10152 Abstracts Collection
Relationships, Objects, Roles, and Queries in Modern Programming Languages
— Dagstuhl Seminar —

Guido Boella[1], Erik Meijer[2], David J. Pearce[3], Friedrich Steimann[4] and Frank Tip[5]

[1] University of Torino, IT
guido@di.unito.it
[2] Microsoft Corp. - Redmond, US
[3] Victoria University of Wellington, NZ
david.pearce@mcs.vuw.ac.nz
[4] Fernuniversität in Hagen, DE
steimann@acm.org
[5] IBM TJ Watson Research Center - Hawthorne, US
ftip@us.ibm.com

Abstract. From 11/04/10 to 16/04/10, the Dagstuhl Seminar 10152 "Relationships, Objects, Roles, and Queries in Modern Programming Languages " was held in Schloss Dagstuhl – Leibniz Center for Informatics. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Relationships, Roles, Software Modelling, Programming Languages

Seminar Topics and Goals

The notions of relationship, object and role are common in many domains such as, for example, sociology, cognitive science, organisational science and linguistics. These notions are also well-supported in many areas of computer science, including: conceptual modeling, database systems, formal ontology and computational linguistics. However, despite their fundamental importance, these notions are still poorly represented in modern programming languages.

More specifically, while modern object-oriented programming languages provide first-class support for objects, they lack any notion of relationships and roles. In such languages, objects are forced to implement relationships and roles using a variety of ad-hoc mechanisms (e.g., pointers and hash-tables) leading to a disconnect between designs and models and their implementation. This leads to numerous problems across the software engineering life cycle. At the implementation level, the reliability and maintainability of software are particularly affected. This is because a single role or relationship may be represented by several code fragments that are scattered widely throughout the source code of an application. In current mainstream languages, programmers are also burdened with having to manage consistency properties on relationships explicitly (e.g., the fact that a relationship is one-to-many), and by the lack of linguistic support for commonly used operations on relationships such as querying and join operations.

To address these issues, a growing number of researchers in the software community are incorporating first-class support for relationships, roles, and querying into modern programming languages. There are several different reasons why researchers are motivated to do this:

- Researchers in programming language design wish to bridge the gap between implementation and design. As discussed above, this is because the lack of relationships, roles, and querying at the implementation level causes a disconnect from the design.
- Researchers in program analysis are interested in raising the level of abstraction in programming languages. This is because the absence of language support for relationships, roles, and querying leads to an increased use of pointers. In turn, this complicates many tasks in program analysis as precise pointer alias information becomes necessary.
- Researchers in databases are keen to bridge the "impedance mismatch" between programs and databases. Many current languages rely on the use of string values for accessing and querying data bases, which is error-prone, and can lead to security vulnerabilities.
- In the absence of language support for querying, many commonly used operations need to be expressed using nested loops and complex conditional statements. This makes code harder to understand, optimize, and parallelize.
- Researchers in modelling languages and knowledge representation want to clarify whether roles and relationships are dual notions, or whether one can be substituted for by the other. This has an impact on programming language design, in that it helps keeping the number of new constructs as small as possible.

The purpose of this workshop is to bring together leading researchers working on relationship-, role- and query-based systems and related areas for a week-long meeting. It is our aim to have presentations from each community on a balanced set of topics that is designed to be accessible and relevant to all involved. In this way, we aim to cross-fertilise important ideas from these communities, and push forward the state-of-the-art in their respective areas.

TALK: Beyond the Steimann Factorization

Uwe Assmann (TU Dresden, DE)

In contrast to natural types, role types are founded and semantically non-rigid, as classified by Guarino.

Based on this distinction, Steimann has suggested a factorization of an object's type. Essentially, a type is factored into a tuple of a natural and several role components, so that polymorphism can be expressed as navigation in the product lattice of the tuple.

This talk presents Steimann's work from this point of view and discusses an extension of this factorization by the notion of facets, which are semantically rigid, but non-founded. Facets fall in one spot of the "type matrix" of Steimann/Guarino, closing the gap between natural and role types to a considerable extent. Phases close another spot, because they are non-rigid and non-founded.

Keywords: Facet classification, role types, phase types, Steimann factorization, variability

TALK: Verification of Relationship-Based Programs

Stephanie Balzer (ETH Zurich, CH)

Programming languages supporting explicit relationships raise the level of abstraction available at the programming language level. The availability of appropriate abstractions at the programming language level raises also the hope that the verification of relationship-based programs becomes simpler. In this talk, we present a verification technique developed in the context of our relationship-based programming language Rumer. The verification technique is a visible states technique and leverages the particular abstractions available in Rumer and their induced program modularization. We discuss the applicability of the technique on an example.

Keywords: Relationship-based programming language, invariants, verification

TALK: The Important Role of Roles and Relationships in Static (Typestate) Analysis

Eric Bodden (TU Darmstadt, DE)

We discuss the important role that roles and object collaborations play in the area of static analysis, especially static typestate verification. In typestate analysis, one can distinguish properties that reason about single objects and properties that reason about combinations of objects. Many interesting properties are of the latter kind. Determining a joint typestate of multiple related objects is challenging because the variables that point to these objects may be aliased and may be spread throughout the program. But typestate properties are often the consequences of roles that objects play in a particular collaboration. We argue that if programming languages properly supported the explicit declaration of collaborations and roles, this could aid static analysis.

TALK: Roles in OO programming and beyond

Guido Boella (University of Torino, IT)

In this talk the model of roles used to develop the OO language powerJava will be illustrated also with reference to its applications to web applications, agent programming languages and ontologies.

Keywords: Roles, object oriented programming, agent oriented programming

See also:

TALK: Unifying Remote Data and Services with Batches

William R. Cook (University of Texas - Austin, US)

Most large-scale applications integrate remote services and/or transactional databases. Yet building software that efficiently invokes distributed service and accesses relational databases is still quite difficult. Existing approaches to these problems are based on the Remote Procedure Call (RPC) and Object-Relational Mapping (ORM). RPCs have been generalized to distributed object systems with remote proxies, a kind of remote object reference. ORM tools generally support a form of query sub-language for efficient object selection. The last 20 years have produced a long litany of technologies based on these concepts, including ODBC, CORBA, DCE, DCOM, RMI, DAO, OLEDB, SQLJ, JDBC, EJB, JDO, Hibernate, XML-RPC, Web Services and LINQ. Even with these technologies, complex design patterns for service facades and/or bulk data transfers must be followed to optimize communication between client and server or client and database, leading to programs that are difficult to modify and maintain. While significant progress has been made, there is no widely accepted solution or even agreement about what the solution should look like. In this talk I present a new unified approach to invocation of distributed services and data access. The solution involves a novel control flow construct that partitions a program block into remote and local computations, while efficiently managing the communication between them. The solution does not require proxies or an embedded query language. Although the result itself is elegant and useful, what is more significant is the realization that the original problems cannot be solved using existing programming language constructs and libraries.

Keywords: Database, Query, Remote Procedure Call

TALK: Session types for access and information flow control

Mariangiola Dezani (University of Torino, IT)

We consider a calculus for multiparty sessions with delegation, enriched with security levels for session participants and data. We propose a type system that guarantees both session safety and a form of access control. Moreover, this type system ensures secure information flow, including controlled forms of declassification.

In particular, the type system prevents leaks that could result from an unrestricted use of the control constructs of the calculus, such as session opening, selection, branching and delegation. We illustrate the use of our type system with a number of examples, which reveal an interesting interplay between the constraints used in classical security type systems and those used in session types to ensure properties like communication safety and session fidelity.

Keywords: Concurrency, communication, session types, secure information flow

TALK: Synergy among Features in Object Teams

Stephan Herrmann (Berlin)

Object Teams provides a playedBy relationship that supports all 15 criteria for role playing as collected by Steimann(2000).

However, the approach draws even more strength from how it integrates roles with existing object-oriented concepts and mechanisms.

Specifically, combinations with class nesting, Java generics and with inheritance will be discussed. In Object Teams each of these concepts has been enhanced compared to what pure Java supports. All these enhancements are specifically geared at creating synergy when used together with and applied to roles.

Thanks to this synergy roles play a central role for a wide range of designs far beyond the textbook examples of Persons and Employees.

TALK: EASY Meta-Programming with Rascal

Paul Klint (CWI - Amsterdam, NL)

Rascal (see www.rascal-mpl.org) is a new meta-programming language that integrates syntax analysis, term rewriting and relational calculus. It supports the Extract-Analyze-SYnthesize paradigm that is suitable for many analysis, modelling and codegeneration tasks. We give an overview of the language and its main applications.

Keywords: Meta-Programming, Software Analysis, Software Transformation, Domain-Specific Languages

TALK: Unrestricted pointers considered harmful

Alan Mycroft (Cambridge University, GB)

Depending on which community we are in, we probably learn about ownership types and/or substructural types (including linear types and separation logic), we may learn about region-based memory allocation. But it seems we first have to pick a community, and then read the material. I also found that there is little tutorial material around so was reduced to reading papers which refer to previous papers. Isn't it time for a textbook?

TUTORIAL: Roles and Relationships

James Noble (Victoria University of Wellington, NZ)

Relationships are well-known as a weak point in object-oriented modeling. In a UML diagram, a relationship or association can be drawn as a line connected two (or more) classes — but implementing these relationships in OO programing languages s is rather more difficult. In this talk, I'll give a survey of approaches to designing relationships (including C# 3.0s LINQ), suggest a set of goals for relationship support, and present some potential directions for future work.

(A version of this talk was an invited keynote at the ECOOP 2007 Workshop on Roles and Relationships in Object-Oriented Programming, Multiagent Systems, and Ontologies)

TUTORIAL: ECMA PCTE OMS

Stephan Herrmann (Berlin)

Presenting the Object Management System (OMS) of the Portable Common Tool Environment (PCTE) as standardized by the European Computer Manufacturers Association (ECMA). While striving to provide an object oriented persistent storage (specialized for Software Engineering Environments) it exhibits a great symmetry between objects and links, both being first class entities.

TALK: Memory Management of Weak Exogenous Associations

Kasper Osterbye (IT University of Copenhagen, DK)

Sometimes one needs to create an association, such as OwnerOf, between existing objects, such as a Person and Car, that do not themselves have fields that allow them to refer to each other. Such an exogenous associa-tion creates a memory management problem: it should not keep a Person object and an associated Car object live unless at least one of them is live for other reasons. Unfortunately, standard weak hash maps are inadequate in the case of cyclic associations – that are encountered if we have an OwnedBy association as well as an OwnerOf association.

In the first part of this paper we show how exogenous associations can be properly maintained on the .NET platform using hash tables, weak refer-ences, and finalizers. The solution is particular interesting for object rela-tional mapping frameworks.

In the second part of the paper we show how the same problem can be solved using ephemerons (Hayes 1997), or equivalently, Haskell's notion of weak reference (which is not available on the Java or .NET platforms). However, the garbage collection of a chain of ephemerons may take time quadratic in the length of the chain. We therefore propose a symmetric version of ephemerons (or Haskell weak references) and how to imple-ment it in a runtime system, and show that this reduces worst-case gar-bage collection time from quadratic to near-linear.

Keywords: Association, garbage collection, joint week array

TALK: Implementing Relationships in Whiley

David J. Pearce (Victoria University of Wellington, NZ)

Whiley is a simple programming language aimed primarily at safety-critical applications. The language provides first-class support for sets, lists and tuples, and provides compile-time checking of constraints. The language has an imperative outer layer, and a functional inner core. Relationships which are inherently behaviour-oriented should be implemented within the imperative layer; in contrast, those which are purely of a structural nature should be implemented within the functional core.

Keywords: Relationships, Associations, Programming Languages, Verification

TALK: Inference of Object Usage Protocols

Michael Pradel (ETH Zurich, CH)

Objects of certain classes are supposed to interact in certain ways. In doing so, each object plays a particular role in an object collaboration. For example, programmers using a collection and an iterator in Java must follow a method call protocol specifying that hasNext() must be called before next(). Violating such protocols can lead to program errors. Unfortunately, the method call protocols of most classes are not explicit, and therefore, checking whether a program violates them is difficult. We present a dynamic program analysis that infers protocols describing typical method call sequences on a set of collaborating objects. The analysis tracks how existing programs use certain classes and infers typical usage protocols. The inferred protocols can be used for API documentation as well as for checking whether a program violates common usage patterns.

Full Paper:
http://mp.binaervarianz.de/ase2009.pdf

TALK: All you need is LINQ (well, almost all)

Friedrich Steimann (Fernuniversität in Hagen, DE)

It has often been complained that object-oriented programming languages lack relationships as first-class language constructs. However, language-integrated querying of objects suggests that relationships are not needed to access and exploit data structures in a relational manner — pointers and collections, the object-oriented way of representing relationships, can be queried using similar expressions. Based on a simple example, we identify two weaknesses of the pointers-and-collections approach to relationships, and sketch an object-relational programming model that eliminates these weaknesses. It turns out that roles, not relationships, are the key abstractions of object-relational programming.

TALK: A Role-based Approach for Modular Language Engineering

Christian Wende (TU Dresden, DE)

Modularisation can reduce the effort in designing and maintaining language specifications. Existing approaches to language modularisation are typically either focused on language syntax or on language semantics. We propose defining composition rules on the level of the abstract syntax metamodel, making it the central artefact in a language module.

In the talk we discuss how role-based metamodelling provides clean interfaces for such language modules-effectively making them language components and how this supports the aspectual modularisation of language semantics and can be integrated with concrete syntax specifications to build self-contained language components. We introduce the implementation of our approach in the LanGems language composition system and show how it is used to provide a modularised definition of the Object Constraint Language (OCL).

Keywords: Language Composition Role Metamodeling Modularisation

TALK: Delegation Reconstructed

Erik Ernst (Aarhus University)

Delegation among objects is a well-known mechanism, e.g., from the language Self, which is capable of expressing many different concepts such as dynamic roles, state changes, inheritance, etc. This talk explains how a certain idiom in the language gbeta can be used to express a form of delegation without direct language support for this mechanism. Built-in support for delegation will be more concise and elegant, but this version serves to illustrate the basic structure of the delegation concept and possibly refresh our view on what it is.

TALK: Relational Meta Modeling

Tijs van der Storm (CWI - Amsterdam)

We propose relations as a concept to reduce the impedance mismatch between object-oriented models and algebraic data types in the context of model-driven engineering (MDE). Initial experiments show that certain types of model analysis and model transformation can be expressed in a very natural way if OO models are encoded relationally.

TALK: Roles & Ownership

Tobias Wrigstad (University of Uppsala)

Role encapsulation aims at controlling accesses to multiple strands of a single object by way of ownership types. Not giving a precise encapsulation property, it is intended to facilitating programmer reasoning, rather than formal or automatic verification. Some ideas, some observations, some questions.

TALK: The Use of Overloading in Java Programs

Keren Lenz (Technion - Haifa)

Method overloading is a controversial language feature, especially in the context of Object Oriented languages, where its interaction with overriding may lead to confusing semantics. One of the main arguments against overloading is that it can be abused by assigning the same identity to conceptually different methods. This talk describes a study of the actual use of overloading in JAVA. To this end, we developed a taxonomy of classification of the use of overloading, and applied it to a large JAVA corpus comprising more than 100,000 user defined types. We found that more than 14% of the methods in the corpus are overloaded. Using sampling and evaluation by human raters we found that about sixty percent of overloaded methods follow one of the ?non ad hoc use of overloading patterns? and that additional twenty percent can be easily rewritten in this form. The most common pattern is the use of overloading as an emulation of default arguments, a mechanism which does not exist in JAVA.

TALK: Multirole Session Types

Sophia Drossopoulou (Imperial College London)

We develop an object oriented calculus for multirole session types. The concept of role is novel in session types, and allows several participants in one role. Example applications are auctions, PC meetings, etc. We support sessions with more than one role, and support conversations between all participants in one role with all other participants of the corresponding roles. We develop a type system. Further issues are participants joining/leaving a session, time-out and exceptions.

TALK: Queries without Anomalies

Frank Tip (IBM TJ Watson Research Center)

Programming languages are starting to provide first-class language support for querying. For example, the C# programming language contains LINQ, a collection of features that enable programmers to write queries over data structures such as collections. The implementation of these queries may depend on programmer-specified notions of equality. When programmers make mistakes, a number of inconsistent behavior may arise, which we call query anomalies. In this presentation, we presented an alternative design based on relation types, a mechanism for specifying object identity in a declarative manner (previously published at ECOOP'07). The proposed design does not suffer from query anomalies.

TALK: Beauty Contest

Friedrich Steimann (Fernuniversität in Hagen, DE)

Aside from short presentations and tutorials, we will also be running a small competition — a beauty contest, if you like — of actual programming language extensions which introduce relationships, roles, queries, or what have you. The idea is that participants who have developed, or are proposing, a particular language extension can demonstrate their contribution on a standardized example, so that results are better comparable. For this purpose, I have prepared a one-page benchmark problem. In this introductory talk, I will highlight the main issues involved.

DEMO: Rumer

Stephanie Balzer (ETH Zurich, CH)

We have designed the programming language Rumer, which provides explicit support for classes (i.e., entities) as well as relationships. Rumer supports contracts (method pre-/postconditions and invariants) as well as heap querying. We have implemented a prototype compiler for the language which offers run-time contract verification. We present the main features of the language and provide an overview of the compiler implementation.

DEMO: Whiley

David J. Pearce (Victoria University of Wellington, NZ)

Whiley is a simple programming language aimed primarily at safety-critical applications. In this demonstration, I will show a number of small programs written in Whiley and demonstrate their constraints are being checked at compile time. I will also show how the system can convert constraints into runtime checks.