

Workshop for PhD Students in Object Oriented Programming

Pedro J. Clemente¹, Miguel A. Pérez¹, Sergio Lujan², and Hans Reiser³

¹ Department of Computer Science. University of Extremadura, Spain[†]
{jclemente, toledano}@unex.es

² Department of Languages and Computer Science. University of Alicante, Spain.
sergio.lujan@ua.es

³ Department of Distributed Systems and Operating Systems.
University of Erlangen-Nürnberg, Germany.
reiser@cs.fau.de

Abstract. The objective of the 13th edition of Ph Doctoral Students in Object-Oriented Systems workshop (PHDOOS) was to offer an opportunity for PhD students to meet and share their research experiences, and to discover commonalities in research and student ship. In this way, the participants may receive insightful comment about their research, learn about related work and initiate future research collaborations. So, PHDOOS is a gauge to detect hot spots in current lines of research and new avenues of work in objects-oriented concepts.

1 Introduction

At its 13th edition, the PHDOOS workshop established the annual meeting of PhD students in object-orientation. The main objective of the workshop is to offer an opportunity for PhD students to meet and share their research experiences, to discover commonalities in research and studentship, and to foster a collaborative environment for joint problem solving.

The workshop also aims at strengthening the International Network of PhD Students in Object-Oriented Systems[17] initiated during the 1st edition of this workshop series at the European Conference on Object Oriented Programming (ECOOP), held in Geneva, in 1991. This network has counts approximately 120 members from all over the world. There is a mailing list and a WWW site, used mainly for information and discussion about OO-related topics. Since its foundation, the International Network of PhD Students has proposed and created a workshop for PhD students in association with ECOOP each year. This 13th edition makes PHDOOS a classical workshop in ECOOP.

At this edition, the workshop was divided into plenary sessions, discussion session and a conference. The sessions were determined according to the research interests of the participants. Potential topics of the workshop were those of the

[†] The organization of this workshop has been partially financed by CICYT, project number TIC02-04309-C02-01

main ECOOP conference, i.e. all topics related to object technology including but not restricted to: analysis and design methods, real-time, parallel systems, patterns, distributed and mobile object systems, aspects oriented programming, frameworks, software architectures, software components, reflection, adaptability, reusability and theoretical foundations. Due to the heterogeneous nature of the topic of the papers received, the workshop conclusions are focused on the interesting research areas and on solving common problems.

The participants had a 20 minute presentation at the workshop (including questions and discussions). The discussion group was based on the research interests of the participants. Finally, the conference featured a speaker who is invited to talk about interesting research, personal experiences or research methodology. This conference was a unique opportunity to hear or ask things not discussed elsewhere, and to have an "unplugged" discussion with a well-known personality from our field. This year, the speaker was the professor Robert E. Filman

This paper is organized into the following points. The next section is focused on the participants of the workshop. In section three, the presented papers are explained and main topics are summarized. The fourth section talks about a conference, and the fifth section includes the final discussion. Finally the paper concludes with workshop conclusions and bibliographic references.

2 PHDOOS Workshop Participants

In previous editions [4,1,2,3] of this workshop there have been more or less twenty participants working for 2 days. However, this year the coincidence with a Doctoral Symposium cut the number of participants with papers to 10, thereby reducing the length of this workshop to 1 day. We can divide the participants into four groups:

- Participants with papers
- Organizing Committee.
- Program Committee
- Invited speaker.

2.1 Participants with Papers

The number of papers received was 11, and 9 were accepted. The attendants with accepted papers were the following:

- M. Devi Prasad. Manipal Center for Information Science, Manipal Academy of Higher Education.
- Hervé Paulino. Department of Computer Science. Faculty of Sciences and Technology. New University of Lisbon.
- Sari R. ElDadah, Nidal Al-Said. Department of Information Systems. Arab Academy For Banking and Financial Sciences.
- Ademar Aguiar. Faculdade de Engenharia da Universidade do Porto.

- Balázs Ugron. Department of General Computer Science. Eötvös Loránd University, Budapest.
- Amparo Navasa Martínez. Quercus Software Engineering Group. University of Extremadura.
- Susanne Jucknath. Institute of Software Engineering and Theoretical Computer Science Technical University of Berlin.
- Jan Wloka. Fraunhofer FIRST.
- Szabolcs Hajdara. Department of General Computer Science. Eötvös Loránd University, Budapest.
- Andreas I. Schmied. Distributed Systems Laboratory. University of Ulm.

2.2 Organizing Committee

The Organizing Committee of Ph Doctoral Object-Oriented Systems is made up of volunteers participants in the previous workshop. Organizers in earlier editions advise these volunteers. This year, the organizers were: Sergio Luján Mora, Sérgio Soares, Hans Reiser, Pedro José Clemente Martín and Miguel Angel Pérez Toledano.

Sergio Luján-Mora is a Lecturer at the Computer Science School at the University of Alicante, Spain. He received a BS in 1997 and a Master in Computer Science in 1998 from the University of Alicante. Currently he is a doctoral student in the Dept. of Language and Information Systems being advised by Dr. Juan Trujillo. His research spans the fields of Multidimensional Databases, Data Warehouses, OLAP techniques, Database Conceptual Modeling and Object Oriented Design and Methodologies, Web Engineering and Web programming.

Sérgio Soares is currently a Ph.D. student at the Computer Science Center of the Federal University of Pernambuco. His current research interests include implementation methods and aspect-oriented programming. He is also a Assistant Professor at the Statistics and Computer Science Department of the Catholic University of Pernambuco

Hans Reiser studied Computer Science at the University of Erlangen-Nürnberg, obtained Dipl. Inf. degree in computer science in spring of 2001. Since 6/2001 he has been employed as researcher at the Department of Distributed Systems and Operating Systems at University of Erlangen-Nürnberg. He is member of the AspectIX research team (a joint group of our department and distributed systems department of University of Ulm), doing research on adaptive middleware for large-scale distributed systems. The primary research focus for PhD is software based fault tolerance in distributed systems.

Pedro José Clemente Martín graduated with a Dipl. Inf. in Computer Sciences from the University of Extremadura (Spain) in 1998. He is a Lecturer at the Computer Science Department at the University of Extremadura, Spain and a member of the Quercus Software Engineering Group. His current research interests are Aspect Oriented Programming and Component based Software Engineering. His PhD is focused on the interconnection of components to build software systems using aspect oriented programming.

Miguel Angel Pérez Toledano graduated with a Dipl. Inf. in Computer Sciences from the Polytechnic University of Cataluña (Spain) in 1993. Now, he is working at the University of Extremadura as Associated Professor. He participated in the organization of the PhD workshop of ECOOP'02, and is a member of the Quercus Software Engineering Group. His current research interests are Semantic Description of Components and Repositories of Software Components. His PhD is focused on the selection and retrieval of components from repositories using semantic descriptions.

2.3 Program Committee

The program committee was composed of senior researchers with a strong background in some object-oriented topic. The review process is designed to ensure that every participant is able to present some relevant and well prepared material. This edition, the program committee was composed by:

- Marcelo Faro do Amaral Lemos (Federal University of Pernambuco, Brazil)
- Márcio Lopes Cornelio (Federal University of Pernambuco, Brazil)
- Juan C. Trujillo (University of Alicante, Spain)
- Fernando Sánchez Figueroa (University of Extremadura, Spain)
- Juan Manuel Murillo Rodríguez (University of Extremadura, Spain)
- Rüdiger Kapitza (University of Erlangen-Nürnberg, Germany)
- Andreas Schmied (University of Ulm, Germany)
- José Samos (University of Granada, Spain)

2.4 Invited Speaker

For this edition, the invited speaker to PhDOOS was professor Robert E. Filman. He is working at the Research Institute for Advanced Computer Science (RIACS), NASA Ames Center. His work about Aspect Software Oriented Development is recognized by the international computer science research community.

3 Research Hot Points

One of the main workshop objectives is to detect the principal research lines following Object Oriented Programming and Programming Technologies. In this sense, the following topics have been presented and have been discussed:

- Aspect Oriented Software Development
- Documentation and Categorization of Software
- Agents Technologies

3.1 Aspect Oriented Software Development

Aspect Oriented Software Development has been the most important point discussed in this workshop due to the fact that sixty percent of the accepted papers dealt with this topic. In this sense, we can find several topics and focuses that allow us to ensure that the area of Aspect Oriented Technologies is currently very important.

The subjects treated included extension and description of Aspect-Oriented Languages, Aspect Oriented Code Visualization, Composition of Aspects, and domain of Aspect-Oriented application.

About *Aspect-Oriented Languages* there are two approaches: to extend an existing language (for example, AspectJ) and to define new languages to describe and compose aspects (for example, using ADL).

Prassat [13] suggests that AspectJ does not treat type-casting as an important operation in programs execution. He demonstrates that Java programs use type casting primarily for retrieving references to new types from distantly related ones. He investigates AspectJs inadequacy in selecting execution points that use type-casting to yield new object or inheritance references. He demonstrates the necessity for a reference creation by type casting joinpoints and argues that its addition makes AspectJs existing model more expressive.

Amparo Navasa [6] claims that to extract the aspect code, crosscutting the functional one makes it easier to implement aspect oriented systems using AOP languages, but it is possible to take away the problem of AO from the implementation phase to the design. In this phase, crosscutting functional code concerns can be treated as independent entities. Her ideas are based on developing aspect oriented systems taking into account the benefits of applying CBSE at the early stages of AO systems development, particularly at architectural design. Concretely, AOSD at the design level as a co-ordination problem, using an ADL to formalize it. This means a new language to define aspects and compose them from the design time viewpoint.

Aspect Oriented Software Visualization presents a growing potential due to the fact that graphic tools do not currently exist to visualize the aspect code execution. In this sense, one intention of Software Visualization is to form a picture in the user's mind of what this software is about, what happens during the execution and what should happen from the programmer's point of view[12]. This is especially helpful with existing software and non-existing documentation. Therefore the amount of existing software is increasing the need to understand this software too.

From *Composition of Aspects* viewpoint, there is special interest in Distributed Systems, because they contain lots of cross-cut and tangled code fragments to fulfill their services. Merging a formerly separated aspect code with a program code by means of aspect-oriented programming is enabled through a couple of available technologies that manipulate program sources. Unfortunately, these tools or aspect weavers both operate on a distinct coarse-grained level (types, methods) and fulfill only a restricted a-priori known set of manipulations.

However, to weave several aspect code fragments which could have been constructed by independent teams for more than one concern simultaneously a composition that not only concatenates aspects, but also manages join effects between them, reveals several complex, possibility interfering weaving demands[7].

The use of *Aspect Oriented* in specific domains or concrete areas in Software Engineering is other growing area. In this sense, Hajdara[10] presented a solution to apply Aspect Oriented technologies to handle different non-functional properties like synchronization specification of parallel systems.

Refactoring and Aspect Orientation (AO) are both concepts for decoupling, decomposition, and simplification of object-oriented code. Refactoring is meant to guide the improvement of existing designs. For this reason it is the main practice in eXtreme Programming to implement 'embrace change' in a safe and reliable way. Aspect orientation on the other hand offers a new powerful encapsulation concept especially for coping with so called crosscutting concerns. Although refactoring and AO have the same goals, their current forms impede each other. Since the development of modular systems has become more and more difficult a combined application of refactoring and AO is still a desirable goal and would be a great help for developers[16].

3.2 Documentation and Categorization of Software

Two papers were presented on this topic during the workshop. The first paper "A Process-based Framework for Automatic Categorization of Web Documents" from Sari R. ElDadah[15], presented the design of a Framework for the development of Automatic Text Categorization applications of Web Documents. The process, composed of 4 activities, (identifying significant categories, finding the best description for each category, classifying the documents into the identified categories and personalizing the categories and their relevant descriptions according to the user preference) is conducted from the various Automatic Text Categorization methods developed so far, and described based on Petri Nets process description language. The paper concluded with notes about the ATC process.

The second paper presented on this topic was titled "A Minimalist Approach to Framework Documentation" from Ademar Aguiar and Gabriel David[5]. This paper proposes a documenting approach for frameworks that aims to be simple and economical to adopt. It reuses existing documentation styles, techniques and tools and combines them in a way that follows the design principles of minimalist instruction theory. The minimalist approach proposes a documentation model, a documentation process, and a set of tools built to support the approach (Extensible Soft Doc).

3.3 Agents Technologies

In this subject, we can include the paper "Mob: a Scripting Language for Programming Web Agents" showed by Hervé Paulino[11]. Mob: a scripting language for programming mobile agents in distributed environments was presented. The

semantics of the language is based on the DiTyCO (Distributed TYped Concurrent Objects) process calculus. Current frameworks that support mobil agents are mostly implemented by defining a set of Java classes that must then be extended to implement a given agent behavior. Mob is a simple scripting language that allows the definition of mobile agents and their interaction, an approach similar to D'Agents. Mob is compiled into a process-calculus based kernel-language, and its semantics can be formally proved correct relative to the base calculus.

4 Workshop Conference

The workshop conference entitled *Aspect Oriented Software Development* was presented by Robert E, Filman.

Robert presented an interesting conference which has four parts:

- Object Infrastructure Frameworks (OIF)
- Aspect-Oriented Software Development
- AOP through Quantification over Events
- Research Remarks

The conference began with an introduction about Robert's latest research projects. He then introduced the concept of *Aspect Oriented Software Development*, and a way to obtain AOP through Quantification over Events. Finally, some remarks about research directions were presented.

Now, we are going to summarize each part of this presentation, because Robert's affirmations are very interesting, above all for students interested in Aspect-Oriented.

4.1 Object Infrastructure Framework (OIF)

Distributed Computing systems is difficult mainly for the following reasons:

- It is hard to archive systems with systematic properties (called *Itities*) like *Reliability, Security, Quality of Service, or Scalability*.
- Distribution is complex for the following reasons: concurrence is complicated, distributed algorithmics are difficult to implement, every policy must be realized in every component, frameworks can be difficult to use, etc.

The introduction of a Component based Architecture require separating the component functionality and the non-functional properties. These non-functional properties should be inserted into components and allow for the interaction (communications) among components.

This idea has been implemented using Object Infrastructure Frameworks (OIF) [14]. OIF allows for the injection of behavior on the communications paths between components, using *injectors* because they are discrete, uniform objects, by object/methods and dynamically configurable. This idea permit the implementation of non-functional properties like *injectors*, and then they can be

applied to the components; for example, injectors can encrypt and decrypt the communications among components.

OIF is an Aspect Oriented Programming mechanism due to the fact that:

- It allows separating concerns into injectors
- It wrapping technology
- It piggy-backs on distributed-object technology (CORBA)

4.2 Aspect-Oriented Software Development

How can we structure our programming languages do help us archive such ilities(Reliability, Security, Quality of Services, Evolvability, etc.)?

Separation of Concerns is an interesting strategy to structure our programming languages because a fundamental engineering principle is that of *separation of concerns*.

Separation of Concerns promises better maintainability, evolvability, Reusability and Adaptability. Concerns occur at both the User/requirements level and Design/implementation level[8].

Concerns cross-cut can be Applied to different modules in a variety of places, and must be composed to build running systems.

In conventional programming, the code for different concerns often becomes mixed together (tangled-code).

Aspect Oriented Programming modularize concerns that would otherwise be tangled. AOP provides mechanisms to weave together the separate concerns.

Implementation Mechanism. The following division allows for the description of the common AOP implementation mechanisms used and the usual platforms used:

- *Wrapping technologies*: Composition filters, JAC
- *Frameworks*: Aspect-Moderator Framework
- *Compilation technologies*: AspectJ, HyperJ
- *Post-processing strategies*: JOIE, JMangler
- *Traversals*: DJ
- *Event-based*: EAOP
- *Meta-level strategies*: Bouraqadi et al., Sullivan, QSOUL/Logic Meta-Programming

4.3 AOP Through Quantification over Events

A single concerns can be applied to many places in the code, but the we need to quantify it.

Concerns can be quantified over the static(lexical) form of the program, semantic (reflective) structure of the program structures and the events that happen in the dynamic execution of a system.

To take the expressiveness in quantification to its extreme is to be able to quantify over all the history of events in a program execution. The events are with respect to the abstract interpreter of a language. However, language definitions do not define their abstract interpreters.

As a consequence, we are able to describe interesting points in the program (lexical structure of the program, reflective structure of the classes and dynamic execution of the system), and then to describe the change in behavior desired at these points. The shadow of a description is the places in the code where the description might happen, for example, the event *invoking a subprogram* represents in a syntactic expression *subprogram calls*. It is necessary to define these events, capture these, and to change the behavior at this point. For more detail, please refer to [9].

4.4 Research Remarks

This section of the conference presents the main research directions about *Aspect Oriented Software Development*, and this information should be useful for current and prospective Phd Students.

Research Regime

- Define a language of events and actions on those events.
- Determine how each event is reflected (or can be made visible) in source code.
- Create a system to transform programs with respect to these events and actions.
- Developing an environment for experimenting with AOP languages (DSL for AOP)

Real AOP Value

- We don't have to define all these policies before building the system
- Developers of tools, services, and repositories can remain (almost) completely ignorant of these issues
- We can change the policies without reprogramming the system
- We can change the policies of a running system

Open Research Directions

Languages for Doing AOP

- Hardly seen the end of this topic
- Join points
- Weaving mechanisms
- Handling conflicts among aspects

The Software Engineering of AOP Systems

- Modeling aspects: From models to executable code
- Debugging aspects independently of the underlying system
- Tools for recognizing and factoring concerns

Applying AOP to Particular Tasks

- Monitoring/debugging
- Version control/provenance
- Web/system services
- User-centered computing
- Reliable systems
- System management

5 Summary of the Discussion Group

Although the workshop has a wide focus it turned out that most participants are working in research areas closely related to aspect oriented programming. Instead of having small subgroup discussions, the group opted for one plenary sessions discussing topics on AOP-related topics. In the process of selecting appropriate topics, we came up with the following main issues:

5.1 Shall Aspects Be Supported as First Class Entities?

The situation today is that most AOP languages do not support aspects as first class entities. This is however due to the simple pragmatic way these languages are implemented. One may anticipate that the next generation of AOP languages will provide support for aspects as first class entities. The main benefits which we expect from such future developments are reusability of aspects, inheritance between aspects, and dynamically adaptive aspects. These areas still offer some research potential.

5.2 Does AOP Make Sense with Non-OOP Paradigms?

This issue was only briefly discussed. The group agreed that in paradigms like functional or imperative programming, separation of concerns is an equally required design method, and AOP is useful technique to support this. However, related to the generalization of aspects discussed later, non-OOP aspect orientation will have somewhat different requirements on potential join-point definition than in the case of OOP.

5.3 What Are Adequate Techniques to Understand AOP Programs?

One major problem with AOP is that while it simplifies things on a rather abstract level, it gets more difficult to understand the concrete behavior of your program at a lower level. Current visualization techniques, already offered in some development environments, are not yet adequate for larger projects. The issues to be supported by visualization techniques are documentation, testing and debugging. The demand for such techniques will further rise, if aspect code and functional code shall be developed independently

5.4 What Purposes Shall Aspects Be Used For?

One widespread use of aspects is to restructure existing software (refactoring to increase modularization), with the goal to improve structure and maintainability. However, we anticipate that in the future, AOP will also be applied for new applications, starting in the design phase. For this purpose, the availability of reusable “aspect components”, addressed in the next item, will be essential. A different question is whether AOP techniques may and shall be used for modifying existing applications, that were developed without considering such later modification. However, we were rather reluctant to consider this as a good AOP technique.

5.5 Is It Feasible to Make Aspects Reusable?

Closely related to the previous topic is this important issue. In our opinion, the most problematic matter is the definition of join points. In current AOP languages, the definition of aspects is always tailored individually to one specific application. Even in such a specific case, existing AOP tools are usable best if the application and the aspects are written by the same developer. Also, even a small modification to one application easily makes aspects unusable. We all agree that this is a highly disappointing situation.

Having reusable aspects is highly desirable, but it requires further research on how this might be done. An important issue in this context is the question of whether aspects can be defined without them? limiting to an specific AOP language. Ultimately, AOP needs to be done already in the design phase of application development.

5.6 Conclusions

In spite of the fact that AOP has matured for over the years, several issues can be found that are still relevant for future research. The most important issue we found are the definition of join points targeting at reusability of aspects, and tool support for visualizing and understanding aspect oriented applications.

References

1. 10th Workshop for Ph Doctoral Students in Objects Oriented Systems. <http://people.inf.elte.hu/phdws/>, 2000.
2. 11th Workshop for Ph Doctoral Students in Objects Oriented Systems. <http://www.st.informatik.tu-darmstadt.de/phdws/>, 2001.
3. 12th Workshop for Ph Doctoral Students in Objects Oriented Systems. <http://www.softlab.ece.ntua.gr/facilities/public/AD/phdoos02/>, 2002.
4. 9th Workshop for Ph Doctoral Students in Objects Oriented Systems. <http://www.comp.lancs.ac.uk/computing/users/marash/PhDOOS99/>, 1999.
5. Gabriel David Ademar Aguiar. A minimalist approach to framework documentation. *13th Workshop for Phd Students in Object Oriented Programming at ECOOP. Darmstadt, Germany.*, 2003.
6. J.M. Murillo A.Navasa, M.A.Pérez. Using an adl to design aspect oriented systems. *13th Workshop for Phd Students in Object Oriented Programming at ECOOP. Darmstadt, Germany.*, 2003.
7. Franz J. Hauck Andreas I. Schmied. Composing non-orthogonal aspects. *13th Workshop for Phd Students in Object Oriented Programming at ECOOP. Darmstadt, Germany.*, 2003.
8. Robert E. Filman and Daniel P. Friedman. Aspect-oriented programming is quantification and obliviousness. *Workshop on Advanced Separation of Concerns. OOP-SLA, Minneapolis*, 2000.
9. Robert E. Filman and Klaus Havelund. Source-code instrumentation and quantification of events. *Workshop on Foundations Of Aspect-Oriented Languages (FOAL) at AOSD Conference. Twente, Netherlands.*, 2002.
10. Szabolcs Hajdara. An example of generating the synchronization code of a system composed by many similar objects. *13th Workshop for Phd Students in Object Oriented Programming at ECOOP. Darmstadt, Germany.*, 2003.
11. Luís Lopes Herve Paulino and Fernando Silva. Mob: a scripting language for programming web agents. *13th Workshop for Phd Students in Object Oriented Programming at ECOOP. Darmstadt, Germany.*, 2003.
12. Susanne Jucknath. Software visualization and aspect-oriented software development. *13th Workshop for Phd Students in Object Oriented Programming at ECOOP. Darmstadt, Germany.*, 2003.
13. M. Devi Prasad. Typecasting as a new join point in AspectJ. *13th Workshop for Phd Students in Object Oriented Programming at ECOOP. Darmstadt, Germany.*, 2003.
14. Diana D. Lee Robert E. Filman, Stu Barrett and Ted Lindero. Inserting ilities by controlling communications. *Communications of the ACM, January*, 45, No 1:118–122, 2002.
15. Nidal Al-Said Sari R. ElDadah. A process-based framework for automatic categorization of web documents. *13th Workshop for Phd Students in Object Oriented Programming at ECOOP. Darmstadt, Germany.*, 2003.
16. Jan Wloka. Refactoring in the presence of aspects. *13th Workshop for Phd Students in Object Oriented Programming at ECOOP. Darmstadt, Germany.*, 2003.
17. International Network for PhD Students in Object Oriented Systems (PhDOOS). <http://www.ecoop.org/phdoos/>, 1991.