Automatically Generated Test Frames from a Q Specification of ICAO Flight Plan Form Instructions

Michael R. Donat
donat@cs.ubc.ca
http://www.cs.ubc.ca/spider/donat
http://www.cs.ubc.ca/formalWARE

April 30, 1998

Abstract

A partially automated process for generating tests has been experimentally applied to a portion of a real world system-level requirements specification. This paper discusses the problems addressed by this process along with how and why this automation was achieved. The requirements were formalized using a notation designed to be readable by a large proportion of requirements stakeholders. This report also addresses traceability of requirements to tests and introduces the requirements specification language Q.

Technical Report TR-98-05
Department of Computer Science
University of British Columbia
Vancouver, B.C.
CANADA

Contents

1 Introduction

This document reports on the semi-automatic generation of a set of 252 test frames from a portion of the ICAO instructions for filling out a flight plan as specified in Appendix 2, Subsection 2 of ICAO's Rules of the Air and Air Traffic Services [6]. Appendices D and E contain a list of 252 test frames which were automatically generated by a software tool from a parseable representation of testable requirements. Figure 1 provides a sample of one of these automatically generated test frames. The test frames for Appendices D and E are generated through different uses of the same requirements specification. Each set of test frames provides complete coverage of all the testable requirements relative to the context in which they are used. The 122 test frames contained in Appendix D are schemas for testing a system that automatically fills out a flight plan. The remaining 130 test frames in Appendix E are schemas for testing a system that validates a given flight plan.

ROIDs: I19ES4	
Stimuli	Response
1. Dinghies are carried	1. insert {Item 19 D} - {number of dinghies carried} 2. insert {Item 19 D} - {total capacity in persons of all dinghies carried} 3. insert {Item 19 D} - {colour of dinghies}

Figure 1: A test frame from Appendix D.

Each test frame specifies a specific combination of conditions corresponding to a single step in a test procedure.[1] The contents of the "Stimuli" field of each test frame are used to determine the contents of the "Stimuli" field of a test step. A test engineer would refine a test frame into a test step by entering appropriate data values into the "Stimuli" and "Responses" fields of the test step that satisfy the "Stimuli" of the test frame. In addition to specifying the contents of the "Stimuli" field, each test frame includes traceability information which may be used by the test engineer to specify Requirement Object IDentifiers (ROIDs) in the "Verified Requirements" field of the test step.

The test frames in this report are provided as an example of a semi-automated process employing a formal yet readable-by-non-specialists requirements specification. It is expected that these 252 test frames could be used directly by test engineers in the development of test procedures for software that produces

[1]A test procedure is a sequence of test steps. Each test step contributes to the demonstration that a specified requirement has indeed been implemented.

a filled out flight plan (Appendix D) and for software validating filled out flight plans (Appendix E). The generation of these 252 test frames was performed by means of an algorithm based on a specific, precisely defined coverage criterion. The ten pages of testable requirements were manually translated into a parseable representation of similar size. This representation was then parsed by the software tool and systematically transformed into test frames. Each step in this derivation is a logical inference. These inferences can be grouped into meta-steps which parallel the steps that would be taken by a test engineer in a manual process.

An overview of the process used to generate the test frames is briefly described in Section 2. Appendix A contains the Q translation of the ICAO flight plan instructions. Appendix C describes the Q specification language. Section 3 of this report outlines a process for the refinement of test frames from Appendix D or E into test steps within a test outline. The coverage criterion determines the number of test frames generated as well as serving as the basis of any claim about the completeness of a test procedure. Section 4 provides a description of the coverage criterion used to generate the 252 test frames in Appendices D and E. Appendix B details the mathemtical definition of the coverage criterion. For each of the test frames, all of the conditions specified in the "Stimuli" field of the test frame are both necessary and sufficient. Section 5 of this report describes an alternate approach which supplements the necessary and sufficient conditions with additional conditions that fully differentiate the test frame from other test frames as a means of helping the test engineer ensure that the expected response has a unique cause. Traceability of requirements to tests is addressed in Section 6. The time required to generate these test frames is described in Section 7. A brief summary of this report is provided in Section 8.

2 Test Frame Generation: Process Overview

This overview provides a brief introduction to the test frame generation process. Details of this process are not essential to the use of the test frames in Appendices D and E. The process used to generate test frames uses a parseable representation of the specification and a test frame generation tool, QTCG. The purpose of this process is to enhance the current manual process through automation while leaving enough flexibility for engineering judgment to be applied. Figure 2 illustrates this process.

Once a set of system-level requirements has been selected, the process of generating test frames involves three steps:

1. Translate the system-level requirements into a parseable representation that can be processed mechanically.

2. Add domain knowledge to document dependencies between conditions. This information is used to eliminate infeasible tests.

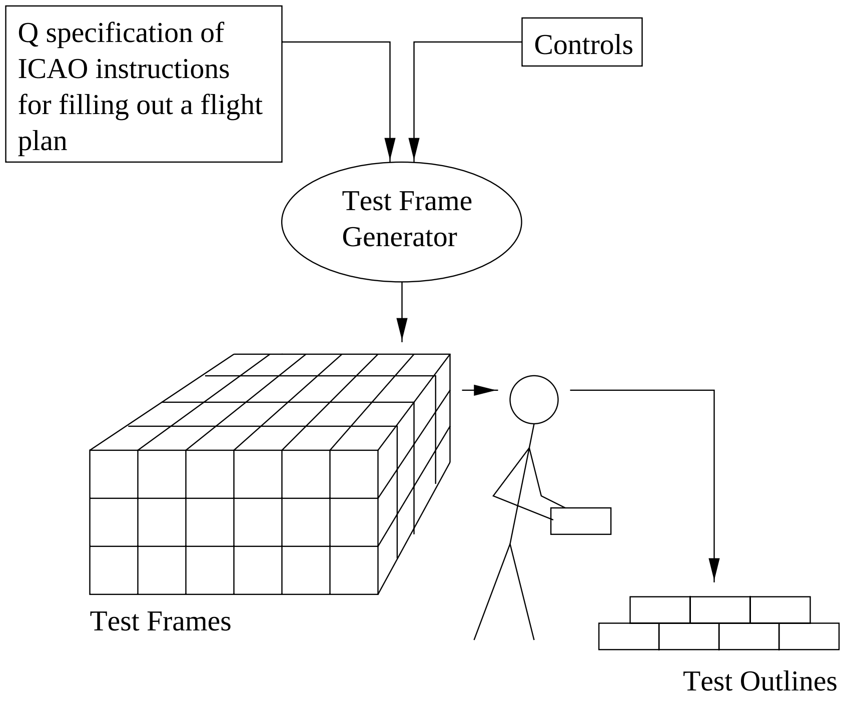
Q specification of ICAO instructions for filling out a flight plan
Controls
Test Frame Generator
Test Frames
Test Outlines

Figure 2: Automatic Generation of Test Frames

3. Use the QTCG tool to generate test frames from the parseable representation. The conversion to a parseable representation is a relatively simple translation task.

To produce the formal specification in Appendix A, text was translated directly from the ICAO flight plan instructions into the parseable representation. The parseable representation is written in a formal language. An important characteristic of the parseable representation is that it is also readable by those unfamiliar with the formal language. The Q fragment below is taken directly from the specification in Appendix A.

```
I19ES4.
if {not {Dinghies are carried}} then {
  cross out {Item 19 D} - {each of {D, C}}}
else {all of {
  insert {Item 19 D} - {number of dinghies carried},

  insert {Item 19 D} -
    {total capacity in persons of all dinghies carried},

  if {not {Dinghies are covered}} then {
    cross out {Item 19 D} - {C}},

  insert {Item 19 D} - {colour of dinghies}
  }}
```

This fragment explicitly expresses the logical relationships between conditions but does not assign any meaning to these conditions. The logical relationships are required for algorithmic test frame generation. The language of this parseable representation is described in more detail in Appendix C.

The parseable representation makes the logical structure containing testable requirements and the alternatives within the requirements explicit. The QTCG tool exploits this structure in the parseable representation to generate test frames. Furthermore, the parseable representation allows requirements to be tagged with identifiers. The QTCG tool preserves these identifiers while generating test frames. The requirement identifiers associated with a test frame indicate which requirements were relevant to its construction. A traceability mapping from test frames to requirements would normally be constructed manually by test engineers as they construct test frames. The QTCG tool provides this mapping automatically.

3 Test Steps from Test Frames

A softcopy of the test frames can be developed into test steps by following the steps below:

1. Sequence the test frames into outlines of test procedures.

2. For each test frame in the outline, select appropriate values that satisfy the stimuli specified by the test frame in a manner compatible with the response in the previous test step.

If it is not possible to select values in step 2, either the outline is infeasible or previously selected values must be adjusted to construct a feasible test procedure.

4 Coverage Criteria

The completeness of a test set is determined by a coverage criterion. The test frames in Appendices D and E were generated using a condition coverage criterion. A simple description of this criterion is that there is at least one test frame for each condition in the Q specification of the requirements. This coverage criterion is based on a mathematical foundation [2]. The precise mathematical definition of this coverage criterion is given in Appendix B. This coverage criterion is intended to be a precise interpretation of the guidance provided in paragraph 6.4.4.1(a)[2] of DO178B [11] that "test cases exist for each software requirement."

This coverage criterion is illustrated by the following example:

The condition R exists if all of the following conditions are satisfied:

1. condition A is true or condition B is true, and
2. condition C is true or condition D is true.

In this example, the letters A, B, C, D, and R are used to symbolically represent a set of conditions. For instance, the letter A may actually be a phrase such as "the total number of persons is known." Given that each of the four conditions A, B, C, and D can be true or false, there are sixteen possible logical combinations of these values. But, of course, it is not practical to generate test steps for each of the possible logical combinations since, in general, the number of test cases would grow exponentially with the number of conditions.

The coverage criterion defined mathematically in Appendix B, requires each requirement to be verified once in the sense that every condition must appear in at least one test procedure step. The coverage criterion also requires the conditions to be both necessary and sufficient. For the above example, these constraints can be satisfied by just two test procedures steps. A step in which condition A and condition C are both true together with step in which condition B and condition D are true would satisfy this coverage criterion. An equally valid combination is a step in which condition A and condition D are both true together with a step in which condition B and condition C are true.

[2]6.4.4.1(b) refers to data selection.

5 Test Frame Styles

The QTCG tool is capable of listing conditions for test frames in one of two styles. The "base style" lists only those conditions that are necessary and sufficient to cause the response. However, this list may not be sufficient to differentiate this cause of the response from that of an overlapping test frame. For this purpose test frame conditions can be listed using the "differentiated style." The style is selected by the test engineer.

The difference between "base style" and "differentiated style" is illustrated in the following example.

Produce response R if any of the following conditions are true:

1. the value of field X is less than 5,
2. the value of field Y is less than 3, or
3. the value of field Z is less than 7.

The test frames for this fragment using a base style are:

–Test Frame 1:

Stimuli	Response
1. $X < 5$	1. R

–Test Frame 2:

Stimuli	Response
1. $Y < 3$	1. R

–Test Frame 3:

Stimuli	Response
1. $Z < 7$	1. R

This style allows for the maximum amount of choice exercised by test engineers in constructing test steps. However, while specifying the test step corresponding to test frame 1, it may be necessary to specify values for Y and Z. The test step corresponding to:

Stimulus	Response
1. $X = 4$	1. R
2. $Y = 2$	
3. $Z = 8$	

does not differentiate between test frames 1 and 2. The differentiated style can assist test engineers by adding constraints to the list of conditions that differentiate the test frames. In this example the set of differentiated test frames is (the extra constraints, or differentiating conditions, are marked with a "•"):

–Test Frame 1:

Stimuli	Response
1. $Y < 3$ 2. $\bullet \neg (X < 5)$ 3. $\bullet \neg (Z < 7)$	1. R

–Test Frame 2:

Stimuli	Response
1. $Z < 7$ 2. $\bullet \neg (X < 5)$ 3. $\bullet \neg (Y < 3)$	1. R

–Test Frame 3:

Stimuli	Response
1. $X < 5$ 2. $\bullet \neg (Y < 3)$ 3. $\bullet \neg (Z < 7)$	1. R

Differentiated frames can be useful in ensuring that test engineers construct test steps that are differentiated. However, in some cases, test frame differentiation takes significant processing time and there may be several alternatives to choose from in order to achieve differentiation. In the QTCG prototype, the choice between alternatives is arbitrary and might not always be appropriate according to best engineering judgment.

As a second example, a base test frame from Section E.2 is:

`--Test Frame 1.14:`

ROIDs: I19P	
Stimuli	Response
1. Number of persons is required by the ATS authority 2. The total number of persons is known 3. NOT (insert {Item 19 P} - {the total number of persons [passengers and crew] on board})	1. report error

while its differentiated form (Section E.3) is:

--Test Frame 1.10:

ROIDs: I19P	
Stimuli	Response
1. Number of persons is required by the ATS authority	1. report error
2. The total number of persons is known	
3. NOT (insert {Item 19 P} - {the total number of persons [passengers and crew] on board})	
4. • Aircraft Identification is correct	
5. • FlightRules and Type of Flight is correct	
6. • Number and Type of Aircraft and Wake Turbulence Category is correct	
7. • Equipment is correct	
8. • Departure Aerodrome and time are correct	
9. • Route is correct	
10. • Destination Aerodrome and Total Estimated Elapsed Time is correct	
11. • Other Information is correct	
12. • insert {Item 19 E} - {the four digit fuel endurance in hours and mi nutes}	
13. • cross out {Item 19 R} - {U}	
14. • cross out {Item 19 R} - {V}	
15. • Emergency location beacon is available	
16. • Polar equipment is carried	
17. • Desert equipment is carried	
18. • Maritime equipment is carried	
19. • Jungle equipment is carried	
20. • cross out {Item 19 J} - {V}	
21. • cross out {Item 19 J} - {U}	
22. • cross out {Item 19 J} - {F_}	
23. • cross out {Item 19 J} - {L}	
24. • Supplementary Information [Part 2] is correct	

The advantage of the differentiated test frame is that the additional conditions ensure there is no overlap with another test frame. The disadvantage is that there may be several different ways to differentiate the test frame, but the current prototype test frame generator takes this flexibility away from the engineer by making an arbitrary choice. It is important to note that test frame style is independent of coverage criteria.

6 Traceability

Traceability is necessary for providing an audit trail to support process monitoring. The QTCG tool supports traceability by keeping track of requirement identifiers inserted into the Q specification by requirements authors. These requirement identifiers are propagated through the logical inferences during the derivation of test frames. This automates the construction of a traceability mapping between requirements and test frames that is currently done manually by test engineers.

7 Processing Times

Computing the base test frames for Appendix D required a total of one minute and 42 seconds[3] on an Ultra-Sparc 60. The base test frames for Appendix E required a total of two minutes and 39 seconds. Computing the differentiated test frames for Appendix E had to be done in pieces and required fifty minutes and seven seconds. Constructing the set of scripts for generating test frames took approximately half an hour.

From the author's exposure to industry practice, a conservative estimate of the time required to construct, review, and produce a traceability map for a single test frame, on average, is one hour.[4] By this estimate, the base test frames in Appendix E that were automatically generated in under three minutes would require approximately three person-weeks to prepare manually. This comparison does not include the translation time due to the expectation that requirements authors would produce original specifications in Q.

8 Summary

This document has reported the production of 252 test frames using a semi-automated process. Test frames can be used during test development to construct test steps. The automatic production of test frames from a parseable interpretation of system-level requirements has the potential to reduce the labour required to produce test steps for logically complex conditions. In addition, the test frames are produced according to a precise definition of coverage which ensures that the coverage provided by the test frames is consistent and homogenous. Conditions for test frames can be listed in one of two styles: 1) necessary and sufficient, or 2) necessary and sufficient along with additional conditions to ensure that no test step can satisfy more than one test frame. Requirement identifier information is automatically propagated to the test frames during their production. This report includes a description of the Q requirements specification language. Further details of this research can be found in [5, 3, 4, 13].

[3]The times given are the elapsed time reported by the unix time utility.

[4]In some cases the estimate is one day.

Acknowledgments

Phil Gray of MacDonald Dettwiler proposed using the ICAO flight plan instructions as the subject of this case study. Some of the ideas for Q are due to Jeffrey Joyce of Raytheon Systems Canada Ltd., and Kendra Cooper, a fellow UBC Ph.D. student. Comments by Jeffrey Joyce have contributed to the presentation of this report.

This work is supported by *formal***WARE**, a university-industry collaborative research project sponsored jointly by the BC Advanced Systems Institute, Raytheon Systems Canada Ltd., MacDonald Dettwiler, The University of British Columbia and The University of Victoria.

http://www.cs.ubc.ca/formalWARE

References

[1] Kendra Cooper. Flex-fix predicates. conversation, June 1997.

[2] Michael R. Donat. Automating formal specification-based testing. In Michel Bidoit and Max Dauchet, editors, *TAPSOFT '97:Theory and Practice of Software Development, 7th International Joint Conference CAAP/FASE*, volume 1214 of *Lecture Notes in Computer Science*, pages 833–847. Springer-Verlag, April 1997.

[3] Michael R. Donat. Automatically generated test frames from an S specification of separation minima for the North Atlantic region. Technical Report TR-98-04, Department of Computer Science, University of British Columbia, Vancouver, B.C., Canada, April 1998.

[4] Michael R. Donat. *A Discipline of Specification-Based Test Generation*. PhD thesis, Department of Computer Science, University of British Columbia, Vancouver, B.C., Canada, 1998. In preparation.

[5] Michael R. Donat and Jeffrey J. Joyce. Applying an automated test description tool to testing based on system level requirements. In *8th Annual Symposium of the International Council on Systems Engineering*, Vancouver, July 1998. International Council on Systems Engineering. http://www.incose.org.

[6] International Civil Aviation Organization, Montréal, Canada. *Rules of the Air and Air Traffic Services (PANS-RAC Doc 4444)*, November 1994. http://www.icao.int.

[7] C. B. Jones. *Systematic Software Development Using VDM (2nd edition)*. Prentice Hall, 1990.

[8] Jeffrey J. Joyce. TCEL. Proprietary document, April 1997.

[9] Jeffrey J. Joyce, Nancy Day, and Michael R. Donat. S: A machine readable specification notation based on higher order logic. In Thomas F. Melham and Juanito Camilleri, editors, *Higher Order Logic Theorem Proving and Its Applications, 7th International Workshop*, volume 859 of *Lecture Notes in Computer Science*, pages 285–299. Springer-Verlag, 1994.

[10] Lawrence C. Paulson. *ML for the Working Programmer.* Cambridge University Press, second, paperback edition, 1992.

[11] RTCA, Inc. and EUROCAE. *DO-178B, Software Considerations in Airbourne Systems and Equipment Certification*, 12B edition, December 1992.

[12] J. Michael Spivey. *Understanding Z: A Specification language and its formal semantics.* Cambridge University Press, 1988.

[13] Kalman Toth, Michael R. Donat, and Jeffrey J. Joyce. Generating test cases from formal specifications. In *6th Annual Symposium of the International Council on Systems Engineering*, Boston, July 1996. International Council on Systems Engineering. http://www.incose.org.