Flavio Oquendo
Brian Warboys
Ron Morrison (Eds.)

LNCS 3047

# Software Architecture

## First European Workshop, EWSA 2004
## St Andrews, UK, May 2004
## Proceedings

Springer

Flavio Oquendo Brian Warboys
Ron Morrison (Eds.)

# Software Architecture

First European Workshop, EWSA 2004
St Andrews, UK, May 21-22, 2004
Proceedings


Springer

Volume Editors

Flavio Oquendo
Université de Savoie
Ecole Supérieure d'Ingénieurs d'Annecy - LISTIC
B.P. 806, 74016 Annecy Cedex, France
E-mail: Flavio.Oquendo@univ-savoie.fr

Brian Warboys
University of Manchester, Department of Computer Science
Manchester M13 9PL, UK
E-mail: bwarboys@cs.man.ac.uk

Ron Morrison
University of St Andrews, School of Computer Science
St Andrews, Fife KY16 9SS, UK
E-mail: ron@dcs.st-andrews.ac.uk

Library of Congress Control Number: 2004105864

CR Subject Classification (1998): D.2

ISSN 0302-9743
ISBN 3-540-22000-3 Springer-Verlag Berlin Heidelberg New York



Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11007821 06/3142 5 4 3 2 1 0

# Lecture Notes in Computer Science 3047

*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. 1–2952

please contact your bookseller or Springer-Verlag

Vol. 3083: W. Emmerich, A.L. Wolf (Eds.), Component Deployment. X, 249 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. 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, Knowledge Management in Electronic Government. XII, 342 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. 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. 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.

Vol. 2956: A. Dengel, M. Junker, A. Weisbecker (Eds.), Reading and Learning. XII, 355 pages. 2004.

Vol. 2954: F. Crestani, M. Dunlop, S. Mizzaro (Eds.), Mobile and Ubiquitous Information Access. X, 299 pages. 2004.

Vol. 2953: K. Konrad, Model Generation for Natural Language Interpretation and Analysis. XIII, 166 pages. 2004. (Subseries LNAI).

# Preface

The last decade has been one of great progress in the field of software architecture research and practice. Software architecture has emerged as an important subdiscipline of software engineering. A key aspect of the design of any software system is its architecture, i.e. the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution (as defined in the Recommended Practice for Architectural Description of Software-Intensive Systems – IEEE Std 1471-2000).

The First European Workshop on Software Architecture (EWSA 2004) provided an international forum for researchers and practitioners from academia and industry to discuss a wide range of topics in the area of software architecture, and to jointly formulate an agenda for future research in this field.

EWSA 2004 distinguished among three types of papers: research papers (which describe authors' novel research work), experience papers (which describe real-world experiences related to software architectures), and position papers (which present concise arguments about a topic of software architecture research or practice).

The Program Committee selected 19 papers (9 research papers, 4 experience papers, and 6 position papers) out of 48 submissions from 16 countries (Australia, Brazil, Canada, Chile, Finland, France, Germany, Italy, Japan, Korea, The Netherlands, Spain, Switzerland, Turkey, UK, USA). All submissions were reviewed by three members of the Program Committee. Papers were selected based on originality, quality, soundness and relevance to the workshop. In addition, the workshop included five invited papers presenting European Union projects related to software architecture: ARCHWARE, CONIPF, FABRIC, MODA-TEL, and OMEGA. Credit for the quality of the proceedings goes to all authors of papers.

We would like to thank the members of the Program Committee (Ilham Alloui, Dharini Balasubramaniam, Jan Bosch, Harald Gall, David Garlan, Mark Greenwood, Valérie Issarny, Volker Gruhn, Philippe Kruchten, Nicole Levy, Radu Mateescu, Carlo Montangero, and Dewayne Perry) for providing timely and significant reviews, and for their substantial effort in making EWSA 2004 a successful workshop.

The EWSA 2004 submission and review process was extensively supported by the Paperdyne Conference Management System. We are indebted to Volker Gruhn and his team, in particular Dirk Peters and Clemens Schäfer, for their excellent support.

The workshop was held in conjunction with the 26th International Conference on Software Engineering (ICSE 2004). We would like to acknowledge Michael Goedicke and the other members of the ICSE Organizing Committee for their assistance, during the organization of EWSA 2004, in creating this co-located event.

We would also like to acknowledge the prompt and professional support from Springer-Verlag, who published these proceedings in printed and electronic form as part of the Lecture Notes in Computer Science series.

May 2004

Flavio Oquendo
Brian Warboys
Ron Morrison

## Program Committee

### Program Chairs

- Flavio Oquendo
  University of Savoie – LISTIC, Annecy, France
  *flavio.oquendo@univ-savoie.fr*
- Brian Warboys
  University of Manchester, UK
  *bwarboys@cs.man.ac.uk*

### Committee Members

- Ilham Alloui
  University of Savoie – LISTIC, Annecy, France
  *ilham.alloui@univ-savoie.fr*
- Dharini Balasubramaniam
  University of St Andrews, UK
  *dharini@dcs.st-and.ac.uk*
- Jan Bosch
  University of Groningen, The Netherlands
  *jan.bosch@cs.rug.nl*
- Harald Gall
  Technical University of Vienna, Austria
  *gall@infosys.tuwien.ac.at*
- David Garlan
  Carnegie Mellon University, USA
  *garlan@cs.cmu.edu*
- Mark Greenwood
  University of Manchester, UK
  *markg@cs.man.ac.uk*
- Valérie Issarny
  INRIA Rocquencourt, France
  *valerie.issarny@inria.fr*
- Volker Gruhn
  University of Leipzig, Germany
  *volker.gruhn@informatik.uni-leipzig.de*
- Philippe Kruchten
  University of British Columbia, Vancouver, Canada
  *kruchten@ieee.org*
- Nicole Levy
  University of Versailles – PRiSM, France
  *nicole.levy@prism.uvsq.fr*
- Radu Mateescu
  INRIA Rhône-Alpes, Grenoble, France
  *radu.mateescu@inria.fr*

- Carlo Montangero
  University of Pisa, Italy
  *carlo.montangero@di.unipi.it*
- Dewayne Perry
  University of Texas at Austin, USA
  *perry@ece.utexas.edu*

## Organizing Committee

- Ron Morrison (Chair)
  University of St Andrews, UK
  *ron@dcs.st-andrews.ac.uk*
- Ferdinando Gallo
  Consorzio Pisa Ricerche, Italy
  *n.gallo@cpr.it*
- Hilary Hanahoe
  Consorzio Pisa Ricerche, Italy
  *h.hanahoe@trust-itservices.com*

## Sponsorship

EWSA 2004 was sponsored by the ArchWare European R&D Project: Architecting Evolvable Software – *www.arch-ware.org*. ArchWare is partially funded by the Commission of the European Union under contract No. IST-2001-32360 in the IST-V Framework Program.

# Table of Contents

## Research Papers



## Experience Papers

## Position Papers



## Invited Papers: European Projects in Software Architecture

# Sotograph - A Pragmatic Approach to Source Code Architecture Conformance Checking

Walter Bischofberger, Jan Kühl, and Silvio Löffler

Software-Tomography GmbH
{wb,jk,sl}@software-tomography.com
www.software-tomography.com

**Abstract.** In our experience the implementation of software systems frequently does not conform very closely to the planned architecture. For this reason we decided to implement source code architecture conformance checking support for Sotograph, our software analysis environment. Besides providing a conformance check for a single version of a system, Sotograph supports also trend analysis. I.e., the investigation of the evolution of a software system and the changes in architecture violations between a number of versions.

## Sotograph

Sotograph extracts information about a number of versions of a software system from source and byte code and manages this information in a relational database. The data model represents the information that can be extracted from C++ and Java. The information in a database can be analyzed and visualized with a closely integrated set of tools supporting source code architecture conformance checking, cycle analysis, metric based analysis, cross referencing between artifacts on different abstraction levels. Custom metrics and custom analyses can be implemented in an SQL extension.

## Supported Architecture Model

From our point of view software architectures describe how the architecture-level artifacts of a software system are intended to cooperate and between which artifacts relationships are forbidden.

The artifacts that are explicitly defined in source code such as classes, packages and name spaces are too fine-grained to serve as architecture-level artifacts. For this reason we aggregate fine-grained artifacts to architecture-level artifacts which we call subsystems. In software systems implemented in Java, subsystems consist typically of a number of packages. In software systems implemented in C subsystems are usually aggregations of the artifacts defined in the source code located in a number of directories. C++ subsystems consist either of aggregations of directories or name spaces.

In defining subsystems it makes sense not just to define the artifacts they consist of, but also the artifacts that form their export interfaces. This permits for a first level of architecture conformance checking.

F. Oquendo et al. (Eds.): EWSA 2004, LNCS 3047, pp. 1–9, 2004.

Architectures can either be described as graphs or as layered architectures. In a graph the arcs define for artifact pairs which references between them are allowed or forbidden. In a layered architecture the layering implicitly forbids upward references and in some cases multi-layer downward references.

We decided to use layered architectures because

- the layers provide a further abstraction level which makes a layered architecture much easier to understand than an arbitrary graph,
- layered architectures are simple to define because the restrictions they impose are inherently given by the layering,
- architects frequently define their top-level architectures as layered systems.

The disadvantage of layered architectures compared to arbitrary graphs is that they are less expressive. This means that sometimes several layered architecture models are needed to describe all relevant architectural restrictions.

## Basic Idea of Architecture Conformance Checking

Figure 1 depicts an example of a subsystem based layered architecture and the kind of illegal relationships a source code architecture conformance checker can automatically detect.

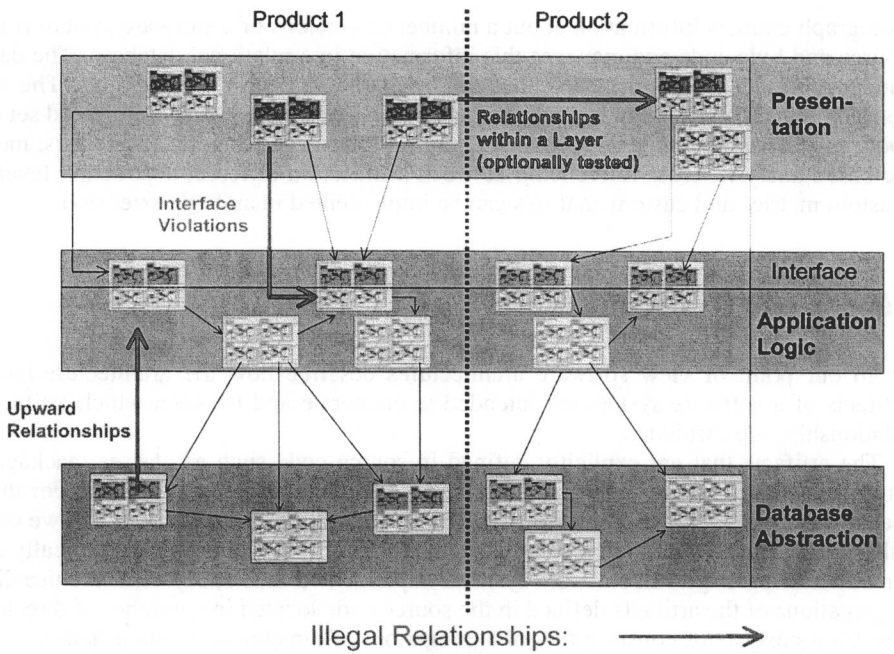


**Fig. 1.** Sample layered architecture.

***Illegal Upward Relationships.*** It is inherent to layered architectures that references from lower to upper layers are not allowed.

***Interface Violations.*** Usage of non interface artifacts of subsystems by other subsystems is not allowed.

***Several Layer Downward Relationships.*** Depending on the kind of modeled architecture this may be legal or illegal. In our example it is illegal that the presentation layer uses the database abstraction layer directly.

***Illegal Relationships within a Layer.*** These can be illegal for several reasons. A typical example is a line of products where products are flexibly configured based on a set of services and tools. In such an architecture groups of services and tools implementing independent functionality should not know each other.

## Architecture Conformance Checking

Architecture conformance checking consists of specifying an architecture which is to be checked, and in checking whether the source code adheres to this architecture.

### Specifying an Architecture

Sotograph represents the following abstraction levels in its databases: Architecture, architecture layer, subsystem, package/directory, file, class, symbol.

All abstraction levels up to the package/directory level can be generated automatically from the source code of the system to be analyzed. On the package/directory level we use packages for java systems and directories for C/C++ systems. In the following we will talk about packages when referring to the package/directory level.

It can make sense to specify several subsystem and architecture models for a code base that emphasize different aspects. Subsystems aggregating packages and architecture layers aggregating subsystems are specified in simple languages described below.

Most often subsystems can be defined with rules specifying the root packages of package trees that should be aggregated into subsystems. The source code below shows an example defining the subsystems Public.util, Public.guiutil, and Public.plugins. The interface of these subsystems consists of their root packages.

```
RuleBasedSubsystem Public {
  InterfacePath "";
  Packages "com.sotogra.'(util|guiutil|plugins)'";
}
```

Typically, only a few exceptions have to be specified by enumerating the packages belonging to subsystems. Enumerating specification makes sense when the subsystem model can be generated, e.g., based on the jar files that are generated for a system or based on project specific information sources.

The definition of an architecture consists of an enumeration of its layers, a few attributes, and the packages they contain. The source code below shows an excerpt of Sotograph's overview architecture model.

```
ArchitectureModel Overview {
  Uses Default; // underlying subsystem model
  ArchitectureLayer Manager {
    // this layer is allowed to use all lower layers
    InterLayerUsage = True;
    // the subsystems pertaining to this layer may us
    // each other
    IntraLayerUsage = True;
    // the list of subsystems pertaining to the layer
    Subsystem Tools.manager;
    Subsystem Access;
  }
  ArchitectureLayer ToolsAndServices {
    InterLayerUsage = True;
    // the subsystems pertaining to this layer
    // may not us each other
    IntraLayerUsage = False;
    // the subsystems pertaining to the layer selected
    // with a regular expression
    Subsystems "Tools.([^m]|met).*";
  }
  ArchitectureLayer ToolInfrastructure {
    InterLayerUsage = True;
    IntraLayerUsage = True;
    Subsystem Base.annotation;
    Subsystem Base.dbupdate;
    Subsystem Base.migration;
  }
```

**Investigating the Correspondence between Architecture and Source Base**

Sotograph can detect the references in a source base that do not correspond to an architecture model and the corresponding subsystem model. Figure 2 shows the number of illegal references on subsystem level for the code bases of Sotograph version 0.95 and 0.96. The rows are sorted according to the difference between the two versions (the last column). This view shows at a glance between which subsystems which kinds of illegal references were added and removed.

This information can then, for example, be visualized to get an overview of how many of the subsystems are illegally used. Figure 3 shows the subsystem inheritance graph with the illegally referenced subsystems marked.

Finally it is possible to zoom in by double clicking the rows of the table displayed in Figure 2. Zooming in means to first investigate the illegal references aggregated on package-level and then on the level of basic references. Figure 4 shows a basic reference-level view, which enumerates the new and existing illegal references between package *architecture* and package *jflex*. The next double click displays the source code defining the illegal reference in the IDE of choice.

Exceptions of Trend Architecture Model Sotograph for v1 = 0.95 and v2 = 0.96

Filter ~ Help

| sub... | subRefing | su... | subRefed | errorKind | v1 | v2 | diff |
|---|---|---|---|---|---|---|---|
| 65965 | Default.Db | 65962 | Default.Base.table | UPWARD | 829 | 788 | -41 |
| 65965 | Default.Db | 65959 | Default.Base.guiutil | UPWARD | 364 | 359 | -5 |
| 65966 | Default.Tools.architecture | 65976 | Default.Tools.subsystem | INTERFACE | 30 | 28 | -2 |
| 65958 | Default.Base.annotation | 65959 | Default.Base.guiutil | INTERFACE | 39 | 39 | 0 |
| 65959 | Default.Base.guiutil | 65972 | Default.Tools.manager | UPWARD | 11 | 11 | 0 |
| 65961 | Default.Base.projecttree | 65969 | Default.Tools.dbview | UPWARD | 1 | 1 | 0 |
| 65962 | Default.Base.table | 65959 | Default.Base.guiutil | INTERFACE | 8 | 8 | 0 |
| 65963 | Default.Base.tool | 65969 | Default.Tools.dbview | UPWARD | 2 | 2 | 0 |
| 65963 | Default.Base.tool | 65971 | Default.Tools.graph | UPWARD | 15 | 15 | 0 |
| 65963 | Default.Base.tool | 65975 | Default.Tools.result | UPWARD | 8 | 8 | 0 |
| 65963 | Default.Base.tool | 65976 | Default.Tools.subsystem | INTERFACE | 2 | 2 | 0 |
| 65964 | Default.Base.util | 65959 | Default.Base.guiutil | UPWARD | 19 | 19 | 0 |
| 65965 | Default.Db | 65958 | Default.Base.annotation | UPWARD | 123 | 123 | 0 |
| 65965 | Default.Db | 65964 | Default.Base.util | INTERFACE | 7 | 7 | 0 |
| 65965 | Default.Db | 65972 | Default.Tools.manager | UPWARD | 26 | 26 | 0 |
| 65965 | Default.Db | 65976 | Default.Tools.subsystem | INTERFACE | 6 | 6 | 0 |
| 65965 | Default.Db | 65977 | Default.Tools.trend | UPWARD | 28 | 28 | 0 |
| 65973 | Default.Tools.metric | 65977 | Default.Tools.trend | INTRA | 139 | 139 | 0 |
| 65975 | Default.Tools.result | 65972 | Default.Tools.manager | UPWARD | 3 | 3 | 0 |
| 65976 | Default.Tools.subsystem | 65964 | Default.Base.util | INTERFACE | 42 | 42 | 0 |
| 65964 | Default.Base.util | 65963 | Default.Base.tool | UPWARD | 0 | 2 | 2 |
| 65974 | Default.Tools.query | 65964 | Default.Base.util | INTERFACE | 94 | 96 | 2 |
| 65964 | Default.Base.util | 65972 | Default.Tools.manager | UPWARD | 56 | 60 | 4 |
| 65966 | Default.Tools.architecture | 65964 | Default.Base.util | INTERFACE | 38 | 42 | 4 |
| 65963 | Default.Base.tool | 65959 | Default.Base.guiutil | INTERFACE | 31 | 36 | 5 |
| 65966 | Default.Tools.architecture | 65977 | Default.Tools.trend | INTRA | 74 | 80 | 6 |
| 65965 | Default.Db | 65963 | Default.Base.tool | UPWARD | 738 | 802 | 64 |

**Fig. 2.** Architecture violations table.

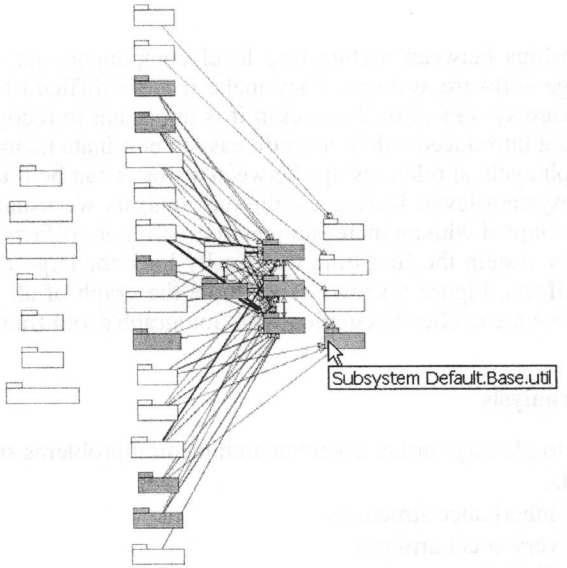


**Fig. 3.** Illegally referenced subsystems marked in subsystem inheritance graph.