greater than maximum time value”. Formalized Requirement using LTL: G(Mistime > Maxim)->X (alarm)
- Req 38: “Time taken to generate path plan shall not be greater than maximum time value”. Formalized Requirement using LTL: G(Pathtime > Maxtime)->X (alarm)

Raising an Alarm if the indicated altitude is localized in Req 1 is greater than the maximum altitude then this warning signal or alarm has to alert the operator so that his attention towards the possible failure is highlighted.
Typical Generic Runtime Monitor Requirements
Considering some the GRM requirements for all the three levels of situational awareness namely perception, comprehension and projection. The Table 7.1 consists of three columns namely requirement id, situational awareness level and GRM requirement. The GRM requirements of all the three SA levels are mentioned.

7.4 Generic Runtime Monitor Requirement Aspects
The three aspects are considered while elicitation of GRM requirements and they are namely:

- Safety
- Performance
- Mission

The priority of the aspects is considered as shown in the Table 7.4. Considering some of the GRM examples for safety, performance and mission shown in Table 7.5.

7.5 Generic Runtime Monitor Requirement Categories
The GRM requirement categories are as follows:

- GRM Inputs
- Events/transitions/Conditions/Transition Guards
- Situational Awareness Types
- Aspect
- GNC Category
- UAV/Manned Aircraft Requirement
- ALFURS Autonomy Levels
- Off-Nominal State Behaviour Formalisation
- Situational Awareness Levels
- Accidents and Incidents elicited

All these above categories were taken into consideration while eliciting each and every requirement. All the GRM requirements are categorized into these levels. This categorisation provides a better representation and classifying the requirements based on any of these above categorize shall become very easy. All these above categories simplified the process of GRM architecture design.
### Table 7.1: GRM Requirements for all 3 SA Levels

<table>
<thead>
<tr>
<th>Req Id</th>
<th>SA Level</th>
<th>GRM Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>Environment temperature shall not be greater than maximum temperature</td>
</tr>
<tr>
<td>2.1</td>
<td>2</td>
<td>Environment temperature shall not be greater than maximum temperature and runway length shall not be smaller than minimum runway length</td>
</tr>
<tr>
<td>2.2</td>
<td>3</td>
<td>Environment temperature shall not be greater than maximum temperature and runway length shall not be smaller than minimum runway length and projected runway length shall not be lesser than reference value</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>Icing accretion shall not be greater than maximum accretion</td>
</tr>
<tr>
<td>5.1</td>
<td>2</td>
<td>Icing accretion shall not be greater than maximum accretion and icing accretion on the non-protected surfaces shall not be greater than minimum allowable accretion</td>
</tr>
<tr>
<td>5.2</td>
<td>3</td>
<td>Icing accretion shall not be greater than maximum accretion and icing accretion on the non-protected surfaces shall not be greater than minimum allowable accretion and projected non-icing altitude value shall not be greater than reference value</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>Weather radar indications shall not be greater than maximum bad weather indications</td>
</tr>
<tr>
<td>6.1</td>
<td>2</td>
<td>Weather radar indications shall not be greater than maximum bad weather indications and wind shear shall not be greater than maximum tolerable wind shear</td>
</tr>
<tr>
<td>6.2</td>
<td>3</td>
<td>Weather radar indications shall not be greater than maximum bad weather indications and wind shear shall not be greater than maximum tolerable wind shear and projected unaffected altitude shall not be greater than reference value</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>Fuel quantity shall not be lower than minimum fuel</td>
</tr>
<tr>
<td>10.1</td>
<td>2</td>
<td>Aircraft fuel shall not be lower than minimum fuel quantity and remaining flight duration shall not be lesser than mission time</td>
</tr>
<tr>
<td>10.2</td>
<td>3</td>
<td>Aircraft fuel shall not be lower than minimum fuel quantity and remaining flight duration shall not be lesser than mission time and projected flight time shall not be lesser than reference value</td>
</tr>
<tr>
<td>Req Id</td>
<td>SA Level</td>
<td>GRM Requirement</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>-----------------</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>Current position of aircraft shall not take more than maximum time to update</td>
</tr>
<tr>
<td>11.1</td>
<td>2</td>
<td>Current position of aircraft shall not take more than maximum time to update and ADS-B shall not be in standby state</td>
</tr>
<tr>
<td>11.2</td>
<td>3</td>
<td>Current position of the UAV shall not take more than maximum time to update and ADS-B shall not be in standby state and projected other aircraft position shall not be less than reference value</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Tail wind shall not be greater than maximum tail wind</td>
</tr>
<tr>
<td>3.1</td>
<td>2</td>
<td>Tail wind shall not be greater than maximum tail wind and runway length shall not be smaller than minimum runway length</td>
</tr>
<tr>
<td>3.2</td>
<td>3</td>
<td>Tail wind shall not be greater than maximum tail wind and runway length shall not be smaller than minimum runway length and projected and required runway shall not be smaller than reference value</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>Absolute altitude (AGL) shall not be lesser than maximum absolute altitude</td>
</tr>
<tr>
<td>4.1</td>
<td>2</td>
<td>Absolute altitude (AGL) shall not be lesser than minimum absolute altitude and rising mountain terrain shall not be greater than maximum flyable altitude (Aircraft Limits)</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>Bird strike radar indications probability distribution shall not be greater than maximum probability distribution</td>
</tr>
<tr>
<td>7.1</td>
<td>2</td>
<td>Bird strike radar indications probability distribution shall not be greater than maximum probability distribution and structural damage to aircraft birdstrike shall not be greater than maximum structural damage</td>
</tr>
<tr>
<td>7.2</td>
<td>3</td>
<td>Bird strike radar indications probability distribution shall not be greater than maximum probability distribution and structural damage to aircraft birdstrike shall not be greater than maximum structural damage and remaining runway length to takeoff after birdstrike shall not be smaller than reference value</td>
</tr>
</tbody>
</table>

Table 7.2: GRM Requirements for all 3 SA Levels
<table>
<thead>
<tr>
<th>Req Id</th>
<th>SA Level</th>
<th>GRM Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
<td>Airspeed shall not be lower than minimum airspeed</td>
</tr>
<tr>
<td>8.1</td>
<td>2</td>
<td>Airspeed shall not be lower than minimum airspeed and airspeed sensor probability of malfunction shall not be 1</td>
</tr>
<tr>
<td>8.2</td>
<td>3</td>
<td>Airspeed shall not be lower than minimum airspeed and airspeed sensor probability of malfunction shall not be 1 and projected time to take over from autopilot mode shall not be greater than minimum time to take over</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>Turn rate shall not be greater than maximum angle</td>
</tr>
<tr>
<td>9.1</td>
<td>2</td>
<td>Turn rate shall not be greater than maximum angle and indicated airspeed shall not be greater than maximum airspeed</td>
</tr>
<tr>
<td>9.3</td>
<td>3</td>
<td>Turn rate shall not be greater than maximum angle and indicated airspeed shall not be greater than maximum airspeed and projected distance to make a particular turn shall not be greater than reference value</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>Current pitch attitude shall not be greater maximum angle</td>
</tr>
<tr>
<td>12.1</td>
<td>2</td>
<td>Current pitch attitude shall not be greater maximum angle and airspeed shall not be lesser than minimum airspeed</td>
</tr>
<tr>
<td>12.2</td>
<td>3</td>
<td>Current pitch attitude shall not be greater maximum angle and airspeed shall not be lesser than minimum airspeed and projected the rate of climb shall not be lower than reference value</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>UAV flight path direction shall not be greater than maximum degrees</td>
</tr>
<tr>
<td>14.1</td>
<td>2</td>
<td>UAV flight path direction deviation shall not be greater than maximum degrees of planned flight path direction</td>
</tr>
<tr>
<td>14.2</td>
<td>3</td>
<td>UAV flight path direction shall not be greater than maximum degrees and wind shall not be greater than maximum wind value and projected the wind correction angle shall not be greater than reference value</td>
</tr>
</tbody>
</table>

Table 7.3: GRM Requirements for all 3 SA Levels
### 7.6 Summary

This chapter discussed about GRM requirements. The requirements elicitation can be done on various methods. The template based method is the semi formalized methodology. The GRM requirements are elicited using the template based method. The template based method provides the structured representation and they also need to follow certain predefined rules. The requirements of generic runtime monitor are elicited from various sources as shown in the figure 7.1. The requirements elicited are documented. Some of the generic requirements are presented in the Table 7.1, 7.2, 7.3. The requirements need to be analyzed based on goals. The generic runtime monitor requirements further need to be formalized. The temporal logics are used to formalize the requirements. In the subsection 7.3.1 the examples of the formalized requirements are mentioned. The three aspects of requirements are considered namely safety, mission and performance. The requirements were elicited considering these three aspects. The Table 7.5 mentions the requirements that consider all the aspects.

**Table 7.4: GRM Requirements Aspect Priority**

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety</td>
<td>1</td>
</tr>
<tr>
<td>Mission</td>
<td>2</td>
</tr>
<tr>
<td>Performance</td>
<td>3</td>
</tr>
</tbody>
</table>

---

CHAPTER 7. GENERIC RUNTIME MONITOR REQUIREMENTS 68
<table>
<thead>
<tr>
<th>Req Id</th>
<th>Aspect</th>
<th>GRM Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Safety</td>
<td>Indicated airspeed shall not be greater than maximum airspeed limits of the UAV</td>
</tr>
<tr>
<td>20</td>
<td>Safety</td>
<td>Distance to Non Fly Zone shall not be lesser than minimum distance</td>
</tr>
<tr>
<td>24</td>
<td>Mission</td>
<td>Absolute altitude shall not be greater than maximum absolute altitude of the mission</td>
</tr>
<tr>
<td>25</td>
<td>Mission</td>
<td>Airspeed shall not be greater than maximum airspeed value of the mission</td>
</tr>
<tr>
<td>27</td>
<td>Performance</td>
<td>Memory usage shall not be greater than maximum memory usage value</td>
</tr>
<tr>
<td>28</td>
<td>Performance</td>
<td>Computation time of tasks shall not be greater than maximum computation time</td>
</tr>
</tbody>
</table>

Table 7.5: GRM Requirements for all 3 Aspects
Chapter 8

Generic Runtime Monitor Prototyping

The generic runtime monitoring architecture, generic runtime monitoring requirements and formalisation of the requirements is all been covered in the above chapters. The further process is the generation of the generic runtime monitor through the LTL formalized requirements. The figure 8.1 shows the process flow involved in generating the monitor. The tools used to translate the LTL to state automaton is shown in the figure Tool. Considering few of the generic requirements from this thesis work. The 7.3.1 shows the GRM Requirements along with their corresponding formalized GRM requirements.

8.1 Linear Temporal Logics to State Automaton Tools

The LTL formalized requirements further need to be converted into the state automaton. The state automaton can be manually drawn by hand or they can be generated automatically through tools such as Spot, BRICS Automaton, LTL2BA etc. In this work the formalized requirements are translated into state automaton by considering Spot and BRICS tools.

Spot tool is automaton manipulation and model checking library written in C++. This tool is an open source and it also provides an online LTL translator. The online translator is an easy way to translate the LTL formulations into automata without the need for the installation of the software.

Spot Tool
The atomic propositions of the SPOT tool are string of characters:

- String of characters are represented between double quotes is an atomic propo-
Figure 8.1: Monitor Generation Process
Figure 8.2: Methods and Tools used to translate LTL to State Automaton

- Any alphanumeric characters that is not a reserved keyword need no require of double quotes
- Any alphanumeric character that starts with ‘F’, ‘G’ or ‘X’ and in the second position if there is a number then there is no need for double quotes.

Boolean Operators for Temporal Formulas
The table 8.1 shows the boolean operations along with the preferred syntax. Considering the ‘f’ and ‘g’ as the two temporal formulas and these two formulas can be combined using boolean operators.

Temporal Operators
The two temporal formulas considering to be ‘f ’ and ‘g ’ and the preferred syntax’s of different operators are mentioned in the table 8.2 Boolean operators for temporal formulas and temporal operators are described above hence now example can be considered.

Req 5: “Icing accretion shall not be greater than maximum accretion and icing accretion on the non protected surfaces shall not be greater than minimum allowable accretion”.
formalized Requirement of LTL: G(iceac<minac)->F(nonprosuac>minacc)

The formalized requirement of the LTL is represented in the short formulas to avoid
### Table 8.1: Boolean Operators for Temporal Formulas

<table>
<thead>
<tr>
<th>Operation</th>
<th>Preferred Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negation</td>
<td>!</td>
</tr>
<tr>
<td>Disjunction</td>
<td>f</td>
</tr>
<tr>
<td>Implication</td>
<td>f-&gt;g</td>
</tr>
<tr>
<td>Exclusion</td>
<td>f xor g</td>
</tr>
<tr>
<td>Equivalence</td>
<td>f&lt;-&gt;g</td>
</tr>
</tbody>
</table>

The SPOT tool provides the user to opt for various desired output for LTL specification. The output desired in this thesis work is the conversion of the LTL specifications into Büchi automaton. Translation of the formula can be done into three different types as you can see in figure 8.3 namely a transition-based generalised Büchi automaton, a state-based de-generalised Büchi automaton and a spin never claim. The figure 8.3 depicts the translation of the LTL into a state-based de-generalised Büchi automaton. The results show state automaton with 2 states which are deterministic, 3 edges and acceptance condition is the Büchi. Now let us consider the desired output of the Büchi automaton by translation of the LTL into a transition-based generalised Büchi automaton. The figure 8.4 indicates the transition-based generalised Büchi automaton. The results show 2 states which are deterministic, 3 edges and acceptance condition with blue coloured circle indicated at the acceptance transition. The major difference between the results of a transition-based generalised Büchi automaton and a state-based degeneralised Büchi automaton is the acceptance condition. The acceptance condition in case of the state-based is the Büchi were the state is indicated with two circles. In case of the transition-based the acceptance condition is the transition with blue coloured circle on top of the state with 1 value. The LTL translation into Spin never claim is shown in the figure 8.5. The results show the Spin code.
<table>
<thead>
<tr>
<th>Operation</th>
<th>Preferred Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next</td>
<td>Xf</td>
</tr>
<tr>
<td>Eventually</td>
<td>Ff</td>
</tr>
<tr>
<td>Always</td>
<td>Gf</td>
</tr>
<tr>
<td>(Strong) Until</td>
<td>f U g</td>
</tr>
<tr>
<td>Weak Until</td>
<td>f W g</td>
</tr>
<tr>
<td>(Weak)Release</td>
<td>f R g</td>
</tr>
<tr>
<td>Strong Release</td>
<td>f M g</td>
</tr>
</tbody>
</table>

Table 8.2: Syntax for Temporal Formulas
All the above three translation from LTL to different Büchi automata was done using the LTL3BA translator algorithm. The other 3 translator algorithms which are available in SPOT tool are Couvrerur/FM, Tauriainen/TAA and Comp.Susp. For the considered example on the icing accretion the results from the different translator algorithms remain to be same. The BRICS automaton tool was also considered for translation of the LTL formalized specification into automaton. In case of the BRICS automaton the LTL formalized requirement shall not be converted into the automaton directly but rather the LTL logic needs to be coded into JAVA. When this code in JAVA is executed it shall produce the automaton run. The automaton run then needs to be copy pasted into Graphviz application which provides the graphical diagram of the automaton. The results obtained after execution of the JAVA code is shown in the figure 8.6. The Netbeans 8.0.2 IDE is used.
8.2 Summary

The requirements after formalized are translated into state automaton. The basics of
the state automaton is discussed in the 2.3.2 The LTL formalized requirements can
be used directly to transform into state automaton. The tool considered in this thesis
to translate them are Spot and BRICS automaton. The Spot tool provides an online
translator also. This means LTL formula can directly be entered online and the results
shall be displayed on the same page. The Spot tool provides options to choose the
translator algorithm, desired output and formal specifications. The LTL formula can
be translated as a transition based generalised büchi automaton as shown in the figure
8.4 The LTL formulas can also be translated into a state based degeneralised büchi
automaton as shown in the figure 8.3 In case of BRICS automaton tool the LTL logic
needs to represented into JAVA code. This JAVA code after running shall output the
digraph automaton. This digraph automaton need to be fed as input to Graphviz tool
which outputs the graphical state automaton.
Figure 8.4: LTL to State Automaton using SPOT tool, A transition-based büchi automaton
Figure 8.5: LTL to State Automaton using SPOT tool, A Spin never claim
Figure 8.6: Diagraph Automaton

```plaintext
digraph Automaton {
  rankdir = LR;
  0 [shape=circle,label=""];
  0 -> 1 [label="p"];  
  1 [shape=doubecircle,label=""];  
  2 [shape=circle,label=""];  
  2 -> 0 [label="c"];  
  2 -> 3 [label="w"];  
  3 [shape=circle,label=""];  
  initial [shape=plaintext,label=""];  
  initial -> 3;  
  3 -> 2 [label="n"];
}
BUILD SUCCESSFUL (total time: 1 second)
```

Figure 8.7: State Automaton
Chapter 9

Taxonomy of GRM Inputs and Health Monitoring Related Terms

A survey of the current literature on terms related to health monitoring identified that a large number of terms relate to health monitoring. The taxonomy of these terms relate to each other and difference in their meaning is illustrated. The taxonomy of GRM inputs used to map to the GRM architecture. The taxonomy helps to classify these inputs and mapping becomes simpler. This section describes the taxonomy of health monitor related terms and its categorisation along with the GRM inputs taxonomy. A common taxonomy of health monitoring terms is been created as a part of the work. There are quite a lot of terms related to health monitoring such as health management, runtime monitoring, status monitoring, online monitor, offline monitoring and so on. But all these terms does not mean the same and hence taxonomy illustration shall make the terms relatable. This illustration delineates were does GRM fall in the categorisation. A taxonomy of inputs provides a clear overview for the reader to know better about categories of the respective inputs. The section 9.1 delineates the Input Taxonomy and further in the section 9.2 describes about the taxonomy of health monitor

9.1 Inputs Taxonomy

The two input taxonomies are considered during this thesis work. During the Initial stages of the thesis work the inputs of GRM were classified using FAA’s pilots handbook of aeronautical knowledge. The inputs were basically classed into 5 categories. As the thesis work progressed the GRM inputs need to be mapped with GRM architecture. This criteria did not hold good with FAA pilot handbook of aeronautical knowledge. Mapping of the categories with GRM architecture was not possible.
Hence the another taxonomy on inputs were considered, GNC categorisation. This GNC categorisation was mapped with the GRM architecture.

### 9.1.1 Taxonomy of Inputs based on FAA’s Pilots Handbook of Aeronautical Knowledge

During the design of the GRM architecture at the initial stages the inputs were categorized mainly into 5 generic categories such as

- Flight Controls
- Aerodynamics of Flight
- Aircraft System
- Weight and Balance
- Weather

The categories considered are generic due to the reason that categories are in general applicable to all the UAV as well as to manned aircraft and also due to source of extraction of categories. These 5 categories were considered from FAA Pilots Handbook of Aeronautical Knowledge. Further while designing the GRM architecture much of changes was needed within the architecture. Further categorization of these inputs were necessary this led to have another categorisation of the GRM inputs which is discussed in further subsection 9.1.2. The taxonomy of inputs based on FAA’s handbook of aeronautical knowledge is not compared with the taxonomy of inputs based on GNC system. The taxonomy of inputs based on FAA’s was developed at the initial stages and this is not considered in mapping the generic runtime monitoring architecture.

### 9.1.2 Taxonomy of Inputs based on GNC System

The classification of inputs was done on the basis on Guidance Navigation and Control System. Each system has subsystems and taking this into consideration the inputs were pictorially represented as one can see in Figure 9.2. The arrows suggest the further classification of that particular system.

### 9.2 Monitor Taxonomy

The taxonomy of monitors is illustrated in the Figure 9.3. The each and individual term has different meaning and the flow of the classification clearly explains the respective position of the different health monitors starting from contingency management. The arrows suggest the further classification of that particular health monitor.

The contingency management is at the top level of the classification. The contingency management not only detected the errors but it also corrects the errors. The errors are reflected back in case of contingency management. The rest of all the other monitoring
factors comes under it. The monitor basics is discussed in the chapter 2. The monitor is further classified into health monitor, non-health monitor and broader approaches. The health monitor shall take into account of remaining fuel of UAV, any kind of malfunctions related to the aircraft equipment and any kind of internal function related error. The non-health monitor includes current and predicted flight parameters, sensor mode, autonomy mode and availability for communication transmissions [16]. In case of broader approaches the system monitor is further categorized into software monitor, hardware monitor and hybrid monitor. The health structural health monitoring can be included under hardware monitor and software health monitoring under software monitor. Both the structural and software health monitoring are discussed along with the related work in chapter 4. Runtime monitor can be used for offline monitoring or online monitoring based on the need of application. Further the online monitor is classified into independent and non-independent monitor. The independent monitor shall not be an integral part of the system to be monitored.

9.3 Summary

The health motoring related terms such as contingency management, health monitor, non health monitor, status monitor etc. all these terms are classified under a tree structure. This taxonomy helps to relate the health motoring terms to each other. The terms are related to each other but their meaning does not remain to be same. The GRM input is categorized based on guidance, navigation and control. The GRM inputs taxonomy helps to map the inputs to the architecture quite easily. The mapping of all the GRM inputs to the architecture is possible due to this taxonomy creation.
CHAPTER 9. TAXONOMY OF GRM INPUTS AND HEALTH MONITORING

RELATED TERMS

Figure 9.1: Input Taxonomy Based On Aeronautical Knowledge
CHAPTER 9. TAXONOMY OF GRM INPUTS AND HEALTH MONITORING

FIGURE 9.2: Input Taxonomy Based On GNC System

Related Terms

Sensing
- Gyroscopes
- Accelerometer

State Estimation
- Planned Cruise Altitude
- Drift Angle
- Indicated Altitude
- Pitch Attitude
- Roll Attitude
- Flight Duration
- Position
- Obstacle
- Absolute Altitude

Waypoint Sequencer
- Sequence Deviation

Mission Planning
- Task Execution
- World Model

Path Planning
- Missed Waypoints

Autopilot
- Initialisation
Figure 9.3: Taxonomy of Health Monitor related terms

CHAPTER 9. TAXONOMY OF GRM INPUTS AND HEALTH MONITORING RELATED TERMS
Chapter 10

Results

The initial results of the thesis work provided the distinction between the different health monitoring terms. Chapter 3 describes the health monitoring terms such as health monitoring, system health monitoring, health management, structural health management, software health management, integrated vehicle health management, and runtime monitoring. The difference between these terminologies is highlighted and the state of art on these terms is put forth. As per one of the objectives of this master thesis, the generic runtime monitoring architecture was designed such that this architecture is extensible. The monitor architecture was designed such that monitor can be integrated into a architecture. The difficulties along with the limits of this monitor integration into architecture are discussed in Chapter 6. The architecture shows how monitor can aggregate different inputs within the architecture. The generic runtime monitor is designed. The monitor detailing within architecture is delineated in earlier Chapter 6. The onboard processing and the off-board processing of the monitor is designed as shown in the figure 6.2 and figure 6.3 respectively. All the available inputs were considered. This runtime monitoring can be used to enable regulatory compliance monitor due to the safety aspect being considered during the design of the monitor. The runtime monitoring is analysed alongside the testing and formal methods such as model checking.

Chapter 4 analyses different methods of runtime monitoring. The generic requirements were elicited which can runtime monitor UAV. The software errors which unnoticed leads to software failures and this leads to lack of situation awareness. As per objective this thesis work combines runtime monitoring along with situation awareness. The generic requirements elicited are not only capable of runtime monitoring of UAV but these are also capable of enhancing situation awareness. The generic requirements sources of elicitation is shown in the figure 7.1. The requirements elicited runtime monitor UAV alongside enhance situation awareness. The generic requirements for all the three levels of situation awareness were extracted. The table 7.1 shows requirements elicited for situation awareness level 1, 2 and 3. Further the requirements were formalized using the linear temporal logics. The typical generic runtime monitoring
requirements are considered and analysed. The requirements aspects and their corresponding priorities were considered as shown in the table 7.5. Requirements based on their corresponding aspects are shown in figure. The requirements were further categorized into GRM inputs, Events/transitions, situation awareness types, aspect, Guidance navigation and control category, UAV/manned aircraft requirement, ALFURS autonomy levels, Off-nominal state behaviour formalisation, situation awareness level and accidents and incidents.

The formalized requirements were further translated into state automaton using Spot tool. The monitor generation process is shown in the figure 8.1. The boolean operator for temporal formulas of the Spot tool is shown in the table 8.1. The syntax of the temporal formulas is shown in the table 8.2. The formalized requirement is translated into state-based de-generalised büchi automaton as shown in the figure 8.3. The formalized requirement further was translated into transition-based generalised büchi automaton as shown in the figure 8.4. The LTL formulae further translated into a Spin never claim as shown in the figure 8.5. The LTL formulas were also translated into the automaton using the BRICS automaton tool. The LTL code shall be represent in the Java code and the run of the Java code provides the digraph automaton as shown in the figure 8.6. This further shall be pasted in Graphviz tool to obtain the state automaton diagram as shown in the figure 8.7.

As per the objective a common taxonomy of terms related to health motoring was created as shown in the figure 9.3. The each and every term related to health monitoring terms has different meaning and the flow of classification explains respective position of different health monitoring terms. Input taxonomy delineates category of individual input category. Two taxonomy of inputs were created during the thesis work that is taxonomy of inputs based on FAA Pilots Handbook of Aeronautical Knowledge as shown in the figure 8.6. This taxonomy was created during the initial stages of the work but later that is it was not used due to necessary changes made in the architecture design. The further taxonomy of inputs was created so as to meet the need of generic architecture. Taxonomy of inputs based on GNC system was created as shown in the figure 9.2.
Chapter 11

Conclusions and Future Scope

The runtime monitoring tackles some of the problems faced by testing and model checking. As the system complexity increases usage of formal methods such as model checking would not be feasible as it faces the state explosion problem. The runtime monitor has a major advantage of its nature of monitoring at runtime. Due to this reason runtime monitoring does not face state explosion problem. The runtime monitoring can be used to monitor online as well as for offline monitoring. The health monitoring related terms are quite a lot. These health monitoring terms does not mean the same. The taxonomy is created so as to clearly relate the terms from one another. The health monitoring related terms considered for taxonomy are health monitor, structural health monitor, health management, system health management, software health management and runtime monitoring. The generic runtime monitor shall detect the errors as soon as possible. The runtime monitoring is cost effective as it verifies only the run of the system. The architecture plays a prominent role in the system design. The Generic (applicable to all UAV’s) runtime monitor architecture is an extensible architecture. The architecture delineates the detailing of the monitor which is automatically generated using the formal specifications. The architecture details both the offboard processing of the data as well on board processing of the data. The inputs are mapped to the system architecture. The requirements were elicited in the textual statements using the template based method. Some of the requirements considered were based on some of the aviation incidents and accidents happened due to loss of situational awareness. The situational awareness is about just getting to know around us. The situational awareness incase of aviation system is to have the environmental awareness along with the internal behaviour of the aircraft.

Situation Awareness (Staying advanced of aircraft) is the driving force for the decision making process. The requirements extracted were from all 3 levels of situation awareness namely perception, comprehension and projection. The three aspects were considered namely safety, mission and performance. The aspects were prioritised considering safety aspect with the highest priority followed by mission aspect and finally performance. The performance aspect was given lowest priority. The requirements
were elicited considering all the three aspects. The requirements were iterated so that the GRM architecture accounts all the requirements. The requirements were categorized into 8 categories. The categories considered are GRM inputs, aspect, events, formalisation of the requirements using LTL and so on. These requirements were further formalized using the temporal logics such as the linear temporal logic. The formalised requirements were translated into monitor automatically using the Spot tool and BRICS automaton tool. The Spot tool provides an online translator also. This means LTL formula can directly be entered online and the results shall be displayed on the same page. The Spot tool provides options to choose the translator algorithm, desired output and formal specifications. The LTL formula can be translated as a transition based generalised Büchi automaton as shown in the figure 8.3. The LTL formulas can also be translated into a state based degeneralised Büchi automaton as shown in the figure 8.3. In case of BRICS automaton tool the LTL logic needs to be represented into Java code. This Java code after running shall output the digraph automaton as shown in figure 8.6. This digraph automaton need to be fed as input to Graphviz tool which outputs the graphical state automaton. The larger goal which was not a part of this thesis work was to correct the possible errors as soon as they are detected. The future work shall be to create a runtime monitor that is capable to correct the possible errors on board. This shall enhance the safety of the system. The generic runtime monitoring architecture is extensible one. Retrofitting into a UAS should be realized in future and this generic runtime monitor shall facilitate retrofitting into UAS. The generic runtime monitor is independent monitor. The system to be verified and the monitoring system are independent. In future GRM shall be able to correct the errors on board and shall also simplify certification process.
Bibliography


[37] DLR German Aerospace Center. Institute of flight systems. DLR, January 2016.


BIBLIOGRAPHY
### Selbstständigkeitserklärung

Ich declare on behalf of the Technische Universität Chemnitz that the thesis is all my own work and uses no external material other than that acknowledged in the text.

This work contains no plagiarism and all sentences or passages directly quoted from other people's work or including content derived from such work have been specifically credited to the authors and sources.

This paper has neither been submitted in the same or a similar form to any other examiner nor for the award of any other degree, nor has it previously been published.