Programming with Views and Collaborations in ObjectTeams/Java

Stephan Herrmann and Christine Hundt
Technische Universität Berlin
{stephan,resix}@cs.tu-berlin.de

This document proposes a tutorial to be held at AOSD 2006 in Bonn.

1 The Instructors

1.1 Dr.-Ing. Stephan Herrmann (primary contaxct)

name:	Dr.-Ing. Stephan Herrmann
position:	Assistant Professor
affiliation/ postal address:	Technische Universität Berlin Sekr. FR 5-6 Franklinstr. 28/29 10587 Berlin Germany
email:	stephan@cs.tu-berlin.de
webpage:	http://swt.cs.tu-berlin.de/~stephan
fone:	+49 30 314 73174
fax:	+49 30 314 73488

1.1.1 Selective Biography

Dr. Herrmann is an Assistant Professor at the Technische Universität Berlin. He received his PhD in 2002, the title of his thesis being "Views and Concerns and Interrelationships – Lessons Learned from Developing the Multi–View Software Engineering Environment PIROL". Also in 2002 he made the first international publication on the AOP language ObjectTeams/Java. Since 2003 Dr. Herrmann leads the joint research project TOPPrax (3 research institutes 2 companies), a publicly funded project for the evaluation of aspect oriented software development in practical application.

Dr. Herrmann was co-organizer of CD'02 (1st International IFIP/ACM Working Conference on Component Deployment), panelist for the Young Research Workshop at GPCE'03, organizer of EIWAS'04 (European Interactive Workshop on Aspects in Software), and of VAR'05 (Views, Aspects and Roles, workshop at ECOOP 2005). He will be an Organizing Co-Chair of ECOOP 2007. He also served as a program committee member for the conferences CD 2002, CD 2003, Net.ObjectDays 2005, and various workshops (e.g., the AOSD workshops ACP4IS, Early Aspects and DAW) and also as a reviewer for several journals.

1.1.2 Teaching experience

Starting in 1997, Dr. Herrmann has continuously taught various classes on software engineering with special focus on object oriented software development. One particular class has been developed by him in 1999. Already in the first edition of this class he gave an outlook to ongoing research in the fields of AOP and SOP. Ever since he has continued to closely link his research and teaching activities. The language ObjectTeams/Java is part of this class since 2003, meaning that one out of about four practical assignments has to be programmed in ObjectTeams/Java.

1.2 Dipl. Inform. Christine Hundt

name:	Dipl. Inform. Christine Hundt
position:	Research Associate
affiliation/ postal address:	Technische Universitt Berlin Sekr. FR 5-6 Franklinstr. 28/29 10587 Berlin Germany
email:	resix@cs.tu-berlin.de
webpage:	http://swt.cs.tu-berlin.de/staff/ChristineHundt.html
fone:	+49 30 314 73419
fax:	+49 30 314 73488

1.2.1 Selective Biography

Dipl. Inform. Hundt finished here studies in early 2003. In here master's work she developed the runtime environment for ObjectTeams/Java, which she is still maintaining. Since fall 2003 she works as a research associate for the TOPPrax project (see above). Within this project she participated in the design of tutorials at our industrial partner, some lessons of which she taught in person. She is also responsible for extending the runtime environment from her master's work towards true runtime weaving. In this field she is currently supervising three students for their master's work.

2 The Tutorial

Title
: Programming with Views and Collaborations in ObjectTeams/Java

Kind
: Half-Day

Level
: Intermediate.

2.1 Abstract

Aspect-oriented programming promises to significantly improve modularity for a specific class of aspects, that cut across the system structure as defined by classes and packages. The TOPPrax project systematically investigates the practical applicability of new programming languages and assesses the benefits for commercial software development.

This tutorial applies the second generation aspect language ObjectTeams/Java. By the collaboration-based approach of ObjectTeams/Java it is possible to structure the design and even the implementation according to the use-cases of an application. This greatly improves the tracebility from requirements down to code and significantly reduces the efforts needed for software maintenance and evolution.

In this tutorial participants will learn how to develop reusable collaboration modules in the vein of collaboration-based design methods. The powerful integration mechanisms of ObjectTeams/Java will be used to demonstrate a-posteriori integration of modules cleanly separating functionality from integration. This is the basis for fundamentally improved modularity yielding easily adaptable architectures and facilitating future evolution.

Participants will also learn, how framework technology can be taken one step further by applying inheritance to a whole collaboration module. They will furthermore learn how to use collaboration instances to dynamically activate/deactivate aspects at runtime, yielding a more dynamic structure of the application including client-specific contexts and software modes. Various examples demonstrate, how aspects can be generalized to views, yielding an improved module structure for a wide range of typical situations in software.

The tutorial has been successfully taught at our industrial partners and at Net.ObjectDays 2005. Practical examples will be shown using the comprehensive, Eclipse based IDE for ObjectTeams/Java, which is freely available at our web site. Participants are expected to have good knowledge of object oriented programming and Java in particular, and should be interested in high-quality software designs.

For further information see `www.ObjectTeams.de`.

3 Synopsis

In this tutorial participants will learn how to apply the programming language ObjectTeams/Java in order to address a variety of modularity issues which cannot be solved in a satisfactory way with traditional object oriented languages. The language ObjectTeams/Java (OT/J for short) emerged from the Aspectual Components model and furthermore integrates other advanced techniques like family polymorphism, confined objects and supports both aspect oriented as well as collaboration based programming with roles.

The relevance of this tutorial lies in the fact, that OT/J is a modern programming language incorporating latest research results and at the same time is intended for real world application. Thus the nature of this language can only be truly experienced by programming a number of different applications in OT/J where each example sheds a light on a specific property of the language.

The tutorial is intended for practitioners who want to learn about current trends in programming technology as well as for teachers who like to include new paradigms in their classes. Also researchers in this field shall benefit from the tutorial as it will provide an in-depth look into a technology that is more than a particular solution to a particular problem, but generalizes over a set of concepts and technologies.

3.1 Overall structure

The tutorial will be given in two parts. Part one introduces the programming language OT/J at a conceptual level showing only small examples. Part two will show patterns for good design with OT/J, thus diving into an emerging method for applying OT/J in practical software development. Both parts will be interspersed with practically demos of running programs also showing the support provided by the tool environment OTDT (the Eclipse-based Object Teams Development Tooling).

3.2 Goals

The audience shall learn how OT/J supports improved modularity in three dimensions.

"Teams" are introduced as a new kind of module that combines desirable properties of classes, packages and components. Teams are like classes in that they are instantiable modules defining fields and methods, but they provide better support for large scale designs. Teams are like packages in that they are containers for classes, but there is much more developers can do with teams than with packages, due to the class-like properties of teams. Finally, teams can be used to program light-weight components, meaning that they allow some encapsulation and a-posteriori integration known from components without the need of an application server to host the components.

Along the second dimension participants will learn about the relationships which can be defined between teams, in order to compose a system from subsystems and modules. On this path team inheritance (with overriding of virtual classes following the concept of family polymorphism) will be shown as means for programming-by-difference at a level of granularity the goes beyond classes. It will be shown how team inheritance takes the idea of object-oriented frameworks to the next level. The second relationship to be explained allows a team to *adapt* an existing application. This adaptation relationship combines aspect-oriented techniques with concepts from binding role objects to their bases. The adaptation relationship support much more flexible designs in terms of decoupling, options for multilplicities not supported by inheritance and various forms of runtime dynamism.

Special focus of this tutorial lies finally on programming with views. Here participants will learn how they can use OT/J such that each part of the system is developed using a most suitable ontology. Different views within a system can be connected by the adaptation relationship mentioned above. Much in the vein of the hyperspaces approach independently developed modules can be composed to a system. Two main differences with respect to the hyperspaces approach are the lack of a language boundary (there is no distinct composition language, everything is done

within the language proper) and a much more dynamic composition model (objects, teams, roles and even their bindings all are dynamic instance based techniques).

Apart from the core concepts participants shall learn about patterns which have been identified within existing programs written in OT/J. The granularity of these patterns ranges from small idioms to medium scale architectures. By presenting these patterns real world applicability of OT/J is demonstrated and participants receive valuable instructions for exploiting the new concepts towards improved designs that are much better evolvable than their plain object oriented counter-parts.

3.3 Syllabus

The tutorial is structured in two parts: core concepts and patterns.

3.3.1 Core concepts

The first unit will introduce the fundamental concepts of Object Teams, the technical realization by compiler, runtime environment and IDE, and provide a glance into the language definition document which has matured over the last years and by now is a stable point of reference regarding OT/J. This unit is framed by the demonstration of a teaser example in which we show how an existing application with GUI and database can a-posteriori be extended by an aspect for input validation. The first showing of the teaser will not show its realization which will be explained at the end of this unit.

3.3.2 Patterns for good design with OT/J

Following the tradition of design patterns, this section of the tutorial will always start with a general problem and some seemingly conflicting forces that restrict the design space. Only from that problem statement specific best-practice solutions in OT/J will be presented giving clear information on the impact of that specific solution and degrees of freedom by which variants of the pattern can be constructed.

Small idioms will be presented at the level of single statements. Examples will be:

- "How can I achieve selective aspect activation for specific registered base objects only?"
- "How do I implement a notification protocol?"

The second example demonstrates how former design patterns (here the Observer pattern) will shrink to mere idioms since the underlying concept is directly supported by the programming language.

As an example for larger scale patterns the Connector architecture will be taught, in which team inheritance and the adaptation relationship are used to connect two independently developed modules. Several medium level patterns will be shown, which help to establish such a Connector architecture. Virtual Association rebuilds the association structure of a base module by another association structure within a team without maintaining redundant links. Several variants of Virtual Restructuring allow to define the structure within a team as a re-mapped view of a base module. By this re-mapping structures may differ significantly and yet entities can be connected in such a way that the program may handle them as different views of the same conceptual entity.

Also, External Variants are presented as a way of constructing variantes of a given module without changing its code. External Variants solve a similar problem as team inheritance but they are supperior over team inheritance when multiple variants are to be supported simultaneuosly, in unanticipated combinations and if variants shall be enabled or disabled at runtime.

3.3.3 Summary

When wrapping up everything that was covered during the tutorial, a medium sized design will be shown which uses teams throughout. This final example demonstrates the scalability of the presented concepts. By scalability we mean the possibility to create nested and layered structures without technical restrictions. The audience will learn that it makes hardly any difference whether an existing application or sub-system only uses traditional object-oriented

techniques or whether it has already been build using OT/J. The same concepts can be applied to teams and roles as well as to regular classes. In fact three styles of creating larger structures will be shown, which we call Nesting, Layering and Stacking.

3.3.4 Schedule

Each of the two parts will take 90 minutes with a coffee break in-between.

3.4 Additional information

3.4.1 Targeted Audience

The tutorial mainly targets at developers with good skills in object-oriented programming, who are well aware of the difficulties of finding and maintaining good designs for complex software. Participants are not required to have a particular background in AOSD, although some knowledge in this field will certainly help. The level of teaching is classified as intermediate, meaning that no prior knowledge about Object Teams is required, but assuming good OO-knowledge we will quickly advance into realistic problems and their solutions.

3.4.2 Previous teaching

After initial versions in classroom and at our industrial project parnter, this tutorial has in its proposed shape successfully been taught at the conference Net.ObjectDays 2005 in Erfurt. In Erfurt it was taught by the same instructors as for this proposed tutorial.

Slides are included from the Erfurt tutorial, which will be adapted and improved further for the next edition of the tutorial.

3.4.3 Equipment

Required equipment includes a video projector and a white board. A second video project would be great, but is not mandatory.

3.4.4 Material for participants.

Participants will be encouraged to download the software including examples from our web page in advance. We will also bring CDs for participants and some handouts.

Hands-on exercises will not be part of the tutorial proper, but we will encourage participants to make first experiments with our tool and the language ObjectTeams/Java during the coffee break and after the tutorial. We will certainly provide help for those first steps on-site.

References

[1] Object Teams home page. http://www.ObjectTeams.org.