Carmen Zannier
Hakan Erdogmus
Lowell Lindstrom (Eds.)

LNCS 3134

Extreme Programming and Agile Methods – XP/Agile Universe 2004

4th Conference on Extreme Programming and Agile Methods
Calgary, Canada, August 2004
Proceedings

Springer

Carmen Zannier Hakan Erdogmus
Lowell Lindstrom (Eds.)

Extreme Programming and Agile Methods – XP/Agile Universe 2004

4th Conference on Extreme Programming and Agile Methods
Calgary, Canada, August 15-18, 2004
Proceedings

Springer

Volume Editors

Carmen Zannier
University of Calgary, Department of Computer Science
2500 University Drive NW, Calgary, Alberta, Canada T2N 1N4
E-mail: zannierc@cpsc.ucalgary.ca

Hakan Erdogmus
National Research Council
Montreal Rd. M50, Ottawa, Ontario, Canada K1A 0R6
E-mail: hakan.erdogmus@nrc-cnrc.gc.ca

Lowell Lindstrom
Object Mentor, Inc.
501 North Riverside Drive, Suite 206, Gurnee, IL 60031, USA
E-mail: lindstrom@objectmentor.com

Library of Congress Control Number: 2004109934

CR Subject Classification (1998): D.1, D.2, D.3, F.3, K.4.3, K.6

ISSN 0302-9743
ISBN 3-540-22839-X Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springeronline.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11312239 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3134

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Preface

It was 1999 when *Extreme Programming Explained* was first published, making this year's event arguably the fifth anniversary of the birth of the XP/Agile movement in software development. Our fourth conference reflected the evolution and the learning that have occurred in these exciting five years as agile practices have become part of the mainstream in software development. These pages are the proceedings of XP Agile Universe 2004, held in beautiful Calgary, gateway to the Canadian Rockies, in Alberta, Canada.

Evident in the conference is the fact that our learning is still in its early stages. While at times overlooked, *adaptation* has been a core principle of agile software development since the earliest literature on the subject. The conference and these proceedings reinforce that principle. Although some organizations are able to practice agile methods in the near-pure form, most are not, reflecting just how radically innovative these methods are to this day. Any innovation must coexist with an existing environment and agile software development is no different. There are numerous challenges confronting IT and software development organizations today, with many solutions pitched by a cadre of advocates. Be it CMM, offshoring, outsourcing, security, or one of many other current topics in the industry, teams using or transitioning to Extreme Programming and other agile practices must integrate with the rest of the organization in order to succeed. The papers here offer some of the latest experiences that teams are having in those efforts.

XP Agile Universe 2004 consisted of workshops, tutorials, papers, panels, the Open Space session, the Educators' Symposium, keynotes, educational games and industry presentations. This wide range of activities was intended to provide an engaging experience for industry practitioners, leading consultants, researchers, academics, and students. Feedback from the 2003 conference was used to adjust the content to better suit the needs of the attendees. The sessions at the conference were selected through the dedicated work of the Track Chairs and the Program Committee, to whom we are extremely grateful. Their names are listed in the pages that follow and the contributions of these individuals to the experience of the attendees of the conference cannot be overstated. Over 100 submissions were received to the various activities, with roughly half accepted into the conference. Each submission was reviewed by at least 3 members of the Program Committee, with an average of just under 5 reviewers per submission. The accepted papers are presented in their entirety in these proceedings. Summaries of the workshops and tutorials are presented as a reference for those who attended the conference. The results of the Open Space session can be accessed via the conference website at xpuniverse.com or agileuniverse.com.

The invited speakers to the conference were Christopher Avery, Robert Biddle, Eric Evans, Alejandro Goyen, Craig Larman, Brian Marick, Robert C. Martin, Mary Poppendieck, and Herb Sutter. These speakers represent the breadth and depth of the conference in terms of three main threads: technical practices; business and project management; and teamwork.

The tutorials and workshops continued the trend beyond the programming trenches, focusing primarily on requirements, project management, and acceptance-testing techniques, with some introductory sessions for attendees new to extreme programming and agile practices. The conference also continued its history with hands-on programming events which allowed attendees to join projects that ran throughout the conference using the tools and practices common on agile teams.

In these proceedings, one can find a rich set of papers reflective of the experiences of leading practitioners. Eighteen technical and research papers, experience reports, and educators' symposium papers were accepted out of a total of 45 submissions, representing an acceptance rate of 40%. A number of papers provide advanced discussion on tools and techniques for testing and the trend towards combining the requirements, testing, and specification activities. Three papers discuss methods for better understanding and expressing the customer or user needs in an agile way. For readers who are confronted with many of the challenges faced by today's environment, such as security concerns, CMM auditing, and offshore development teams, there are representative papers describing the use of agile development techniques in those environments.

We are deeply indebted to the organizing committee and the conference sponsors for providing the infrastructure for making the conference happen. The content of the conference and these proceedings would not have been possible without the submissions and all of the effort that goes into them. For those courageous enough to submit their work to the conference, we thank and salute you. But mostly, we thank the attendees, for supporting the conference, giving it its positive energy, and making it the magical gathering that it has become.

August 2004 Lowell Lindstrom and Hakan Erdogmus

Program Chairs

Hakan Erdogmus	National Research Council Canada
Lowell Lindstrom	Object Mentor

Track Chairs

Tutorials	Brian Button, Agile Solutions Group
Workshops	Dave Astels, ThoughtWorks, Inc. Grigori Melnik, University of Calgary
Panels, Short Activities	Mike Cohn, Mountain Goat Software
Educators' Symposium	Rick Mercer, University of Arizona
Open Space, Birds-of-a-Feather	Ann Anderson, First Data Corporation William Wake, Independent Consultant

Organizing Committee

General Chair	Lance Welter, Object Mentor
Conference Coordinator	Jennifer Goodsen, RADsoft
Calgary Coordinator	Janet Gregory, Wireless Matrix
Social Fun Committee	John Goodsen, RADsoft
Sponsorship and Exhibits	Lance Welter, Object Mentor
Web Master	Micah Martin, Object Mentor
Community Liaisons	US – Alex Viggio Canada – Shaun Smith
Academic Liaison	Frank Maurer, University of Calgary
Proceedings Coordinator	Carmen Zannier, University of Calgary
On-site Logistics	Janet Gregory, Wireless Matrix
Information Coordinators	Talisha Jefferson, Object Mentor Susan Rosso, Object Mentor

Educators' Symposium Committee

Rick Mercer	University of Arizona
Joe Bergin	Pace University
Robert Biddle	Carleton University
Jim Caristi	Valparaiso University
Jutta Eckstein	Independent Consultant
James Grenning	Object Mentor
Diana Larsen	Industrial Logic
Grigori Melnik	University of Calgary
Rick Mugridge	University of Auckland
Daniel Steinberg	O'Reilly Networks
Eugene Wallingford	University of Northern Iowa

Open Space and Birds-of-a-Feather Committee

Ann Anderson	First Data Corporation
Lisa Crispin	Fast401k
J.B. Rainsberger	Diaspar Software Services
William Wake	Independent Consultant

Program Committee

Ann Anderson	First Data Corporation
Jennitta Andrea	Clearstream Consulting
Ken Auer	Role Model Software
Mike Beedle	e-Architects Inc.
Robert Biddle	Carleton University
Jim Coplien	Vrije Universiteit Brussel
Alain Desilets	National Research Council Canada
Dwight Deugo	Carleton University
Armin Eberlein	American University of Sharjah
Jutta Eckstein	Independent Consultant
Alejandro Goyen	Microsoft
John Favaro	Consulenza Informatica
Steve Fraser	Independent Consultant
John Grundy	University of Auckland
John Goodsen	RADSoft
Philip Johnson	University of Hawaii
Brian Hanks	University of California
Chet Hendrickson	HendricksonXP
Michael Hill	Independent Consultant
Paul Hodgetts	Agile Logic
Andy Hunt	The Pragmatic Programmers
Ron Jeffries	XPProgramming
Bil Kleb	NASA
Jykri Kontio	Helsinki University of Technology
Philippe Kruchten	University of British Columbia
Tom Kubit	Gene Codes Forensics
Manfred Lange	Independent Consultant
Diana Larsen	Industrial Logic
Jim Leask	Sybase
Tim Mackinnon	Connextra
Brian Marick	Testing Foundations
Robert C. Martin	Object Mentor
Frank Maurer	University of Calgary
Pete McBreen	Software Craftsmanship
Todd Medlin	SAS Institute

Grigori Melnik	University of Calgary
Steve Mellor	Project Technology
Granville Miller	Borland
Maurizio Morisio	Politecnico di Torino
Rick Mugridge	University of Auckland
Gary Pollice	Worcester Polytechnic Institute
Linda Rising	Independent Consultant
Ken Schwaber	Agile Alliance
David Stotts	University of North Carolina
Shaun Smith	Sandbox Systems
Oryal Tanir	Bell Canada
Dave Thomas	The Pragmatic Programmer
Dave Thomas	Bedarra Research Labs
Jim Tomayko	Carnegie Mellon University
Marco Torchiano	Politecnico di Torino
David Trowdridge	Microsoft
Jay Turpin	Intel
William Wake	Independent Consultant
Don Wells	ExtremeProgramming.org
Frank Westphal	Independent Consultant
Laurie Williams	North Carolina State University
William Wood	NASA

Sponsoring Institutions

Galaxy Class	Object Mentor
	Microsoft
	ThoughtWorks, Inc.
	Valtech Technologies, Inc.
Star Class	ClearStream Consulting, Inc.
	Rally Software Development
	BrightSpot Consulting
Satellite Class	VersionOne
	RADSoft
Media Partners	Software Development Magazine
	DevTown Station
	Integration Developer News
	Java Developer's Journal
	Better Software Magazine
	Linux Journal
	Cutter Consortium

Lecture Notes in Computer Science

For information about Vols. 1–3053

please contact your bookseller or Springer-Verlag

Vol. 3172: M. Dorigo, M. Birattari, C. Blum, L. M.Gambardella, F. Mondada, T. Stützle (Eds.), Ant Colony, Optimization and Swarm Intelligence. XII, 434 pages. 2004.

Vol. 3158: I. Nikolaidis, M. Barbeau, E. Kranakis (Eds.), Ad-Hoc, Mobile, and Wireless Networks. IX, 344 pages. 2004.

Vol. 3157: C. Zhang, H. W. Guesgen, W.K. Yeap (Eds.), PRICAI 2004: Trends in Artificial Intelligence. XX, 1023 pages. 2004. (Subseries LNAI).

Vol. 3156: M. Joye, J.-J. Quisquater (Eds.), Cryptographic Hardware and Embedded Systems - CHES 2004. XIII, 455 pages. 2004.

Vol. 3153: J. Fiala, V. Koubek, J. Kratochvíl (Eds.), Mathematical Foundations of Computer Science 2004. XIV, 888 pages. 2004.

Vol. 3152: M. Franklin (Ed.), Advances in Cryptology – CRYPTO 2004. XI, 579 pages. 2004.

Vol. 3148: R. Giacobazzi (Ed.), Static Analysis. X, 393 pages. 2004.

Vol. 3146: P. Érdi, A. Esposito, M. Marinaro, S. Scarpetta (Eds.), Computational Neuroscience: Cortical Dynamics. XI, 161 pages. 2004.

Vol. 3144: M. Papatriantafilou, P. Hunel (Eds.), Principles of Distributed Systems. XI, 246 pages. 2004.

Vol. 3143: W. Liu, Y. Shi, Q. Li (Eds.), Advances in Web-Based Learning – ICWL 2004. XIV, 459 pages. 2004.

Vol. 3142: J. Diaz, J. Karhumäki, A. Lepistö, D. Sannella (Eds.), Automata, Languages and Programming. XIX, 1253 pages. 2004.

Vol. 3140: N. Koch, P. Fraternali, M. Wirsing (Eds.), Web Engineering. XXI, 623 pages. 2004.

Vol. 3139: F. Iida, R. Pfeifer, L. Steels, Y. Kuniyoshi (Eds.), Embodied Artificial Intelligence. IX, 331 pages. 2004. (Subseries LNAI).

Vol. 3138: A. Fred, T. Caelli, R.P.W. Duin, A. Campilho, D.d. Ridder (Eds.), Structural, Syntactic, and Statistical Pattern Recognition. XXII, 1168 pages. 2004.

Vol. 3134: C. Zannier, H. Erdogmus, L. Lindstrom (Eds.), Extreme Programming and Agile Methods - XP/Agile Universe 2004. XIV, 233 pages. 2004.

Vol. 3133: A.D. Pimentel, S. Vassiliadis (Eds.), Computer Systems: Architectures, Modeling, and Simulation. XIII, 562 pages. 2004.

Vol. 3131: V. Torra, Y. Narukawa (Eds.), Modeling Decisions for Artificial Intelligence. XI, 327 pages. 2004. (Subseries LNAI).

Vol. 3130: A. Syropoulos, K. Berry, Y. Haralambous, B. Hughes, S. Peter, J. Plaice (Eds.), TEX, XML, and Digital Typography. I, 123 pages. 2004.

Vol. 3129: Q. Li, G. Wang, L. Feng (Eds.), Advances in Web-Age Information Management. XVII, 753 pages. 2004.

Vol. 3128: D. Asonov (Ed.), Querying Databases Privately. IX, 115 pages. 2004.

Vol. 3127: K.E. Wolff, H.D. Pfeiffer, H.S. Delugach (Eds.), Conceptual Structures at Work. XI, 403 pages. 2004. (Subseries LNAI).

Vol. 3126: P. Dini, P. Lorenz, J.N.d. Souza (Eds.), Service Assurance with Partial and Intermittent Resources. XI, 312 pages. 2004.

Vol. 3125: D. Kozen (Ed.), Mathematics of Program Construction. X, 401 pages. 2004.

Vol. 3124: J.N. de Souza, P. Dini, P. Lorenz (Eds.), Telecommunications and Networking - ICT 2004. XXVI, 1390 pages. 2004.

Vol. 3123: A. Belz, R. Evans, P. Piwek (Eds.), Natural Language Generation. X, 219 pages. 2004. (Subseries LNAI).

Vol. 3121: S. Nikoletseas, J.D.P. Rolim (Eds.), Algorithmic Aspects of Wireless Sensor Networks. X, 201 pages. 2004.

Vol. 3120: J. Shawe-Taylor, Y. Singer (Eds.), Learning Theory. X, 648 pages. 2004. (Subseries LNAI).

Vol. 3118: K. Miesenberger, J. Klaus, W. Zagler, D. Burger (Eds.), Computer Helping People with Special Needs. XXIII, 1191 pages. 2004.

Vol. 3116: C. Rattray, S. Maharaj, C. Shankland (Eds.), Algebraic Methodology and Software Technology. XI, 569 pages. 2004.

Vol. 3115: P. Enser, Y. Kompatsiaris, N.E. O'Connor, A.F. Smeaton, A.W.M. Smeulders (Eds.), Image and Video Retrieval. XVII, 679 pages. 2004.

Vol. 3114: R. Alur, D.A. Peled (Eds.), Computer Aided Verification. XII, 536 pages. 2004.

Vol. 3113: J. Karhumäki, H. Maurer, G. Paun, G. Rozenberg (Eds.), Theory Is Forever. X, 283 pages. 2004.

Vol. 3112: H. Williams, L. MacKinnon (Eds.), Key Technologies for Data Management. XII, 265 pages. 2004.

Vol. 3111: T. Hagerup, J. Katajainen (Eds.), Algorithm Theory - SWAT 2004. XI, 506 pages. 2004.

Vol. 3110: A. Juels (Ed.), Financial Cryptography. XI, 281 pages. 2004.

Vol. 3109: S.C. Sahinalp, S. Muthukrishnan, U. Dogrusoz (Eds.), Combinatorial Pattern Matching. XII, 486 pages. 2004.

Vol. 3108: H. Wang, J. Pieprzyk, V. Varadharajan (Eds.), Information Security and Privacy. XII, 494 pages. 2004.

Vol. 3107: J. Bosch, C. Krueger (Eds.), Software Reuse: Methods, Techniques and Tools. XI, 339 pages. 2004.

Vol. 3105: S. Göbel, U. Spierling, A. Hoffmann, I. Iurgel, O. Schneider, J. Dechau, A. Feix (Eds.), Technologies for Interactive Digital Storytelling and Entertainment. XVI, 304 pages. 2004.

Vol. 3104: R. Kralovic, O. Sykora (Eds.), Structural Information and Communication Complexity. X, 303 pages. 2004.

Vol. 3103: K. Deb, e. al. (Eds.), Genetic and Evolutionary Computation – GECCO 2004. XLIX, 1439 pages. 2004.

Vol. 3102: K. Deb, e. al. (Eds.), Genetic and Evolutionary Computation – GECCO 2004. L, 1445 pages. 2004.

Vol. 3101: M. Masoodian, S. Jones, B. Rogers (Eds.), Computer Human Interaction. XIV, 694 pages. 2004.

Vol. 3100: J.F. Peters, A. Skowron, J.W. Grzymała-Busse, B. Kostek, R.W. Świniarski, M.S. Szczuka (Eds.), Transactions on Rough Sets I. X, 405 pages. 2004.

Vol. 3099: J. Cortadella, W. Reisig (Eds.), Applications and Theory of Petri Nets 2004. XI, 505 pages. 2004.

Vol. 3098: J. Desel, W. Reisig, G. Rozenberg (Eds.), Lectures on Concurrency and Petri Nets. VIII, 849 pages. 2004.

Vol. 3097: D. Basin, M. Rusinowitch (Eds.), Automated Reasoning. XII, 493 pages. 2004. (Subseries LNAI).

Vol. 3096: G. Melnik, H. Holz (Eds.), Advances in Learning Software Organizations. X, 173 pages. 2004.

Vol. 3095: C. Bussler, D. Fensel, M.E. Orlowska, J. Yang (Eds.), Web Services, E-Business, and the Semantic Web. X, 147 pages. 2004.

Vol. 3094: A. Nürnberger, M. Detyniecki (Eds.), Adaptive Multimedia Retrieval. VIII, 229 pages. 2004.

Vol. 3093: S.K. Katsikas, S. Gritzalis, J. Lopez (Eds.), Public Key Infrastructure. XIII, 380 pages. 2004.

Vol. 3092: J. Eckstein, H. Baumeister (Eds.), Extreme Programming and Agile Processes in Software Engineering. XVI, 358 pages. 2004.

Vol. 3091: V. van Oostrom (Ed.), Rewriting Techniques and Applications. X, 313 pages. 2004.

Vol. 3089: M. Jakobsson, M. Yung, J. Zhou (Eds.), Applied Cryptography and Network Security. XIV, 510 pages. 2004.

Vol. 3087: D. Maltoni, A.K. Jain (Eds.), Biometric Authentication. XIII, 343 pages. 2004.

Vol. 3086: M. Odersky (Ed.), ECOOP 2004 – Object-Oriented Programming. XIII, 611 pages. 2004.

Vol. 3085: S. Berardi, M. Coppo, F. Damiani (Eds.), Types for Proofs and Programs. X, 409 pages. 2004.

Vol. 3084: A. Persson, J. Stirna (Eds.), Advanced Information Systems Engineering. XIV, 596 pages. 2004.

Vol. 3083: W. Emmerich, A.L. Wolf (Eds.), Component Deployment. X, 249 pages. 2004.

Vol. 3080: J. Desel, B. Pernici, M. Weske (Eds.), Business Process Management. X, 307 pages. 2004.

Vol. 3079: Z. Mammeri, P. Lorenz (Eds.), High Speed Networks and Multimedia Communications. XVIII, 1103 pages. 2004.

Vol. 3078: S. Cotin, D.N. Metaxas (Eds.), Medical Simulation. XVI, 296 pages. 2004.

Vol. 3077: F. Roli, J. Kittler, T. Windeatt (Eds.), Multiple Classifier Systems. XII, 386 pages. 2004.

Vol. 3076: D. Buell (Ed.), Algorithmic Number Theory. XI, 451 pages. 2004.

Vol. 3074: B. Kuijpers, P. Revesz (Eds.), Constraint Databases and Applications. XII, 181 pages. 2004.

Vol. 3073: H. Chen, R. Moore, D.D. Zeng, J. Leavitt (Eds.), Intelligence and Security Informatics. XV, 536 pages. 2004.

Vol. 3072: D. Zhang, A.K. Jain (Eds.), Biometric Authentication. XVII, 800 pages. 2004.

Vol. 3071: A. Omicini, P. Petta, J. Pitt (Eds.), Engineering Societies in the Agents World. XIII, 409 pages. 2004. (Subseries LNAI).

Vol. 3070: L. Rutkowski, J. Siekmann, R. Tadeusiewicz, L.A. Zadeh (Eds.), Artificial Intelligence and Soft Computing - ICAISC 2004. XXV, 1208 pages. 2004. (Subseries LNAI).

Vol. 3068: E. André, L. Dybkjær, W. Minker, P. Heisterkamp (Eds.), Affective Dialogue Systems. XII, 324 pages. 2004. (Subseries LNAI).

Vol. 3067: M. Dastani, J. Dix, A. El Fallah-Seghrouchni (Eds.), Programming Multi-Agent Systems. X, 221 pages. 2004. (Subseries LNAI).

Vol. 3066: S. Tsumoto, R. Słowiński, J. Komorowski, J.W. Grzymała-Busse (Eds.), Rough Sets and Current Trends in Computing. XX, 853 pages. 2004. (Subseries LNAI).

Vol. 3065: A. Lomuscio, D. Nute (Eds.), Deontic Logic in Computer Science. X, 275 pages. 2004. (Subseries LNAI).

Vol. 3064: D. Bienstock, G. Nemhauser (Eds.), Integer Programming and Combinatorial Optimization. XI, 445 pages. 2004.

Vol. 3063: A. Llamosí, A. Strohmeier (Eds.), Reliable Software Technologies - Ada-Europe 2004. XIII, 333 pages. 2004.

Vol. 3062: J.L. Pfaltz, M. Nagl, B. Böhlen (Eds.), Applications of Graph Transformations with Industrial Relevance. XV, 500 pages. 2004.

Vol. 3061: F.F. Ramos, H. Unger, V. Larios (Eds.), Advanced Distributed Systems. VIII, 285 pages. 2004.

Vol. 3060: A.Y. Tawfik, S.D. Goodwin (Eds.), Advances in Artificial Intelligence. XIII, 582 pages. 2004. (Subseries LNAI).

Vol. 3059: C.C. Ribeiro, S.L. Martins (Eds.), Experimental and Efficient Algorithms. X, 586 pages. 2004.

Vol. 3058: N. Sebe, M.S. Lew, T.S. Huang (Eds.), Computer Vision in Human-Computer Interaction. X, 233 pages. 2004.

Vol. 3057: B. Jayaraman (Ed.), Practical Aspects of Declarative Languages. VIII, 255 pages. 2004.

Vol. 3056: H. Dai, R. Srikant, C. Zhang (Eds.), Advances in Knowledge Discovery and Data Mining. XIX, 713 pages. 2004. (Subseries LNAI).

Vol. 3055: H. Christiansen, M.-S. Hacid, T. Andreasen, H.L. Larsen (Eds.), Flexible Query Answering Systems. X, 500 pages. 2004. (Subseries LNAI).

Vol. 3054: I. Crnkovic, J.A. Stafford, H.W. Schmidt, K. Wallnau (Eds.), Component-Based Software Engineering. XI, 311 pages. 2004.

Table of Contents

Papers

Testing and Integration

Managing Requirements and Usability

Pair Programming

Foundations of Agility

Process Adaptations

Educators' Symposium

Workshop Summaries

Panels

Tutorials

Combining Formal Specifications with Test Driven Development*

Hubert Baumeister

Institut für Informatik
Ludwig-Maximilians-Universität München
Oettingenstr. 67, D-80538 München, Germany
baumeist@informatik.uni-muenchen.de

Abstract. In the context of test driven development, tests specify the behavior of a program before the code that implements it, is actually written. In addition, they are used as main source of documentation in XP projects, together with the program code. However, tests alone describe the properties of a program only in terms of examples and thus are not sufficient to completely describe the behavior of a program. In contrast, formal specifications allow to generalize these example properties to more general properties, which leads to a more complete description of the behavior of a program. Specifications add another main artifact to XP in addition to the already existent ones, i.e. code and tests. The interaction between these three artifacts further improves the quality of both software and documentation. The goal of this paper is to show that it is possible, with appropriate tool support, to combine formal specifications with test driven development without loosing the agility of test driven development.

1 Introduction

Extreme Programming advocates test driven development where tests are used to specify the behavior of a program before the program code is actually written. Together with using the simplest design possible and intention revealing program code, tests are additionally used as a documentation of the program. However, tests are not sufficient to completely define the behavior of a program because they are only able to test properties of a program by example and do not allow to state general properties. The latter can be achieved using formal specifications, e.g. using Meyer's design by contract [21].

As an example we consider the function `primes`, that computes for a given natural number n a list containing all prime numbers up to and including n. Tests can only be written for special arguments of the primes function, e.g. that `primes(2)` should produce the list with the number 2 as its only element, and that `primes(1553)` is supposed to yield the list of prime numbers from 2 up to 1533. Actually, a program that behaves correctly w.r.t. these tests could have the set of prime numbers hard coded for these particular inputs and return arbitrary lists for all other arguments. One solution is to move from tests to specifications, which allow to generalize the tested properties. For example, the behavior of `primes` would be expressed by a formal specification stating that the result of the function `primes(n)` contains exactly the prime numbers from 2 up to n, for all natural numbers n.

* This research has been partially sponsored by the EC 5th Framework project AGILE: Architectures for Mobility (IST-2001-32747)

C. Zannier et al. (Eds.): XP/Agile Universe 2004, LNCS 3134, pp. 1–12, 2004.

This example shows that formal specifications provide a more complete view on the behavior of programs than tests alone. However, while it is easy to run tests to check that a program complies with the tests, the task of showing that a program satisfies a given specification is in general more complex. To at least validate a program w.r.t. a specification, one can use the specification to generate run-time assertions and use these to check that the program behaves correctly.

The study of formal methods for program specification and verification has a long history. Hoare and Floyd pioneered the development of formal methods in the 1960s by introducing the Hoare calculus for proving program correctness as well as the notions of pre-/postconditions, invariants, and assertions [13, 10]. Their ideas were gradually developed into fully fledged formal methods geared towards industrial software engineering, e.g. the Vienna Development Method (VDM) developed at IBM [17], Z [23], the Java Modeling Language (JML) [19] and, more recently, the Object Constraint Language (OCL) [25] – which again originated at IBM – used to specify contraints on objects in UML diagrams. For an overview of formal methods and their applications refer to the WWW virtual library on formal methods [5].

An important use of formal specifications is the documentation of program behavior without making reference to an implementation. This is often needed for frameworks and libraries, where the source code is not available in most cases and the behavior is only informally described. In general, the documentation provided by a formal specification is both more precise and more concise compared to the implementation code because the implementation only describes the algorithm used by a method and not what it achieves. Not only the literature on formals methods, but also in the literature on the pragmatics of programming, e.g. [15, 20], recommends to make explicit the assumptions on the code using specifications because this improves the software quality.

The goal of this paper is to show that it is possible, with appropriate tool support, to combine formal specifications with test driven development without loosing the agility of the latter. This is done by using the tests, that drive the development of the code, also to drive the development of the formal specification. By generating runtime assertions from the specification it is possible to check for inconsistencies between code, specifications, and tests. Each of the three artifacts improves the quality of the other two, yielding better code quality and better program documentation in the form of a validated formal specification of the program.

Our method is exemplified by using the primes example with Java as the programming language, JUnit[1] as the testing framework, and the Java Modeling Language (JML) [19] for the formulation of class invariants and pre- and postconditions for methods. We use JML since JML specifications are easily understood by programmers, and because it comes with a runtime assertion checker, [6], which allows to check invariants and pre- and postconditions of methods at runtime.

2 Formal Specifications and Tests

As with test driven development, in our proposed methodology, tests are written before the code. Either now or after several iterations of test and code development, the prop-

[1] www.junit.org