Frank van der Linden (Ed.)

Software Product-Family Engineering

LNCS 3014

5th International Workshop, PFE 2003
Siena, Italy, November 2003
Revised Papers

Springer

Frank van der Linden (Ed.)

Software Product-Family Engineering

5th International Workshop, PFE 2003
Siena, Italy, November 4-6, 2003
Revised Papers

Springer

Volume Editor

Frank van der Linden
Philips Medical Systems N.V., BL Medical IT
QV-1, Veenpluis 4-6, PO Box 10000, 5680 DA Best
The Netherlands
E-mail: Frank.van.der.linden@philips.com

Library of Congress Control Number: 2004105721

CR Subject Classification (1998): D.2.11, K.6, D.2

ISSN 0302-9743
ISBN 3-540-21941-2 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 10999257 06/3142 5 4 3 2 1 0

Preface

This book contains the proceedings of the 5th International Workshop on Product Family Engineering, PFE-5. This workshop was held in Siena, Italy, November 4–6, 2003. This workshop was the fifth in the series, with the same subject, software product family engineering. These workshops have been held initially irregularly about every 18 months since 1996. Since 1999 the workshop has been held every second year in the fall. The proceedings of the second, third and fourth workshops were published as Springer LNCS volumes 1429, 1951 and 2290.

The workshops were organized within co-operation projects of European industry. The first two were organized by ARES (Esprit IV 20.477) 1995–1999; this project had 3 industrial and 3 academic partners, and studied software architectures for product families. Some of the partners continued in the ITEA project if99005 ESAPS (1999–2001). ITEA is the software development programme (Σ! 2023) within the European Eureka initiative. ITEA projects last for 2 years, and ESAPS was succeeded by CAFÉ (ITEA if00004) for 2001–2003 and FAMILIES (ITEA if02009). This fifth workshop was initially prepared within CAFÉ and the preparation continued in FAMILIES.

As usual Henk Obbink was the workshop chair, and Linda Northrop and Sergio Bandinelli were the co-chairs.

The programme committee was recruited from a collection of people who have shown interest in the workshop on earlier occasions:

Felix Bachmann
Sergio Bandinelli
Len Bass
Joe Bauman
Günter Böckle
Jan Bosch
Paul Clements
Jean-Marc DeBaud
Stefania Gnesi
André van den Hoek
Kari Känsälä
Peter Knauber
Philippe Kruchten
Frank van der Linden
Alessandro Maccari
Nenad Medvidovic
Robert Nord
Henk Obbink
Rob van Ommering
Dewayne Perry
Serge Salicki
Juha Savolainen
Klaus Schmid
Steffen Thiel
David Weiss

This workshop attracted many more papers than the previous ones. This is an indication that product family engineering has spread across the world, and has become an accepted way of doing software engineering. Many of those ahs to be rejected, and even now there were more papers accepted than what is good for a 3-day workshop. Even though we had enough discussions. Only authors of accepted papers were invited to the workshop. This time we had about 55 participants. However, we have to think about the format, and we may change it for the next occasion, planned for the fall of 2005.

The meeting place was again excellent. The weather was fine for that time of the year. The medieval city of Siena has a nice atmosphere and the famous Chianti wine is produced in the neighboring countryside. During the week of the workshop the first local wine of 2003 was opened in Siena. Alessandro Fantechi of the University of Florence, and Alessandro Maccari of Nokia acted as local hosts. It was done perfectly.

February 2004

Frank van der Linden

Lecture Notes in Computer Science 3014

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

Editorial Board

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Lecture Notes in Computer Science

For information about Vols. 1–2956

please contact your bookseller or Springer-Verlag

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

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

Vol. 3053: C. Bussler, J. Davies, D. Fensel, R. Studer (Eds.), The Semantic Web: Research and Applications. XIII, 490 pages. 2004.

Vol. 3052: W. Zimmermann, B. Thalheim (Eds.), Abstract State Machines 2004. Advances in Theory and Practice. XII, 235 pages. 2004.

Vol. 3047: F. Oquendo, B. Warboys, R. Morrison (Eds.), Software Architecture. X, 279 pages. 2004.

Vol. 3046: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications - ICCSA 2004. LIII, 1016 pages. 2004.

Vol. 3045: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications – ICCSA 2004. LIII, 1040 pages. 2004.

Vol. 3044: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications – ICCSA 2004. LIII, 1140 pages. 2004.

Vol. 3043: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications – ICCSA 2004. LIII, 1180 pages. 2004.

Vol. 3042: N. Mitrou, K. Kontovasilis, G.N. Rouskas, I. Iliadis, L. Merakos (Eds.), NETWORKING 2004, Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications. XXXIII, 1519 pages. 2004.

Vol. 3035: M.A. Wimmer (Ed.), Knowledge Management in Electronic Government. XII, 326 pages. 2004. (Subseries LNAI).

Vol. 3034: J. Favela, E. Menasalvas, E. Chávez (Eds.), Advances in Web Intelligence. XIII, 227 pages. 2004. (Subseries LNAI).

Vol. 3033: M. Li, X.-H. Sun, Q. Deng, J. Ni (Eds.), Grid and Cooperative Computing. XXXVIII, 1076 pages. 2004.

Vol. 3032: M. Li, X.-H. Sun, Q. Deng, J. Ni (Eds.), Grid and Cooperative Computing. XXXVII, 1112 pages. 2004.

Vol. 3031: A. Butz, A. Krüger, P. Olivier (Eds.), Smart Graphics. X, 165 pages. 2004.

Vol. 3029: B. Orchard, C. Yang, M. Ali (Eds.), Innovations in Applied Artificial Intelligence. XXI, 1272 pages. 2004.

Vol. 3028: D. Neuenschwander, Probabilistic and Statistical Methods in Cryptology. X, 158 pages. 2004.

Vol. 3027: C. Cachin, J. Camenisch (Eds.), Advances in Cryptology - EUROCRYPT 2004. XI, 628 pages. 2004.

Vol. 3026: C. Ramamoorthy, R. Lee, K.W. Lee (Eds.), Software Engineering Research and Applications. XV, 377 pages. 2004.

Vol. 3025: G.A. Vouros, T. Panayiotopoulos (Eds.), Methods and Applications of Artificial Intelligence. XV, 546 pages. 2004. (Subseries LNAI).

Vol. 3024: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 621 pages. 2004.

Vol. 3023: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 611 pages. 2004.

Vol. 3022: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 621 pages. 2004.

Vol. 3021: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 633 pages. 2004.

Vol. 3019: R. Wyrzykowski, J. Dongarra, M. Paprzycki, J. Wasniewski (Eds.), Parallel Processing and Applied Mathematics. XIX, 1174 pages. 2004.

Vol. 3015: C. Barakat, I. Pratt (Eds.), Passive and Active Network Measurement. XI, 300 pages. 2004.

Vol. 3014: F. van der Linden (Ed.), Software Product-Family Engineering. IX, 486 pages. 2004.

Vol. 3012: K. Kurumatani, S.-H. Chen, A. Ohuchi (Eds.), Multi-Agnets for Mass User Support. X, 217 pages. 2004. (Subseries LNAI).

Vol. 3011: J.-C. Régin, M. Rueher (Eds.), Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. XI, 415 pages. 2004.

Vol. 3010: K.R. Apt, F. Fages, F. Rossi, P. Szeredi, J. Váncza (Eds.), Recent Advances in Constraints. VIII, 285 pages. 2004. (Subseries LNAI).

Vol. 3009: F. Bomarius, H. Iida (Eds.), Product Focused Software Process Improvement. XIV, 584 pages. 2004.

Vol. 3008: S. Heuel, Uncertain Projective Geometry. XVII, 205 pages. 2004.

Vol. 3007: J.X. Yu, X. Lin, H. Lu, Y. Zhang (Eds.), Advanced Web Technologies and Applications. XXII, 936 pages. 2004.

Vol. 3006: M. Matsui, R. Zuccherato (Eds.), Selected Areas in Cryptography. XI, 361 pages. 2004.

Vol. 3005: G.R. Raidl, S. Cagnoni, J. Branke, D. W. Corne, R. Drechsler, Y. Jin, C.G. Johnson, P. Machado, E. Marchiori, F. Rothlauf, G.D. Smith, G. Squillero (Eds.), Applications of Evolutionary Computing. XVII, 562 pages. 2004.

Vol. 3004: J. Gottlieb, G.R. Raidl (Eds.), Evolutionary Computation in Combinatorial Optimization. X, 241 pages. 2004.

Vol. 3003: M. Keijzer, U.-M. O'Reilly, S.M. Lucas, E. Costa, T. Soule (Eds.), Genetic Programming. XI, 410 pages. 2004.

Vol. 3002: D.L. Hicks (Ed.), Metainformatics. X, 213 pages. 2004.

Vol. 3001: A. Ferscha, F. Mattern (Eds.), Pervasive Computing. XVII, 358 pages. 2004.

Vol. 2999: E.A. Boiten, J. Derrick, G. Smith (Eds.), Integrated Formal Methods. XI, 541 pages. 2004.

Vol. 2998: Y. Kameyama, P.J. Stuckey (Eds.), Functional and Logic Programming. X, 307 pages. 2004.

Vol. 2997: S. McDonald, J. Tait (Eds.), Advances in Information Retrieval. XIII, 427 pages. 2004.

Vol. 2996: V. Diekert, M. Habib (Eds.), STACS 2004. XVI, 658 pages. 2004.

Vol. 2995: C. Jensen, S. Poslad, T. Dimitrakos (Eds.), Trust Management. XIII, 377 pages. 2004.

Vol. 2994: E. Rahm (Ed.), Data Integration in the Life Sciences. X, 221 pages. 2004. (Subseries LNBI).

Vol. 2993: R. Alur, G.J. Pappas (Eds.), Hybrid Systems: Computation and Control. XII, 674 pages. 2004.

Vol. 2992: E. Bertino, S. Christodoulakis, D. Plexousakis, V. Christophides, M. Koubarakis, K. Böhm, E. Ferrari (Eds.), Advances in Database Technology - EDBT 2004. XVIII, 877 pages. 2004.

Vol. 2991: R. Alt, A. Frommer, R.B. Kearfott, W. Luther (Eds.), Numerical Software with Result Verification. X, 315 pages. 2004.

Vol. 2989: S. Graf, L. Mounier (Eds.), Model Checking Software. X, 309 pages. 2004.

Vol. 2988: K. Jensen, A. Podelski (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. XIV, 608 pages. 2004.

Vol. 2987: I. Walukiewicz (Ed.), Foundations of Software Science and Computation Structures. XIII, 529 pages. 2004.

Vol. 2986: D. Schmidt (Ed.), Programming Languages and Systems. XII, 417 pages. 2004.

Vol. 2985: E. Duesterwald (Ed.), Compiler Construction. X, 313 pages. 2004.

Vol. 2984: M. Wermelinger, T. Margaria-Steffen (Eds.), Fundamental Approaches to Software Engineering. XII, 389 pages. 2004.

Vol. 2983: S. Istrail, M.S. Waterman, A. Clark (Eds.), Computational Methods for SNPs and Haplotype Inference. IX, 153 pages. 2004. (Subseries LNBI).

Vol. 2982: N. Wakamiya, M. Solarski, J. Sterbenz (Eds.), Active Networks. XI, 308 pages. 2004.

Vol. 2981: C. Müller-Schloer, T. Ungerer, B. Bauer (Eds.), Organic and Pervasive Computing – ARCS 2004. XI, 339 pages. 2004.

Vol. 2980: A. Blackwell, K. Marriott, A. Shimojima (Eds.), Diagrammatic Representation and Inference. XV, 448 pages. 2004. (Subseries LNAI).

Vol. 2979: I. Stoica, Stateless Core: A Scalable Approach for Quality of Service in the Internet. XVI, 219 pages. 2004.

Vol. 2978: R. Groz, R.M. Hierons (Eds.), Testing of Communicating Systems. XII, 225 pages. 2004.

Vol. 2977: G. Di Marzo Serugendo, A. Karageorgos, O.F. Rana, F. Zambonelli (Eds.), Engineering Self-Organising Systems. X, 299 pages. 2004. (Subseries LNAI).

Vol. 2976: M. Farach-Colton (Ed.), LATIN 2004: Theoretical Informatics. XV, 626 pages. 2004.

Vol. 2973: Y. Lee, J. Li, K.-Y. Whang, D. Lee (Eds.), Database Systems for Advanced Applications. XXIV, 925 pages. 2004.

Vol. 2972: R. Monroy, G. Arroyo-Figueroa, L.E. Sucar, H. Sossa (Eds.), MICAI 2004: Advances in Artificial Intelligence. XVII, 923 pages. 2004. (Subseries LNAI).

Vol. 2971: J.I. Lim, D.H. Lee (Eds.), Information Security and Cryptology -ICISC 2003. XI, 458 pages. 2004.

Vol. 2970: F. Fernández Rivera, M. Bubak, A. Gómez Tato, R. Doallo (Eds.), Grid Computing. XI, 328 pages. 2004.

Vol. 2968: J. Chen, S. Hong (Eds.), Real-Time and Embedded Computing Systems and Applications. XIV, 620 pages. 2004.

Vol. 2967: S. Melnik, Generic Model Management. XX, 238 pages. 2004.

Vol. 2966: F.B. Sachse, Computational Cardiology. XVIII, 322 pages. 2004.

Vol. 2965: M.C. Calzarossa, E. Gelenbe, Performance Tools and Applications to Networked Systems. VIII, 385 pages. 2004.

Vol. 2964: T. Okamoto (Ed.), Topics in Cryptology – CT-RSA 2004. XI, 387 pages. 2004.

Vol. 2963: R. Sharp, Higher Level Hardware Synthesis. XVI, 195 pages. 2004.

Vol. 2962: S. Bistarelli, Semirings for Soft Constraint Solving and Programming. XII, 279 pages. 2004.

Vol. 2961: P. Eklund (Ed.), Concept Lattices. IX, 411 pages. 2004. (Subseries LNAI).

Vol. 2960: P.D. Mosses (Ed.), CASL Reference Manual. XVII, 528 pages. 2004.

Vol. 2959: R. Kazman, D. Port (Eds.), COTS-Based Software Systems. XIV, 219 pages. 2004.

Vol. 2958: L. Rauchwerger (Ed.), Languages and Compilers for Parallel Computing. XI, 556 pages. 2004.

Vol. 2957: P. Langendoerfer, M. Liu, I. Matta, V. Tsaoussidis (Eds.), Wired/Wireless Internet Communications. XI, 307 pages. 2004.

Table of Contents

Product Derivation

Transition to Family Development

Industrial Experience

Evolution

Decisions and Derivation

Research Topics and Future Trends

Jan Bosch[1], Henk Obbink[2], and Alessandro Maccari[3]

[1] University of Groningen, Department of Computing Science
PO Box 800, NL 9700 AV Groningen, The Netherlands
jan.bosch@cs.rug.nl
[2] Philips Research, Eindhoven
henk.obbink@philips.com
[3] Nokia Mobile Software, Web Service Technologies
PO Box 100, FIN 00045 Nokia Group, Finland
alessandro.maccari@nokia.com

1 Introduction

The PFE conference ended with a plenary debate revolving around research topics and future trends in the area of product family engineering. The organizers of the panel asked the conference participants to write down what they thought were the most interesting areas to research on. We took some of them for discussion during the plenary, and came out with a number of relevant trends. This paper summarizes the main findings, outlining the main research and development issues, and proposing, where appropriate, some hints for solutions.

2 Future Trends

The session started by summarizing the relevant future trends, as forecast by companies nowadays. The discussion highlighted a number of facts, listed below.

The *size* of software systems is on the continuous increase, both in the number of functional elements and in the amount of work that is necessary to develop a single product. An internal study by Philips indicates that the workload for development of an average product increases by a factor of 10 every 7-8 years. Moreover, the trend shows an increase in the number of errors that need to be corrected during the testing phase. This can be quantified in an order of magnitude every 10 years approximately. Studies like this also show that the trend is posed to continue in the future, as software is the core of innovation, and often constitutes the competitive edge, especially for companies that make mass-market products. The increase in size and errors is an obvious result of an increase in overall complexity.

The amount of *variability* in software product families is also visibly increasing, for two main reasons: i) there is a tendency to move variability from hardware to software, thus increasing the flexibility of the system configuration and decreasing the cost of variance; ii) design decisions are usually delayed as much as possible during the software development process. Often, variability is totally resolved only at the moment of installation of the software system. Both these trends imply that binding time is also continuously pushed as closer to runtime as possible. There is speculation that in the future all binding will be done at runtime, and a considerable amount of research has focused on this topic in recent times.

F. van der Linden (Ed.): PFE 2003, LNCS 3014, pp. 1–5, 2004.

3 General Trends, New Technologies, Methods and Processes

During the discussion, several new technologies were mentioned as being "hot". Dynamic software architectures, multi-dimensional separation of concerns, model-driven architecture (MDA), grid computing and server farms are the object of an increasing number of research papers, as well as the centre of many development projects. The recent standardization effort that several IT companies have carried out has put web services in the centre of attention. Many define web services as *the* new paradigm in distributed computing, and all major companies are trying to agree on a successful business model that would ramp up the adoption of the technology.

As concerns new methods and processes, software variability management is definitely a key part of the development and maintenance activities for product families. Also, agile software development and evolution methods seem to be among the most debated topics, at least in the research world, and their application to organizations that develop product families must be investigated (for instance, by means of experience reports and case studies). Finally, to reflect the importance of web services, the concept of web service centric software engineering is gaining popularity, and a vast take-up can be forecast if web services really turn out to be a major development in computer science and business alike.

These new methods, however, are refinements (at different levels of detail) of the "traditional" software engineering methods that are already used by organizations that develop software product family. The general agreement was that the community hasn't recently witnessed any radical technological innovation of a substantial importance. In contrast, it was remarked that the need for innovation, at least on the European front, seems to be pressing. Software development companies have a tendency to outsource most of their development (and therefore most of their knowledge) to countries where programmers' wages are low. The general trend for companies based in the European Union is to buy infrastructure (e.g. operating systems) from the United States, and develop the software applications that run on top of it outside the Union. This generates a lack of innovation and drive that is threatening Europe's position as one of the world's leading countries for software development. The participants to the conference remarked that the research and development community should tackle this problem, and find solutions that keep competitiveness in the continent.

The discussion then switched to a number of technical topics that were deemed important for future research. The following sections briefly analyze the main ones.

3.1 Variability Management

The conference participants felt that the topic of variability management, though extensively treated in research and even dedicated workshops, still left some important research questions unanswered. Dependencies between variation points and variants, for instance, have not yet been understood. A parallel problem is feature interaction, which looks at the same issues, but centered on requirements. This brings up research questions on the relationship among variability points across abstraction levels, and along the product lifecycle.

Another topic that deserves attention is unsystematic variability. Do the same methods apply when the variability factors cannot be assumed to behave in the same way across time?

In general, there was a general feeling that the phenomenon of variability needs to be better understood, perhaps by means of practical or empirical research. People remarked that variation is still not (easily enough) manageable, and that a definition and implementation of appropriate variation mechanisms is needed before the topic is fully understood and companies can apply the corresponding methodologies.

3.2 Product Family Lifecycle (Requirements, Architecture, Implementation)

The discussion covered the interesting topic lifecycle management for product family. We know that managing the development and evolution of a product family requires somewhat different approaches and methodologies than those used with a self-standing software system. In particular, the conference delegates underlined the importance of architectural assessment before evolutionary maintenance activities take place. People also advocated a unified approach and technology for all the activities in a product family lifecycle (requirements, architecture, design, code, documentation, test cases, and so forth in a product family).

In general, there was a common agreement on the need to narrow the gap between research on new features and applications and the actual implementation of these features and applications in real products. The community should get closer to the "real" world, and continuously test the methods, processes and features that are object of research against the needs of companies and non-profit organizations that operate in the sector.

3.3 Product Derivation and Tool Support

In this line of discussion, it was remarked that some sort of "expert-system" derivation of product family instances would help the maintenance and evolution process. Certainly, in many cases tools can effectively support the management complexity (especially for architectural model generation and reverse engineering).

During the discussing it was emphasized that, in order to be effective and widely adopted, tools must be simple and must not require more work than what they are supposed to save.

3.4 Evolution

The conference participants spent some time discussing the topic of product family evolution. It was remarked that, especially in the requirements and architecture areas, evolution patterns haven't been studied well enough, and mechanisms that help companies evolve a product line are still relatively unexplored. More research is advocated on this topic.

3.5 Validation of Product Family Benefits

Validation was felt to be an essential issue. A relevant slice of the software product family methodologies that this community has researched in the past years still lack experimental validation. This holds for qualitative aspects, as the impact of product family engineering approaches on software quality has not yet been estimated. However, we also miss an extensive quantitative estimation of the (mostly economical) trade-off between the costs that the introduction of a product family approach bears and the benefits it brings. It was remarked that economic models should focus on cost, but not limit to it.

Finally, some people observed that most of the techniques and methods that the community has investigated in the last few years have not been experimentally validated in the large scale. With the exception of a few large companies, most of the case studies that are usually presented in research conferences concern small and medium organizations. The scalability of the approach is not automatically guaranteed, as the increase in the company size brings additional complexity and dependencies. This should be further investigated.

Nevertheless, remarks were heard about the absence of a fully established practice for product family engineering for small/medium enterprises. In such companies, the

3.6 Organization and Management

The presence of several people with experience in industrial software product family development may have been the reason behind the large amount of discussion concerning organization and management.

One of the most compelling issues was felt to be the management of product populations, or multi-level product lines, where the variation points are hierarchically distributed. Also, the debate dwelled on methods that enable to make product families a success in a cross-organizational setting. One possible solution, which hasn't been looked into to a sufficient extent, is incremental introduction of product family engineering. Especially for large organizations, it was felt that it could help the companies realize the importance of the approach, and measure the benefits in a gradual way.

Interesting subjects of debate were the social issues related to product family management. In practice, a large part of the management work, especially in large company, involves contacts with other people. Social sciences have helped understand the constraints and relationships that rule the daily life of organizations, so they could be of help in this specific case.

Finally, it was remarked that we should focus some research effort on defining funding and business models to support strategic reuse across products in a family.

4 Conclusion

We conclude by listing some of the research topics that, according to the general opinion of the community, need urgent attention. One of the keynote speakers enumerated model-driven architecture, generative programming, and aspect-oriented programming as three key techniques that need development. Concerning aspect-

oriented modeling, some people remarked that it had already been listed as a topic during the ESAPS project, but no satisfying results were achieved. In fact, the partners couldn't even agree on a common definition. We should make a new start with a fresh approach in this research thread.

The conference ended with a discussion on what the future may bring. A number of basic laws have governed the evolution of computing technology, proving to be true until now despite large skepticism: Moore's law (transistor density doubles every 18 months), Gilder's law (communication bandwidth doubles every 12 months) and the storage law (storage capacity doubles every 9 months).

Assuming these laws continue to be valid, in 100 years computers will be billions of times faster. Humans will not be able to fully exploit the features of such systems, nor to write programs that master their complexity. Now, the mathematical foundations of our techniques and the techniques themselves are not progressing at the same speed as the machines that they are supposed to help us program and manage. We can conservatively forecast that already in 10 or 15 years we will not be able to handle such complexity. This leaves us with a lot of unused computational power.

One way to exploit this gap in computational power will be by using it in favour of easier programming paradigms. We will use the new power to simplify programming languages and the general user interface of these new machines. One possible outcome is the integration of the methodologies we have debated here into programming languages, thus getting them closer to our way of thinking.

For sure, performance optimization techniques are likely to disappear in the long term. However, what is to be of this new programming environment still remains the biggest open issue in the field.

Testing Variabilities in Use Case Models*

Erik Kamsties, Klaus Pohl, Sacha Reis, and Andreas Reuys

Software Systems Engineering, University of Duisburg-Essen
Schützenbahn 70, 45117 Essen, Germany
{kamsties,pohl,reis,reuys}@sse.uni-essen.de

Abstract. The derivation of system test cases for product families is difficult due to variability in the requirements, since each variation point multiplies the number of possible behaviors to be tested. This paper proposes an approach to develop domain test cases from use cases that contain variabilities and to derive application test cases from them. The basic idea to avoid combinatorial explosion is to preserve the variability in domain test cases. New strategies to capture variability in test cases are suggested, which in combination help dealing with all basic types of variability in a use case and in its relationships (e.g., <<include>>).

1 Introduction

System testing in the context of a product family has the same goal as in the context of a single system, checking the quality of software systems. Product family testing can be separated into *domain testing*, i.e., testing the core assets of a product family, and *application testing*, i.e., testing the artifacts of a particular customer-specific product. Domain system testing mainly comprises the production of reusable test artifacts. Application system testing comprises the reuse and adaptation of those artifacts and the development of additional ones due to customer-specific requirements. These test artifacts are used to carry out system testing on a customer-specific product.

In domain testing, variability prevents from directly applying test techniques known from single system engineering, because the variation points introduce another magnitude of complexity to the possible behaviors to be tested. The most obvious solution to this problem is to bypass domain testing and focus on application testing. However, it cannot be afforded to develop test cases from scratch for each application. But products of a family differ to some extent thus the problem arises how test artifacts developed for one product can be reused to ensure *systematic* testing of the following product.

The goal of the approach proposed in this paper is to derive system test cases for domain and application testing from use cases that contain variabilities. Efficient derivation of application test cases is accomplished by preserving the variabilities that

* This work was partially funded by the CAFÉ project "From Concept to Application in System Family Engineering"; Eureka Σ! 2023 Programme, ITEA Project ip00004 (BMBF, Förderkennzeichen 01 IS 002 C) and the state Nord-Rhein-Westfalia.

F. van der Linden (Ed.): PFE 2003, LNCS 3014, pp. 6–18, 2004.