

Lecture Notes in Artificial Intelligence 1994

Subseries of Lecture Notes in Computer Science

Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science

Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Singapore

Tokyo

Jürgen Lind

Iterative Software Engineering for Multiagent Systems

The MASSIVE Method



Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Author

Jürgen Lind
iteratec GmbH
Inselkammerstr. 4, 82008 Unterhaching, Germany
E-mail: Juergen.Lind@iteratec.de

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Lind, Jürgen:

Iterative software engineering for multiagent systems : the MASSIVE
method / Jürgen Lind. - Berlin ; Heidelberg ; New York ; Barcelona ;
Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 2001
(Lecture notes in computer science ; 1994 : Lecture notes in
artificial intelligence)
ISBN 3-540-42166-1

CR Subject Classification (1998): I.2.11, D.2, I.2, C.2.4, D.1

ISBN 3-540-42166-1 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springer.de>

© Springer-Verlag Berlin Heidelberg 2001
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN 10782125 06/3142 5 4 3 2 1 0

Foreword

Agent-based techniques are beginning to be used to develop a wide range of commercial and industrial applications. This take up is occurring because the agent-based approach offers a natural and powerful means of conceptualising, designing and building complex, distributed systems. The key conceptual components from which this new approach to software engineering derives its power are: (i) the autonomous components (agents) that can achieve their objectives in flexible ways; (ii) the high-level interactions (e.g., cooperation, coordination and negotiation) in which these agents can engage; and (iii) the organisational structures (e.g., teams, coalitions and various forms of hierarchy) into which the agents can arrange themselves. When taken together, the agents represent the application's basic units of computation, the interactions represent the inter-connections between these units and the organisational structures define the way the components relate to one another.

Although the agent-based approach appears to offer a promising new paradigm for building complex distributed systems, to date, the majority of the agent-based applications that have been developed have been built by researchers who specialise in agent-based computing. However, if agent-based computing is to become anything more than a niche technology practised by the few, then the base of people who can successfully use the approach needs to be broadened. A crucial step in this broadening endeavour is to find mechanisms by which professional software engineers can gain access to the philosophy, the concepts and the methods of agent-based computing without having to immerse themselves in the research community. Perhaps the key mechanism for achieving this is to develop methodologies for agent-oriented software engineering. Such methodologies should assist developers in the analysis, design and development of their application; particularly, they should identify the key steps that are involved, the key models that need to be built at the various steps, and how the different models and stages relate to one another.

Against this background, this book, *The MASSIVE Method: Software Engineering for Multiagent Systems*, presents one of the first coherent attempts to develop such a methodology for a broad class of agent-based systems. In particular, it provides a clear introduction to the key issues in the field of agent-oriented software engineering and provides a comprehensive overview

of the state of the art. It then describes and illustrates the application of the MASSIVE methodology to a number of real-world applications. When taken together, these components make the book an important contribution to the fledgling field of agent-oriented software engineering and, as such, essential reading for both researchers and practitioners alike.

August 2000

Nick Jennings

Contents

Foreword	V
List of Figures	XIV
List of Process Models	XV
1. Introduction	1
2. Agents, Multiagent Systems and Software Engineering	9
2.1 Intelligent Agents	9
2.1.1 What's an Agent, anyway?	9
2.1.2 Roles	12
2.1.3 Architectures	13
2.1.4 Agents, Roles and Architectures	14
2.2 Systems of Agents	15
2.2.1 Interaction	16
2.2.2 The Social Dimension	17
2.3 Related Fields in Computer Science	19
2.4 Agent-Oriented Software Engineering	21
2.4.1 Aspects of Programming Paradigms	21
2.4.2 A Historic Perspective	28
2.4.3 The Bottom Line	30
2.4.4 Where Next?	32
2.5 Summary	33
3. Basic Concepts in Software Engineering	35
3.1 Cognitive Aspects of Software Engineering	35
3.1.1 Basic Human Information Processing	36
3.1.2 Software Engineering as a General Design Task	38
3.1.3 Designs and Models	40
3.1.4 A General Model of Engineering	41
3.1.5 The Basic Engineering Cycle	43
3.1.6 Basic Skills in Software Engineering	46
3.2 Requirements for Software Engineering Support	50

3.3	A General Model of Software Engineering	51
3.4	Software Engineering Product Models	53
3.4.1	A Generic Product Model	54
3.4.2	Software Blueprints: The Unified Modeling Language .	55
3.5	Software Engineering Process Models	57
3.5.1	Classical Process Models	58
3.5.2	Novel Trends in Software Engineering	67
3.5.3	Development Methods for Multiagent Systems	78
3.5.4	Discussion	91
3.6	Quality Management and Systematic Learning	91
3.6.1	The Quality Improvement Paradigm	92
3.6.2	Experience Factory	92
3.7	Summary	95
4.	The Conceptual Framework of MASSIVE	97
4.1	The Foundations of MASSIVE	97
4.2	Knowbbles	99
4.3	Views	101
4.3.1	What and Why?	102
4.3.2	View-Oriented Analysis	106
4.3.3	A View System for Multiagent Systems	108
4.4	Iterative View Engineering.....	114
4.5	Putting It All Together	117
4.6	Summary	120
5.	MASSIVE Views	121
5.1	A Brief Introduction to Train Coupling- and Sharing (TCS) .	122
5.2	Environment View	125
5.2.1	Developers Perspective	125
5.2.2	Systems Perspective	129
5.3	Task View	130
5.3.1	Use Case Analysis	130
5.3.2	Functional Requirements	131
5.3.3	Nonfunctional Requirements	138
5.4	Role View	138
5.4.1	Role Definition	139
5.4.2	Role Assignment	144
5.5	Interaction View	144
5.5.1	Intent Layer	145
5.5.2	Protocol Layer	148
5.5.3	Transport Layer	165
5.6	Society View	167
5.6.1	Characterization of Social Systems	167
5.6.2	Designing Social Systems	169
5.7	Architecture View	174

5.7.1	System Architecture	175
5.7.2	The Architectural Feature Space	177
5.7.3	Agent Architecture	184
5.8	System View	190
5.8.1	User Interface Design	190
5.8.2	Exception Handling	194
5.8.3	Performance Engineering	196
5.8.4	Deployment	201
5.9	Summary	203
6.	Further Case Studies	205
6.1	The TEAMWORK LIBRARY	205
6.1.1	Environment View	205
6.1.2	Task View	206
6.1.3	Role View	208
6.1.4	Interaction View	210
6.1.5	Society View	214
6.1.6	Architecture View	214
6.1.7	System View	217
6.2	Personal Travel Assistant: Intermodal Route Planning	219
6.2.1	Environment View	220
6.2.2	Task View	222
6.2.3	Role View	226
6.2.4	Interaction View	228
6.2.5	Society View	231
6.2.6	Architecture View	231
6.2.7	System View	234
6.3	Summary	241
7.	Conclusion	243
A.	Toolkits for Agent-Based Applications	247
A.1	SIF	247
A.2	ZEUS	251
A.3	Swarm	253
A.4	Summary	254
B.	Basic Problem Solving Capabilities of TCS Agents	255
B.1	Planing Algorithm for a Single Task	255
B.2	Plan Integration Operator	256
B.3	Decision Functions	259
B.4	Plan Execution Simulation	259
C.	Protoz Specification of the Contract-Net Protocol	261

Bibliography	265
Glossary	281
Index	283

List of Figures

2.1	Perceive-Reason-Act Cycle.....	10
2.2	A Single Agent.....	10
2.3	A Container Stock Area.....	11
2.4	Agents and Roles.....	14
2.5	A Generic Agent Application Architecture.....	15
2.6	Sociogram.....	18
2.7	MAS-Related fields in Computer Science.....	20
2.8	Levels of Abstraction.....	22
2.9	Programming Concepts.....	29
3.1	Memory Model.....	37
3.2	Representations.....	40
3.3	Design and Model.....	40
3.4	Cognitive model of Software Engineering.....	41
3.5	The Basic Engineering Cycle.....	45
3.6	General Design Tree.....	46
3.7	Design Trees as Projections.....	47
3.8	A General Model of Software Engineering.....	52
3.9	The Structure of an Ideal Development Method.....	52
3.10	A Generic Product Model.....	55
3.11	The Ideal Software Development Process.....	57
3.12	Waterfall Model.....	59
3.13	Iterative Enhancement.....	60
3.14	Product-States.....	61
3.15	The V-Model.....	62
3.16	The Cleanroom Model.....	63
3.17	Classes of Prototypes.....	65
3.18	The Spiral Model.....	67
3.19	Round-trip Engineering.....	68
3.20	The DSDM Method.....	70
3.21	Booch's Macro Process.....	71
3.22	Booch's Micro Process.....	72
3.23	Cost-of-Change Curves.....	73
3.24	The Generic DESIRE Agent Architecture.....	84
3.25	Hierarchical Set of Models.....	86

3.26	Model Relations in Gaia	89
3.27	The Quality Improvement Paradigm	93
3.28	The Experience Factory	94
4.1	Balanced Approach	98
4.2	Connections between Design and Implementation	100
4.3	Knowbble refinement	101
4.4	A Generic View System	102
4.5	Aspect Weaver	103
4.6	Views	104
4.7	Knowbbles and Views	105
4.8	Shared Knowbbles and Knowbble Refinement	106
4.9	Example for a Generic View Analysis	107
4.10	Coverage of MAS Models	108
4.11	MASSIVE Views	110
4.12	A View System Tree	113
4.13	Models of Iteration	114
4.14	Micro Processes	115
4.15	Iterative View engineering	116
4.16	The MASSIVE Method	117
4.17	The MASSIVE Method (UML)	118
5.1	Views and Multiagent Systems	122
5.2	Hierarchical freight haulage	123
5.3	Train Coupling and Sharing (TCS)	124
5.4	Perspectives on the Environment	125
5.5	Use case example from the TCS domain	131
5.6	Example for a Task Tree	133
5.7	TCS Simulator Workflow (UML)	134
5.8	Example railroad network	135
5.9	Task Tree of the TCS Domain	136
5.10	Hybrid Role Identification	140
5.11	Layers of Abstraction in Interaction Design	145
5.12	Task Decomposition	145
5.13	Centralized Market	146
5.14	Distributed Market	147
5.15	Structural Elements of UML Activity Diagrams	151
5.16	Synchronization Point	152
5.17	Extended Activity Diagram	153
5.18	Defining Macros	154
5.19	English Auction	155
5.20	Contract-Net Protocol	157
5.21	Contract-Net Protocol (UML)	158
5.22	Simulated Trading	159
5.23	Simulated Trading (UML)	160

5.24	Contract-Net Manager	164
5.25	Contract-Net Bidder	164
5.26	Information Exchange	166
5.27	Generic Blackboard Architecture	166
5.28	Example for a hierarchical society	168
5.29	Social System Performance	170
5.30	Clustering	172
5.31	Generic Software Architectures	175
5.32	Architectural Design Space	178
5.33	System Architecture	179
5.34	System Architecture (UML)	179
5.35	Message Passing in TCS/MAS	181
5.36	Message Passing in TCS/MAS	182
5.37	InteRRaP	187
5.38	Union Agents	188
5.39	Union Agents (UML)	189
5.40	Elements of the TCS/MAS GUI	193
5.41	The Performance Engineering Process	196
5.42	The Performance Improvement Cycle	196
5.43	Deployment Life cycle	201
5.44	The MASSIVE View System	204
6.1	Team Meeting	211
6.2	Broadcast Protocol	213
6.3	Team Structure	214
6.4	Teamwork Agent Architecture	216
6.5	Broadcast Statistic Tool	218
6.6	IMRP Example	221
6.7	IMRP Broker Architecture	229
6.8	Distributed IMRP Architecture	229
6.9	Basic Society Structure	231
6.10	Hierarchical IMRP Organization	232
6.11	MECCA Agent Architecture	234
6.12	IMRP Graphical User Interface	234
6.13	Generic Selection Function	237
6.14	Example 1	239
6.15	Example 2	239
6.16	Example 3	240
7.1	MADS View System Selection Process	245
7.2	MADS Supported View Construction	245
A.1	The EMS Idea	248
A.2	The EMS Model	248
A.3	Conceptual Agent Model of SIF	249

XIV List of Figures

A.4	Information and Control Flow in SIF	249
A.5	SIF User Interaction	250
A.6	ZEUS Agent Architecture	251
A.7	ZEUS Screen shot	252
A.8	Swarm Hierarchy	253
B.1	Time Window Propagation	256
B.2	Planer (UML)	256
B.3	Action Scheduling	258

Our ability to imagine complex applications will always exceed our ability to develop them.

Grady Booch

List of Process Models

1	The Basic Engineering Cycle (BEC)	44
2	Cleanroom	64
3	Burmeister	81
4	Kinny and Georgeff	82
5	DESIRE	84
6	MAS-CommonKADS	88
7	Gaia	90
8	Quality Improvement Paradigm	93
9	View-oriented Analysis	107
10	MASSIVE	119
11	Use Case Analysis	131
12	MASSIVE Role Modeling	141
13	Protoz Protocol Design	163
14	MASSIVE Society Design	171
15	MASSIVE User Interface Design	192
16	Performance Engineering	197
17	System Deployment	203

Our ability to imagine complex applications will always exceed our ability to develop them.

Grady Booch