

Contents

List of Figures	xv
List of Tables	xvii
1 Introduction	1
2 Motivation	5
2.1 Background	5
2.2 Problem Description	12
2.3 Goals	21
2.4 Synopsis	23
3 Requirements for Automated Failure Detection	25
3.1 Failure Description and Classification	26
3.1.1 Purpose	26
3.1.2 Scope	26
3.2 Automated Detection	27
3.2.1 Contextual Requirements	28
3.2.2 Data Collection Requirements	29
3.2.3 Data Analysis Requirements	31
3.2.4 Failure Visualisation Requirements	32
3.3 Summary	33
4 Concurrent Programming in Java	35
4.1 Thread Principles	36
4.2 Thread Lifecycle	42
4.3 Unsynchrosised Interaction	49
4.4 Synchronised Interaction	51
4.5 Summary	60

5	Liveness Failures and Potentials in Java	61
5.1	Terminology	62
5.1.1	Error, Mistake, Fault, and Failure	62
5.1.2	Potential for Failure	64
5.1.3	Failure and Symptom	67
5.2	Liveness Failures	68
5.2.1	Liveness and Safety	68
5.2.2	Concurrent Java Liveness Failures	70
5.2.3	Potentials for Java Liveness Failures	82
5.2.4	Classification of Java Liveness Failures and Potentials	84
5.2.5	Failures in Concurrent Application Logic	88
5.3	Summary	88
6	A Model of Thread Synchronisation	89
6.1	Model Requirements	90
6.1.1	Concurrent Liveness Failure Characteristics	90
6.1.2	Dynamics of Thread Synchronisation	91
6.1.3	Control Flow States	93
6.1.4	Summary of Model Requirements	95
6.2	Related Work	95
6.2.1	The Java Language Specification	96
6.2.2	Formal Java Semantics	97
6.3	A Statechart Model for Thread Synchronisation	98
6.3.1	The UML Statechart Approach	99
6.3.2	Classification of Thread Lifecycle Methods	102
6.3.3	Principles for Designing States and Transitions	104
6.3.4	The Resulting Statechart	111
6.3.5	System State and History	119
6.4	Formalising Liveness Failures and Potentials	122
6.4.1	Formal Description of Failures	122
6.4.2	Formal Description of Potentials	124
6.5	Summary	126
7	Trace-based Data Collection	129
7.1	Tracing Requirements	130
7.1.1	Format, Schema, and Encoding	131
7.1.2	Trace Generation	133
7.2	Related Work	135
7.2.1	Basic Techniques	135
7.2.2	Trace Formats	139
7.2.3	Code Instrumentation	139

7.2.4	Runtime APIs	140
7.2.5	Debuggers	143
7.2.6	Trace-based Tools	145
7.2.7	Comparison	146
7.3	JAVIS-Tracer for Concurrent Java Programs	148
7.3.1	Trace Format	148
7.3.2	Trace File Generation	150
7.3.3	Tracer Architecture	153
7.4	Summary	155
8	Trace-based Failure and Potential Analysis	157
8.1	Analysis Requirements	157
8.1.1	Functional Requirements	157
8.1.2	Time and Space Complexity	159
8.2	Related Work	159
8.2.1	Deadlock Detection	159
8.2.2	Deadlock Potential Detection	160
8.2.3	Failures and Potentials Involving <code>wait()</code> or <code>join()</code>	162
8.3	JAVIS-Algorithms	162
8.3.1	Cycle Detection	162
8.3.2	Missed Notification	165
8.3.3	Potentials for Cyclic Dependencies	166
8.4	Summary	166
9	Trace-based Failure and Potential Visualisation	167
9.1	Visualisation Requirements	167
9.1.1	Trace Visualisation	168
9.1.2	Failure and Potential Visualisation	170
9.1.3	Visualisation Environment	171
9.2	Related Work	171
9.2.1	Visualisation of Concurrent Programs	172
9.2.2	Object-Oriented Visualisation	173
9.2.3	UML-based Visualisation	177
9.2.4	Comparison	180
9.3	JAVIS-Visualisation	181
9.3.1	UML Profile for Java Traces, Failures, and Potentials	181
9.3.2	Visualisation Architecture	189
9.4	Summary	189

10 Using the JAVIS Prototypes	191
10.1 Example for Automated Failure Detection	191
10.1.1 The Banking Example Revisited	193
10.1.2 Tracing	194
10.1.3 Automated Deadlock Detection	195
10.1.4 Trace and Deadlock Visualisation	195
10.2 Example for General Purpose Tracing	199
10.2.1 A Simulation Software Example	199
10.2.2 Tracing and Visualising JEVOX	201
10.3 Summary	201
11 Conclusion	203
11.1 Contributions	203
11.2 Evaluation	204
12 Outlook	209
12.1 Remaining Issues	209
12.2 Evolution	210
12.3 Related Domains	211
Bibliography	213
Index	223

List of Figures

2.1	Banking Example	8
2.2	Banking Example with Deadlock	9
2.3	Cyclic Resource Dependency and Wait-For Graph	11
2.4	Source Code View	14
2.5	Call Stack View	15
2.6	Monitor View	16
2.7	Monitor View with Callstacks	17
2.8	Trace Example	19
2.9	Thesis Goals	22
3.1	Requirements for Automated Failure Detection	27
3.2	First Refinement of Goals	33
4.1	Banking Simulation Classes	37
4.2	Unsynchronised Access of Accounts	40
4.3	Lost Update	41
4.4	Deriving New Threads	43
5.1	Error Terminology	63
5.2	Extending the Basic Terminology with Potential	64
5.3	Potential for Deadlock	65
5.4	Deriving Potential Conditions from Failure Conditions	66
5.5	Safety and Liveness Terminology	69
5.6	Deadlock	71
5.7	Missed Notification	73
5.8	Balancing <code>wait()</code> and <code>notify()</code>	74
5.9	Nested Monitor Lockout	76
5.10	Circular Join	77
5.11	Self Join	77
5.12	Join-induced Deadlock	78
5.13	Livelock	79

6.1	Failures and Potentials as Legal System Behaviour	92
6.2	Thread Lifecycle Statechart	113
6.3	Thread Lifecycle Statechart with Guards and Actions	115
6.4	Object Synchronisation Behaviour Statechart	116
7.1	Requirements for Data Collection	130
7.2	Levels of Instrumentation	137
7.3	The Java Platform Debugger Architecture (JPDA)	142
7.4	Trace Format Example	149
7.5	Class Diagram of the Tracer	154
8.1	Requirements for Data Analysis	158
8.2	Cycle in Graph Structure	163
9.1	Requirements for Data Visualisation	168
9.2	GThread History View	172
9.3	GThread Mutex View	173
9.4	Jinsight Execution View	174
9.5	Jinsight Execution View - Zoom	175
9.6	Jinsight Reference Pattern	176
9.7	Tagged Values for Messages in Interaction Diagrams	185
9.8	Graphical Notation for Stereotypes	186
9.9	Cyclic Failures using Stereotypes	188
9.10	Class Diagram of the Together Visualisation	190
10.1	Banking Application Class Diagram	192
10.2	Sequence Diagram Generated by Together	192
10.3	Trace Generation Dialogue	193
10.4	Part of the Trace	194
10.5	Importing a Trace in Together	195
10.6	Sequence Diagram of a Trace	196
10.7	Collaboration Diagram with Involved Methods	198
10.8	Collaboration Diagram with Deadlock	199
10.9	Complete Trace Visualisation in Together	200
10.10	Part of the Remote Trace	202

List of Tables

4.1	Signature of <code>sleep()</code>	45
4.2	Signature of Interrupt Methods	51
4.3	Signature of <code>join()</code>	52
4.4	Keyword <code>synchronized</code>	53
4.5	Signature of <code>wait()</code> and <code>notify/All()</code>	56
4.6	Java Concurrency Concepts	59
5.1	Liveness Failures	86
5.2	Potential Liveness Failures	87
6.1	Mapping Java Behaviour to Statechart Events	109
7.1	Trace Collection Approaches	147
9.1	UML Profile for Concurrent Java Traces: Tags	184
9.2	UML Profile for Concurrent Java Traces: Stereotypes	187