Ain Shams University
Faculty of Science
Mathematics Dept.

Software Verification

~~004~~ 17-5h

A Thesis
Submitted to the Department of Mathematics
in Computer Science

In Partial Fulfilments for the Degree of
Master of Science

By
Hala Shawky Abd El Gawad

51776

Supervised By

Dr. **Moustafa Samy Mahmoud**
Ass. Prof. of Comp. Sc.
Suez Canal University

Dr. **Nashat Fared Mahomed**
Mathematics Dept. Faculty of Science
Ain Shams University

Cairo, 1994

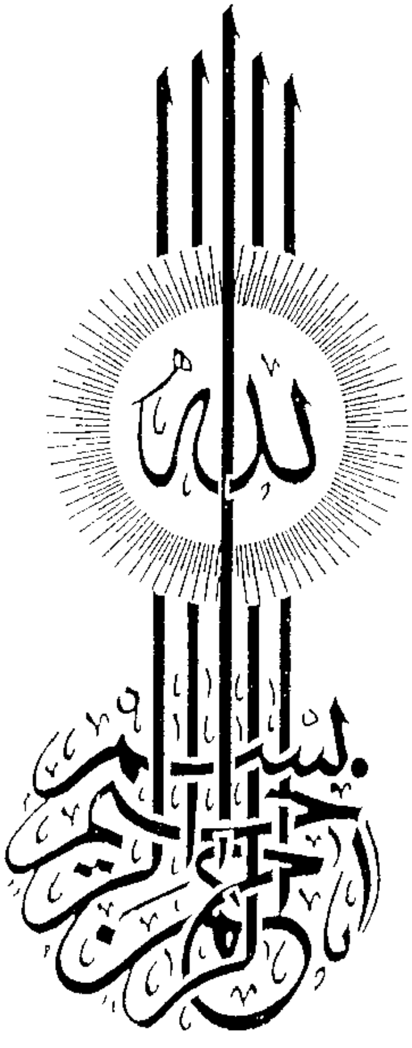
software verification

****

thesis advisors	Approval
Dr.Moustafa samy mahmoud **Ass. Prof. of Comp.SC.** **Sues Canal University**	([illegible])
Dr. Nashat Fared Mahomed **Ass.Prof. in mathematics Dept.** **Faculty of science** **Ain Shams University**	(N. Farid)

ا.د انتصارات محمد حسن الشبكي

رئيس قسم الرياضيات
كلية العلوم جامعة عين شمس



Central Library - Ain Shams University

Acknowedgement

ACKNOWLEDGEMENT

I would like to express my deep gratitude to Dr. Moustafa Samy Associated Prof. Mathematics Departement, Faculty of Science , Kanal El swes Uinveristy, for suggesting the problem, valuable guidance through his supervision of this work and for the following up every aspect of the thesis.

I much indebted to Dr. Nashat Fared , Associated Prof. , Mathematics Departement, Faculty of Science , Ain Shams University, for his continuous guidance and the following up every aspect of the thesis

Contents

CONTENTS

Introduction 1

Chapter I : Software System Development Life Cycle

1.1 Software Crisis 3

1.1.1 Software Maintenance 3

1.1.2 Problems 5

1.2 Software Engineering 7

1.3 Planning phase 8

1.4 Requirement Phase 9

1.4.1 S.R Format 11

1.4.2 Structured Languages for Requirements Definition 13

1.4.2.1 RSL\REVS 13

1.4.2.2 ADA Programming Language 14

1.5 Software Specification Phase (S.S.) 15

1.5.1 Interface Specification 17

1.5.2 Operational Specification 17

1.5.3 Data Specification 18

1.5.4 Formal Specification Techniques 18

1.5.4.1 Formal Notations 19

1.6 Software Design Phase 25

1.6.1 Design Methodologies 28

1.7 Implementation Phase 29

1.7.1 Programming Methodology 29

1.7.2 Information Hiding 31

1.7.3 Programming Style 31

1.7.3.1 Program Name 31

1.7.3.2 Program Control Constructs 32

1.7.3.3 Program Layout 32

1.7.4 Programming Environments 32

1.7.5 Programming Portability 33

1.7.6 Machine Architecture Dependencies 33

1.7.7 Operating System Dependencies 33

ChapterII :Software Verification and Validation

2.1 Quality Assurance Activity 35

2.2 Definitions 36

2.2.1 Software Verification 36

2.2.2 Software Validation 36

2.3 Verification and Validation Techniques 39

2.3.1 Inspection 39

2.3.1.1 The Advantage of The Inspection 39

2.3.2 Walk Through 40

2.3.2.1 Advantage of Walk Through 40

2.3.3 Reviews 40

2.4 Software Testing 41

2.4.1 Integration Testing Philosophies 41

2.3.2 Static Analysis 43

2.5 Formal Verification Techniques 44

2.5.1 Input/Output asserations 44

2.5.2 Weakest Precondation 44

2.5.3 Structure Induction 45

2.6 Details of each Major Verification and Validation Phase 45

2.6.1 Requirements Verification 45

2.6.2 Design Verification 46

2.6.3 Code Verification 46

2.7 Verification and Validation as management Support 46

2.8 Definition of Effective Verification and Validation 47

2.9 How to Select Appropriat Verification and Validation Tools 48

Chapter III : Software Modelling Tools

3.1 Flow Chart 50

3.2 SADT 51

3.2.1 How to Validate SADT Model 54

3.3 Data Flow Diagram (DFD) 55

3.3.1 How to Validate DFD Model 58

3.4 Petri Net 59

3.4.1 Fundamental Principles of Petri Nets 59

3.4.2 A formal Structure of petri net 60

3.4.3 Properties of Petri Net 64

3.4.3.1 Safeness 65

3.4.3.2 Boundedness 65

3.4.3.3 Conservation 65

3.4.3.4 Liveness 65

3.4.3.5 Deadlock 66

3.4.3.6 Reachability Problem 66

3.4.3.7 Coverability Problem 66

3.5 Modelling of Conventional Problem in Complex Systems 68

3.5.1 Deadlock 69

3.5.1.1 Deadlock Avoidance 70

3.5.2 Concurrent Process 71

3.5.2.1 Fork and Join Constructs 71

3.5.2.2 Parbegin/Parend 72

3.5.2 Synchronization 73

3.5.2.1 Mutual Exclusion 73

3.5.2.2. Producer/Consumer 74

Chapter IV : Software System Modelling Using High-Level Petri net

4.1 High-level Petri net 77

4.2 Colored Petri net 78

4.2.1 Multi-sets and expressions 80

4.2.2 Variables and Expression 81

4.2.3 Formal Definition of CP-Net 82

4.2.4 Informal execution of CPN 83

4.2.5 Formal behavior of CPN 84

4.2.6 Firing rule of CPN 86

4.3 Analysis techniques for CPN 87

4.3.1 Simulation 87

4.3.2 Reachability tree 87

4.3.2.1 Covering marking 87

4.3.2.2.Duplication marking 88

4.3.2.3.Equivalent marking 88

4.3.3.Reduction 89

4.4 why CPN is approprate system modelling tool 89

4.4.1 Hierarchical CPN 90

4.4.1.1 Sbstitution of transitions 92

4.4.1.1.1 Semantic of substitution transition 93

4.4.1.2 Substitution places 93

4.4.1.2.1 Semantic of substitution places 93

4.4.1.3 Invocation transition 94

4.4.1.3.1 Semantcs of invocation transition 94

4.4.1.4 Fusion of places 95

4.4.1.5 Fusion of transition 96

Chapter V RElation between CPN and Abstract data types

5.1 Fundamentals of Algebraic Specification 98

5.1.1 Signature 99

5.1.2 Algebra 99

5.1.3 SIG-Algebra 100

5.1.4 Variables and Terms 100

5.1.5 Terms of a signature with variables 100

5.1.6 Equations of terms 101

5.1.7 Multi sets of terms 102

5.1.8 Equation and validity 103

5.1.9 Specification and SPEC-Algebra 103

5.1.10 Specification of abstract data type 104

5.2 Algebraic form of CPN 107

5.3 Abstract CPN 110

Chapter VI Simple Integration Design and Implementation of a Verification System

6.1 Advantages of Object-Oriented Approach 115

6.2 An Integrated Design Methodology 117

6.3 Idea Behind 119

6.4 Model Description 121

6.5 Implementation 125

Conclusion 127

Appendex 128

References 136

Introduction

Introduction

The software development process can realize a successful result if a suitable judgment on the successive steps is applied. This is in fact not an easy task since, the different development steps have a set of various factors affecting its realization. Sometimes these factors are even contradicting the verification of the software model which assists to great extent in detecting and overcoming the most important problems before the implemented system is put into work. The main objective of the thesis is to present the development process from software engineering point of view and to focus on the tools applied for the software verification. The formal tools are more interesting due to their great power and their ability to express the model systems in a mathematical form. One of the most important tools is the high-level Petri net. This thesis is organized as follows :

In chapter (I) we shall discuss software crisis, introducing different definitions of the term software engineering, and illustrate the software system development life cycle and the activity of each phase of this life cycle . **Chapter (II** starts by giving an overview of the quality assurance activity, concerned with verification and validation activity, discusses its fundamental definitions, and its techniques and tools.

The aim of chapter (III) is to discuss different modelling tools classified from a low structured primitive tool (flow chart) to highly structured modelling tool (Petri nets), and illustrates its modelling and decision power. **In chapter (IV)** the basic structure of high level Petri net, especially colored Petri net will be introduced . **In chapter (V)**, an algebric form of colored Petri net will be introduced to combine the strengths of abstract data type for data representation with the strenths of colored Petri net (synchronization, concurrence, graphic representation) in the same algebraic framework. **In chapter (VI)** the design and the implementation of a simple integrated tool, realized in C^{++}, is introduced. This tool is a real adoption of the concepts concerned by this research. A relation between the structure and the processing of a colored Petri net-based model and the concept of object-oriented language is discussed . This work is followed by two appendices, the first one introduces the relation between colored Petri Net and abstract data types and the second appendix contains the main code of C^{++} package . Finally the list of the used references is attached at the end of this thesis.

Chapter I Software System Development Life Cycle