Come, Let's Play

Springer-Verlag Berlin Heidelberg GmbH

David Harel Rami Marelly

Come, Let's Play:

Scenario-Based Programming Using LSCs and the Play-Engine

With 185 Figures and CD-ROM

Springer

David Harel
Rami Marelly

The Weizmann Institute of Science
Faculty of Mathematics and Computer Science
Rehovot 76100, Israel

Library of Congress Cataloging-in-Publication Data
Harel, David, 1950–
Come, let's play: scenario-based programming using LSCs and the play-engine/
David Harel, Rami Marelly.
p.cm.
ISBN 978-3-642-62416-2 ISBN 978-3-642-19029-2 (eBook)
DOI 10.1007/978-3-642-19029-2
1. Software engineering. 2. System design. 3. Object-oriented programming (Computer science) 4. Visual programming languages (Computer science) I. Marelly, Rami, 1967– II. Title
QA76.758.H365 2003
005.1–dc21 2003045546

ACM Computing Classification (1998): D.2, C.2.0, B.4.0, D.0, D.1.5, D.3, I.6.5, I.6.8

http://www.springer.de

Originally published by Springer-Verlag Berlin Heidelberg New York in 2003

Cover design: KünkelLopka, Heidelberg
Typesetting: Camera-ready by authors
Printed on acid-free paper 45/3142 GF– 543210

For my precious children, Sarit, Hadas, Efrat, Yair and Tamar,
with whom playing has always been so much fun.
And for my beloved wife Michal,
soul-mate, playmate, teammate and partner.

(D.H.)

For my dear parents, Terry and Aaron Marelly,
who taught me how to play.
And for my beloved wife and daughters, Tali, Noa and Adi,
with whom I enjoy playing each and every day.

(R.M.)

Preface

This book does not tell a story. Instead, it is *about* stories. Or rather, in technical terms, it is about scenarios. Scenarios of system behavior. It concentrates on **reactive systems**, be they software or hardware, or combined computer-embedded systems, including distributed and real-time systems.

We propose a different way to program such systems, centered on inter-object scenario-based behavior. The book describes a language, two techniques, and a supporting tool. The language is a rather broad extension of **live sequence charts** (**LSCs**), the original version of which was proposed in 1998 by W. Damm and the first-listed author of this book. The first of the two techniques, called **play-in**, is a convenient way to 'play in' scenario-based behavior directly from the system's graphical user interface (GUI). The second technique, **play-out**, makes it possible to execute, or 'play out', the behavior on the GUI as if it were programmed in a conventional intra-object state-based fashion. All this is implemented in full in our tool, the **Play-Engine**.

The book can be viewed as offering improvements in some of the phases of known system development life cycles, e.g., requirements capture and analysis, prototyping, and testing. However, there is a more radical way to view the book, namely, as proposing an alternative way to program reactivity, which, being based on inter-object scenarios, is a lot closer to how people think about systems and their behavior.

We are excited by the apparent potential of this work. However, whether or not it is adopted and becomes really useful, what kinds of systems are the ideas most fitting for, and how we should develop methodologies for large-scale applications, all remain to a large extent open questions. Whatever the case, we hope that the book triggers further research and experimentation.

David Harel and Rami Marelly
Rehovot, February 2003

Note on the Software

The Play-Engine tool is available with the book for free, and the attached CD contains most of the files needed for using the software. However, some of the ideas behind the play-in and play-out methods are patent-pending, and both the relevant intellectual property and the Play-Engine software itself are owned by the Weizmann Institute of Science. In order to obtain the remaining parts of the software, please visit the book's website, `www.wisdom.weizmann.ac.il/~playbook`, where you will be asked to sign an appropriate license agreement and to register your details.

A reservation is in order here: the Play-Engine is not a commercial product (at least at the time of writing), and should be regarded as a research-level tool. Thus, its reliability and ease of use are less than what would be expected from professional software, and we cannot be held responsible for its performance. Nevertheless, we will make an effort to correct bugs and to otherwise improve and modify the tool, posting periodical updates on the website. In fact, we have already made some slight changes in the software since the text of the book was finalized a few weeks ago; these are all documented in the User Guide that is attached to the software. We would be very grateful if readers would use the appropriate locations on the website to report any errors, both in the software and in the text of the book.

So, please do come and visit the site, sign in, download, play, and enjoy...

Acknowledgments

Our first thanks go to Werner Damm from the University of Oldenburg. His 1997–98 collaboration with the first-listed author — to which he brought both the scientist's skill and the pragmatist's insight — yielded the original version of the LSCs language, which turned out to be the crucial prerequisite of our work.

A very special thanks goes to Hillel Kugler and Na'aman Kam for being such demanding, yet tolerant users of the Play-Engine. Many of their comments and suggestions have found their way into the material presented here. Hillel's work on smart play-out (cosupervised by Amir Pnueli) is the most promising follow-up research project related to the topic of this book, and we would like to further thank him for his help in our research on symbolic instances, and for his dedication in the effort of writing Chap. 18. Na'aman also supplied the examples from the C. elegans nematode model given in that chapter.

We received helpful suggestions and insightful comments from several other people during our work. They include Liran Carmel, Sol Efroni, Yael Kfir, Jochen Klose, Yehuda Koren, Anat Maoz, David Peleg, Amir Pnueli, Ehud Shapiro, Gera Weiss and an anonymous referee.

Thanks go to Dan Barak for his work on connecting multiple Play-Engines (the SEC project) and for helping with the implementation of external objects. Evgeniy Bart and Maksim Frenkel helped develop an early version of GUIEdit.

The first-listed author would also like to thank the Verimag research center in Grenoble, and its director, Joseph Sifakis, for a generous part-time visiting position in 2002, during which parts of the book were written.

The second-listed author would like to thank Orna Grumberg and Moti Kehat for helping him make the right choices at the right times.

Contents

Part I. Prelude

1. Introduction 3
1.1 What Are We Talking About? 3
1.2 What Are We Trying to Do? 6
1.3 What's in the Book? 7

2. Setting the Stage 9
2.1 Modeling and Code Generation 9
2.2 Requirements 12
2.3 Inter-Object vs. Intra-Object Behavior 14
2.4 Live Sequence Charts (LSCs) 16
2.5 Testing, Verification and Synthesis 17
2.6 The Play-In/Play-Out Approach 21

3. An Example-Driven Overview 25
3.1 The Sample System 25
3.2 Playing In 26
3.3 Playing Out 39
3.4 Using Play-Out for Testing 43
3.5 Transition to Design 44
3.6 Time 46
3.7 Smart Play-Out 47

Part II. Foundations

4. The Model: Object Systems 55
4.1 Application Types 55
4.2 Object Properties 56
4.3 And a Bit More Formally 58

5. The Language: Live Sequence Charts (LSCs) 59
5.1 Constant LSCs 60
5.2 Playing In 62
5.3 The General Play-Out Scheme 65
5.4 Playing Out 68
5.5 Combining Locations and Messages 71
5.6 And a Bit More Formally 73
5.7 Bibliographic Notes 81

6. The Tool: The Play-Engine 83
6.1 Bibliographic Notes 87

Part III. Basic Behavior

7. Variables and Symbolic Messages 91
7.1 Symbolic Scenarios 91
7.2 Enriching the Partial Order 94
7.3 Playing Out 97
7.4 And a Bit More Formally 99
7.5 Bibliographic Notes 103

8. Assignments and Implemented Functions 105
8.1 Using Implemented Functions 105
8.2 Assignments 108
8.3 Playing Out 111
8.4 And a Bit More Formally 114

9. Conditions 119
9.1 Cold Conditions 119
9.2 Hot Conditions 120
9.3 Playing In 121
9.4 Playing Out 126
9.5 And a Bit More Formally 128
9.6 Bibliographic Notes 132

10. Branching and Subcharts 133
10.1 The If-Then-Else Construct 133
10.2 Subcharts 134
10.3 Nondeterministic Choice 135
10.4 Playing In 136
10.5 Playing Out 138

10.6 And a Bit More Formally ... 141
10.7 Bibliographic Notes ... 146

Part IV. Advanced Behavior: Multiple Charts

11. Executing Multiple Charts ... 149
11.1 Simultaneous Activation of Multiple Charts ... 149
11.2 Overlapping Charts ... 154
11.3 And a Bit More Formally ... 157

12. Testing with Existential Charts ... 159
12.1 Specifying Test Scenarios ... 159
12.2 Monitoring LSCs ... 160
12.3 Recording and Replaying ... 163
12.4 On-line Testing ... 164
12.5 Executing and Monitoring LSCs in the Play-Engine ... 165
12.6 And a Bit More Formally ... 166
12.7 Bibliographic Notes ... 171

Part V. Advanced Behavior: Richer Constructs

13. Loops ... 175
13.1 Using Loops ... 175
13.2 Playing In ... 176
13.3 Playing Out ... 178
13.4 Using Variables Within Loops ... 179
13.5 Executing and Monitoring Dynamic Loops ... 181
13.6 And a Bit More Formally ... 183
13.7 Bibliographic Notes ... 187

14. Transition to Design ... 189
14.1 The Design Phase ... 189
14.2 Incorporating Internal Objects ... 190
14.3 Calling Object Methods ... 193
14.4 Playing Out ... 196
14.5 External Objects ... 198
14.6 And a Bit More Formally ... 201
14.7 Bibliographic Notes ... 205

15. Classes and Symbolic Instances 209
15.1 Symbolic Instances 209
15.2 Classes and Objects 210
15.3 Playing with Simple Symbolic Instances 212
15.4 Symbolic Instances in the Main Chart 213
15.5 Quantified Binding 215
15.6 Reusing a Scenario Prefix 216
15.7 Symbolic Instances in Existential Charts 218
15.8 An Advanced Example: NetPhone 218
15.9 And a Bit More Formally 221
15.10 Bibliographic Notes 227

16. Time and Real-Time Systems 229
16.1 An Example 229
16.2 Adding Time to LSCs 230
16.3 Hot Timing Constraints 231
16.4 Cold Timing Constraints 234
16.5 Time Events 235
16.6 Playing In 236
16.7 Playing Out 237
16.8 Unification of Clock Ticks 239
16.9 The Time-Enriched NetPhone Example 240
16.10 And a Bit More Formally 242
16.11 Bibliographic Notes 247

17. Forbidden Elements 251
17.1 Example: A Cruise Control System 251
17.2 Forbidden Messages 252
17.3 Generalized Forbidden Messages 255
17.4 Symbolic Instances in Forbidden Messages 256
17.5 Forbidden Conditions 258
17.6 Scoping Forbidden Elements 262
17.7 Playing Out 264
17.8 Using Forbidden Elements with Time 266
17.9 A Tolerant Semantics for LSCs 267
17.10 And a Bit More Formally 268
17.11 Bibliographic Notes 277

Part VI. Enhancing the Play-Engine

18. Smart Play-Out (with H. Kugler) 281
18.1 Introduction 281
18.2 Being Smart Helps 283
18.3 The General Approach 287
18.4 The Translation 289
18.5 Current Limitations 299
18.6 Satisfying Existential Charts 301
18.7 Bibliographic Notes 307

19. Inside and Outside the Play-Engine 309
19.1 The Engine's Environment 309
19.2 Playing In 310
19.3 Playing Out 311
19.4 Recording Runs and Connecting External Applications 313
19.5 Additional Play-Engine Features 313

20. A Play-Engine Aware GUI Editor 317
20.1 Who Needs a GUI Editor? 317
20.2 GUIEdit in Visual Basic 318
20.3 What Does GUIEdit Do? 318
20.4 Incorporating Custom Controls 321
20.5 GUIEdit As a Proof of Concept 321
20.6 Bibliographic Notes 322

21. Future Research Directions 323
21.1 Object Refinement and Composition 323
21.2 Object Model Diagrams, Inheritance and Interfaces 325
21.3 Dynamic Creation and Destruction of Objects 326
21.4 Structured Properties and Types 327
21.5 Linking Multiple Engines 328

Part VII. Appendices

A. Formal Semantics of LSCs 333
A.1 System Model and Events 333
A.2 LSC Specification 336
A.3 Operational Semantics 342

B. XML Description of a GUI Application 357

C. The Play-Engine Interface 361
C.1 Visual Basic Code 361

D. The GUI Application Interface 363
D.1 Visual Basic Code 364

E. The Structure of a (Recorded) Run 367

References 369

Index 375