Testing Quick Reference Handbooks in Flight Simulators

Anthony Berg (200871682)
Supervisor: Dr Leo Freitas

12th May 2024

Preface

Abstract

This is an abstract.

Declaration

I declare that this dissertation represents my own work except where otherwise stated.

Acknowledgements

This is the acknowledgements.

Contents

Chapter 1

Introduction

1.1 Scene

- Designing Emergency Checklists is difficult
- Procedures in checklists must be tested in simulators [1], which usually means trained pilots test it, as the tests need to work consistently [2] (making sure it's not lengthy, concise and gets critical procedures)
- Checklists are usually carried out in high workload environments, especially emergency ones

1.2 Motivation

- Testing procedures in checklists are often neglected [1]
- There are some checklists that may not be fit for certain scenarios - e.g. ditching (water landing) checklist for US Airways Flight 1549 assumed at least one engine was running [3], but in this scenario, there were none
- Some checklists may make pilots 'stuck' - not widely implemented, could be fixed with 'opt out' points. e.g. US Airways 1549, plane below 3000ft, could have skip to later in the checklist to something like turn on APU, otherwise plane will have limited control [3].
- Checklists may take too long to carry out - Swissair 111

1.3 Aim

The goal of this project is to test checklists in Quick Reference Handbooks (QRH) for flaws that could compromise the aircraft and making sure that the tests can be completed in a reasonable amount of time by pilots. It is also crucial to make sure that the tests are reproducible in the same flight conditions and a variety of flight conditions.

1.4 Objectives

1. Research current checklists that may be problematic and are testable in the QRH tester being made
2. Implement a formal model that runs through checklists, with the research gathered to produce an accurate test
   (a) Understand the relative states of the aircraft that need to be captured
   (b) Ensure that the results of the checklist procedures are consistent
3. Implement a QRH tester manager that
   - Runs the formal model and reacts to the output of the formal model

- Connect to a flight simulator to run actions from the formal model
- Implement checklist procedures to be tested, run them, and get feedback on how well the procedure ran

Chapter 2

Background

2.1 Hypothesis

- Checklists can be tested in a simulated environment to find flaws in checklist for things like
  - Can be done in an amount of time that will not endanger aircraft
  - Provides reproducible results
  - Procedures will not endanger aircraft or crew further (Crew referring to Checklist Manifesto with the cargo door blowout)
- Results in being able to see where to improve checklists

2.2 Safety in Aviation

2.2.1 History

- 70-80% of aviation accidents are attributed to human factors [4]

2.2.2 Checklists

- Checklists have been shown to aid in minimising human errors [2]
- However, checklists can be misleading and compromise the safety of the aircraft due to them being either too confusing or taking too long to complete [1]
- That is why testing checklists are important to avoid these situations

2.3 Formal Methods

2.3.1 History

2.3.2 Application

2.4 Solution Stack

- There would be around 3 main components to this tester
  - Formal Model
  - Flight Simulator plugin
  - Checklist Tester (to connect the formal model and flight simulator)
- As VDM-SL is being used, it uses VDMJ to parse the model [5]. This was a starting point for the tech stack, as VDMJ is also open source.
- VDMJ is written in Java [5], therefore to simplify implementing VDMJ into the Checklist Tester, it would be logical to use a Java virtual machine (JVM) language.

2.4.1 Formal Model

- There were a few ways of implementing the formal model into another application
- Some of these methods were provided by Overture [6]
  - RemoteControl interface
  - VDMTools API [7]
- However, both of these methods did not suit what was required as most of the documentation for RemoteControl was designed for the Overture Tool IDE. VDMTools may have handled the formal model differently
- The choice was to create a VDMJ wrapper, as the modules are available on Maven

2.4.2 Checklist Tester

JVM Language

- There are multiple languages that are made for or support JVMs [8]
- Requirements for language
  - Be able to interact with Java code because of VDMJ
  - Have Graphical User Interface (GUI) libraries
  - Have good support (the more popular, the more resources available)
- The main contenders were Java and Kotlin [9]
- Kotlin [9] was the choice in the end as Google has been putting Kotlin first instead of Java. Kotlin also requires less boilerplate code (e.g. getters and setters) [10]

Graphical User Interface

- As the tester is going to include a UI, the language choice was still important
- There are a variety of GUI libraries to consider using
  - JavaFX [11]
  - Swing [12]
  - Compose Multiplatform [13]
- The decision was to use Compose Multiplatform in the end, due to time limitations and having prior experience in using Flutter [14]
- Compose Multiplatform has the ability to create a desktop application and a server, which would allow for leeway if a server would be needed

2.4.3 Flight Simulator Plugin

- There are two main choices for flight simulators that can be used for professional simulation
  - X-Plane [15]
  - Prepar3D [16]
- X-Plane was the choice due to having better documentation for the SDK, and a variety of development libraries for the simulator itself
- For the plugin itself, there was already a solution developed by NASA, X-Plane Connect [17] that is more appropriate due to the time limitations and would be more likely to be reliable as it has been developed since 2015

Chapter 3

Design/Implementation

3.1 Components

Splitting up the project into multiple components has been useful for

- Aiding in planning to make the implementation more efficient
- Delegating specific work tasks
- Making the project modular, for example, allowing for a different simulator to be implemented with minimal need to refactor other parts of the codebase

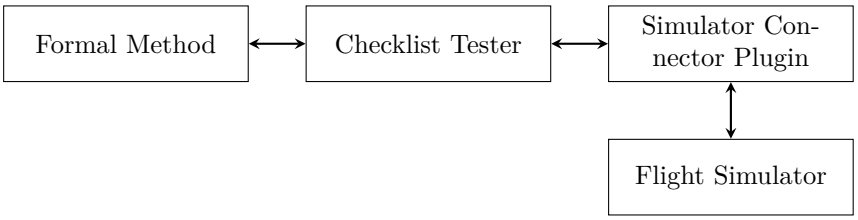

Figure 3.1: Abstract layout of components

Each of the components in Figure 3.1 will be covered in detail in this chapter.

3.2 Model

- Formal modelling is the heart of the logic for testing checklists
- Formal model created using VDM-SL
- It allows to test that the checklists have been completed in a proper manner - and that it is provable
- Model keeps track of
  - Aircraft state
  - Checklist state
- If an error were to occur in the model, this can be relayed that there was something wrong with running the test for the checklist, such as:
  - Procedure compromises integrity of aircraft
  - There is not enough time to complete the procedure
  - There is a contradiction with the steps of the checklist

3.3 Scenarios

- Use a Quick Reference Handbook (QRH) to find potential list of checklists to test
- Look at previous accident reports that had an incident related to checklists and test it with my tool to see if it will pick it up
- These previous accident reports can be good metrics to know what statistics to look out for

3.4 Decisions

Chapter 4

Results

4.1 Problems Found

4.2 LOC?

4.3 Reflection

4.4 Time Spent

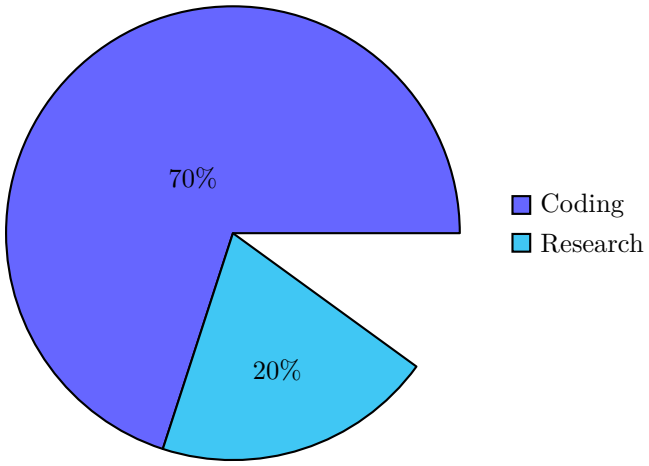

Figure 4.1: Time spent on sections of project

Chapter 5

Conclusion

5.1 Changes

5.2 Objectives

5.3 What Next

References

[1] Immanuel Barshi, Robert Mauro, Asaf Degani et al. *Designing Flightdeck Procedures*. eng. Ames Research Center, Nov. 2016. URL: https://ntrs.nasa.gov/citations/20160013263.

[2] Atul Gawande. *The Checklist Manifesto: How To Get Things Right*. Main Edition. Profile Books, July 2010. ISBN: 9781846683145.

[3] National Tranportation Safety Board. *Loss of Thrust in Both Engines After Encountering a Flock of Birds and Subsequent Ditching on the Hudson River*. Technical Report PB2010-910403. May 2010. URL: https://www.ntsb.gov/investigations/Pages/DCA09MA026.aspx.

[4] William R. Knecht and Michael Lenz. *Causes of General Aviation Weather-Related, Non-Fatal Incidents: Analysis Using NASA Aviation Safety Reporting System Data*. Tech. rep. DOT/FAA/AM-10/13. FAA Office of Aerospace Medicine Civil Aerospace Medical Institute, Sept. 2010.

[5] Nick Battle. *VDMJ*. URL: https://github.com/nickbattle/vdmj (visited on 21/04/2024).

[6] Peter Gorm Larsen, Kenneth Lausdahl, Peter Jørgensen et al. *Overture VDM-10 Tool Support: User Guide*. TR-2010-02. Apr. 2013. Chap. 16, pp. 81–98. URL: https://raw.github.com/overturetool/documentation/editing/documentation/UserGuideOvertureIDE/OvertureIDEUserGuide.pdf.

[7] Kyushu University. *The VDM Toolbox API*. Version 1.0. 2016. URL: https://github.com/vdmtools/vdmtools/raw/stable/doc/api-man/ApiMan_a4E.pdf.

[8] Raoul-Gabriel Urma. 'Alternative Languages for the JVM'. In: *Java Magazine* (July 2014). URL: https://www.oracle.com/technical-resources/articles/java/architect-languages.html (visited on 05/05/2024).

[9] JetBrains s.r.o. *Kotlin Programming Language*. URL: https://kotlinlang.org/ (visited on 21/04/2024).

[10] Google LLC. *Kotlin and Android | Android Developers*. URL: https://developer.android.com/kotlin (visited on 21/04/2024).

[11] OpenJFX. *JavaFX*. URL: https://openjfx.io/ (visited on 21/04/2024).

[12] FormDev Software GmbH. *FlatLaf - Flat Look and Feel | FormDev*. URL: https://www.formdev.com/flatlaf/ (visited on 21/04/2024).

[13] JetBrains s.r.o. *Compose Multiplatform UI Framework | JetBrains | JetBrains: Developer Tools for Professionals and Teams*. URL: https://www.jetbrains.com/lp/compose-multiplatform/ (visited on 21/04/2024).

[14] Google LLC. *Flutter - Build apps for any screen*. URL: https://flutter.dev/ (visited on 21/04/2024).

[15] Laminar Research. *X-Plane | The world's most advanced flight simulator*. URL: https://www.x-plane.com/ (visited on 21/04/2024).

[16] Lockheed Martin Corporation. *Prepar3D – Next Level Training. World class simulation. Be ahead of ready with Prepar3D*. URL: https://www.prepar3d.com/ (visited on 21/04/2024).

[17] NASA Ames Research Center Diagnostics and Prognostics Group. *X-Plane Connect*. URL: https://github.com/nasa/XPlaneConnect (visited on 21/04/2024).

[18] Barbara Burian. 'Design Guidance for Emergency and Abnormal Checklists in Aviation'. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50 (Oct. 2006). DOI: 10.1177/154193120605000123.

[19] Quinn Kennedy, Joy Taylor, Daniel Heraldez et al. 'Intraindividual Variability in Basic Reaction Time Predicts Middle-Aged and Older Pilots' Flight Simulator Performance'. In: *The Journals of Gerontology: Series B* 68.4 (Oct. 2012), pp. 487–494. ISSN: 1079-5014. DOI: 10.1093/geronb/gbs090. eprint: https://academic.oup.com/psychsocgerontology/article-pdf/68/4/487/1520662/gbs090.pdf.

[20] Civil Aviation Authority. *Aircraft Emergencies: Considerations for air traffic controllers*. CAP745. Mar. 2005. URL: https://www.caa.co.uk/cap745.

[21] The Overture Project. *The Vienna Development Method*. URL: https://www.overturetool.org/method/ (visited on 23/02/2024).

[22] Google LLC. *Jetpack Compose UI App Development Toolkit - Android Developers*. URL: https://developer.android.com/develop/ui/compose (visited on 21/04/2024).