10
References

ABREU, F.; GOULÃO, M.; ESTEVES, R. (1995) Toward the Design Quality Evaluation of Object-Oriented Software Systems, **Proceedings of the 5th International Conference on Software Quality**, Austin, Texas, USA, October 1995.

AOSD-EUROPE PROJECT. **European Network of Excellence on Aspect-Oriented Software Development.** http://www.aosd-europe.net/, 2007.

THE ASPECTJ TEAM. **The AspectJ™ Programming Guide.** http://www.eclipse.org/aspectj. December 2007.

BALDWIN, C.; CLARK, K. **Design Rules – Volume 1: The Power of Modularity**, The MIT Press, March 2000.

BACHMANN, F.; BASS, L.; KLEIN, M. Deriving Architectural Tactics: A Step Toward Methodical Architectural Design, **Technical Report CMU/SEI-2003-TR-004**, Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University, 2003.

BELLIFEMINE, F.; RIMASSA, G.; POGGI, A. JADE: A FIPA-compliant agent framework, **Proceedings of the 4th International Conference on the Practical Applications of Agents and Multi-Agent Systems (PAAM'99)**, UK, pp.97–108, 1999.

BARBOSA, R. M.; GOLDMAN, A. MobiGrid, **Proceedings of the International Workshop on Mobility Aware Technologies and Applications (MATA 2004)**, LNCS 3284, pp.147–157, 2004.

BARTOLOMEI, T.; GARCIA, A.; SANT'ANNA, C.; FIGUEIREDO, E. Towards a Unified Coupling Framework for Measuring Aspect-Oriented Programs, **Proceedings of the Workshop on Software Quality Assurance (SOQUA)**. Portland, 2006.

BASS L.; CLEMENTS, P.; KAZMAN, R. Software **Architecture in Practice**, Addison Wesley, 2nd Edition, 2003.

BASS, L., KLEIN, M., NORTHROP, L. Identifying Aspects Using Architectural Reasoning. **Proceedings of the Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design**, held in conjunction with AOSD'04, 2004.

BELADY, L. A.; LEHMAN, M. M. A model of large program development. **IBM Systems Journal**. 15(3), pp. 225–252, 1976.

BOOCH, G. **Object-oriented Analysis and Design with Applications**, 2nd ed., Benjamin/Cummings, 1994.

BRIAND, L.; MORASCA, S.; BASILI, V. Measuring and Assessing Maintainability at the End of High Level Design, **Proceedings of the IEEE Conference of Software Maintenance**, 1993.

BRIAND, L.; DALY, J.; WUEST, J. A Unified Framework for Cohesion Measurement in Object-Oriented Systems, **Empirical Software Engineering - An International Journal**, 3(1), pp. 65-117, 1998.

BRIAND, L.; DALY, J.; WUST, J. A Unified Framework for Coupling Measurement in Object-Oriented Systems. **IEEE Transactions on Software Engineering**, 25 (1), pp. 91-121, 1999.

BUSCHMANN, F.; MEUNIER, R.; ROHNERT, H.; SOMMERLAD, P.; STAL, M. **Pattern-Oriented Software Architecture:** A System of Patterns. John Wiley, 1996.

CACHO, N.; SANT'ANNA, C.; FIGUEIREDO, E.; GARCIA, A.; BATISTA, T.; LUCENA, C. Composing Design Patterns: A Scalability Study of Aspect-Oriented Programming, **Proceedings of the 5th International Conference on Aspect-Oriented Software Development (AOSD'06)**, Bonn, Germany, 20-24, pp. 109-121, 2006a.

CACHO, N.; BATISTA, T.; GARCIA, A.; SANT'ANNA, C; BLAIR, G. Improving Modularity of Reflective Middleware with Aspect-Oriented Programming, **International Workshop on Software Engineering for Middleware (SEM'06)**, Portland, Oregon, November, 2006b.

CACHO, N.; BATISTA, T.; GARCIA, A.; SANT'ANNA, C. Aspect Open-ORB: An Aspect-Oriented Reflective Middleware. **Proceedings of the Brazilian Symposium on Computer Networks and Distributed Systems (SBRC 2007)**, 2007.

CECCATO, M.; TONELLA, P. Measuring the Effects of Software Aspectization, **Proceedings of the 1st Workshop on Aspect Reverse Engineering (WARE 2004)**, November 2004.

CHAVEZ, C.; LUCENA, C. Design-level support for aspect-oriented software development, **Proceedings of the Workshop on Advanced Separation of Concerns in Object-Oriented Systems**, held in conjunction with OOPSLA'01, 2001.

CHAVEZ, C.; GARCIA, A.; KULESZA, U.; SANT'ANNA, C.; LUCENA, C. Taming Heterogeneous Aspects with Crosscutting Interfaces, **Proceedings of the 19th Brazilian Symposium on Software Engineering**, Uberlândia, pp. 216-231, 2005.

CHIDAMBER, S.; KEMERER, C. Towards a Metrics Suite for Object Oriented design, in A. Paepcke, (ed.) **Proceedings of the Conference on Object-Oriented Programming: Systems, Languages and Applications (OOPSLA'91)**, October 1991. Published in SIGPLAN Notices, 26 (11), pp. 197-211, 1991.

CHIDAMBER, S.; KEMERER, C. A Metrics Suite for Object Oriented Design, **IEEE Transactions on Software Engineering**, 20 (6), pp. 476-493, 1994.

CHURCHER, N.; SHEPPERD, M. Towards a Conceptual Framework for Object Oriented Software Metrics, **Software Engineering Notes**, 20 (2), pp. 69-76, 1995.

CLEMENTS, P.; NORTHROP, L. **Software Product Lines: Practices and Patterns**. Addison-Wesley, 2002.

CLEMENTS, P.; KAZMAN, R.; KLEIN, M. **Evaluating Software Architectures: Methods and Case Studies**, USA: Addison-Wesley, 2002

CLEMENTS, P.; BACHMANN, F.; BASS, L.; GARLAN, D.; IVERS, J.; LITTLE, R.; NORD, R.; STAFFORD, J. **Documenting Software Architectures: Views and Beyond**, Addison Wesley, 2003.

DIJKSTRA, E. **A Discipline of Programming**. Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

DOBRICA, L.; NIEMELA, E. A Survey on Software Architecture Analysis Methods, **IEEE Transactions on Software Engineering**, 28 (7), pp. 638-653, July 2002.

DUCASSE, S.; GIRBA, T.; KUHN, A. Distribution Map, **Proceedings of the Int'l Conference on Software Maintenance (ICSM)**, Philadelphia, pp. 203-212 2006.

EADDY, M.; AHO, A.; MURPHY, G. Identifying, Assigning, and Quantifying Crosscutting Concerns, **Proceedings of the Workshop on Assessment of Contemporary Modularization Techniques (ACoM)**, held in conjunction with ICSE'07, 2007.

EADDY, M.; ZIMMERMANN, T.; SHERWOOD, K.; GARG, V.; MURPHY, G.; NAGAPPAN, N.; AHO, A. Do Crosscutting Concerns Cause Defects?, **IEEE Transactions on Software Engineering** (to appear), 2008a.

EADDY, M.; AHO, A.; ANTONIOL, G.; GUÉHÉNEUC, Y. Cerberus: Tracing Requirements to Source Code Using Static, Dynamic, and Semantic Analysis, **International Conference on Program Comprehension (ICPC)** (to appear), Amsterdam, The Netherlands, June 10-13, 2008b.

ECLIPSE FOUNDATION. **Eclipse Project.** http://www.eclipse.org/, 2007a.

ECLIPSE FOUNDATION. **The Eclipse CVS Platform.** http://www.eclipse.org/eclipse/platform-cvs/, 2007b.

EICK, S.; GRAVES, T.; KARR, A.; MARRON, J.; MOCKUS, A. Does Code Decay? Assessing the Evidence from Change Management Data, **IEEE Transactions on Software Engineering**, 27(1), pp. 1-12, January 2001.

ELRAD, T.; FILMAN, R.; BADER, A. Aspect-Oriented Programming, **Communication of the ACM**, 44 (10), pp. 29-32, October 2001

EARLY ASPECTS AT ICSE. **Workshop on Aspect-Oriented Requirements Engineering and Architecture Design**, in conjunction with the International Conference on Software Engineering (ICSE'2007), Minneapolis, USA. http://www.early-aspects.net/, 2007.

FENTON, N.; PFLEEGER, S. **Software Metrics: A Rigorous and Practical Approach**, 2.ed. London: PWS, 1997.

FIGUEIREDO, E.; GARCIA, A.; LUCENA, C. AJATO: an AspectJ Assessment Tool, **Proceedings of the ECOOP.06**, Demo Session, Nantes, France, 2006.

FIGUEIREDO, E.; SANT'ANNA, C.; GARCIA, A.; BARTOLOMEI, T.; CAZZOLA, W. MARCHETTO, A. On the Maintainability of Aspect-Oriented Software: A Concern-Oriented Measurement Framework. **Proceedings of the 12th European Conference on Software Maintenance and Reengineering (CSMR'08)**, Athens, Greece, 2008a.

FIGUEIREDO, E.; CACHO, N.; SANT'ANNA, C.; MONTEIRO, M.; KULESZA, U.; GARCIA, A.; SOARES, S.; FERRARI, F.; KHAN, S.; FILHO, F.; DANTAS, F. Evolving Software Product Lines with Aspects: An Empirical Study on Design Stability. **Proceedings of the 30th International Conference on Software Engineering (ICSE'08)**, Leipzig, Germany, pp. 10-18, 2008b.

FIGUEIREDO, E.; CACHO, N.; SANT'ANNA, C.; MONTEIRO, M.; KULESZA, U.; GARCIA, A.; SOARES, S.; FERRARI, F.; KHAN, S.; FILHO, F.; DANTAS, F. **Evolving Software Product Lines with Aspects**, http://www.lancs.ac.uk/postgrad/figueire/spl/icse08/, 2008c.

FILHO, F.; CACHO, N.; FIGUEIREDO, E.; MARANHÃO, R.; GARCIA, A.; RUBIRA, C. Exceptions and Aspects: The Devil is in the Details, **Proceedings of the 14th International Symposium on Foundations of Software Engineering (FSE)**, Portland, Oregon, USA, pp. 152-162, 2006.

FILHO, F.; GARCIA, A.; RUBIRA, C. Extracting Error Handling to Aspects: A Cookbook. **Proceedings of the 23rd IEEE International Conference on Software Maintenance (ICSM'07)**, Paris, IEEE Computer Society, pp. 134–143, October 2007.

FILMAN, R.; ELRAD, T.; CLARKE, S.; AKSIT, M. **Aspect-Oriented Software Development**. Addison-Wesley, 2005.

FOWLER, M. **Refactoring: Improving the Design of Existing Code**. Addison-Wesley, Reading, MA, USA, 1999.

GAMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley, Reading, 1995.

GARCIA, A. **From Objects to Agents: An Aspect-Oriented Approach**, PhD thesis, Pontifical Catholic University of Rio de Janeiro, Brazil, April, 2004.

GARCIA, A.; SANT'ANNA, C.; CHAVEZ, C.; SILVA, V.; LUCENA, C.; STAA, A. Separation of Concerns in Multi-Agent Systems: An Empirical Study, In Lucena C. et al (Eds). **Software Engineering for Multi-Agent Systems II**. Springer-Verlag, LNCS 2940, 2004a.

GARCIA, A.; KULESZA, U.; LUCENA, C. Aspectizing Multi-Agent Systems: From Architecture to Implementation. In Choren, R. et al (Eds.) **Software Engineering for Multi-Agent Systems (SELMAS 2004)**, Springer, LNCS 3390, 2004, pp.121–143, 2004b.

GARCIA, A.; LUCENA, C.; COWAN, D.D. Agents in Object-Oriented Software Engineering. **Software: Practice and Experience**, 34(5), pp. 489–521, 2004c.

GARCIA, A.; SANT'ANNA, C.; FIGUEIREDO, E.; KULESZA, U.; LUCENA, C.; STAA, A. Modularizing Design Patterns with Aspects: A Quantitative Study. **Proceedings of the 4th International Conference on Aspect-Oriented Software Development (AOSD'05)**, Chicago, USA, March 2005.

GARCIA, A.; CHAVEZ, C.; BATISTA, T.; SANT'ANNA, C.; KULESZA, U.; RASHID, A.; LUCENA, C, On the Modular Representation of Architectural Aspects, **Proceedings of the 3rd European Workshop on Software Architecture**, Nantes, France, 2006a.

GARCIA, A.; SANT'ANNA, C.; FIGUEIREDO, E.; KULESZA, U.; LUCENA, C.; STAA, A. Modularizing Design Patterns with Aspects: A Quantitative Study. **Transactions on Aspect-Oriented Software Development I**, LNCS 3880, Springer, pp. 36-74, 2006b.

GARCIA, A.; LUCENA, C. Taming Heterogeneous Agent Architectures with Aspects. **Communications of the ACM**, May 2008 (accepted to appear).

GARLAN, D.; MONROE, R.; WILE, D. ACME: An Architecture Description Interchange Language, **Proceedings of CASCON'97**, Toronto, Ontario, pp. 169-183, 1997.

GHEZZI, C.; JAZAYERI, M.; MANDRIOLI, D. **Fundamentals of Software Engineering**, Englewood Cliffs, NJ., USA: Prentice Hall, 1991.

GREENWOOD, P.; BARTOLOMEI, T.; FIGUEIREDO, E.; DOSEA, M.; GARCIA, A.; CACHO, N.; SANT'ANNA, C.; SOARES, S.; BORBA, P.; KULESZA, U.; RASHID, A. On the Impact of Aspectual Decompositions on Design Stability: An Empirical Study, **Proceedings of the 21st European Conference on Object-Oriented Programming (ECOOP.07)**, Germany, 2007a.

GREENWOOD, P.; GARCIA, A.; RASHID, A.; FIGUEIREDO, E.; SANT'ANNA, C.; CACHO, N.; SAMPAIO, A.; SOARES, S.; BORBA, P.; DOSEA, M.; RAMOS, R.; KULESZA, U.; BARTOLOMEI, T.; PINTO, M.; FUENTES, LIDIA.; GAMEZ, N.; MOREIRA, A.; ARAUJO, J.; BATISTA, T.; MEDEIROS, A.; DANTAS, F.; FERNANDES, L.; WLOKA, J.; CHAVEZ, C.; FRANCE, R.; BRITO, I. On the Contributions of an End-to-End AOSD Testbed. **Proceedings of the Early Aspects at ICSE: Workshops in Aspect-Oriented Requirements Engineering and Architecture Design**, 2007b.

GREENWOOD, P.; BARTOLOMEI, T.; FIGUEIREDO, E.; DOSEA, M.; GARCIA, A.; CACHO, N.; SANT'ANNA, C.; SOARES, S.; BORBA, P.; KULESZA, U.; RASHID, A. **Aspect Interaction and Design Stability: An Empirical Study**, Available from http://www.comp.lancs.ac.uk/computing/users/greenwop/ecoop07, 2007c.

GRISWOLD, W.; YUAN, J.; KATO, Y. Exploiting The Map Metaphor In A Tool For Software Evolution, **Proceedings of the 23rd International Conference on Software Engineering**, pp. 265–274, 2001.

HANNEMANN, J.; KICZALES, G. Overcoming the Prevalent Decomposition in Legacy Code, **Proceedings of the ICSE Workshop on Advanced Separation of Concerns in Software Engineering**, 2001.

HANNEMANN, J., KICZALES, G. Design Pattern Implementation in Java and AspectJ, **Proceedings of the Conference on Object-Oriented Programming: Systems, Languages and Applications (OOPSLA'02)**, pp. 161-173, 2002.

HARRISON, W.; OSSHER, H.; SUTTON JR., S.; TARR, P. **Concern Modeling in the Concern Manipulation Environment**. Technical Report RC23344, IBM Research, 2004.

IEEE, **IEEE Standard Glossary of Software Engineering Terminology**, IEEE Std 610.12-1990, 1990.

KICZALES, G.; LAMPING, J.; MENDHEKAR, A.; MAEDA, C.; LOPES, C.; LOINGTIER, J-M.; IRWIN, J. Aspect-oriented programming, **Proceedings of the 11th European Conference on Object-Oriented Programming**, LNCS 1241, Springer-Verlag, pp.220–242, 1997.

KICZALES, G.; HILSDALE, E.; HUGUNIN, J.; KERSTEN, M.; PALM, J.; GRISWOLD, G. Getting started with AspectJ, **Communications of the ACM**, October, 44(10), pp.59–65, 2001.

KITCHENHAM, B.; PFLEEGER, S.; FENTON, N. Towards a Framework for Software Measurement Validation, **IEEE Transactions on Software Engineering**, 21(12), 1995.

KULESZA, U..; GARCIA, A.; LUCENA, C. Towards a Method for the Development of Aspect-Oriented Generative Approaches, **Workshop on Early Aspects, OOPSLA'04**, Vancouver, Canada, November 2004.

KULESZA, U.; SANT'ANNA, C.; GARCIA, A.; COELHO, R.; STAA, A.; LUCENA, C. Quantifying the Effects of Aspect-Oriented Programming: A Maintenance Study. **Proceedings of the 22nd IEEE International Conference on Software Maintenance (ICSM'06)**, Philadelphia, USA, September 2006.

LANGE, D.B.; MITSURU, O. **Programming and Deploying Java Mobile Agents Aglets**, USA: Addison-Wesley Longman Publishing Co., Inc, 1998.

LANZA, M.; MARINESCU, R. **Object-Oriented Metrics in Practice**. Springer, 2006.

LI, W.; HENRY, S. Object-Oriented Metrics that Predict Maintainability, **Journal of Systems and Software**, 23(2), pp. 111-122, 1993.

LINDVALL, M.; TESORIERO, R.; COSTA, P. Avoiding Architectural Degeneration: An Evaluation Process for Software Architecture, **Proceedings of the 8th International Symposium on Software Metrics (Metrics 2002)**, pp. 77-86, June 2002.

LEE, Y.-S.; LIANG, B.-S.; WU, S.-F.; WANG, F.-J. Measuring the Coupling and Cohesion of an Object-Oriented Program Based on Information Flow, **Proceedings of the International Conference on Software Quality**, Maribor, Slovenia, 1995.

LOBATO, C.; GARCIA, A.; KULESZA, U.; STAA, A.; LUCENA, C. Evolving and Composing Frameworks with Aspects: The MobiGrid Case. **Proceedings of the 7th IEEE Intl. Conference on Composition-Based Software Systems (ICCBSS 2008)**, Madrid, Spain, pp. 53-62, 2008.

LORENZ, M.; KIDD, J. **Object-Oriented Software Metrics**. Prentice-Hall Object-Oriented Series, Englewood Cliffs, NY, 1994.

LUNG, C.; KALAICHELVAN, K. An Approach to Quantitative Software Architecture Sensitivity Analysis, **Proceedings of the Int'l Conference on Software Eng & Knowledge Eng.**, pp. 185-192, 1998.

MARIN, M; MOONEN, L.; DEURSEN, A. SoQueT: Query-Based Documentation of Crosscutting Concerns, **Proceedings of the 29th International Conference on Software Engineering (ICSE'07)**, pp. 758-761, 2007.

MARINESCU, R. **Measurement and Quality in Object-Oriented Design**, PhD thesis, Politechnica University of Timisoara, 2002.

MARINESCU, R. Detection Strategies: Metrics-Based Rules for Detecting Design Flaws, **Proceedings of the 20nd IEEE International Conference on Software Maintenance (ICSM'04)**, pp. 350–359, 2004.

MARTIN, R. Stability, **C++ Report**, Feb. 1997.

MEDVIDOVIC, N.; TAYLOR, R. N. A Classification and Comparison Framework for Software Architecture Description Languages. **IEEE Transactions on Software Engineering**, 26(1), January, 2000, pp.70–93.

B. MEYER, **Object-Oriented Software Construction**, 2nd ed. Upper Saddle River, NJ, EUA: Prentice Hall PTR, 1997.

MONTEIRO, M.; FERNANDES, J. Towards a Catalog of Aspect-Oriented Refactorings. **Proceedings of the AOSD**, Chicago, pp. 111-122, 2005.

MYERS, G. Characteristics of Composite Design. **Datamation**, 19(9), pp. 100-102, September 1973.

NAVASA, A.; PREZ, M.; MURILLO, J.; HERNNDEZ, J. Aspect Oriented Software Architecture: a Structural Perspective. **Workshop on Early Aspects**, AOSD'02, April 2002.

OBJECT TECNOLOGY INTERNATIONAL, INC. **Eclipse Platform Technical Overview**. White Paper. 2001.

OMG. **Unified Modeling Language (UML) Specification: Infrastructure Version 2.0**, July 2005. http://www.omg.org/uml/, 2005.

PARNAS, A. On the Criteria to Be Used in Decomposing Systems into Modules, **Communications of the ACM**, 15(12), pp.1053–1058, December 1972.

PERRY, D.; WOLF, A. Foundations for the Study of Software Architecture, **Software Engineering Notes**, 17(4), pp. 40-52, October 1992.

PESSEMIER, N.; SEINTURIER, L.; DUCHIEN, L. Components, ADL and AOP: Towards a Common Approach. **ECOOP Workshop on Reflection, AOP and Meta-Data for Software Evolution (RAMSE04)**, June 2004.

PINTO, M.; FUENTES, L.; TROYA, J. DAOP-ADL: An Architecture Description Language for Dynamic Component and Aspect-Based Development, **Proceedings of the 2nd International Conference on Generative Programming and Component Engineering**, Erfurt, Germany, Generative Programming and Component Engineering, vol. 48, Springer-Verlag, New York, NY, pp. 118-137, 2003.

PINTO, M.; FUENTES, L. AO-ADL: An ADL for Describing Aspect-Oriented Architectures. **Early Aspect Workshop at AOSD'07**, 2007.

PINTO, M.; GAMEZ, N.; FUENTES, L. Towards the Architectural Definition of the Health Watcher System with AO-ADL. **Proceedings of the Early Aspects at**

ICSE: Workshops in Aspect-Oriented Requirements Engineering and Architecture Design, 2007.

POHL, K.; BÖCKLE, G.; LINDEN, F. J. **Software Product Line Engineering: Foundations, Principles and Techniques**. Springer-Verlag New York, Inc, 2005.

PRESSMAN, R. **Software Engineering: A Practitioner's Approach** (European Adaptation), 5th ed: McGraw-Hill Book'Company, 2000.

RIEL, A. **Object-Oriented Design Heuristics**. Addison-Wesley, 1996.

ROBILLARD, M; MURPHY, G. Representing Concerns in Source Code. **ACM Transactions on Software Engineering and Methodology**, 16(1), Article 3, February 2007.

SAHRAOUI H.; GODIN R.; MICELI T. Can Metrics Help to Bridge the Gap Between the Improvement of OO Design Quality and Its Automation?, **Proceedings of the International Conference on Software Maintenance (ICSM'00)**, 2000.

SAMPAIO, A.; GREENWOOD, P.; GARCIA, A.; RASHID, A. A Comparative Study of Aspect-Oriented Requirements Engineering Approaches. **Proceedings of the International Symposium on Empirical Software Engineering and Measurement** (ESEM.07), Madrid, Spain, 2007.

SANT'ANNA, C.; GARCIA, A.; CHAVEZ, C.; LUCENA, C. STAA, C. On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework, **Proceedings of the Brazilian Symposium on Software Engineering (SBES'03)**, pp. 19-34, October 2003.

SANT'ANNA, C.; GARCIA, A.; KULESZA, U.; LUCENA, C.; STAA, A. Design Patterns as Aspects: A Quantitative Assessment, **Journal of the Brazilian Computer Society** (SBES'04 Best Paper Award), 10(2), Porto Alegre, Brazil, November 2004.

SANT'ANNA, C.; LOBATO, C.; KULESZA, U.; GARCIA, A.; CHAVEZ, C.; LUCENA, C. On the Quantitative Assessment of Modular Multi-Agent System Architectures, **Proceedings of the Special Track on Multiagent Systems and Software Architecture at Net.ObjectDays**, 2006.

SANT'ANNA, C.; FIGUEIREDO, E.; GARCIA, A.; LUCENA, C. On the Modularity Assessment of Software Architectures: Do my architectural concerns count? **Proceedings of the International Workshop on Aspects in Architecture Descriptions (AARCH.07)**, AOSD.07 Conference, Vancouver, Canada, 2007a.

SANT'ANNA, C.; FIGUEIREDO, E.; GARCIA, A.; LUCENA, C. On the Modularity of Software Architectures: A Concern-Driven Measurement Framework, **Proceedings of the 1st European Conference on Software Architecture**, September 24-26, Madrid, Spain, 2007b.

SANT'ANNA, C.; LOBATO, C.; KULESZA, U.; GARCIA, A.; CHAVEZ, C.; LUCENA, C. On the Modularity Assessment of Aspect-Oriented Multiagent Architectures: a Quantitative Study, **International Journal of Agent-Oriented Software Engineering**, 2(1), pp. 34-61, 2008.

SILVA, L.; BATISTA, T.; GARCIA, A.; MEDEIROS, A.; MINORCA, L. On the Symbiosis of Aspect-Oriented Requirements and Architectural Descriptions,

Proceedings of the 10th Workshop on Early Aspects - Aspect-Oriented Requirements Engineering and Architecture Design, in conjunction with AOSD'07, Vancouver, Canada, 2007.

SOARES, S.; LAUREANO, E.; BORBA, P. Implementing Distribution and Persistence Aspects with AspectJ, **Proceedings of the 17th ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA' 02**, ACM Press, pp. 174–190, November 2002.

SOLOWAY, E.; PINTO, J.; LETOVSKY, S.; LITTMAN, D.; LAMPERT, R. Designing Documentation to Compensate for Delocalized Plans. **Communications of ACM**, 31(11), pp. 1259–1267, 1988.

SOMMERVILLE, I. **Software Engineering**, 6th ed: Addison-Wesley Longman, 2000.

STEVENS, W.; MYERS, G.; CONSTANTINE, L. Structured design. In: Yourdon, E. (ed.), **Classics in Software Engineering**, ACM Classic Books Series, Yourdon Press, Upper Saddle River, NJ, pp. 205-232, 1979.

SUN MICROSYSTEMS. **Java ME**. http://java.sun.com/javame/index.jsp. 2007.

TARR, P.; OSSHER, H.; HARRISON, W.; SUTTON, S. N. N Degrees of Separation: Multi-Dimensional Separation of Concerns, **Proceedings of the 21st International Conference on Software Engineering.** ACM Press, Los Angeles, USA, pp. 107-119, 1999.

TEKINERDOĞAN, B.; GARCIA, A.; SANT'ANNA, C.; FIGUEIREDO, E.; PINTO, M.; FUENTES, L. Approach for Modeling Aspects in Architectural Views. University of Twente, Twente. AOSD-Europe Deliverable D77, AOSD-Europe-UT-D77. 2006, p. 1-57.

TEKINERDOĞAN, B.; AKŞIT, M. Providing Automatic Support for Heuristic Rules of Methods. In: Demeyer, S., & Bosch, J. (eds.), **Object-Oriented Technology**, LNCS 1543, Springer-Verlag, pp. 496-499, 1998.

TYREE, J.; AKERMAN, A. Architecture Decisions: Demystifying Architecture. **IEEE Software**, pp. 19-27, 2005.

VALENZUELA, J.; FERNÁNDEZ, L.; PINTO, M.; FUENTES, L. Deriving AO Software Architectures Using the AO-ADL Tool Suite. **Proceedings of the 7th International Conference on Aspect-Oriented Software Development**. Demo Session. Brussels, Belgium, 2008.

VAN GURB, J.; BOSCH, J. 2001. Design Erosion: Problems and Causes. **Journal of Systems and Software**, 61(2), pp. 105–119, 2002.

WONG, W.; GOKHALE, S.; AND HORGAN, J. Quantifying the Closeness between Program Components and Features. **Journal of Systems and Software**, pp. 87-98, 2000.

WOODS, E.; ROZANSKI, N. Using Architectural Perspectives. **Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05),** IEEE Computer Society, Washington, DC, pp. 25-35, 2005.

YOUNG, T.; MURPHY, G. Using AspectJ to Build a Product Line for Mobile Devices. **Proceedings of AOSD'05** (demo session), Chicago, 2005.

YOUNG, T. **Using AspectJ to Build a Software Product Line for Mobile Devices**. MSc dissertation, University of British Columbia, 2005.

ZHANG, C.; JACOBSEN, H. Resolving Feature Convolution in Middleware Systems, **Proceedings of the 19th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications (OOPSLA'04)**, pp. 188–205, ACM Press, 2004.

ZHAO, J. **Towards a Metrics Suite for Aspect-Oriented Software**. TR SE200213625, Inf. Proc. Society of Japan, 2002.

ZHAO, J, Measuring Coupling in Aspect-Oriented Systems, **Proceedings of the 10th International Software Metrics Symposium (METRICS'2004)**, September 2004.

ZHAO, J.; XU, B. Measuring Aspect Cohesion. **Proceedings of the Conference on Fundamental Approaches to Software Engineering (FASE'04)**, LNCS 2984, Barcelona, pp. 54-68, 2004.

Appendix A –
Mobile Media Architecture Description

The appendix presents the component-and-connector view of the eight releases of Mobile Media AO and non-AO architectures. Figure 63 to Figure 68 show the non-AO architectures. The graphical description of releases 1 and 2 are identical (Figure 63). They differ in the interfaces details. Releases 3 and 4 also have the same representation (Figure 64).

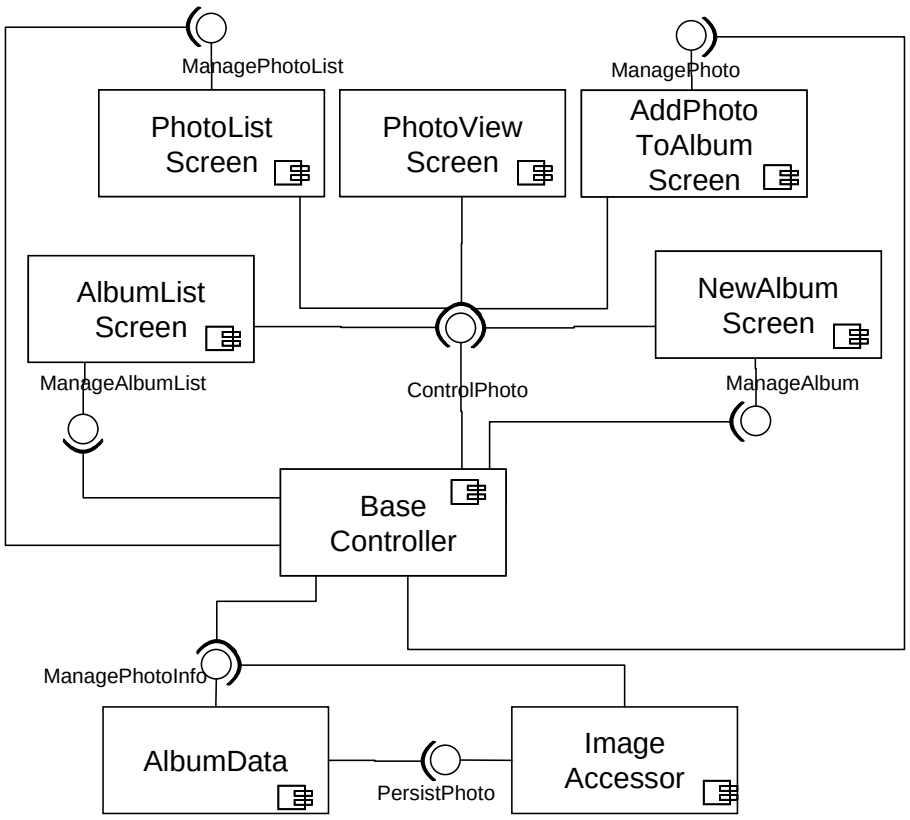

Figure 63: Non-AO architecture of the Mobile Media product line – Releases 1 and 2

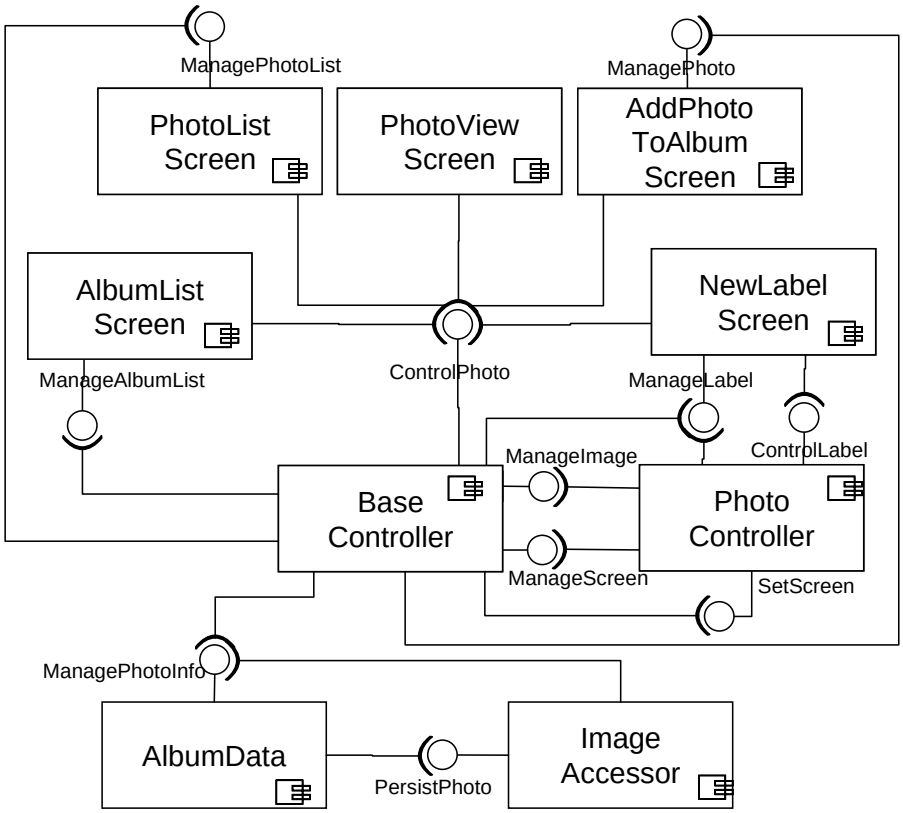

Figure 64: Non-AO architecture of the Mobile Media product line – Releases 3 and 4

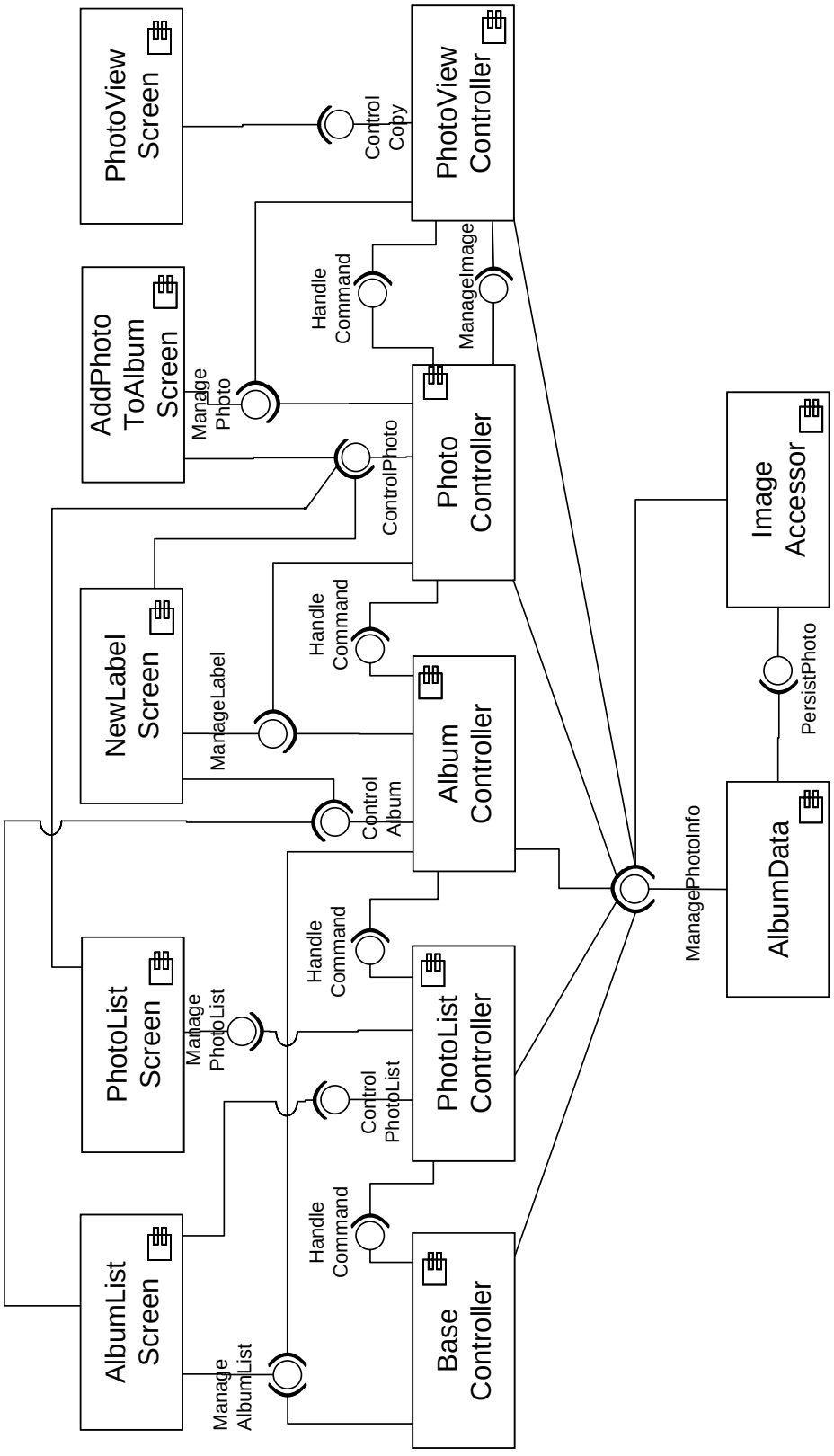

Figure 65: Non-AO architecture of the Mobile Media product line – Release 5

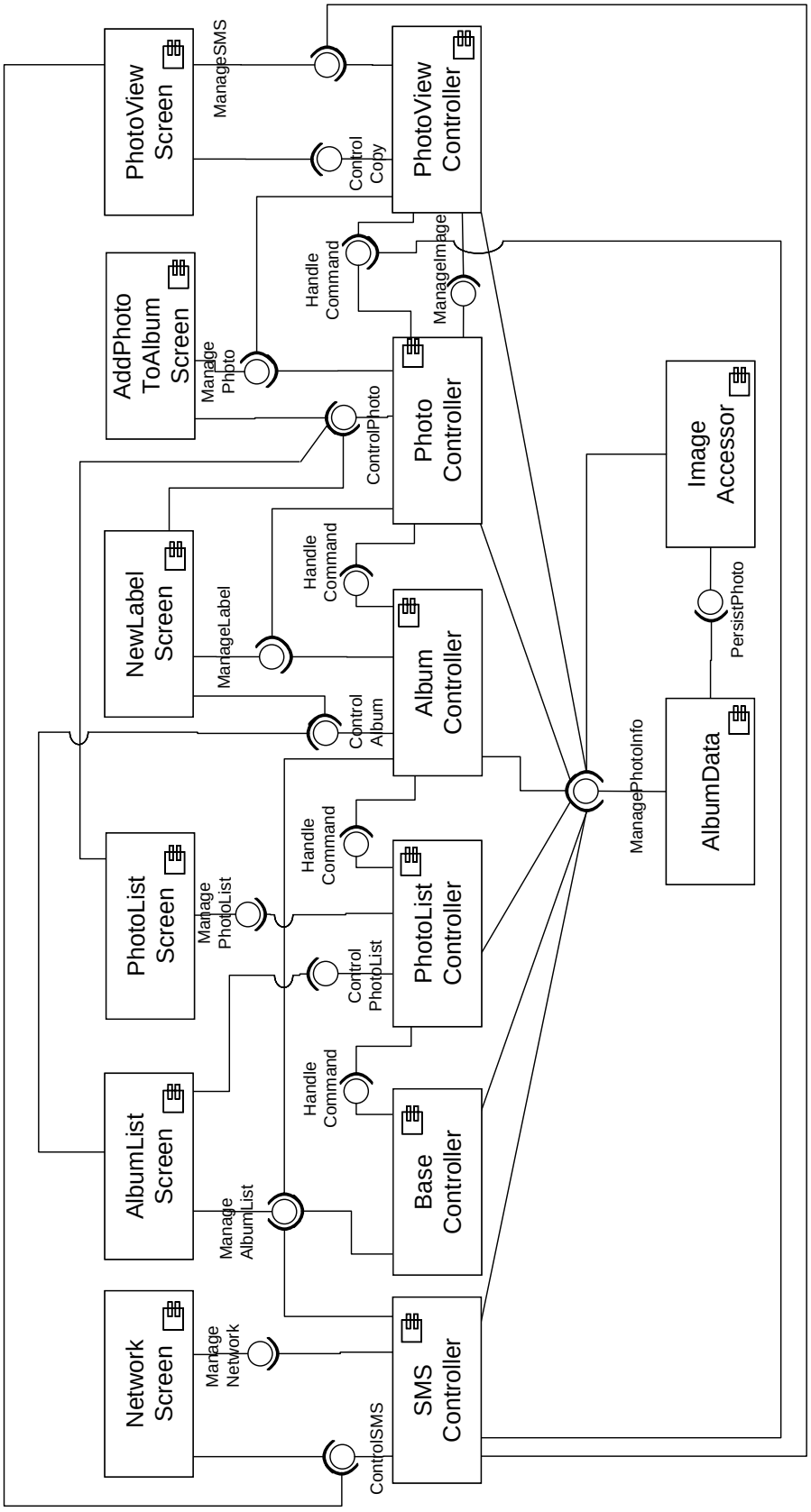

Figure 66: Non-AO architecture of the Mobile Media product line – Release 6

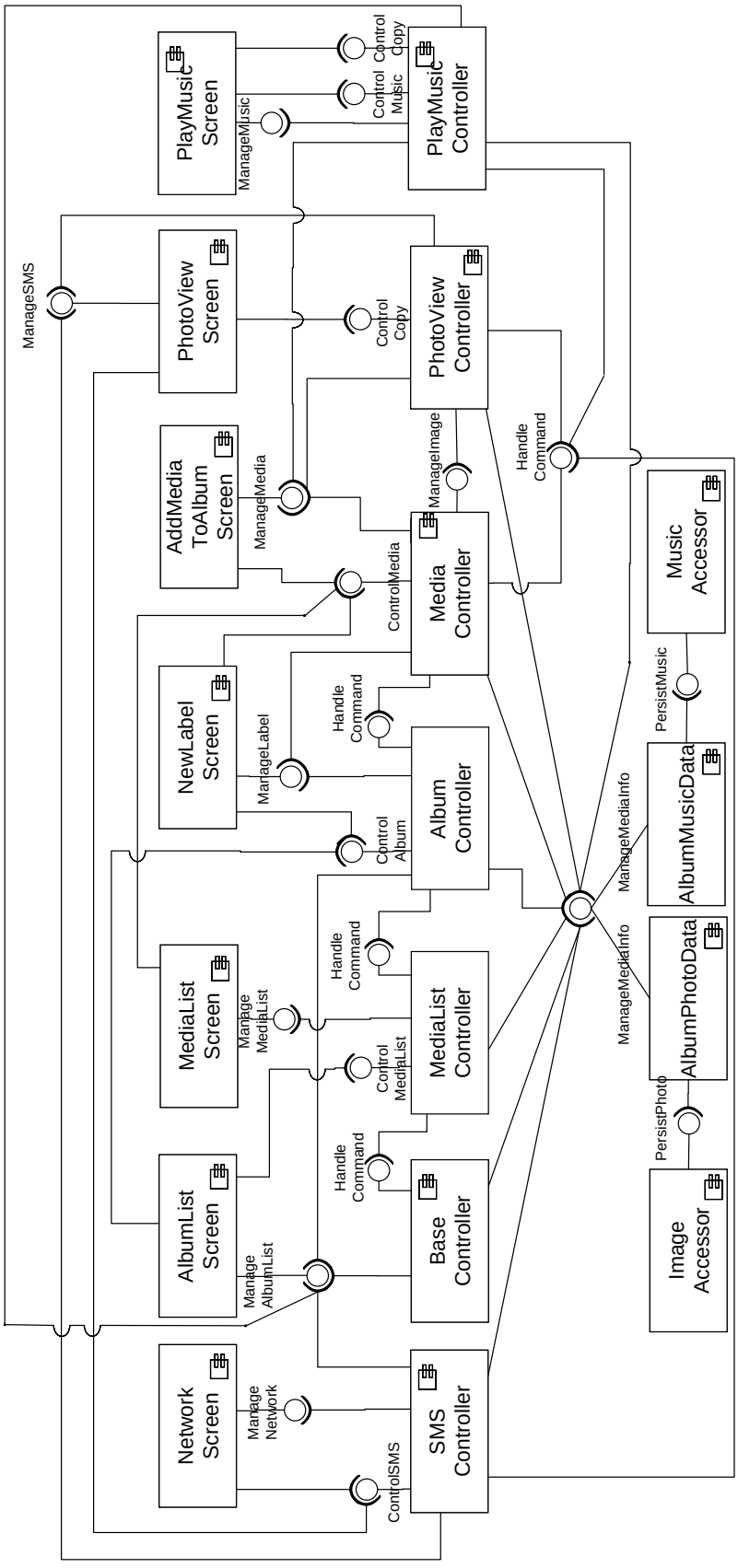

Figure 67: Non-AO architecture of the Mobile Media product line – Release 7

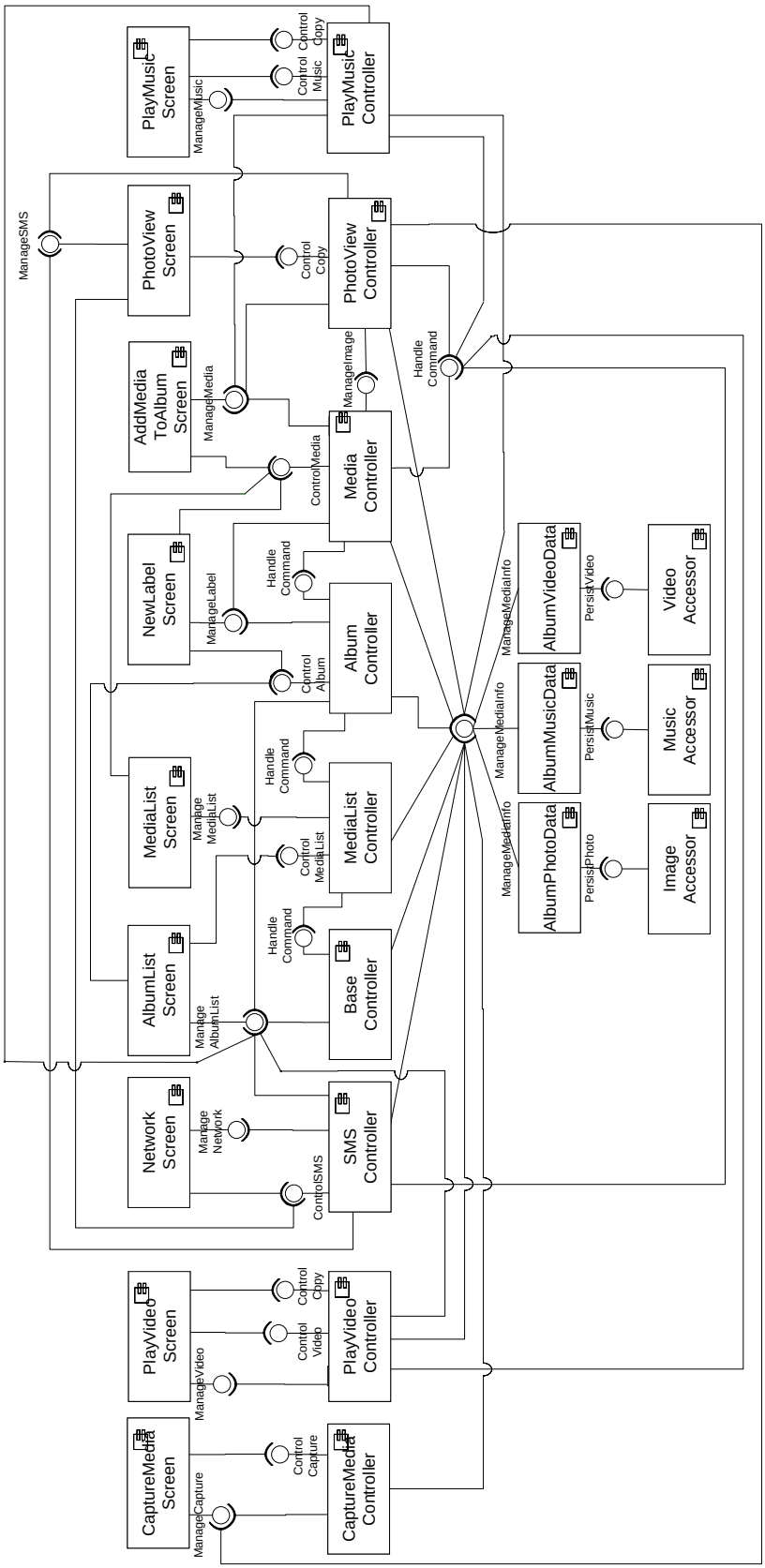

Figure 68: Non-AO architecture of the Mobile Media product line – Release 8

Figure 69 to Figure 75 show the AO architectures of the Mobile Media releases. In fact, there is no AO version for the first release of Mobile Media. Aspectual components were included as the change scenarios were implemented, i.e., from the second version on. Therefore, we present here the AO architecture of releases two to eight.

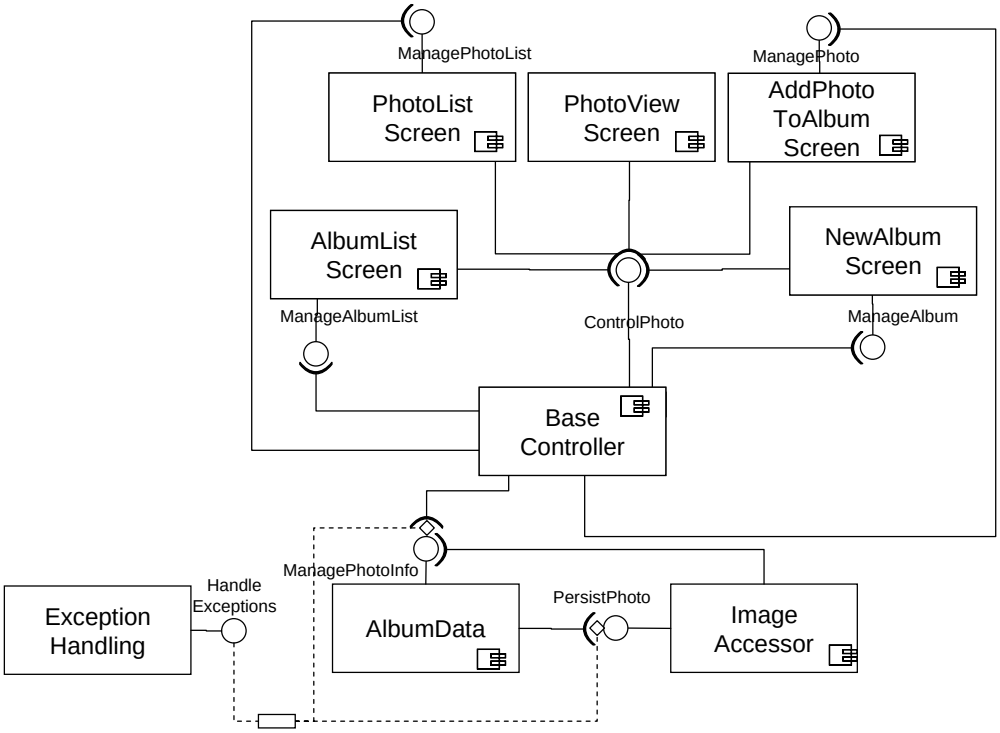

Figure 69: AO architecture of the Mobile Media product line – Release 2

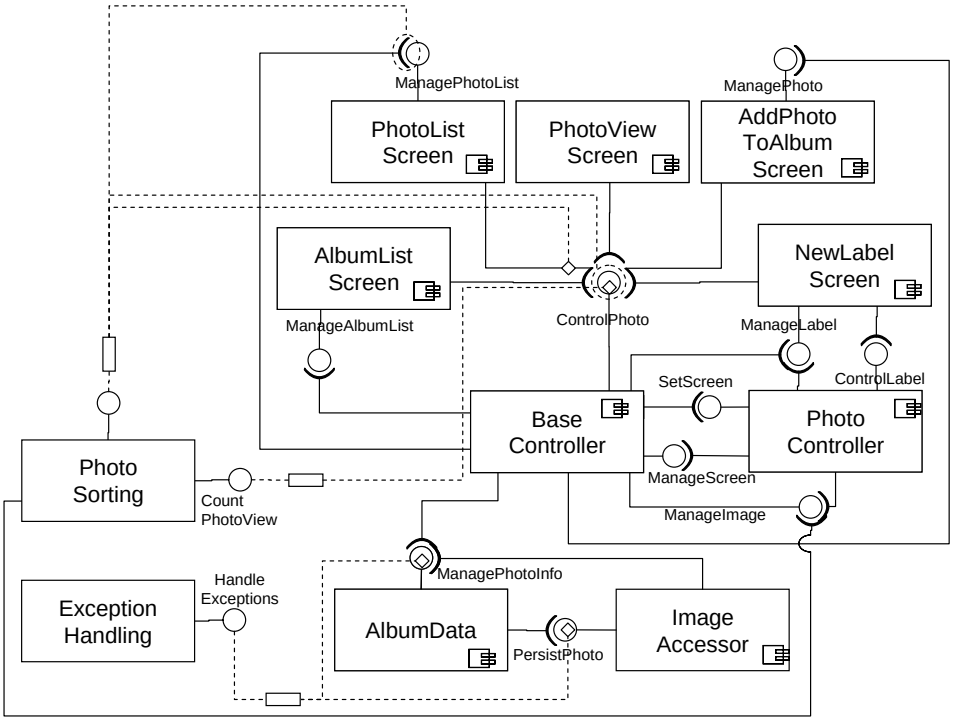

Figure 70: AO architecture of the Mobile Media product line – Release 3

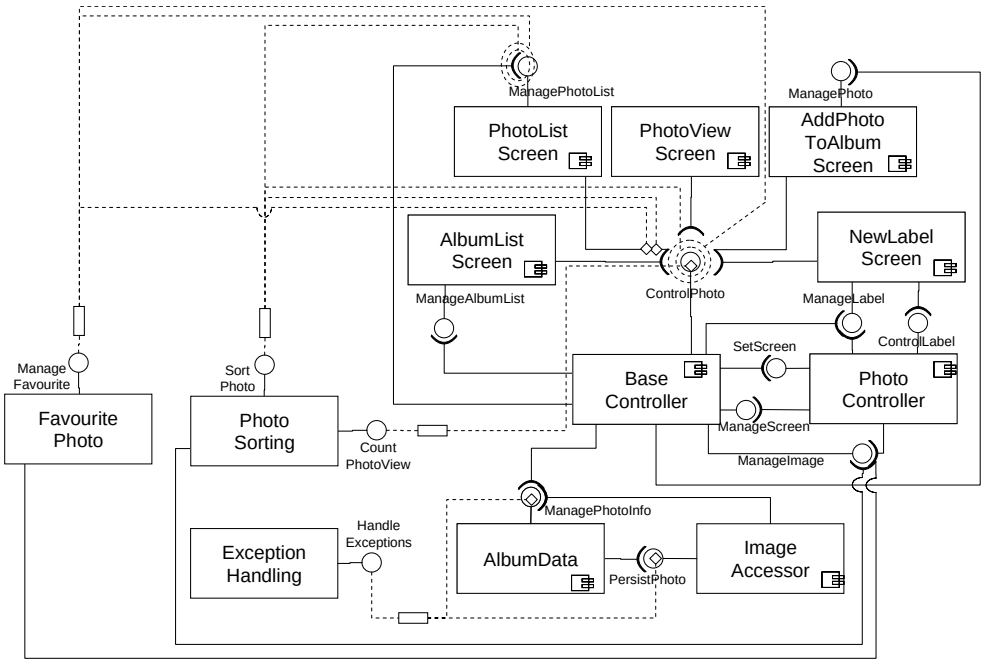

Figure 71: AO architecture of the Mobile Media product line – Release 4

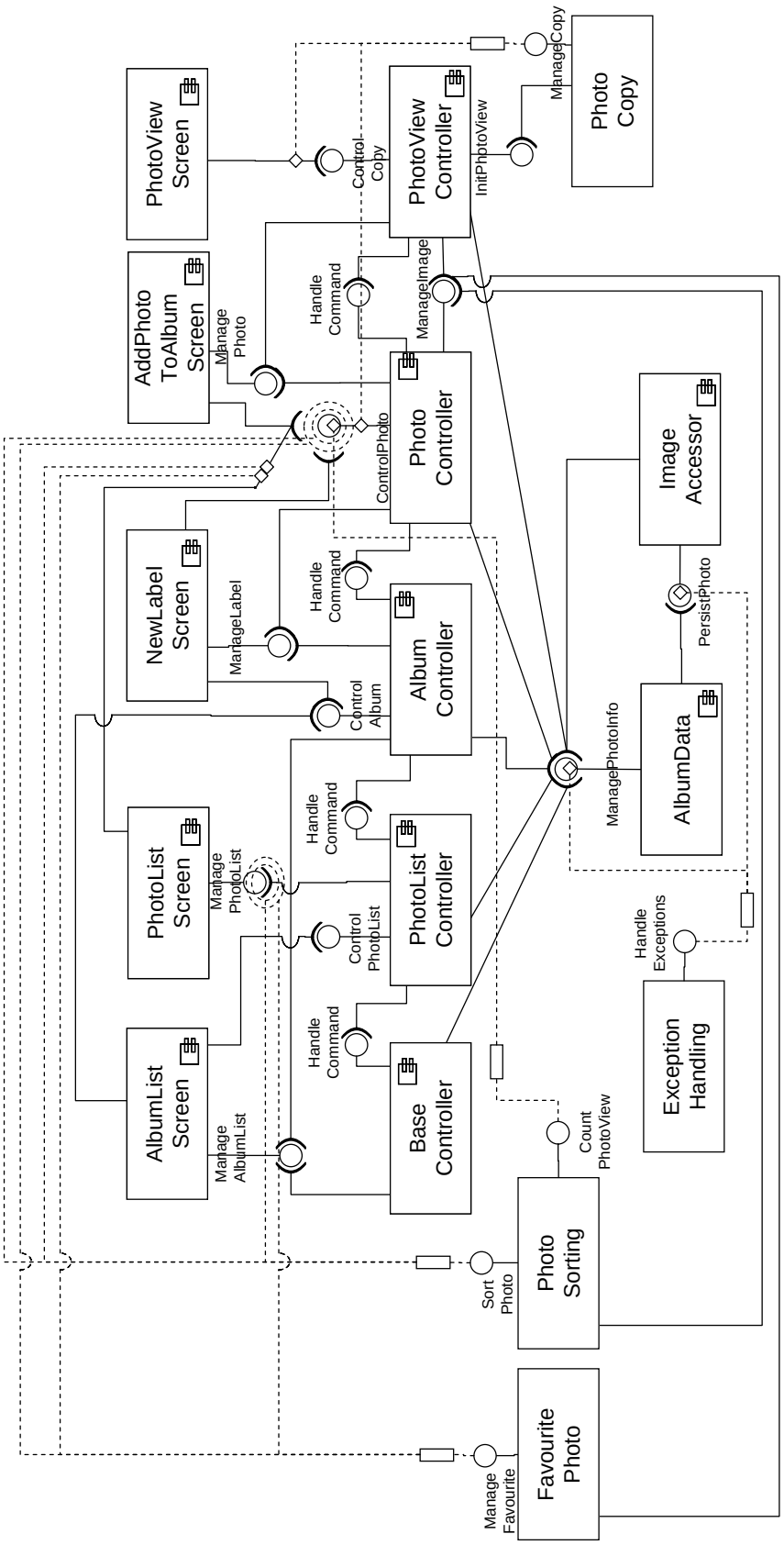

Figure 72: AO architecture of the Mobile Media product line – Release 5

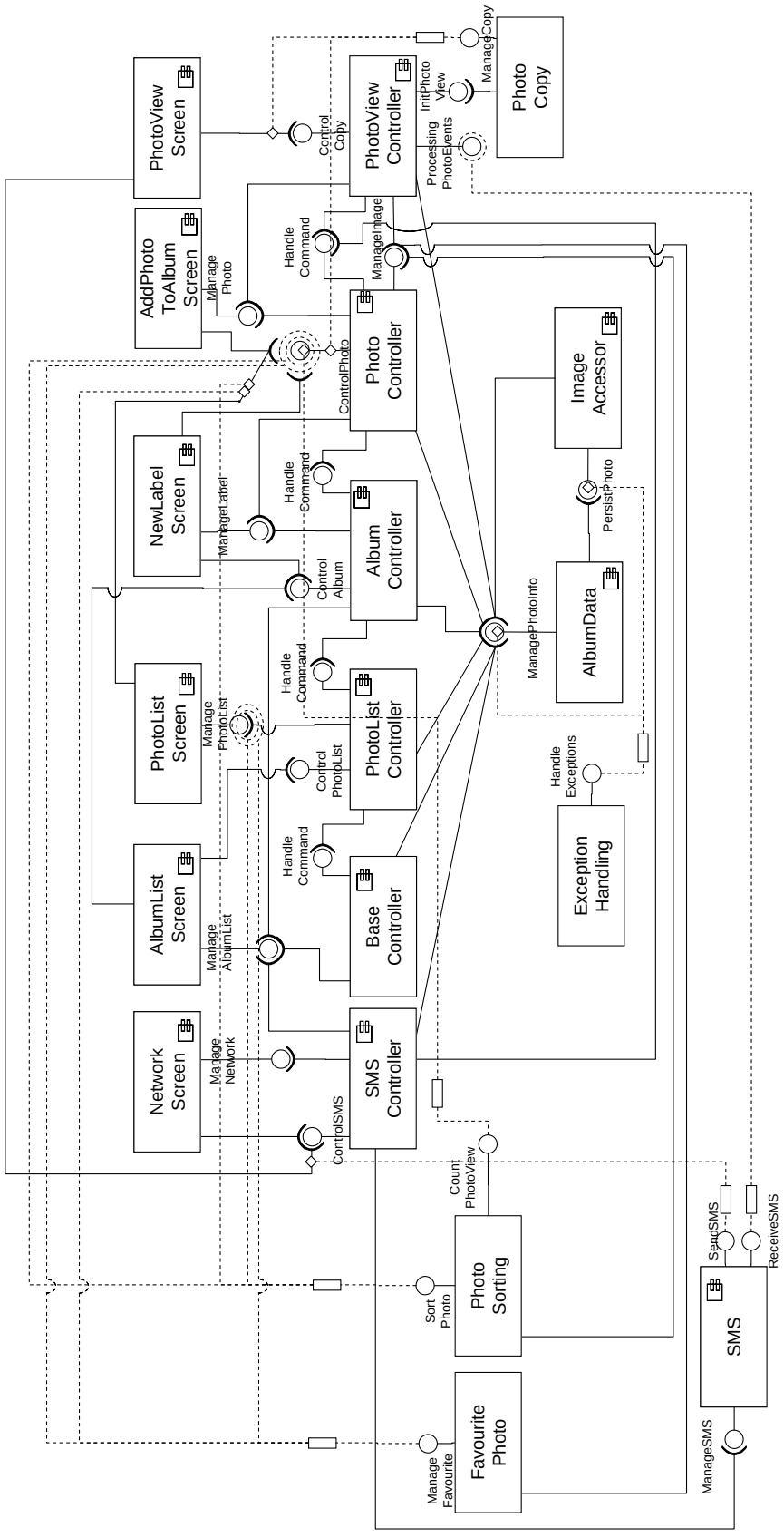

Figure 73: AO architecture of the Mobile Media product line – Release 6

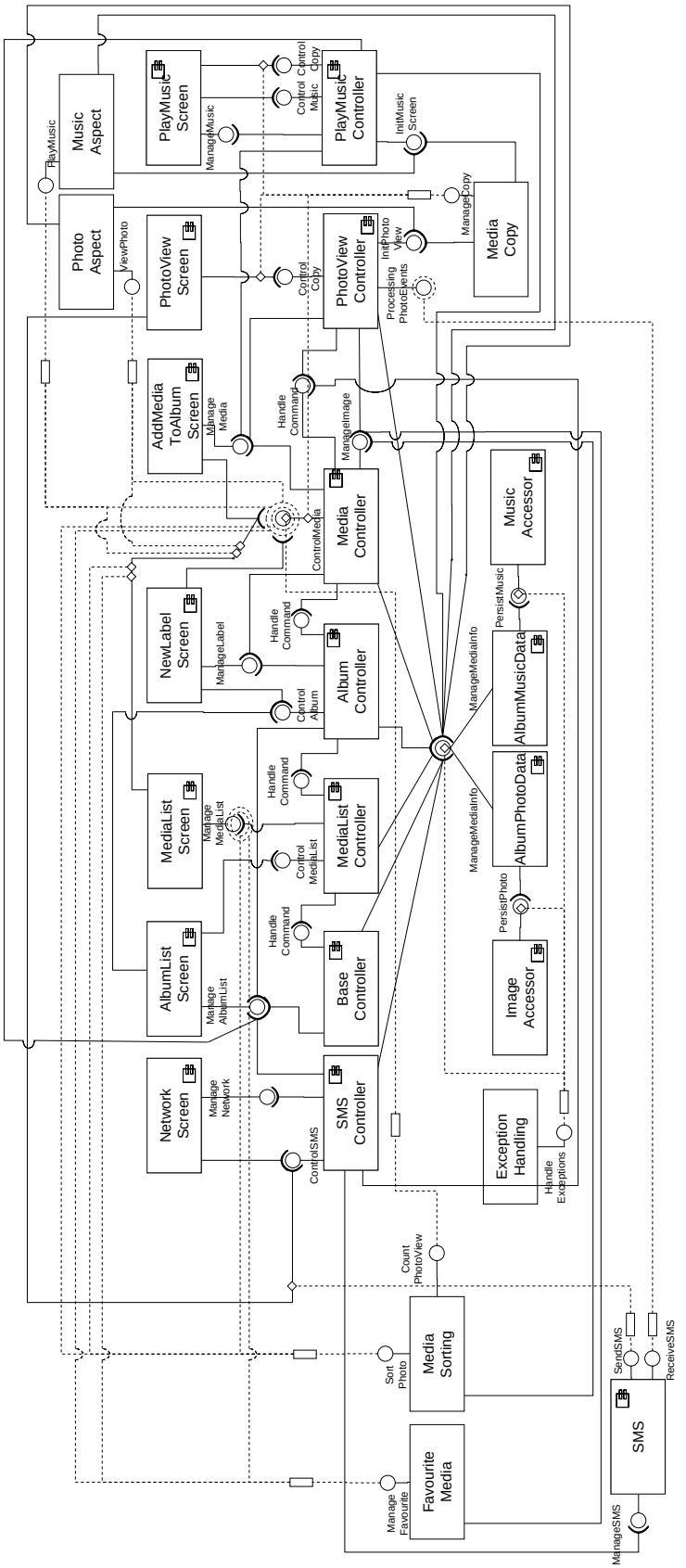

Figure 74: AO architecture of the Mobile Media product line – Release 7

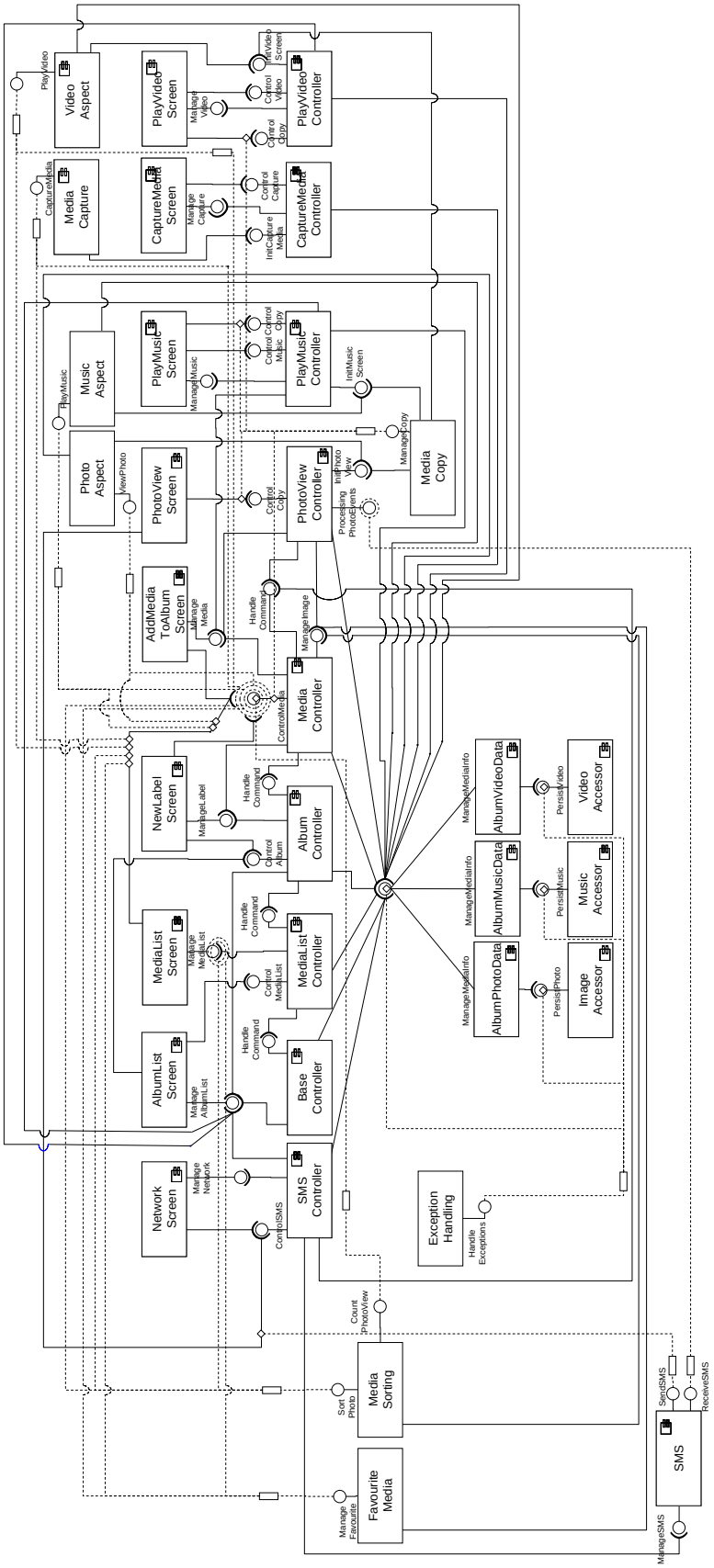

Figure 75: AO architecture of the Mobile Media product line – Release 8

Appendix B –
Detailed Design Metrics Study: Form and Metrics Values

This appendix presents the form provided to the students involved in the detailed design metrics study which contains information about the target design as well as the guidelines they should follow in order to conduct the study. In addition, we also present here the values of the metrics used by the students. Table 30 and Table 31 present these results.

Class	Lack of Cohesion in Methods	Coupling Between Object Classes	Number of Attributes	Number of Operations	Weighted Methods per Class	Lack of Concern-based Cohesion
Complaint	228	19	14	25	46	1
ComplaintRecord	0	11	1	6	11	3
ComplaintRepositoryRDB	0	23	7	20	43	2
Employee	4	17	3	8	15	2
EmployeeRecord	0	9	2	4	8	3
EmployeeRepositoryRDB	0	10	2	6	12	2
HealthUnit	16	12	3	9	14	1
HealthUnitRecord	0	8	1	7	12	3
HealthUnitRepositoryRDB	0	15	3	10	18	2
HealthWatcherFacade	0	21	4	23	36	4
HealthWatcherFacadeInit	71	29	5	18	29	3
HWServlet	1	21	1	2	4	1
IComplaintRepository	0	9	0	6	11	2
IEmployeeRepository	0	8	0	5	10	2
IFacade	0	15	0	16	27	4
IHealthUnitRepository	0	9	0	8	14	2
IPersistenceMechanism	0	11	0	7	7	3
PersistenceMechanism	63	8	11	14	23	3
ServletInsertEmployee	0	8	0	1	3	4
ServletSearchComplaintData	0	13	0	1	3	4
ServletUpdateComplaintData	0	9	0	1	3	4
ServletUpdateHealthUnitData	0	5	0	1	3	4

Table 30: Results of the metrics for the Health Watcher design used in the study about detailed design metrics (Section 8.2) - values obtained per component.

Concern	Concern Diffusion over Classes	Concern Diffusion over Operations
Concurrency	8	42
Distribution	49	76
Exception Handling	73	294
Persistence	41	158
Business	37	222
View (GUI)	21	44

Table 31: Results of the metrics for the Health Watcher design used in the study about detailed design metrics (Section 8.2) - values obtained per concern.

In the following we present the form provided to the students with the instructions to conduct the study.

Group Members
1. _____
2. _____

Experimental Steps

Step 1) The group start to follow the guidelines and answer the questions. The answers are provided by the **group** (just one form per group).

Step 2) Start to read the description of the Health Watcher system provided below.

IMPORTANT: before you *start* to read the description, indicate here what time

it is now: **_:_ _pm**

Health Watcher System – Design

The Health Watcher (HW) system allows a citizen to register complaints to the health public system. Figure 1 shows a diagram representing some of the classes and interfaces of the Health Watcher system. This diagram is actually a simplification of the HW design. In the HW system, complaints are registered, updated and queried through a Web client. The system is structured with the goal of decoupling different parts of the system in order to make it easy to change them separately. Therefore, the user interface part of the system is decoupled from the part which implements the business logic which, in turn, is also decoupled from the database management.

The user interface is represented by the Servlet classes, such as ServletInsertEmployee and ServletSearchComplaintData (Figure 1). Accesses to the HW services are made through the IFacade interface, which is implemented by the HealthWatcherFacade. One of the Health Watcher's requirements is to allow several customers to access the system at the same time. Therefore, a client-server approach is used to distribute part of the execution, which is implemented by the classes HealthWatcherFacade and the

HealthWatcherFacadeInit. The HealthWatcherFacadeInit works as a portal to access the business collections, such as ComplaintRecord and EmployeeRecord. Records access the data also using interfaces, like IComplaintRepository, which decouple the business logic from the specific type of data management in use. Figure 1 shows the classes that serve the purpose of implementing a repository for a relational database: HealthUnitRepositotyRDB, EmployeeRepositoryRDB and ComplainRepositoryRDB. Table 1 summarizes the main responsibilities associated with each class in the HW system. Table 2 describes the main design concerns.

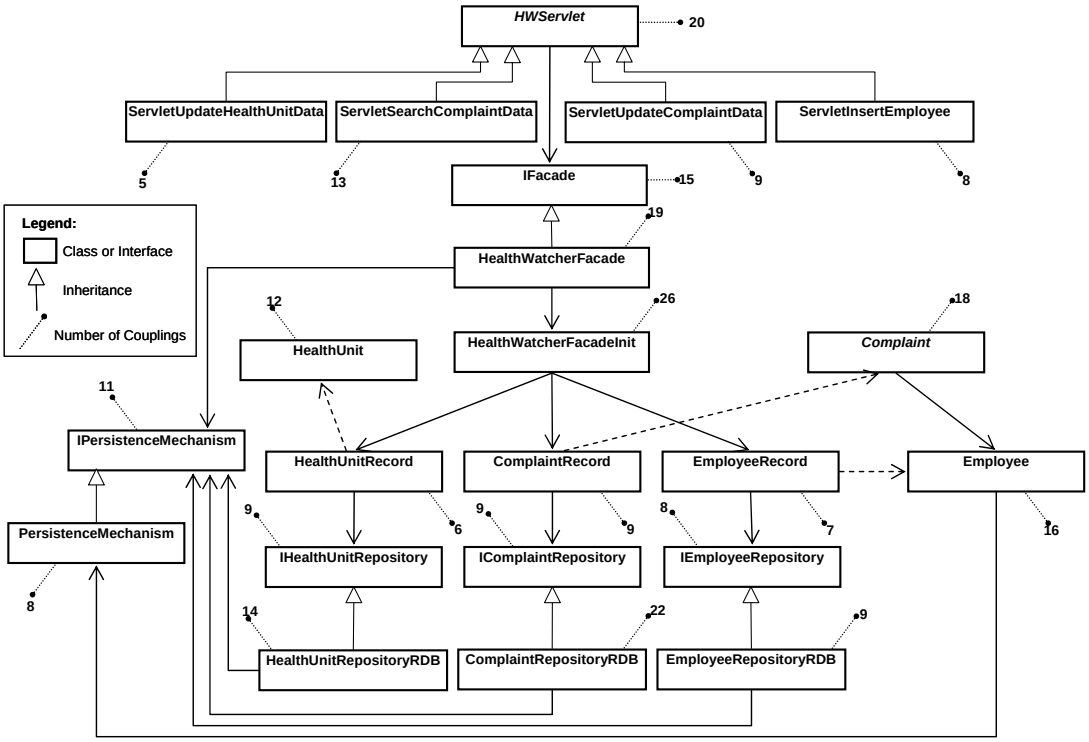

Figure 1. A Partial View of the HW Design

Table 2. Main Design Concerns

Concern	**Description**
Business	It defines the business elements and rules.
Concurrency	It provides a control for avoiding inconsistency in the information manipulated by the system.
Distribution	It is responsible for externalizing the system services at the server side and supporting their distribution to the clients.
Exception Handling	Ii supports error recovery.
Persistence	It is responsible for storing the information manipulated by the system.
View (GUI)	It provides a Web interface for the system.

Table 1. Class Responsibilities

Classes	What they do
HWServlet ServletInsertEmployee, ServletSearchComplaintData, ServletUpdateComplaintData, ServletUpdateHealthUnitData	These classes implement the user interface of the system.
IFacade, HealthWatcherFacade, HealthWatcherFacadeInit	These classes provide a simple interface to all services of the system.
HeathUnit, Employee, Complaint	These classes represent business basic concepts of the Health Watcher system domain.
HealthUnitRecord, EmployeeRecord, ComplainRecord	These classes represent a grouping of objects from a basic class. For instance, EmployeeRecord groups objects of Employee.
HealthUnitRepositotyRDB, EmployeeRepositoryRDB, ComplainRepositoryRDB	These classes contain methods for manipulating persistent objects of the business basic classes. The code of these classes depends on a specific API for accessing some persistence platform.
IHealthUnitRepositoty, IEmployeeRepository, IComplainRepository	These interfaces establish a decoupled relationship between the "Record" classes and "RepositoryRDB" classes. The goal is to make the business code more independent from the data access code and from a specific persistence platform.
PersistenceMechanism	This class implements services such as connecting to and disconnecting from the database, transaction mechanism, concurrency mechanism.

Step 3) The designers of the Health Watcher system need your help to identify **which classes** contain the bad smell "Divergent Change". Start to read the description of the bad smell and the measures. Remember that it is important to reason about the metrics and identify which of them (one, some, or all) are relevant indicators based on the bad smell description.

IMPORTANT: before you *start* to read the text and measures, indicate here what time it is now: **_:_ _pm**

Step 4) Your **group** should now answer the following questions:

a) Which are the classes with the highest probability of having the bad smell "Divergent Change"? You should rank your list of classes; the ones with highest probability should come first.

IMPORTANT: when you *finish* to answer this question, indicate here what time it is now: **_:_ _pm**

b) Explain which metrics you have used for detecting the bad smell, and how they were useful to identify the classes mentioned in the previous question. Which ones were not useful at all?

Step 5) The goal now is to detect the presence of "Shotgun Surgery" bad smells. So identify **which classes** when changed are more likely to propagate changes to other classes. Start to read the description of the bad smell and the measures. Remember that it is important to reason about the metrics and identify which of them (one, some, or all) are relevant indicators based on the bad smell description.

IMPORTANT: before you *start* to read the text and measures, indicate here what time it is now: **_:_ _pm**

Step 6) Your **group** should now answer the following questions:

a) Which are the classes (when changed) that have the highest probability of propagating changes to other classes? You should rank your list of classes; the ones with highest probability should come first.

IMPORTANT: when you *finish* to answer this question, indicate here what time it is now: **_:_ _pm**

b) Explain which metrics you have used for detecting the bad smell, and how they were useful to identify the classes mentioned in the previous question. Which ones were not useful at all?

Step 7) Answer the questionnaire about the experiment design issues.