IMPROVEMENTS TO THE TRACE ASSERTION METHOD FOR SOFTWARE ENGINEERING

By

ANDREW C. P. DOOKHAN, B.Sc.Hon.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Master of Engineering

McMaster University

MASTER OF ENGINEERING (1999) McMASTER UNIVERSITY
(Computer Engineering) Hamilton, Ontario, Canada

TITLE: Improvements to the Trace Assertion Method for Software Engineering

AUTHOR: Andrew C. P. Dookhan, B.Sc.Hons. (Queen's University)

SUPERVISORS: Dr. David Lorge Parnas

NUMBER OF PAGES: xvii, 214

Abstract

Many formal methods have been suggested for documenting software work assignments, some are discussed in [19]. This thesis presents a new version of **T**race **A**ssertion **M**ethod [1,10]. We refer to this new version as ITAM. TAM originated in 1977, [1,2]. Although TAM is sound, TAM specifications seem inappropriately complex. The complexity of a specification is not proportional to the complexity of the concepts being described. In this thesis we reintroduce TAM to document software work assignments, (i.e. module interface specifications).

This thesis suggests improvements to TAM (ITAM) for programmers and specification writers. In doing this we :–

(a) allow specification method writers to represent the state of an object, using sets.

(b) view the collection of objects implemented by one module, as a single object (refer to as a *primary* object,) which may be composed of other objects.

(c) presents objects' state representation semantics and a syntax that is more in tune with programmers' perceptions.

(d) provide a tool that checks the syntax describing the canonical representation of an object's state.

(e) provide commonly used predefined auxiliary functions to reduce redundant mathematical notation.

(f) provide predefined auxiliary functions to be used when it is necessary to "check type" and "check availability" of an object, before performing operations on an object.

(g) provide a method of abbreviating, long and duplicate, mathematical expressions.

(h) use tables which can be easily checked for completeness and consistency, when we

describe state changes of a module's objects, caused by invocations of that module's programs, (*operation tables*).

(i) show that it is possible for ITAM to document polymorphic and non–deterministic module specifications.

(j) provide a common compact and systematic document format for ITAM's module interface specifications.

Acknowledgements

I would like to express my sincere thanks to Dr. David L. Parnas for the guidance, freedom of thought and the tremendous support he has given me throughout this thesis.

I would also like to thank Dr. Mark Lawford and Dr. Ryszard Janicki for their helpful comments, suggestion and solutions after reading the thesis, Dr. Martin von Mohrenschildt for his comments and help with the syntax checking tool, Dr. Emil Sekerinsksi and Dr. Ridha Khedri for their mathematical input, Dennis Peters for our long discussions, Cui Feng, Li Puili for being at the pre–seminar rehearsal, and the other graduate students of McMaster Software Engineering Research Group for their comments and discussions.

My appreciation goes to the staff members Doris Burns, Cheryl Gies for their help with some administrative matters during my work on this thesis.

I would like to express my deepest appreciation to Linda Cox, Ron Dolson, Dr. Irene Siotis and Dr. David Parnas, an invaluable team, who provided a special kind of support that helped me to maintain my focus on this thesis. Special thanks to Jane Schouten for her moral support.

Finally, I would like to acknowledge the Natural Sciences and Engineering Research Council of Canada, Bell Canada, the Telecommunications Research Institute of Ontario (TRIO/CITO) and DEC for the use of resources that they provided.

I would like to dedicate this thesis to my dearest sisters Bernice, Jennifer and her two children Bendita and Antoine, my mother Ruby and my brother Trevor.

Table of Contents

List of Figures

List of Tables