Tabular Expression-Based Testing Strategies: A Comparison

Xin Feng, David Lorge Parnas
Software Quality Research Laboratory
Faculty of Informatics and Electronics
The University of Limerick, Ireland
Xin.Feng@ul.ie, David.Parnas@ul.ie

T.H. Tse
Department of Computer Science
The University of Hong Kong
Pokfulam, Hong Kong
thtse@cs.hku.hk

Abstract

Tabular expressions were proposed as a documentation tool that can be used to document software precisely and unambiguously. This paper explores the applications of four testing strategies in tabular expression-based specifications and further compares the strategies on a mathematical basis.

1. Introduction

Tabular expressions [1] have proven to be very useful in industrial experiences with A-7, Nuclear Power Station, Bell Labs, Dell, and so on. However, little work has been done on how to generate test cases from the tabular expression-based specifications. Testers need guidelines in selecting effective testing strategies. While more than one testing strategies can be directly or indirectly applied to the specifications, what the testers care are "how to apply the strategies" and "which strategies are more effective".

2. Comparison

We select four testing strategies that can make use of the features of the tabular expressions. One Cell One Test Case (S^O) [2] is a testing method specifically proposed for the tabular expressions. Decision Table-Based Testing (S^D) [3] generates test data by describing the correspondences between actions and conditions of inputs. Basic Meaningful Impact Strategy (S^B) [4] and Fault-Based Testing (S^F) [5] are based on Boolean expression-based specifications. We have formulated four algorithms for applying the respective strategies to tabular expression-based specifications. We compare the relative effectiveness using mathematical theorems based on sets of abstract test case constraints obtained from the algorithms.

Table 1 shows the relative effectiveness of the four testing strategies. The symbols '▶', '▷', and '~' denote "unconditionally subsume", "conditionally subsume", and "incomparable", respectively. DSP and NDSP stand for specifications with and without duplicated expressions, respectively.

Table 1. Relative effectiveness

	S^O	S^D	S^B	S^F
S^O	=	-	-	-
S^D	▶(NDSP) or ~ (DSP)	=	-	-
S^B	▶(NDSP) or ~ (DSP)	▶	=	▷
S^F	▷(NDSP) or ~ (DSP)	▷	▷	=

3. Conclusions

In the four testing strategies, no one strategy is the strongest for all specifications. Testers need to select testing strategies based on the features of the specifications. The comparison helps testers to analyze the specifications and to select the best testing strategies. Moreover, researchers can use these results to improve current testing strategies.

References

[1] R. Janicki, D.L. Parnas, and J. Zucker, "Tabular Representations in Relational Documents", in *Software Fundamentals: Collected Papers by David L. Parnas*, Chapter 4, Ed. D. M. Hoffman, D.M. Weiss, Addison-Wesley, 2001, pp. 71-85.

[2] S. Liu, "Generating Test Cases from Software Documentation", *M.Sc. Thesis*, McMaster Univ., 2001.

[3] P.C. Jorgensen, *Software Testing: a Craftsman's Approach, 2nd edition*, CRC Press: Boca Raton, 2002.

[4] E.J. Weyuker, T. Goradia, and A. Singh, "Automatically Generating Test Data from a Boolean Specification", *IEEE Trans. Softw. Eng.*, 20(5), 1994, pp. 353-363.

[5] M.F. Lau and Y.T. Yu, "An Extended Fault Class Hierarchy for Specification-Based Testing", *ACM Trans. Softw. Eng. Method.*, 14(3), 2005, pp. 247-276.

0-7695-2984-4/07 $25.00 © 2007 IEEE
DOI 10.1109/TAIC.PART.2007.19