

ECOOP 2005

19th European Conference on Object-Oriented Programming

SECC, Glasgow 25–29 July 2005

<http://2005.ecoop.org/>

Technical Programme Index

AITO Dahl-Nygaard prizes	<i>SECC: Lomond Auditorium</i>	3
Timetable of Events		3
Abstracts of Invited Talks		3
Formal Announcement of 2005 Awards		4
Technical Paper Sessions	<i>SECC: Lomond Auditorium</i>	5
Wednesday 27th July		5
<i>Conference Opening</i>		5
S1: Testing		5
S2: Aspects and Modularity I		5
S3: Aspects and Modularity II		5
<i>Invited Talk: Bertrand Meyer ETH Zurich & Eiffel Software</i>		5
Thursday 28th July		6
<i>Programme Committee Report</i>		6
<i>Invited Talk: Gail Murphy UBC Vancouver</i>		6
S4: Language Design		6
S5: Java		6
S6: Concurrency		6
Friday 29th July		7
S7: Program Analysis		7
S8: Types		7
<i>Conference Close</i>		7
Doctoral Symposium	<i>Moat House: Boardroom</i>	8
Workshops	<i>SECC & Moat House</i>	9
Monday 25th July		9
W1: Exception Handling in Object Oriented Systems: Developing Systems that Handle Exceptions		9
W3: Second International Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT'05)		9
W5: Eleventh MOS Workshop: Stretching the Object Model to its Limits in Big and Small Environments		9
W7: Ninth ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2005)		9
W9: Workshop on Object Technology for Ambient Intelligence (OT4AmI)		9
W11: Parallel/High-Performance OO Scientific Computing		10
W13: RAM-SE'05: Reflection, AOP and Meta-Data for Software Evolution		10
W15: Ninth Workshop on Pedagogies and Tools for the Teaching and Learning of Object Oriented Concepts		10
W17: Tenth International Workshop on Component-Oriented Programming (WCOP 2005)		10
W19: Views, Aspects and Roles (VAR)		10

Workshops continued	SECC & Moat House	11
Tuesday 26th July		11
W4: 6th International Workshop on OO Reengineering (WOOR)		11
W6: Practical Problems of Programming in the Large (PPPL)		11
W8: Architecture-Centric Evolution		11
W10: Second European Lisp and Scheme Workshop		11
W12: Building Systems Using Patterns: Examine the Illustrious Claim		11
W14: Formal Techniques for Java-like Programs (FTfJP)		11
W16: Models and Aspects - Handling Crosscutting Concerns in MDSD		11
W18: Second ECOOP Workshop on Programming Languages and Operating Systems		11
Birds of a Feather Sessions		12
Tutorials	Moat House	13
Monday 25th July		13
T1: Service-Oriented Architectures (SOA) - The Missing Link Between Business and Technology		13
T3: Implementing Domain-Specific Modelling Languages and Generators		14
T4: Design by Contract and Extended Static Checking for Java with JML and ESC/Java2		14
T6: Patterns of Service-Oriented Architectures		15
T7: From Requirements to Implementation using Model Driven Development		15
T8: Patterns in Functional Programming		15
Tuesday 26th July		16
T10: Architectures on Demand for Any Domain Using Stable Software Patterns		16
T11: Architectural Patterns for Enabling Application Security		16
T12: Basic Patterns from Aspect Oriented Projects		16
T15: Adaptive Object-Model Architecture: How to Build Systems That Can Dynamically Adapt to Changing Requirements		17
Demonstrations	SECC: Dochart	18
Demonstration Timetable		18
D1: MetaEdit+: Domain Specific Modelling for Full Code Generation		18
D2: The Intensional View Environment: Co-evolving source-code and design		19
D3: Ambient Oriented Programming in Ambient Talk		20
D4: Advanced Refactorings for Java in Eclipse		20
Posters	SECC: Hall 1	21
Poster Session Timetable		21
P1: Supporting Object-Oriented Design: a Requirements Elicitation CASE Tool		21
P2: A Distributed AOP System for J2EE		21
P3: GluonJ: An AOP System Dealing with Dependency among Components		22
P4: A Pattern-Driven Software Development Framework		22

The First AITO Dahl-Nygaard Prizes

ECOOP 2005 is delighted to host the first awards of the AITO Dahl-Nygaard prizes. Information about these prizes, including the process for making nominations for future years, is available from the AITO website at www.aito.org

Ole-Johan Dahl and Kristen Nygaard jointly received the ACM A.M.Turing Award in 2001 “for ideas fundamental to the emergence of object oriented programming, through their design of the programming languages Simula I and Simula 67”. In 2002, they shared the IEEE John von Neumann Medal “for the introduction of the concepts underlying object-oriented programming through the design and implementation of SIMULA 67”. A virtual exhibition honouring Dahl, Nygaard and Dijkstra, all of whom died in the summer of 2002, is available online at

<http://cs-exhibitions.uni-klu.ac.at/exhibitions/index.php?id=2>

Timetable of Events for the AITO Dahl-Nygaard prizes

Wednesday:	09:15 – 09:45	Presentation of the First AITO Dahl-Nygaard Prizes
	16:30 – 18:00	Bertrand Meyer — 2005 Senior Prize — Invited Talk
Thursday:	09:15 – 10:15	Gail Murphy — 2005 Junior Prize — Invited Talk

Abstracts of Invited Talks by the 2005 AITO Dahl-Nygaard Prize Recipients

- **Attached Types and Their Application to Three Open Problems of OO Programming**
Bertrand Meyer *ETH Zurich & Eiffel Software*

The three problems of the title - the first two widely discussed in the literature, the third less well known but just as important for further development of object technology - are:

- Eradicating the risk of void calls: $x.f$ with, at run time, the target x not denoting any object, leading to an exception and usually a crash.
- Eradicating the risk of ‘catcalls’: erroneous run-time situations, almost inevitably leading to crashes, resulting from the use of covariant argument typing.
- Providing a simple way, in concurrent object-oriented programming, to lock an object handled by a remote processor or thread of control, or to access it without locking it, as needed by the context and in a safe way.

A single language-based mechanism provides a combined solution to all three issues.

This mechanism also allows new solutions to two known problems: how to check that a certain object has a certain type, and then use it accordingly (‘Run-Time Type Identification’ or ‘downcasting’), for which it may provide a small improvement over previously proposed techniques; and how to provide a .once per object. facility, permitting just-in-time evaluation of certain object properties.

The solution relies on a small extension to the type system involving a single symbol, the question mark. The idea is to declare certain types as ‘attached’ (not permitting void values), enforce some new validity rules that rule out void calls, and validate a number of common programming schemes as ‘Certified Attachment Patterns’ guaranteed to rule out void calls. (In addition, the design replaced an existing type-querying construct by a simpler one.)

The mechanism is completely static: all checks can be performed by compilers as part of normal type system enforcement. It places no undue burden on these compilers (in particular, does not require dataflow analysis) and can be fairly quickly explained to programmers. Existing code, if reasonably well-written, will usually continue to work without change; for exceptions to this rule, often reflecting real risks of run-time crashes, backward-compatible options and a clear transition path are available.

- **The Emergent Structure of Development Tasks**

Gail Murphy, Mik Kersten *UBC*, Martin Robillard *McGill* & Davor Čubranić *U.Victoria*

Integrated development environments have been designed and engineered to display structural information about the source code of large systems. When a development task lines up with the structure of the system, the tools in these environments do a great job of supporting developers in their work. Unfortunately, many development tasks do not have this characteristic. Instead, they involve changes that are scattered across the source code and various other kinds of artifacts, including bug reports and documentation. Today's development environments provide little support for working with scattered pieces of a system, and as a result, are not adequately supporting the ways in which developers work on the system. Fortunately, many development tasks do have a structure. This structure emerges from a developer's actions when changing the system. In this paper, we describe how the structure of many tasks crosscuts system artifacts, and how by capturing that structure, we can make it as easy for developers to work on changes scattered across the system's structure as it is to work on changes that line up with the system's structure.

AITO 2005 Announcement of Recipients

The following text is taken from the AITO announcement of the 2005 awards:

AITO is very proud to announce that the first Dahl-Nygaard Prizes will be given in 2005 to Bertrand Meyer (ETH, Zurich) and to Gail Murphy (University of British Columbia). Professor Meyer is being recognized for his career contributions as a senior researcher and Professor Murphy for her achievements as a junior researcher.

The AITO Dahl-Nygaard Prizes are named for Ole-Johan Dahl and Kristen Nygaard, two pioneers in the area of programming and simulation. Their foundational work on object-oriented programming, made concrete in the Simula language, is one of the most important inventions in software engineering. Their key ideas were expressed already around 1965, but took over 20 years to be absorbed and appreciated by the broader software community. After that, object-orientation has profoundly transformed the landscape of software design and development techniques.

It was a great loss to our community that both Ole-Johan Dahl and Kristen Nygaard passed away in 2002. In remembrance of their scholarship and enthusiastic encouragement of young researchers, AITO has established a prize to be awarded annually to a senior researcher with outstanding career contributions and a younger researcher who has demonstrated great potential for following in the footsteps of these pioneers. Among many excellent candidates proposed to the prize committee for this first time, Bertrand Meyer and Gail Murphy were chosen. Both of them have successfully combined theory and practice in their work, like Dahl and Nygaard.

Bertrand Meyer has been one of the most influential researchers in the eighties, in the initial period of object-oriented programming. He designed the Eiffel language, which pioneered the concept of design by contract. He provided strong arguments for object-oriented software architecture in his book "Object-Oriented Software Construction", which remains a very influential work in object technology. Many of his contributions have proved to be of lasting value.

Like Nygaard, Meyer has not backed away from controversy and has consistently followed his own vision of object orientation. His principle of design by contract established an essential bridge between axiomatic specification approaches and object-oriented programming. Currently his research on trusted components continues to explore challenging problems in software engineering.

Gail Murphy has shown promising potential as a young researcher by proposing innovative ideas and by proving that these are conceptually sound and realistically implementable. She focuses her research and teaching on software engineering, and she has made contributions to understanding and reducing the problems associated with evolving large software systems.

Like Dahl and Nygaard, Murphy challenges students to look at new things, be it aspects or performance measurement, with a disciplined questioning eye. She encourages the development of sound theories backed with the practice of prototype implementations in preparing a new generation of researchers.

Technical Paper Sessions

Conference Opening

Wednesday 27 July 09:00–10:00 SECC: Lomond Auditorium
including: Presentation of AITO Dahl-Nygaard awards

S1: Testing

Chair: *Yvonne Coady*

Wednesday 27 July 10:00–11:00 SECC: Lomond Auditorium

- **Eclat: Automatic Generation and Classification of Test Inputs**
Carlos Pacheco, Michael Ernst
- **Lightweight Defect Localization for Java**
Valentin Dallmeier, Christian Lindig, Andreas Zeller

S2: Aspects and Modularity I

Chair: *Shigeru Chiba*

Wednesday 27 July 11:30–13:00 SECC: Lomond Auditorium

- **Separation of Concerns with Procedures, Annotations, Advice and Pointcuts**
Gregor Kiczales *UBC*, Mira Mezini *TU Darmstadt*
- **Expressive Pointcuts for Increased Modularity**
Klaus Ostermann, Mira Mezini, Christoph Bockisch
- **Sustainable System Infrastructure and Big Bang Evolution: Can Aspects Keep Pace?**
Celina Gibbs, Chunjian Robin Liu, Yvonne Coady

S3: Aspects and Modularity II

Chair: *Christophe Dony*

Wednesday 27 July 14:30–16:00 SECC: Lomond Auditorium

- **Aspect-Oriented Programming Beyond Dependency Injection**
Shigeru Chiba, Rei Ishikawa
- **Open Modules: Modular Reasoning about Advice**
Jonathan Aldrich
- **Evaluating Support for Features in Advanced Modularization Technologies**
Roberto Lopez-Herrejon, Don Batory, William Cook

Invited Talk: Bertrand Meyer *ETH Zurich & Eiffel Software*

Attached types and their application to three open problems of object-oriented programming

Chair: *Dave Thomas*

Wednesday 27 July 16:30–18:00 SECC: Lomond Auditorium

Programme Committee Report

Andrew Black (ECOOP'05 PC chair) *Portland State Univ.*

Thursday 28 July 09:00–09:15

SECC: Lomond Auditorium

Invited Talk: Gail Murphy *UBC Vancouver*

The Emergent Structure of Development Tasks

Chair: Jean Bézivin

Thursday 28 July 09:15–10:15

SECC: Lomond Auditorium

S4: Language Design

Chair: Erik Ernst

Thursday 28 July 10:45–12:45

SECC: Lomond Auditorium

- **First-class Relationships in an Object-Oriented Language**
Gavin Bierman *Microsoft Research*, Alisdair Wren *University of Cambridge*
- **The Essence of Data Access in C ω**
Gavin Bierman, Erik Meijer, Wolfram Schulte
- **Prototypes with Multiple Dispatch: An Expressive and Dynamic Object Model**
Lee Salzman, Jonathan Aldrich
- **Efficient Multimethods in Smalltalk-80**
Brian Foote, Ralph Johnson *UIUC*, James Noble *Victoria University of Wellington*

S5: Java

Chair: Martin Odersky

Thursday 28 July 14:00–15:30

SECC: Lomond Auditorium

- **Loosely-Separated “Sister” Namespaces in Java**
Yoshiki Sato, Shigeru Chiba
- **Efficiently Refactoring Java Applications to Use Generic Libraries**
Robert Fuhrer, Frank Tip *IBM*, Adam Kiezun *MIT*, Julian Dolby, Markus Keller *IBM*
- **Sharing the Runtime Representation of Classes across Class Loaders**
Laurent Daynès, Grzegorz Czajkowski

S6: Concurrency

Chair: Eric Jul

Thursday 28 July 16:00–17:30

SECC: Lomond Auditorium

- **Extending JML for Modular Specification and Verification of Multi-Threaded Programs**
Edwin Rodríguez *KSU*, Matthew Dwyer *UNL*, Cormac Flanagan *UCSC*, John Hatcliff *KSU*, Gary Leavens *Iowa State*, Robby *KSU*
- **Derivation and Evaluation of Concurrent Collectors**
Martin Vechev *U. Cambridge*, David Bacon, Perry Cheng, David Grove *IBM*
- **Static Deadlock Detection for Java Libraries**
Amy Williams, William Thies, Michael Ernst

S7: Program Analysis

Chair: Allen Wirfs-Brock

Friday 29 July 09:00–10:30

SECC: Lomond Auditorium

- **Interprocedural Analysis for Privileged Code Placement and Tainted Variable Detection**
Marco Pistoia *IBM*, Robert Flynn *Polytechnic University*, Larry Koved, Vugranam Sreedhar *IBM*
- **State Based Ownership, Reentrance, and Encapsulation**
Anindya Banerjee *KSU*, David Naumann *Stevens Inst.Tech.*
- **Consistency Checking of Statechart Diagrams of a Class Hierarchy**
Vitus Lam, Julian Padget

S8: Types

Chair: Andrew Black

Friday 29 July 11:00–12:30

SECC: Lomond Auditorium

- **Towards Type Inference for JavaScript**
Christopher Anderson *Imperial College*, Paola Giannini *U. Piemonte Orientale*,
Sophia Drossopoulou *Imperial College*
- **Chai: Traits for Java-like languages**
Charles Smith, Sophia Drossopoulou
- **A Type System for Reachability and Acyclicity**
Yi Lu, John Potter

Conference Close

Friday 29 July 12:30–12:35

SECC: Lomond Auditorium

Doctoral Symposium

Monday 25 July 08:30–18:00

Moat House: Boardroom

The combined Doctoral Symposium (DS) and 15th ECOOP PhDOOS Workshop (WS) will be held on the first day of ECOOP 2005. Participation in this event normally requires prior submission and acceptance by the organisers.

The Doctoral Symposium and PhDOOS Workshop is sponsored by AITO.

Programme

08:30 – 09:00		Welcome
09:00 – 09:40	DS1: Inspecting Object-Oriented Code from the Behavioural Perspective Neil Walkinshaw	<i>Panel: Roel Wuyts & Awais Rashid</i>
09:40 – 10:20	DS2: Understanding Feature Modularity in Feature-Oriented Programming and its Implications for AOP Roberto Erick Lopez-Herrejon	<i>Panel: Roel Wuyts, Awais Rashid & Jonathan Aldrich</i>

10:20 – 10:40 *Coffee Break*

10:40 – 11:20	DS3: A Framework for Automating the Performance Management of Component-Based Enterprise Systems Ada Diaconescu	<i>Panel: Yvonne Coady & Sotirios Terzis</i>
11:20 – 12:00	DS4: Automatically Optimizing Context Management in Contextual Composition Frameworks Mircea Trofi n	<i>Panel: Yvonne Coady & Sotirios Terzis</i>
12:00 – 13:15	<i>Invited Talk: Tales from Dissertationland and the Job Hunt</i> Dr Jonathan Aldrich, CMU	

13:15 – 14:00 *Lunch Break*

14:00 – 14:40	DS5: Dynamic Updates of Existing Distributed Applications Robert Bialek	<i>Panel: John Murphy</i>
14:40 – 15:15	WS1: Writing Reusable Infopipes Using DirectFlow Chuan-kai Lin	
15:15 – 15:50	WS2: Towards a Practical and Efficient Process for Software Reuse Eduardo Santana de Almeida	

15:50 – 16:10 *Coffee Break*

16:10 – 16:45	WS3: Middleware Support for Data-flow Distribution in Web Services Composition Lucian-Mircea Patcas	
16:45 – 17:20	WS4: Contracted Persistent Object Programming Stephanie Balzer	
17:20 – 17:55	WS5: Jimmy McGibney	
17:55 – 18:00		Close

19:30 – 22:00 *Welcome Reception at Glasgow Science Centre*

Workshops:

Monday 25th July

- **W1: Exception Handling in Object Oriented Systems:
Developing Systems that Handle Exceptions**
Alexander Romanovsky *U. Newcastle upon Tyne*, Christophe Dony *U. Montpellier-II*,
Joergen Lindskov Knudsen *Mjolner Informatics A/S*, Anand Tripathi *U. Minnesota*
<http://homepages.cs.ncl.ac.uk/alexander.romanovsky/home.formal/ehoos2005.htm>
Monday 25th July 09:00–17:00 Moat House: Castle I

- **W3: Second International Workshop on Coordination and
Adaptation Techniques for Software Entities (WCAT'05)**
Steffen Becker *U. Oldenburg*, Carlos Canal *U. Málaga*, Juan Manuel Murillo *Univ. Extremadura*,
Pascal Poizat *U. Evry*, Massimo Tivoli *U. of L'Aquila*
<http://wcat05.unex.es/>
Monday 25th July 09:00–17:00 SECC: Boisdale 1

- **W5: Eleventh MOS Workshop: Stretching the Object Model
to its Limits in Big and Small Environments**
Ciarán Bryce *U. Geneva*, Grzegorz Czajkowski *Sun Microsystems Laboratories*
<http://cui.unige.ch/~ecoopws>
Monday 25th July 09:00–17:00 SECC: Carron 1

- **W7: Ninth ECOOP Workshop on Quantitative Approaches
in Object-Oriented Software Engineering (QAOOSE 2005)**
Fernando Brito e Abreu *Lisbon New Univ.*, Coral Calero *Castilla-La Mancha Univ.*,
Michele Lanza *U. Lugano*, Geert Poels *Ghent Univ.*, Houari A. Sahraoui *U. Montreal*
<http://www.iro.umontreal.ca/~sahraouh/qaoose2005/>
Monday 25th July 09:00–17:00 SECC: Dochart 1

- **W9: Workshop on Object Technology for Ambient Intelligence (OT4AmI)**
Holger Mügge *U. Bonn*, Pascal Cherrier *France Telecom*, Pascal Costanza *Vrije Univ. Brussel*,
Robert Hirschfeld *DoCoMo EuroLabs*
<http://roots.iai.uni-bonn.de/OT4AmI/>
Monday 25th July 09:00–17:00 SECC: Carron 2

Workshops:

Tuesday 26th July

- **W4: 6th International Workshop on OO Reengineering (WOOR)**
Serge Demeyer *U.Antwerp*, Stéphane Ducasse *U.Berne*, Kim Mens *U.Catholique de Louvain*,
Roel Wuyts *U. Libre de Bruxelles*
<http://smallwiki.unibe.ch/WOOR>
Tuesday 26th July 09:00–17:00 SECC: Carron 1
- **W6: Practical Problems of Programming in the Large (PPPL)**
Ralf Reussner *U.Oldenburg*, Wolfgang Weck *Independent Software Architect*
<http://se.informatik.uni-oldenburg.de/PPPL>
Tuesday 26th July 09:00–17:00 SECC: Carron 2
- **W8: Architecture-Centric Evolution**
Paris Avgeriou *IPSI Fraunhofer*, Uwe Zdun *Vienna Univ. of Economics*
<http://wi.wu-wien.ac.at/~uzdun/ACE2005>
Tuesday 26th July 09:00–17:00 Moat House: Staffa
- **W10: Second European Lisp and Scheme Workshop**
Pascal Costanza *Vrije Univ. Brussel*, Theo D'Hondt *Vrije Univ. Brussel*,
Arthur Lemmens *Independent Consultant*, Manuel Serrano *Inria Sophia-Antipolis*
<http://lisp-ecoop05.bknr.net/>
Tuesday 26th July 09:00–17:00 Moat House: Castle I
- **W12: Building Systems Using Patterns: Examine the Illustrious Claim**
Mohamed E. Fayad *San José State Univ.*, Haitham S. Hamza *U.Nebraska, Lincoln*
<http://www.engr.sjsu.edu/fayad/workshops/ECOOP05/>
Tuesday 26th July 09:00–17:00 SECC: Boisdale 1
- **W14: Formal Techniques for Java-like Programs (FTfJP)**
Susan Eisenbach *Imperial College*, Gary T. Leavens *Iowa State Univ.*, Peter Muller *ETH Zurich*,
Arnd Poetzsch-Heffter *U.Kaiserslautern*, Erik Poll *Raboud Univ. Nijmegen*, Jan Vitek *Purdue*,
Sophia Drossopolou *Imperial College*, Francesco Logozzo *École Polytechnique*
<http://www.cs.ru.nl/ftfjp/>
Tuesday 26th July 09:00–17:00 SECC: Dochart 1
- **W16: Models and Aspects - Handling Crosscutting Concerns in MDSD**
Markus Voelter *Independent Consultant*, Christa Schwanninger *Siemens AG*,
Iris Groher *Darmstadt Univ. of Technology*
<http://www.st.informatik.tu-darmstadt.de:8080/ecoop2005/maw/>
Tuesday 26th July 09:00–17:00 SECC: Dochart 2
- **W18: Second ECOOP Workshop on Programming Languages and Operating Systems**
Andreas Gal *U.California*, Christian Probst *U.California*
<http://www.ecoop-plos.org/>
Tuesday 26th July 09:00–17:00 Moat House: Castle III

Birds of a Feather Sessions

Monday 25th July – Friday 29th July

ECOOP 2005 wishes to promote discussion to the greatest extent possible, and to that end we can provide facilities for impromptu “Birds of a Feather” (BoF) sessions.

BoFs can be used to continue a discussion that began in a workshop, to allow more detailed follow-up to a paper or demonstration, as a way for a group to get together to pursue a common interest, or for any other similar purpose. If you want a room for any sort of discussion or presentation please just book a BoF. BoFs may be “open” or “closed”. Open BoFs have an open-door policy, and all ECOOP participants are welcome to attend. Closed BoFs are essentially private meetings and the organiser will make all arrangements with the participants directly.

Rooms are available throughout the week for Birds of a Feather sessions, and can be booked for one or more sessions of one or more hours each. To book a BoF, simply go to the Information Desk in the Loch Suite Foyer and a room will be allocated for your meeting. Rooms for up to 50 people are available; if you want a larger room we can do it, but may need a little warning. If our capacity to host BoFs is exceeded, we will give preference to the Open BoFs, directing the organisers of Closed BoFs to quiet areas around the SECC and Moat House that may serve their purpose for a meeting space.

Each Open BoF should be advertised with a notice on the BoFs board in Hall 1. We will endeavour to assist in the advertising of Open BoFs; for example from Wednesday to Friday the Open BoFs for that day (if any) will be announced in the main auditorium.

If you need a digital projector for your BoF please let us know at the time of booking. Other meeting aids may also be available, simply ask and we will try to satisfy your requests.

Friday afternoon BoFs are possible, but please be aware that many delegates will have planned to depart or go sight-seeing after lunch and we must vacate the SECC promptly at the end of the day on Friday.

Tutorials:

Monday 25th July

- T1: **Service-Oriented Architectures (SOA) - The Missing Link Between Business and Technology**

Mohamed Fayad *San José State University*

Intermediate

Monday 25th July (whole day)

Moat House: Barra

SOA is a rapidly emerging technology allowing the aggregation of business functions from coarse-grained components. SOAs promote flexibility by allowing multiple business processes to share the same implementation of common individual steps where the processes and implementations vary independently – services become the digital representation of business capabilities. The emergence of SOA concepts lends urgency to answering questions such as: What is the right way to architect a business into components? Should these designs be specific to a company, or can they be shared across companies within an industry? What is the ROI for companies to move to an SOA? Will the payoff be great enough for this movement to occur rapidly, or will this take a long time to roll out? How will IT infrastructure and application vendors respond to the emergence of SOAs? Will they likewise decompose their applications? What business model for software will emerge to support SOAs?

Dealing with SOA forms a real challenge, as developers need to deeply understand the core knowledge of each artifact of SOA so they can apply them correctly and in the right context. In this tutorial we provide a collection of patterns organized in a Pattern Language (PL) that encompasses the knowledge of both the syntax and the semantics of an SOA. The objective of the PL is to develop the core knowledge of an SOA and present it in a way that can be effectively reused in business. It will serve as a base for dealing with business activities. The PL is developed based on the concepts of the software stability model (SSM), which is a new methodology to develop stable systems. The concept of Software Stability provides a set of guidelines for developing patterns that will be simple to learn and will provide a large feature set that can be used to quickly hook multiple applications with minimal changes.

- **T3: Implementing Domain-Specific Modelling Languages and Generators**

Steven Kelly & Juha-Pekka Tolvanen *MetaCase Consulting*

Intermediate

Monday 25th July (morning)

Moat House: Shuna

Domain-Specific Modelling (DSM) languages provide a viable solution for improving development productivity by raising the level of abstraction beyond coding. With DSM, the models are made up of elements representing concepts that are part of the problem domain world, not the code world. These languages follow domain abstractions and semantics, allowing developers to perceive themselves as working directly with domain concepts. In many cases, full final product code can be automatically generated from these high-level specifications with domain-specific code generators. This tutorial introduces DSM and looks at how it differs from modelling languages like UML, which focus more on the level of the code world. This is followed by real-life examples of DSM from various fields of software product development. The main part of the tutorial addresses the guidelines for implementing DSM: how to identify the necessary language constructs, and different ways of building code generation. Participants will be able to try their hand and learn these skills in practice in group exercises.

- **T4: Design by Contract and Extended Static Checking for Java with JML and ESC/Java2**

Joe Kiniry, Erik Poll & David Cok *UCD, Radboud University & Eastman Kodak*

Basic

Monday 25th July (morning)

Moat House: Jura

This tutorial introduces ESC/Java2 and the JML annotation language. The Java Modeling Language (JML) is a behavioral interface specification language that can be used to specify the behavior of Java modules. It combines the Design by Contract approach of Eiffel and the model-based specification approach of the Larch family of interface specification languages, with some elements of the refinement calculus. JML has a Java-based syntax and semantics, thus is easy to learn for Java programmers.

ESC/Java2 is a tool that checks that a program is consistent with its annotation. It also detects, at compile time, common programming errors that ordinarily are not detected until run time, and sometimes not even then; for example, null dereference errors, array bounds errors, type cast errors, and race conditions. While ESC/Java uses a theorem prover, it feels to a programmer more like a powerful type checker. Because JML is familiar to Java programmers, and ESC/Java2 just feels like a typechecker, we believe that they are an excellent way to gently introduce programmers to formal methods.

- **T6: Patterns of Service-Oriented Architectures**

Uwe Zdun *Vienna University of Economics*

Intermediate

Monday 25th July (afternoon)

Moat House: Jura

This tutorial explains service-oriented architectures in a technology-neutral way using software patterns. It aims at explaining the fundamental concepts of service-oriented architectures in an architecture-centric way. At the same time, by using patterns as a technique to convey these concepts, the tutorial also provides a practical guideline how to apply these concepts in a concrete technology setting. The tutorial aims at providing architectural guidance for architects, developers, and researchers to both understand and develop sustainable serviceoriented architectures.

- **T7: From Requirements to Implementation using Model Driven Development**

Petter Graff & Richard Mitchell *InferData*

Intermediate/Advanced

Monday 25th July (afternoon)

Moat House: Shuna

Today, application development still remains a laborious process with relatively little reuse and automation. Application programmers have to manually map their high-level analysis models to target platform architectures such as J2EE and .NET and eventually to code. Rather than focusing on the problem domain, they have to deal with the complex details of the target platforms. The analysis and design models are often not properly maintained and making applications hard to evolve later. Retargeting an application to a new platform is almost as difficult as writing it from scratch.

Model-Driven Architecture (MDA) is a framework for model-based development being standardized by the Object Management Group (OMG) that aims at solving the above-mentioned problems. In MDA, models are the primary source of an application. All other artifacts such as code, tests, and documentation are (mostly) automatically derived from models. In this tutorial, we will take a critical look at MDA and clearly distinguish what is possible today and the visions for tomorrow.

After explaining basic MDA concepts such as metamodeling and model transformations, we'll discuss a methodology for capturing requirements and discuss how these refinement models can be refined into MDA ready models. We'll also demo transformations from these models into running J2EE and .NET applications.

- **T8: Patterns in Functional Programming**

Jeremy Gibbons *University of Oxford*

Basic/Intermediate

Monday 25th July (afternoon)

Moat House: Hebrides

The aim of this tutorial is to draw together ideas from the Design Patterns community (the Gang of Four: Gamma, Helm, Johnson, Vlissides) and the Functional Programming world (eg Bird, Meertens, Hughes). In particular, the thesis is that whereas design patterns must be expressed extralinguistically (as prose, diagrams, examples) in object-oriented languages, they may be captured directly as abstractions using higher-order operators in functional programming languages. Therefore, they may be reasoned about, type-checked, applied and reused, just as any other abstractions may be. We argue this case by developing the idea of higher-order operators, specifically for capturing patterns of computation in programs. We then bring this around to show how the intentions behind a number of the Gang of Four patterns - such as Composite, Visitor, Iterator, and Builder - have higher-order operators as their analogues in functional languages.

Tutorials:

Tuesday 26th July

- **T10: Architectures on Demand for Any Domain Using Stable Software Patterns**

Mohamed Fayad *San José State University*

Basic/Intermediate Tuesday 26th July (all day) Moat House: Barra

The rapid growth of technology coupled with the tightened development time and production cost constraints have imposed a tremendous pressure and desire for software enterprises to create new and innovative designs to respond to a rapidly changing business environment. Enterprises must invest in building stable architectures that can be ready to be adapted in many different ways to meet the new challenges.

These kinds of architectures are called architectures on demand as they can be adapted accordingly to meet the future requirements and changes in the system. The primary focus of this tutorial is to show how software stability concepts are used to develop on-demand architectures.

- **T11: Architectural Patterns for Enabling Application Security**

Joseph Yoder *The Refactory*

Intermediate Tuesday 26th July (morning) Moat House: Jura

Systems are often developed without security in mind. This omission is primarily because the application programmer is focusing more on trying to learn the domain rather than worrying about how to protect the system. In these cases, security is usually the last thing he or she needs or wants to worry about. When the time arrives to deploy these systems, it quickly becomes apparent that adding security is much harder than just adding a password protected login screen. This tutorial will present a collection of patterns to be used when dealing with application security. Secure Access Layer provides an interface for applications to use the security of the systems on which they are built. Single Access Point limits entry into the application through one single point. Check Point gives the developer a way to handle an unknown or changing security policy. Groups of users have different Roles that define what they can and cannot do. The global information about the user is distributed throughout the application with a Session. Finally, users are presented with either a Limited View of legal options or are given a Full View With Errors. These patterns work to provide a security framework for building applications.

- **T12: Basic Patterns from Aspect Oriented Projects**

Arno Schmidmaier Lohweg *AspectSoft*

Basic/Intermediate Tuesday 26th July (morning) Moat House: Shuna

Objects have been quite successful in the past to modularize and organize most problems. OO technology created new architectures, solved problems and provided a sound base for modern middleware platforms. However due to the increasing complexity of modern programs non business problems got more and more important. Typical examples are transaction handling, persistency, tracing, contract validation. Object technology failed to bind these additional concerns in a modular way to the objects of the business domain. Aspect oriented programming (AOP) proved quite successful. Adopting AOP creates huge possibilities for overcoming these daily problems, and offers new features and design ideas for future software development. Patterns guide the developers, architects and managers around the common pitfalls and guide them to the way of building applications based on AOP, which will stand the test of time. This tutorial presents several patterns mined from several commercial projects live in production.

- **T15: Adaptive Object-Model Architecture: How to Build Systems That Can Dynamically Adapt to Changing Requirements**

Joseph Yoder *The Refactory*

Advanced

Tuesday 26th July (afternoon)

Moat House: Shuna

Architectures that can dynamically adapt to changing requirements are sometimes called reflective or meta architectures. We call a particular kind of reflective architecture an Adaptive Object-Model (AOM) architecture. An Adaptive Object-Model is a system that represents classes, attributes, relationships, and behavior as metadata. It is a model based on instances rather than classes. Users change the metadata (object model) to reflect changes to the domain model. These changes modify the system's behavior. In other words, it stores its Object-Model in XML files or in a database and interprets it. Consequently, the object model is adaptive; when the descriptive information for the object model is changed, the system immediately reflects those changes. We have noticed that the architects of a system with Adaptive Object-Models often claim this is the best system they have ever created, and they brag about its flexibility, power, and eloquence.

At the same time, many developers find them confusing and hard to work with. This is due in part because the developers do not understand the architecture. This tutorial will give a description of the Adaptive Object-Model architectural style and will make it easier for developers to understand and build systems that need to adapt to changing requirements.

Timetable of Demonstrations:

All demonstrations take place in the Dochart rooms in the SECC Loch Suite (upper floor). Wherever possible, demonstrations are timed to coincide with breaks in the main conference programme, however time has been allowed for rapid lunchbreaks for those wishing to visit the demonstrations and each demonstration is scheduled twice, to maximise availability.

Wednesday:	11:00 – 11:30	D1: MetaEdit+: Domain Specific Modelling for Full Code Generation
	13:30 – 14:30	D4: Advanced Refactorings for Java in Eclipse
	16:00 – 16:30	D3: Ambient Oriented Programming in Ambient Talk
Thursday:	10:15 – 10:45	D3: Ambient Oriented Programming in Ambient Talk
	13:00 – 14:00	D2: The Intensional View Environment: Co-evolving source-code and design
	15:30 – 16:00	D1: MetaEdit+: Domain Specific Modelling for Full Code Generation
Friday:	10:00 – 11:00	D2: The Intensional View Environment: Co-evolving source-code and design
	11:30 – 12:30	D4: Advanced Refactorings for Java in Eclipse

Demonstration Descriptions:

- **D1: MetaEdit+: Domain Specific Modelling for Full Code Generation**

Steven Kelly *MetaCase*

Wednesday 11:00–11:30 SECC: Dochart

Thursday 15:30–16:00 SECC: Dochart

Domain-Specific Modeling suggests higher productivity can be obtained from a modeling language tailored for a specific domain, company or project, than from a generic modeling language. Domain-specific code generators allow full production code based on previous hand-written code to be generated automatically from models. MetaEdit+ allows creating new modeling languages and generators efficiently and without programming, through the power of object technology.

Domain-Specific Modeling (DSM) raises the level of abstraction beyond programming by specifying systems directly using domain concepts. In many cases, the final products can be generated from these high-level specifications. This automation is possible because both the language and generators need fit the requirements of only one company and domain. Industrial experience consistently shows DSM increases productivity by a factor of 5-10.

This demonstration introduces DSM by showing real world cases from various fields of software development: enterprise application development for Symbian smartphones, financial product definition for a B2B J2EE web site, voice menu development for 8-bit microcontroller and MMS/SMS telecom service configuration. These cases illustrate how DSM, giving first class support for modeling, can prevent incorrect or unwanted designs at early stages of the development, how underlying platform complexity is hidden, and how full code can be generated.

Previously creating DSM languages and tools has required a significant investment, hindering the adoption of DSM. Many contemporary metamodeling approaches still require a considerable amount of manual programming for finalizing the language definition. Issues such as language constraints, representational elements and graphical editing tools are often left for manual coding: unfortunately, these are generally the most time-consuming parts.

The second part of the demonstration will show in an interactive manner how DSM languages, their supporting tools and code generators can be built efficiently. We will present MetaEdit+, a metamodeling environment that enables metamodeling without programming, allowing new modeling languages to be created in hours instead of weeks or months. Using MetaEdit+ for metamodeling, we extend an existing modeling language and its generator. Language extensions include adding domain constraints, rules and new concepts. For the generator, we will show how it is based on the organisation's existing code, so generated code integrates well with and looks the same as previous hand-written code. Features from the research prototype will give the audience a look into the future.

- **D2: The Intensional View Environment: Co-evolving source-code and design**

Kim Mens & Andy Kellens *Univ. Catholique de Louvain* Thursday 13:00–14:00 SECC: Dochart
Friday 10:30–11:30 SECC: Dochart

Maintaining the source code of an evolving software system requires adequate documentation of its design and architecture. However, due to its constant evolution, it is difficult to keep source code and design / architecture synchronized. Intensional source-code views have been proposed as an active documentation technique to alleviate this problem.

- They increase our ability to understand, modularize and browse the source code by grouping together source-code entities that address a same concern.
- They facilitate software development and evolution, because alternative descriptions of the same intensional view can be checked for consistency and because relations among intensional views can be defined and verified.
- They enable us to document and verify knowledge developers have about source code that is not captured by traditional documentation mechanisms.

To define and verify intensional views and their interrelationships we have built a series of tools which we grouped in what we call The Intensional View Environment. All these tools were implemented in VisualWorks Smalltalk. Some of these tools are :

- The Intensional View Editor : a tool for defining, modifying and storing intensional views and checking them against the source code;
- The Relation Browser : a tool for defining, modifying and storing relations between intensional views and checking them against the source code;
- The Relation Inspector : a tool for getting fine-grained feedback on the validity of high-level relationships between intensional views;
- The Deduce Tool : a tool for automatically deducing new interesting relationships between a set of known intensional views.

Recently we reimplemented most of the tools, to improve the storage mechanism and to integrate them seamlessly with StarBrowser2, an advanced source code browser for VisualWorks Smalltalk. In addition to having the Prolog-like query language SOUL as underlying language in which to express the intensional views and relations, we also added support for using Smalltalk itself as query language. Perhaps most importantly, we added support for visualizing intensional views and relations, by relying on CodeCrawler, a reverse engineering tool which combines metrics and software visualization.

The goal of this demonstration is, using the running example of the SmallWiki case study, to illustrate the power of intensional views and relations, and their associated tools, to co-evolve the high-level design and source code of a medium-sized software system. We start by documenting the architecture of an early version of SmallWiki and show how this documentation helps us in discovering some minor irregularities in the source code. We then apply this verifiable documentation to a slightly more recent version of SmallWiki and draw conclusions about how SmallWiki evolved, and about the consequences of this evolution on the documented architecture. Finally, we verify the documentation again on yet a more recent version of SmallWiki and draw conclusions about the usefulness of intensional views and relations to document the high-level structure of an evolving software system.

- **D3: Ambient Oriented Programming in Ambient Talk**

Jessie Dedecker *Vrije Univ. Brussels*, Stijn Mostinckx *Vrije Univ.*, Tom Van Cutsem *Vrije Univ.*, Sebastian Gonzalez *Univ. Catholique de Louvain*, Theo D'Hondt *Vrije Univ.*

Wednesday 16:00–16:30 SECC: Dochart

Thursday 10:15–10:45 SECC: Dochart

Software development for mobile devices (such as smart phones and PDAs) is given a new impetus with the advent of mobile networks. Mobile networks surround a mobile device equipped with wireless technology and are demarcated dynamically as users move about. Mobile networks turn the applications running on mobile devices from mere isolated programs into smart applications that can cooperate with their environment. As such, mobile networks take us one step closer to the world of ubiquitous computing envisioned by Weiser; a world where (wireless) technology is gracefully integrated into the everyday lives of its users. Mobile networks that surround a device have several properties that distinguish them from other types of networks. The most important ones are that connections are volatile (because the communication range of the wireless technology is limited) and that the network is open (because devices can appear and disappear unexpectedly). This puts extra burden on software developers. Although low-level system software and networking libraries providing uniform interfaces to these wireless technologies (such as JXTA and M2MI) have matured in the last couple of years, developing application software for mobile networks still remains difficult. One of the main reasons for this is that current-day programming languages lack abstractions that deal with the mobile hardware characteristics. This observation justifies the need for a new Ambient-Oriented Programming paradigm (AmOP for short) that consists of programming languages that explicitly incorporate potential network failures in the very heart of their basic computational steps.

The demo showcases AmbientTalk, a first scion of this AmOP programming language family and is conceived as an alteration between:

- showing code snippets that illustrate the simplicity and expressive power of AmbientTalk. Meanwhile participants can become acquainted with AmbientTalks concurrent object model, as well as its system of first-class mailboxes to deal with the mobile hardware characteristics.
- demonstrating how these programs actually behave in a real-life context.

This will be done by showing the execution of the programs on several portable devices such as Laptops and smart phones.

- **D4: Advanced Refactorings for Java in Eclipse**

Robert Fuhrer, Markus Keller, Adam Kiezun, Frank Tip *IBM* Wednesday 13:30–14:30 SECC: Dochart
Friday 12:35–13:35 SECC: Dochart

We will demonstrate several advanced type-related refactorings for Java that have been recently added to Eclipse 3.x. These refactorings are typical of the sort of transformations OO programmers perform manually to refine class hierarchies in order to promote reuse, clarity, and extensibility.

The automation of such refactorings involves nontrivial static analyses, as well as challenging usability issues from a software engineering perspective, which reflect the complications that programmers face in performing the transformations manually. As a result, during each demonstration, we will give a brief overview of the technical challenges that the refactoring presents, along with highlights of the approach we took in addressing some of the more difficult ones.

The refactorings we will present will include:

- Extract Interface - extracts a user-selected portion of the API for a given class and, more interestingly, determines which sites that reference the original class can be modified to refer to the newly-created interface.
- Generalize Type - determines what super-types can be substituted in the declaration of a given entity, maintaining the static type-correctness of the program.
- Infer Type Arguments - aids in migrating existing Java code to Java 1.5 parametric classes ("generics") by determining what element types flow into each container entity in the original program, modifying declarations and allocations sites where possible, and removing casts that have been rendered redundant.
- Introduce Type Parameter - adds a new type parameter to a class to be used in place of the presently-declared type of a user-specified declaration, and performs all other necessary changes, possibly including adding new type parameters to related classes.

Timetable for Poster Sessions

The poster display runs from Tuesday afternoon until Friday in Hall 1 of the SECC, the area in which most of the daytime catering will be provided. Some of the workshops and demonstrations will also be supplemented by posters, which will stand alongside the “presented” posters.

The “presented” posters will be displayed throughout the conference. In addition, you are invited to discuss the poster contents and ask questions of the presenters during the poster sessions.

Wednesday:	16:00 – 16:30	SECC: Hall 1
Thursday:	15:30 – 16:00	SECC: Hall 1

Poster Descriptions

- **P1: Supporting Object-Oriented Design: a Requirements Elicitation CASE Tool**

Ke Li, Rob Pooley, Rick Dewar *Heriot-Watt University*

This poster highlights a problem in design and the needs of designers in terms of constructing system design. To address the problem and meet designers’ needs, we propose and represent an incremental and iterative approach that could deliver quality requirements and object oriented analysis, hence to provide better assistance to designers. A simple example is used to demonstrate how it works.

In the case that incomplete and ambiguous requirements are delivered, designers might need to communicate back to customers or domain experts in order to clarify. We aim to improve the overall quality of requirements by offering ambiguity detection, requirements interpretation, domain modelling, requirements specification and traceability. In this poster, we represent how an integrated requirements elicitation and analysis system could address the issues derived from incomplete and ambiguous requirements in object-oriented design. The primary goal is to assist designer to build a better design solution more easily and quickly

- **P2: A Distributed AOP System for J2EE**

Muga Nishizawa & Shigeru Chiba *Tokyo Institute of Technology*

There have been a few AOP systems for J2EE applications before. However, those AOP systems basically provide language mechanisms inherited from AspectJ but they do not provide mechanisms newly developed for J2EE. For example, JBoss AOP is an AOP system integrated into J2EE but its programming model is almost equivalent to AspectJ. Indeed, JBoss AOP can be solely used as a standalone AOP system without the J2EE framework since it does not have a mechanism dedicated only for J2EE. A crosscutting concern of J2EE applications is often distributed on several hosts. Such a concern is not well modularized as a simple aspect by the mechanisms inherited from AspectJ. Rather, it tends to be modularized as an aspect that consists of multiple distributed sub-components. The developers should be able to modularize such a concern as a simple and non-distributed aspect. Moreover such an aspect should be woven into the rest of the program by the EJB container on each participating host. Existing AOP systems for J2EE do not support the automatic delivery of an aspect for weaving on each host. Therefore the users of the systems need to manually deliver an aspect to each participating host. It is not a simple task. We illustrate these problems by showing an example, which is a test program of a J2EE application.

To address these problems, we present a new AOP system for J2EE, named DJcutter. By comparison with existing AOP systems, it is integrated into J2EE and provides AOP mechanisms designed for J2EE. First, DJcutter provides a new mechanism called remote pointcuts [1]. Although a pointcut in AspectJ identifies execution points on the local host, a remote pointcut can identify them on a remote host transparently. Remote pointcuts allows the users to write a distributed crosscutting concern as a simple aspect that is not distributed on each participating host. Moreover, DJcutter supports load-time weaving and the automatic delivery of compiled aspects to each participating host. The weaver of DJcutter is implemented as one of the services (MBean) of an EJB container. Furthermore, the prototype of DJcutter implemented on top of JBoss supports restricted dynamic weaving by using the hot deployment of JBoss.

- **P3: GluonJ: An AOP System Dealing with Dependency among Components**

Rei Ishikawa & Shigeru Chiba *Tokyo Institute of Technology*

Our poster shows details and demos of GluonJ, which is our aspect oriented programming system proposed in our paper “Aspect-Oriented Programming (AOP) beyond Dependency Injection” presented during the technical paper sessions. GluonJ is an AOP system for reducing the dependency among components written in Java. GluonJ allows developers to explicitly construct an aspect instance and associate it with aspect targets. The aspect instance is a component implementing a crosscutting concern and the aspect targets are components that the concern cuts across.

Our poster shows that GluonJ provides the flexibility to express component relationships by a pointcut-advice pair and thereby reduce the inter-component dependency. GluonJ provides better flexibility than other AOP systems like AspectJ with respect to constructing an aspect instance and associating it with the aspect targets. An aspect instance of other AOP systems is implicitly constructed and associated with aspect targets. They let developers select a type of construction and association from limited number of descriptors like singleton, perthis, and so on. these options cover only limited kinds of relations among components. For example, AspectJ does not provide a descriptor for associating an aspect instance with a group of the aspect targets. On the other hand, GluonJ can make this association since GluonJ allows explicitly describing the construction and association.

Our goal is to provide an AOP-based solution for reducing inter-component dependency, which is partly solved by dependency injection. The dependency injection is a technology used by a new generation of component frameworks like the Spring Framework. It moves the code for instantiating sub-components from a component to a component framework. The developers do not have to describe a component that depends on a particular platform that supplies subcomponents. However, since existing component frameworks using dependency injection are implemented with a normal language, mostly in Java, their ability is unsatisfactory. Our AOP-based solution, that is, GluonJ provides better ability.

- **P4: A Pattern-Driven Software Development Framework**

Haitham S. Hamza *Univ. Nebraska-Lincoln*, Yi Chen *Microsoft*,
Mohamed E. Fayad *San José State University*

Software patterns have emerged as a promising technique for documenting and communicating experience among developers. Analysis patterns are conceptual models that can be reused to analyze similar and related problems, and hence, they provide a valuable resource for domain modeling. Design patterns, on the other hand, can provide a solution to frequently recurring design problems, leading to less costly high-quality design.

Despite the existence of analysis and design patterns for more than a decade, less attention has been devoted to integrate the use of analysis patterns into the traditional software development life-cycles. Moreover, there has been no attempt to integrate the use of patterns across the different development phases. Existing techniques for using patterns to develop software systems are focused on selecting and integrating architectural and design patterns to evolve systems’ models. Analysis phase, on the other hand, is performed using traditional techniques.

To our knowledge, the Pattern-Oriented Software Analysis and Design (POAD) method proposed in is the only approach that emphasizes the use of patterns in both analysis and design. POAD performs analysis by viewing the system as a collection of components and identifying responsibilities and functionalities of each of these components. The design step is conducted by selecting different design patterns to implement the different identified components. The main difference between our approach and the POAD approach is that our approach uses analysis patterns to the analysis and modeling of the problem, while POAD does not.

We use analysis patterns and design patterns to analyze the problem and to construct the design model. The resultant design model is then implemented. The approach starts by analyzing the problem and decomposes it into smaller problems. These problems are then matched against a repository of analysis patterns and conceptual models. Each analysis pattern is used to guide the selection of the appropriate design patterns to develop the design model. To so, the following main activities are applied: (1) Extract association relationships between the different classes in the analysis pattern, and identify responsibilities associated with these relationships; (2) Identify the nature of the design pattern that can be used to realize the identified responsibilities; (3) Identify specific design patterns to be used; and (4) Construct the identified design patterns to develop a design model. At the end, a composition process takes place to integrate the design models developed for the different analysis patterns.