Applying an Automated Test Description Tool to Testing Based on System Level Requirements

Michael R. Donat
University of British Columbia
Department of Computer Science
2366 Main Mall
Vancouver, British Columbia
Canada V6T 1Z4

Jeffrey J. Joyce
Raytheon Systems Canada Ltd.
International Airspace Management Systems Division
13951 Bridgeport Road
Richmond, British Columbia
Canada V6V 1J6

Abstract. A partially automated process for generating test procedures has been experimentally applied to a portion of the Software Requirements Specification for an Air Traffic Management system. This process uses algorithms based on formal logic to automate some of the more tedious and error prone aspects of deriving test procedures from requirements. This approach is particularly well suited to functional requirements involving complex decisions within stimulus-response relationships. In addition to the potential improvement to requirements-based test generation methodology, this process may also be used to improve requirements authoring.

INTRODUCTION

This paper presents a partially automated process for generating tests from system level requirements specifications. To assess the practical usefulness of this process, it was experimentally applied to a portion of the Software Requirements Specification (SRS) for the Canadian Automated Air Traffic System (CAATS) being developed by Raytheon Systems Canada Ltd. This process is designed to be integrated with current industrial practices such as DO178B and MIL-STD-498. It directly addresses the issues of coverage consistency, traceability of requirements to tests, and minimum imposition of tools on current practice. This process is the fruition of concepts originally laid out in (Toth et. al. 1996) and incorporates the theory detailed in (Donat 1997).

This paper demonstrates that a formal representation of a set of system requirements can be parsed and algorithmically transformed into a set of test descriptions called *test frames*. These test frames are complete with respect to an objective definition of coverage and include accurate tracing information. Although this process requires a formal specification, this task is more like formatting than mathematical modelling.

The context of this process is discussed in the next section. The following two sections then discuss problems with the current manual process and issues involved in automating it. The automated process is then introduced. Two coverage criteria are briefly described. This is followed by a description of the application of this process to a portion of the CAATS software requirements. Conclusions and future applications of this work are also discussed.

CONTEXT

This paper focuses on aspects of the planning phase of system-level requirements-based testing for the purpose of requirements verification.

Following a methodology consistent with standard industry practice, descriptions of test steps form the basis for defining test procedures. In this paper these descriptions are referred to as *test frames*. Each test procedure contains a sequence of test steps. Each test step specifies the list of stimuli to be given to the system, the list of conditions describing the state of the system at the time of the stimulus, and the expected response of the system at that step in the test procedure. Successful execution of these test procedures along with documentation that the test steps provide adequate coverage of the requirements constitutes verification of the requirements. A test frame such as,

Stimulus	Condition	Response
altitude check	{altitude} is greater than 38,000 feet	issue a warning

might be instantiated[1] as the following step in a test procedure:

Stimulus	Condition	Response
altitude check	{altitude} is 39,000 feet	issue a warning

[1] This instantiation of test frames to test steps is difficult to automate in some cases and can be impossible in others. Our research focuses on the derivation of test frames from requirements. We have not considered the separate problem of deriving test steps from test frames.

Test frames provide a reliable basis for test planning while avoiding the risks of committing resources to constructing detailed test steps early in the development cycle.

PROBLEMS

The following problems exist in the process of deriving test frames from requirements specifications:

1. Extracting test frames from the requirements specification requires a great deal of effort. Requirements specifications are often large and complex. This complexity arises through the use of decisions referring to several conditions and the negation of such decisions. As a result, manual test frame derivation is a tedious, routine, and error prone task. The nature of this task requires that additional effort be spent in reviews to ensure that the test frames satisfy certain properties. The work presented in this paper addresses this extraction process without placing unreasonable constraints on other aspects of system-level requirements-based testing where engineering judgement and skill are required.
2. Re-working test procedures as a result of specification changes is costly not only due to the effort involved, but also due to the impact on schedules.
3. The lack of well-defined coverage criteria for system-level requirements-based testing places too much dependence on engineering judgement and experience. This accentuates differences in performance within the test team to the point where it becomes a management issue. While documents such as DO178B, DOD-STD-2167A, ANSI/IEEE 829-1983, and MIL-STD-498 provide some general guidelines for requirements-based testing, they do not provide the specific detail required to objectively decide if a particular test frame is missing or redundant.
4. Ensuring that traceability exists between requirements and tests assists in addressing problems 2 and 3 above by providing an index that can be used to facilitate the appropriate review tasks. However, these problems are not solved by traceability alone.
5. The act of producing test frames often uncovers anomalies in the specification. However, the loose connection between specification authoring and test planning causes this feedback to be delayed until late in the process of authoring requirements.

CHALLENGES FOR AUTOMATION

The previous section noted problems that stem from a typical test derivation process. This section discusses additional issues that arise when attempting to automate it.

The first challenge is to choose an approach that ensures that the test frames produced are consistent with the specification. It will also be necessary to detect dependencies between conditions within the specification so that no infeasible test frames are generated and that redundant conditions are removed. One approach is the use of formal rules of logic as a medium for calculating test frames from the specification. This is analogous to the use of statistical analysis in the calculation of marketing statistics by a spreadsheet program.

This leads to the second challenge. An automated approach based on formal logic requires an amount of formal structure in the specification. Since it is undesirable to incur the costs of maintaining both a layman-readable specification and a formal specification, it is necessary to have a readable specification that contains enough structure for the purposes of test frame generation.

To be usable, the syntax of the specification language should impose as few restrictions on specification authors as possible. It may be necessary to produce test frames for a specification that is an arbitrary relation between any number of stimuli, conditions, and responses.

Information that allows traceability from requirements to tests will always be necessary for auditing purposes. Any automated approach will need to keep track of the source of the test frame components as the test frames are being constructed.

THE AUTOMATED PROCESS

Figure 1 illustrates the process used to automatically generate test frames from a requirements specification together with information about dependencies and a particular coverage scheme.

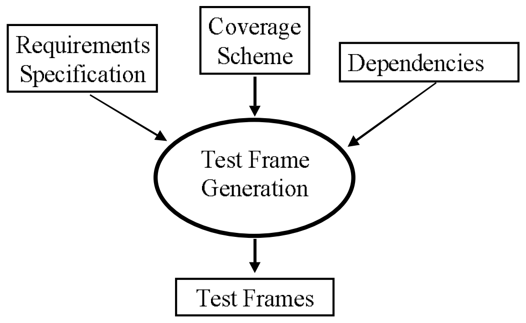

Figure 1. Automated Test Frame Generation

The specification language provides a concise syntax for denoting the logical relationships and alternatives within the requirements while also providing a natural language appeal. For example, the fragment,

```
Either the leading aircraft or the
trailing aircraft is supersonic.
```

may appear as part of the conditions in the requirements for the separation of two aircraft. The logical structure of this fragment is formally represented in our approach as,

```
{any of {the leading aircraft, the
trailing aircraft}} is supersonic.
```

The brackets impose a parsable structure on the requirements. The semantics of the language constructs, such as "any of", allows the test frame generator to calculate the logically equivalent expression, which in this case is:

```
{{the leading aircraft} is super-
sonic} or {{the trailing aircraft}
is supersonic}.
```

Once these constructs are expanded into their logical equivalents, test frames are produced according to the algorithm given in (Donat 1997). This algorithm uses well-defined rules of formal logic to calculate a set of feasible test frames that satisfies the coverage scheme. A variety of coverage criteria that govern test selection are supported. Two of these are discussed in the next section. Along with selection of the coverage scheme, the test frame generator also provides several other means by which processing can be controlled by the test engineers.

Traceability is achieved by attaching tags from the specification to the corresponding identifiers in the underlying logical expressions. In order to maintain the link between requirements and test frames, these tags are propagated by the rewrite system which performs the logical manipulations during test frame production.

COVERAGE CRITERIA

In this paper, the Disjunctive Normal Form (DNF) and Term Coverage schemes shall be discussed. The test frame generator groups test frames based on the particular stimulus/response behaviour, referred to as a test class, from which the test frames were generated.

DNF Coverage means that the disjunction of the stimulus and conditions portions of the test frames form a disjunctive normal form of the stimulus and conditions portion of the associated test class. Several test generation techniques have been based on a DNF approach (Dick and Faivre 1993, Weyuker et. al. 1994). In contrast, Term Coverage means only that each stimulus and condition in the test class is listed in at least one test frame. Term Coverage is similar to condition/decision coverage (Chilenski 1994).

Depending on the complexity of the specification, DNF and Term Coverage produce dramatically different test suite sizes. DNF Coverage is desirable to achieve, especially in safety critical portions of the system, due to the number of different combinations tested. However, in the worst case, test suites with DNF Coverage grow exponentially with the number of stimuli and conditions while growth is linear with Term Coverage.

RESULTS

The example reported in this paper is taken from a portion of the CAATS software requirements that refers to separation rules. The separation rules form a set of complex conditions under which certain responses occur. The specification of the separation rules is composed of several subsections dealing with different aspects of separation. The portion of the specification used in this example contains 177 requirements designated as testable requirements[2].

In our evaluation of this process we chose not to produce a test suite with DNF Coverage for this specification due to the large number of test frames that would have resulted. The specification refers to the separation rules in both a negative (the aircraft are not separated) and a positive (the aircraft are separated) context. This results in two corresponding test classes. The numbers of test frames constituting DNF Coverage are estimated to be approximately 1000 for the positive case and roughly 10^{24} for the negative case.

Test frames were generated using a Term Coverage scheme. This resulted in approximately 130 test frames for the positive case and approximately 230 test frames for the negative case.

Table 1 gives one of the test frames generated by our automated process. This test frame is only an example and was generated from a representation of only a portion of the CAATS software requirements that was used to evaluate the usefulness of this process. Any errors or omissions in this test frame are due to the way in which this portion was extracted by the authors.

[2] In addition to requirements that can be verified through testing, requirements specifications often contain requirements that cannot be verified through a test program and must be addressed by other means that are beyond the scope of this paper.

ROIDs	84672 224215 226547 226549 226550	
Stimulus	**Conditions**	**Responses**
{ACC operator} requests planned clearance	1. {planned clearance} exists for the flight 2. the source of the {planned clearance} is an aerodrome control tower with a tower method of operation of complex 3. the aircraft state is not AIRBORNE 4. {intruder} is using {altimeter setting} 5. {planned clearance} is using {altimeter setting} 6. the lowest altitude in the protected altitude band for {intruder} is at or below {FL 290} 7. the lowest altitude in the protected altitude band for {planned clearance} is at or below {FL 290} 8. the protected altitude band for {intruder} is vertically separated from the protected altitude band for {planned clearance} by {1000} feet or more 9. (NOT {planned clearance} is dumping fuel) 10. (NOT {intruder} is dumping fuel)	1. {ATA} shall commit {planned clearance}

Table 1. An Automatically Generated Test Frame

DISCUSSION

This paper has discussed the application of an automated test frame generation process to a real world system-level requirements specification. This demonstrates the effectiveness of the concepts and techniques discussed in (Toth et. al. 1996) and (Donat 1997). This process automatically produces test frames logically consistent with the specification. These test frames also satisfy a given coverage scheme. Since this process is based on a mathematical system the need for reviewing the correctness and completeness of the test frames is reduced. Should such a review be desired, tracing information is provided.

An important result of this work is that test frames are produced with a consistency of coverage that is difficult to achieve manually. In the manual process, different levels of experience within the test team may produce tests with uneven coverage. This lack of quality is eliminated by the test frame generator.

Beyond the test frame generation aspects of this automated process there are other possible benefits to be exploited:

1. Reviewing test frames can play a role in validation. Since each test frame is a theorem of the specification, the identification of a test frame specifying undesirable or unexpected behaviour implies an error in the specification. The trace information will aid the reviewer in determining the source of the error.

2. Making the test frame generator available to specification authors may provide benefits similar to those of integrated product teams. An important aspect of team integration ideas (Browning 1997) is an appreciation for how one's work is used by other individuals. By providing authors with the means of "compiling" their specification into test frames they will be able to review the results before handing their work to other individuals.

A problem that arose during the development of this example was the capacity of the test frame generator to construct intermediary results from which test frames are selected. It is expected that an increase in capacity would result in smaller test suites satisfying the same coverage criteria. For this example, this problem was overcome by iteratively expanding the detail of selected portions of the separation rule definitions and combining the results of each iteration.

It is important to note that the success of this example was due to the following essential qualities:

1. The consistency of the test frames, the assurance of proper coverage, and the accuracy of the tracing information are due to the mathematical underpinnings of the algorithms used.

2. The formal version of the software requirements fragment contained enough mathematical structure to facilitate test frame generation while still being readable.

3. Conditions were relatively independent, which allowed for a simple encoding of the existing condition dependencies.

The test frame generator does not yet include the capability of dealing with changes to the specification. This is the subject of future enhancements.

ACKNOWLEDGEMENTS

Our colleague, Kendra Cooper has also influenced aspects of the specification language. This work is supported by ***formal*WARE** (Joyce et. al. 1998), 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.

BIOGRAPHY

Michael R. Donat is completing his Ph.D. thesis in Computer Science at the University of British Columbia. His supervisor is Jeffrey J. Joyce.

Jeffrey J. Joyce was director of the ***formal*WARE** research project until its completion at the end of March 1998. He is currently employed by Raytheon Systems Canada Ltd.

REFERENCES

Browning, Tyson R., "Mechanisms for interteam integration: Findings from five case studies." In Lisa Hritz and Judith Peach, editors, *Systems Engineering: A Necessary Science, Proceedings of the Seventh Annual International Symposium of the International Council on Systems Engineering*, volume 1, August 1997, pages 649-656.

Chilenski, John Joseph and Newcomb, Philip H., "Formal specification tools for test coverage analysis." *KBSE'94 Knowledge-Based Software Engineering*, 1994, pages 59-68.

Dick, Jeremy and Faivre, Alain, "Automating the generation and sequencing of test cases from model-based specifications." In *Formal Methods Europe '93*, volume 670 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993, pages 268-284.

Donat, Michael R., "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*. Springer-Verlag, April 1997, pages 833-847.

Joyce, Jeffrey J., et. al., `http://www.cs.ubc.ca/formalWARE`.

Toth, K., Donat, M. R. and Joyce, J. J., "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/`.

Weyuker, E., Goradia, T., and Singh, A., "Automatically generating test data from a Boolean specification." *IEEE Transactions on Software Engineering*, 20(5):353-363, May 1994.