IN PARTNERSHIP WITH:
Centrum Wiskunde & Informatica

Activity Report 2012

Project-Team ATEAMS

Analysis and Transformation based on rEliAble tool coMpositionS

RESEARCH CENTER
Lille - Nord Europe

THEME
Programs, Verification and Proofs

Table of contents

Project-Team ATEAMS

Keywords: Programming Languages, Formal Methods, Domain-Specific Languages, Software Engineering, Meta-modeling

ATEAMS is a joint research team with SEN1 of Centrum Wiskunde & Informatica (CWI) in Amsterdam, The Netherlands. The group is funded equally —as a principle— both by CWI and Inria. We are proud to represent this very close collaboration between our two excellent European research institutes. The research results of ATEAMS/SEN1 appear in yearly reports of both CWI and Inria alike.

ATEAMS/SEN1 has merged with another group from CWI (INS3) in 2011. The results of this part of our team are not included in the current report because they are not related to the research topic of ATEAMS/SEN1.

Creation of the Project-Team: July 01, 2009 .

1. Members

Research Scientists

Paul Klint [Team Leader, CWI Research Fellow, Director Master Software Engineering at Universiteit van Amsterdam]
Jurgen Vinju [Group Leader SEN1, Teacher Universiteit van Amsterdam]
Jan van Eijck [Senior Researcher CWI, Professor Universiteit van Amsterdam]
Tijs van der Storm [Senior Researcher CWI, Teacher Universiteit van Amsterdam]
Mark Hills [Post-doc CWI]
Michael Godfrey [Research Visitor University of Waterloo, Canada]
Sunil Simon [Post-doc CWI]
Vadim Zaytsev [Post-doc CWI]

Engineers

Bert Lisser [Scientific Programmer CWI]
Maarten Dijkema [Technical support CWI]

PhD Students

Atze van der Ploeg [Junior Researcher CWI]
Davy Landman [Junior Researcher CWI]
Paul Griffioen [Junior Researcher CWI]
Anastasia Izmaylova [Junior Researcher CWI]
Jeroen van den Bos [Junior Researcher CWI]
Riemer van Rozen [Researcher Hogeschool van Amsterdam]
Floor Sietsma [Junior Researcher CWI]
Michael Steindorfer [Junior Researcher CWI]
Ali Afroozeh [Junior Researcher CWI]

Administrative Assistant

Susanne van Dam [Secretary CWI]

2. Overall Objectives

2.1. Presentation

Software is still very complex, and it seems to become more complex every year. Over the last decades, computer science has delivered various insights how to organize software better. Via structured programming, modules, objects, components and agents, software systems are these days more and more evolving into "systems of systems" that provide services to each other. Each system is large, uses incompatible — new, outdated or non-standard — technology and above all, exhibits failures.

It is becoming more and more urgent to analyze the properties of these complicated, heterogeneous and very large software systems and to refactor and transform them to make them simpler and to keep them up-to-date. With the plethora of different languages and technology platforms it is becoming very difficult and very expensive to construct tools to achieve this.

The main challenge of ATEAMS is to address this combination of a need for all kinds of novel analysis and transformation tools and the existence of the diversity of programming environments. We do this by investing in a virtual laboratory called “Rascal”. It is a domain specific programming language for source code analysis, transformation and generation. Rascal is programming language parametric, such that it can be used to analyze, transform or generated source code in any language. By combining concepts from both program analysis and transformation into this language we can efficiently experiment with all kinds of tools and algorithms.

We now focus on three sub-problems. Firstly, we study software analysis: to extract information from existing software systems and to analyze it. The extracted information is vital to construct sound abstract models that can be used in further analysis. We apply these extraction techniques now to analyze (large bodies of) source code: finding bugs and finding the causes of software complexity.

Secondly, we study refactoring: to semi-automatically improve the quality of a software system without changing its behavior. Refactoring tools are a combination of analysis and transformations. Implementations of refactoring tools are complex and often broken. We study better ways of designing refactorings and we study ways to enable new (more advanced and useful) refactorings. We apply these refactorings now to isolate design choices in large software systems and compare systems that are equal except a single design choice.

Finally, we study code generation from domain specific languages (DSLs). Here we also find a combination of analysis and transformation. Designing, implementing and, very importantly, maintaining DSLs is costly. We focus on application areas such as Computational Auditing and Digital Forensics to experiment with this subject. In Computational Auditing we are focusing on type system extensions and in Digital Forensics on optimization and specialization.

2.2. Highlights of the Year

Paul Klint was awarded the CWI Fellowship, for lifetime contributions to science and CWI in particular. This distinction is given to prominent researchers at Centrum Wiskunde & Informatica (CWI) in Amsterdam for their contribution to CWI’s research and administration.

Floor Sietsma defended her PhD on December 13, 2012. This makes her the *youngest PhD in Dutch academic history*, at 20 years old. Remarkably, Floor Sietsma has still two years of research time to go, for her thesis preparation took her about half of the allotted four years. NWO granted her a personalized grant on account of her unusual talents. Sietsma will stay at CWI and use the rest of her research grant to expand her research on the formal analysis of communication, exploring connections with data stream analysis, cryptography and agent technology in artificial intelligence.

3. Scientific Foundations

3.1. Research method

We are inspired by formal methods and logic to construct new tools for software analysis, transformation and generation. We try and proof the correctness of new algorithms using any means necessary.

Nevertheless we mainly focus on the study of existing (large) software artifacts to validate the effectiveness of new tools. We apply the scientific method. To (in)validate our hypothesis we often use detailed manual source code analysis, or we use software metrics, and we have started to use more human subjects (programmers).

Note that we maintain ties with the CWI spinoff “Software Improvement Group” which services most of the Dutch software industry and government and many European companies as well. This provides access to software systems and information about software systems that is valuable in our research.

3.2. Software analysis

This research focuses on source code; to analyze it and to transform it. Each analysis or transformation begins with fact extraction. After the we may analyze specific software systems or large bodies of software systems. Our goal is to improve software systems by learning to understand the causes of complexity.

The mother and father of fact extraction techniques are probably Lex, a scanner generator, and AWK, a language intended for fact extraction from textual records and report generation. Lex is intended to read a file character-by-character and produce output when certain regular expressions (for identifiers, floating point constants, keywords) are recognized. AWK reads its input line-by-line and regular expression matches are applied to each line to extract facts. User-defined actions (in particular print statements) can be associated with each successful match. This approach based on regular expressions is in wide use for solving many problems such as data collection, data mining, fact extraction, consistency checking, and system administration. This same approach is used in languages like Perl, Python, and Ruby. Murphy and Notkin have specialized the AWK-approach for the domain of fact extraction from source code. The key idea is to extend the expressivity of regular expressions by adding context information, in such a way that, for instance, the begin and end of a procedure declaration can be recognized. This approach has, for instance, been used for call graph extraction but becomes cumbersome when more complex context information has to be taken into account such as scope information, variable qualification, or nested language constructs. This suggests using grammar-based approaches as will be pursued in the proposed project. Another line of research is the explicit instrumentation of existing compilers with fact extraction capabilities. Examples are: the GNU C compiler GCC, the CPPX C++ compiler, and the Columbus C/C++ analysis framework. The Rigi system provides several fixed fact extractors for a number of languages. The extracted facts are represented as tuples (see below). The CodeSurfer source code analysis tool extracts a standard collection of facts that can be further analyzed with built-in tools or user-defined programs written in Scheme. In all these cases the programming language as well as the set of extracted facts are fixed thus limiting the range of problems that can be solved.

The approach we want to explore is the use of syntax-related program patterns for fact extraction. An early proposal for such a pattern-based approach is described in: a fixed base language (either C or PL/1 variant) is extended with pattern matching primitives. In our own previous work on RScript we have already proposed a query algebra to express direct queries on the syntax tree. It also allows the querying of information that is attached to the syntax tree via annotations. A unifying view is to consider the syntax tree itself as “facts” and to represent it as a relation. This idea is already quite old. For instance, Linton proposes to represent all syntactic as well as semantic aspects of a program as relations and to use SQL to query them. Due to the lack of expressiveness of SQL (notably the lack of transitive closures) and the performance problems encountered, this approach has not seen wider use.

Another approach is proposed by de Moor and colleagues and uses path expressions on the syntax tree to extract program facts and formulate queries on them. This approach builds on the work of Paige and attempts to solve a classic problem: how to incrementally update extracted program facts (relations) after the application of a program transformation.

Parsing is a fundamental tool for fact extraction for source code. Our group has longstanding contributions in the field of Generalized LR parsing and Scannerless parsing. Such generalized parsing techniques enable generation of parsers for a wide range of real (legacy) programming languages, which is highly relevant for experimental research and validation.

3.2.1. Goals

The main goal is to replace labour-intensive manual programming of fact extractors by automatic generation from annotated grammars or other concise and formal notation. There is a wide open scientific challenge here: to create a uniform and generic framework for fact extraction that is superior to current more ad-hoc approaches. We expect to develop new ideas and techniques for generic (language-parametric) fact extraction from source code and other software artifacts.

Given the advances made in fact extraction we are starting to apply our techniques to observe source code and analyze it in detail.

3.3. Relational paradigm

For any source code analysis or transformation, after fact extraction comes elaboration, aggregation or other further analyses of these facts. For fact analysis, we base our entire research on the simple formal concept of a "relation".

There are at least three lines of research that have explored the use of relations. First, in SQL, n-ary relations are used as basic data type and queries can be formulated to operate on them. SQL is widely used in database applications and a vast literature on query optimization is available. There are, however, some problems with SQL in the applications we envisage: (a) Representing facts about programs requires storing program fragments (e.g., tree-structured data) and that is not easy given the limited built-in datatypes of SQL; (b) SQL does not provide transitive closures, which are essential for computing many forms of derived information; (c) More generally, SQL does not provide fixed-point computations that help to solve sets of equations. Second, in Prolog, Horn clauses can be used to represent relational facts and inference rules for deriving new facts. Although the basic paradigm of Prolog is purely declarative, actual Prolog implementations add imperative features that increase the efficiency of Prolog programs but hide the declarative nature of the language. Extensions of Prolog with recursion have resulted in Datalog in many variations [AHV95]. In F(p)–L a Prolog database and a special-purpose language are used to represent and query program facts.

Third, in SETL, the basic data type was the set. Since relations can easily be represented as sets of tuples, relation-based computations can, in principle, be expressed in SETL. However, SETL as a language was very complicated and has not survived. However, work on programming with sets, bags and lists has continued well into the 90's and has found a renewed interested with the revival of Lisp dialects in 2008 and 2009.

We have already mentioned the relational program representation by Linton. In Rigi, a tuple format (RSF) is introduced to represent untyped relations and a language (RCL) to manipulate them. Relational algebra is used in GROK, Crocopat and Relation Partition Algebra (RPA) to represent basic facts about software systems and to query them. In GUPRO graphs are used to represent programs and to query them. Relations have also been proposed for software manufacture, software knowledge management, and program slicing. Sometimes, set constraints are used for program analysis and type inference. More recently, we have carried out promising experiments in which the relational approach is applied to problems in software analysis and feature analysis. Typed relations can be used to decouple extraction, analysis and visualization of source code artifacts. These experiments confirm the relevance and viability of the relational approach to software analysis, and also indicate a certain urgency of the research direction of this team.

3.3.1. Goals

- New ideas and techniques for the efficient implementation of a relation-based specification formalism.
- Design and prototype implementation of a relation-based specification language that supports the use of extracted facts (Rascal).
- We target at uniform reformulations of existing techniques and algorithms for software analysis as well as the development of new techniques using the relational paradigm.
- We apply the above in the reformulation of refactorings for Java and domain specific languages.

3.4. Refactoring and Transformation

The final goal, to be able to safely refactor or transform source code can be realized in strong collaboration with extraction and analysis.

Software refactoring is usually understood as changing software with the purpose of increasing its readability and maintainability rather than changing its external behavior. Refactoring is an essential tool in all agile software engineering methodologies. Refactoring is usually supported by an interactive refactoring tool and consists of the following steps:

- Select a code fragment to refactor.
- Select a refactoring to apply to it.
- Optionally, provide extra parameter needed by the refactoring (e.g., a new name in a renaming).

The refactoring tool will now test whether the preconditions for the refactoring are satisfied. Note that this requires fact extraction from the source code. If this fails the user is informed.The refactoring tool shows the effects of the refactoring before effectuating them. This gives the user the opportunity to disable the refactoring in specific cases.The refactoring tool applies the refactoring for all enabled cases. Note that this implies a transformation of the source code. Some refactorings can be applied to any programming language (e.g., rename) and others are language specific (e.g., Pull Up Method). At http://www.refactoring.com an extensive list of refactorings can be found.

There is hardly any general and pragmatic theory for refactoring, since each refactoring requires different static analysis techniques to be able to check the preconditions. Full blown semantic specification of programming languages have turned out to be infeasible, let alone easily adaptable to small changes in language semantics. On the other hand, each refactoring is an instance of the extract, analyze and transform paradigm. Software transformation regards more general changes such as adding functionality and improving non-functional properties like performance and reliability. It also includes transformation from/to the same language (source-to-source translation) and transformation between different languages (conversion, translation). The underlying techniques for refactoring and transformation are mostly the same. We base our source code transformation techniques on the classical concept of term rewriting, or aspects thereof. It offers simple but powerful pattern matching and pattern construction features (list matching, AC Matching), and type-safe heterogenous data-structure traversal methods that are certainly applicable for source code transformation.

3.4.1. Goals

Our goal is to integrate the techniques from program transformation completely with relational queries. Refactoring and transformation form the Achilles Heel of any effort to change and improve software. Our innovation is in the strict language-parametric approach that may yield a library of generic analyses and transformations that can be reused across a wide range of programming and application languages. The challenge is to make this approach scale to large bodies of source code and rapid response times for precondition checking.

3.5. The Rascal Meta-programming language

The Rascal Domain Specific Language for Source code analysis and Transformation is developed by ATeams. It is a language specifically designed for any kind of meta programming.

Meta programming is a large and diverse area both conceptually and technologically. There are plentiful libraries, tools and languages available but integrated facilities that combine both source code analysis and source code transformation are scarce. Both domains depend on a wide range of concepts such as grammars and parsing, abstract syntax trees, pattern matching, generalized tree traversal, constraint solving, type inference, high fidelity transformations, slicing, abstract interpretation, model checking, and abstract state machines. Examples of tools that implement some of these concepts are ANTLR, ASF+SDF, CodeSurfer, Crocopat, DMS, Grok, Stratego, TOM and TXL. These tools either specialize in analysis or in transformation, but not in both. As a result, combinations of analysis and transformation tools are used to get the job done. For instance, ASF+SDF relies on RScript for querying and TXL interfaces with databases or query tools. In other approaches, analysis and transformation are implemented from scratch, as done in the Eclipse JDT. The TOM tool adds transformation primitives to Java, such that libraries for analysis can be used directly. In either approach, the job of integrating analysis with transformation has to be done over and over again for each application and this requires a significant investment.

We propose a more radical solution by completely merging the set of concepts for analysis and transformation of source code into a single language called Rascal. This language covers the range of applications from pure analyses to pure transformations and everything in between. Our contribution does not consist of new concepts or language features *per se*, but rather the careful collaboration, integration and cross-fertilization of existing concepts and language features.

3.5.1. Goals

The goals of Rascal are: (a) to remove the cognitive and computational overhead of integrating analysis and transformation tools, (b) to provide a safe and interactive environment for constructing and experimenting with large and complicated source code analyses and transformations such as, for instance, needed for refactorings, and (c) to be easily understandable by a large group of computer programming experts. Rascal is not limited to one particular object programming language, but is generically applicable. Reusable, language specific, functionality is realized as libraries.

4. Software

4.1. Derric

Participants: Tijs van der Storm, Jeroen van den Bos [correspondent].

Characterization: A-2-up3, SO-4, SM-2-up3, EM-3, SDL-3-up4, OC-DA-3-CD-3-MS-3-TPM-3.

WWW: http://www.derric-lang.org

Objective: Encapsulate all the variability in the construction of so-called "carving" algorithms, then generate the fastest and most accurate implementations. Carving algorithms recover information that has been deleted or otherwise scrambled on digital media such as hard-disks, usb sticks and mobile phones.

Users: Digital forensic investigation specialists

Impact: Derric has the potential of revolutionizing the carving area. It does in 1500 lines of code what other systems need tens of thousands of lines for with the same accuracy. Derric will be an enabler for faster, more specialized and more successful location of important evidence material.

Competition: Derric competes in a small market of specialized open-source and commercial carving tools.

Engineering: Derric is a Rascal program of 1.5 kloc designed by two persons.

Publications: [8][32], [14]

In 2012 Derric was validated on a large body of image files taken from wikipedia, and a new approach to software optimization via model transformation was developed for optimizing Derric code. We released Derric 1.0 in 2012.

4.2. Basic Voting Theory

Participants: Jan van Eijck [correspondent], Floor Sietsma.

Characterization: A1, SO-3, SM-1, EM-1, SDL-2, OC-DA-3-CD-3-MS-3-TPM-3.

WWW: http://homepages.cwi.nl/~jve/software

Objective: Demonstrate the basic concepts of voting theory.

Users: Students and researchers interested in voting theory.

Impact: This is a demonstrator and a tool for teaching.

Competition: None.

Engineering: Haskell program.

4.3. Rascal

Participants: Paul Klint, Jurgen Vinju [correspondent], Tijs van der Storm, Jeroen van den Bos, Mark Hills, Bert Lisser, Atze van der Ploeg, Vadim Zaytsev, Anastasia Izmaylova, Michael Steindorfer, Ali Afroozeh.

Characterization: A5, SO-4, SM-4, EM-4, SDL-4-up5, OC-DA-3-CD-3-MS-3-TPM-3.

WWW: http://www.rascal-mpl.org

Objective: Provide a completely integrated programming language parametric meta programming language for the construction of any kind of meta program for any kind of programming language: analysis, transformation, generation, visualization.

Users: Researchers in model driven engineering, programming languages, software engineering, software analysis, as well as practitioners that need specialized tools.

Impact: Rascal is making the mechanics of meta programming into a non-issue. We can now focus on the interesting details of the particular fact extraction, model, source analysis, domain analysis as opposed to being distracted by the engineering details. Simple things are easy in Rascal and complex things are manageable, due to the integration, the general type system and high-level programming features.

Competition: There is a plethora of meta programming toolboxes and frameworks available, ranging from plain parser generators to fully integrated environments. Rascal is distinguished because it is a programming language rather than a specification formalism and because it completely integrates different technical domains (syntax definition, term rewriting, relational calculus). For simple tools, Rascal competes with scripting languages and for complex tools it competes context-free general parser generators, with query engines based on relational calculus and with term rewriting and strategic programming languages.

Engineering: Rascal is about 100 kLOC of Java code, designed by a core team of three and with a team of around 8 phd students and post-docs contributing to its design, implementation and maintenance. The goal is to work towards more bootstrapping and less Java code as the project continues.

Publications: [23], [22], [11], [21], [22]

4.3.1. Novelties

- Statically typed access to external data-sources [21]. This includes access to CVS files, spreadheets, databases, etc.
- Significant improvements to online documentation and inter-active tutor environment.
- Full transparent support for Unicode codepoints.
- Added language-supported quickcheck-style random testing facility (by Wietse Venema, intern), including bridge to JUnit testing framework and IDE support.
- Revived access libraries to CVS, SVN and Git VCSs.
- Added support for JSON export and import, towards Rascal webservices.
- Totally re-implemented and extended debugging interface.
- Priority and associativity mechanism for context-free grammars was completed, such that it can not be used to accidentally remove sentences from a language anymore.
- Reimplementation of the except disambiguation filter with much higher efficiency.
- Improved module import times.
- Reimplemented URI encoding/decoding mechanism for correctness and portability.
- Added semi-automated exam generation and grading feature to the Rascal tutor environment.
- Experimented with strategies for error recovery in context-free general top-down parser.
- Added MissGrant and SuperAwesomeFighter language workbench demonstrations.
- Structured re-design of menus and menu options in the IDE

- Added bindings to Apache statistics libraries
- Created Rascalopedia, a glossary of concepts and terms that are relevant for metaprogrammers. The descriptions are aiming at under-graduate students.
- Two previously designed programmable transformation languages for grammars in a broad sense: the unidirectional XBGF and the bidirectional ΞBGF — have been reimplemented as libraries in Rascal.
- Improved general stability and efficiency.

4.4. IDE Meta-tooling Platform

Participants: Jurgen Vinju [correspondent], Michael Steindorfer.

IMP, the IDE meta tooling platform is an Eclipse plugin developed mainly by the team of Robert M. Fuhrer at IBM TJ Watson Research institute. It is both an abstract layer for Eclipse, allowing rapid development of Eclipse based IDEs for programming languages, and a collection of meta programming tools for generating source code analysis and transformation tools.

Characterization: A5, SO-3, SM4-up5, EM-4, SDL-5, DA-2-CD-2-MS-2-TPM-2

WWW: http://www.eclipse.org/imp

Objective: The IDE Meta Tooling Platform (IMP) provides a high-level abstraction over the Eclipse API such that programmers can extend Eclipse with new programming languages or domain specific languages in a few simple steps. IMP also provides a number of standard meta tools such as a parser generator and a domain specific language for formal specifications of configuration parameters.

Users: Designers and implementers of IDEs for programming languages and domain specific languages. Also, designers and implementers of meta programming tools.

Impact: IMP is popular among meta programmers especially for it provides the right level of abstraction.

Competition: IMP competes with other Eclipse plugins for meta programming (such as Model Driven Engineering tools), but its API is more general and more flexible. IMP is a programmers framework rather than a set of generators.

Engineering: IMP is a long-lived project of many contributors, which is managed as an Eclipse incubation project at `eclipse.org`. Currently we are moving the project to Github to explore more different ways of collaboration.

Publications: [3]

4.5. Ensō

Participant: Tijs van der Storm [correspondent].

Characterization: A5, SO-4, SM-3-up-4, EM-2-up-4, SDL-4, OC-DA-4-CD-4-MS-4-TPM-4

WWW: http://www.enso-lang.org

Objective: Together with Prof. Dr. William R. Cook of the University of Texas at Austin, and Alex Aloh, Tijs van der Storm has been designing and implementing a new programming system, called Ensō. Ensō is theoretically sound and practical reformulation of model-based development. It is based on model-interpretation as opposed to model transformation and code generation. Currently, the system already supports models for schemas (data models), web applications, context-free grammars, diagram editors and security.

Users: All programmers.

Impact: Ensō has the potential to revolutionize the activity of programming. By looking at model driven engineering from a completely fresh perspective, with as key ingredients interpreters and partial evaluation, it may make higher level (domain level) program construction and maintenance as effective as normal programming.

Competition: Ensō competes as a programming paradigm with model driven engineering tools and generic programming and languages that provide syntax macros and language extensions.

Engineering: Ensō is less than 7000 lines of (bootstrapped) Ruby code.

4.6. Turing language

Participants: Davy Landman [correspondent], Tijs van der Storm, Jeroen van den Bos, Vadim Zaytsev, Paul Klint.

Characterization: A3, SO-2, SM-1, EM-1, SDL-5, DA-3-CD-3-MS-3-TPM-3

WWW: http://www.legoturingmachine.org

Objective: This software is used to program the Lego Turing Machine which was built as a piece for the Turing Centennial exposition at CWI. The software features a full fledged Eclipse based IDE for a small programming language which is compiled to Turing machine instructions that run on a Lego machine.

Users: People interested in learning about computation and programming languages.

Impact: the Lego Turing Machine and it's software have reached more than 3 million people via the internet (slashdot, vimeo, youtube) and all Dutch national newspapers.

Competition: none.

Engineering: the hardware is Lego and the software is fully generated from Rascal syntax definitions and IDE construction functions.

4.7. Lua AiR

Participant: Riemer van Rozen [correspondent].

Characterization: A2-up, SO-4, SM-2-up, EM-3-up, SDL-4, DA-3-CD-3-MS-3-TPM-3

WWW: https://github.com/cwi-swat/Lua_AiR

Objective: This system provides IDE integrated support for static analysis of Lua code. Lua is a scripting language used in game development.

Users: Game programmers and game designers

Impact: Lua AiR is currently a research prototype designed to experiment with and validate the static analysis of a highly dynamic scripting language.

Competition: none.

Engineering: LuA AiR is fully implemented in Rascal.

5. New Results

5.1. Programming language support for statically type access to external resources

One of the open issues in programming is how to obtain typed access, including its beneficial IDE support, to data sources that have not been modeling with the programming language's data modeling facilities. Rather most data is modeled externally or not modeled at all. Mark Hills, Jurgen Vinju and Paul Klint proposed, designed and validated a programming language design where meta models for external data are imported and/or inferred at compile-time. These models are then used to generate source code to represent these models natively in the idiom of the programming language.

5.2. Statically analyzing PHP code

Tool support in IDEs for PHP code is limited due to the dynamic nature of the language. Mark Hills, Jurgen Vinju and Paul Klint produced a principled yet pragmatic infrastructure for analyzing PHP code nevertheless. The analyses first use crude but effective over-approximations of the PHP semantics to limit the search spaces and improve accuracy, then exploit information from user-manuals, and then use state-of-the-art static analysis techniques in a fixed point abstract interpretation to arrive at accurate results.

5.3. Modular Language Parametric Refactoring Framework

Anastasia Izmaylova with Jurgen Vinju produced a prototype implementation of a framework for specifying refactoring tools based on type constraints that is open to unpredicted language extensions. The problem with the co-evolution of programming language and their supporting refactoring tools is complexity. Often existing refactorings are not retro-fitted with the new language semantics and new opportunities for refactoring tools are not fulfilled. Anastasia designed a solution based on monad transformers that allows the kind of invasive extensibility needed to adapt complex existing implementation of language semantics with additional features that interact in many ways with the existing features.

5.4. Communication Action Emulation

CAE is a novel epistemic model for describing and evaluating the equivalence of communication models by Floor Sietsma and Jan van Eijck.

5.5. Notation-Parametric Grammar Recovery

Vadim Zaytsev generalized the algorithm for recovering context-free grammars from legacy language documentation. This facilitated the recovery of more grammars to be used in the study of grammarware and software language engineering.

5.6. (In)Validating Domain Knowledge Existence in Legacy Source Code

Davy Landman conducted a large experiment in comparing an extensive domain model to the information present in source code of applications that are used in the domain in question. The project management domain was chosen for this. The experiment is still in progress. Big steps were made in setting up the experiment, which includes reporting comprehensively on a large number of design decisions, in a traceable and reproducible manner.

5.7. Ensō

Ensōis a new programming system based on interpretation of domain-specific modeling languages. The system is co-designed and authors by Tijs van der Storm in collaboration with William Cook and Alex Loh. The two foundations of the Ensōsystem are managed data and object grammars. Managed data provides modular strategies for customizing how programming languages represent and provide access to data.

Object grammars form the second foundation: they facilate declarative, compositional, and bidirectional mappings from textual syntax to object graphs. Domain-specific models in Ensōare parsed and rendered using object grammars, and represented, in memory as managed data. Together they combine into a highly flexible and modular platform for model-driven development.

6. Partnerships and Cooperations

6.1. National Initiatives

6.1.1. Master Software Engineering

ATEAMS is the core partner in the Master Software Engineering at Universiteit van Amsterdam. This master is a collaboration between SWAT/ATEAMS, Universiteit van Amsterdam, Vrije Universiteit and Hogeschool van Amsterdam.

6.1.2. Early Quality Assurance in Software Production

The EQUA project is a collaboration among Hogeschool van Amsterdam (main partner) Centrum Wiskunde & Informatica (CWI), Technisch Universiteit Delft, Laboratory for Quality of Software (LaQuSo), Info Support, Software Improvement Group (SIG), and Fontys Hogeschool Eindhoven.

6.1.3. Model-Driven Engineering in Digital Forensics

In this project ATEAMS works with the Dutch National Forensics Institute on next generation carving software for recovering evidence from damaged or erased data storage media.

6.1.4. Next Generation Auditing: Data-assurance as a service

This collaboration between Centrum Wiskunde & Informatic (CWI) PriceWaterhouseCoopers (PWC), Belastingdienst (National Tax Office), and Computational Auditing, is to enable research in the field of computational auditing.

6.2. European Initiatives

6.2.1. FP7 Projects

OSSMETER aims to extend the state-of-the-art in the field of automated analysis and measurement of open-source software (OSS), and develop a platform that will support decision makers in the process of discovering, comparing, assessing and monitoring the health, quality, impact and activity of open-source software. The project started in October 2012. ATEAMS contributes to this project by focusing on software analysis and related areas.

6.3. International Research Visitors

6.3.1. Visits of International Scientists

- Michael W. Godfrey, PhD, Associate professor - David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada. (full year visit)
- Alex Loh - University of Texas, Austin, U.S.A. (three month internship for excellent PhD students)
- William Cook - University of Texas, Austin, U.S.A.
- Erik Meijer - Microsoft Research, Seattle, U.S.A.
- Oege de Moor - Semmle & Oxford University
- Krzystof Czarnecki - University of Waterloo, Canada
- Stéphane Ducasse - Inria Lille, France
- Ralf Lämmel - University of Koblenz-Landau, Germany
- Magne Haveraaen - University of Bergen, Norway
- Anya Helene Bagge, PhD - University of Bergen, Norway
- Vlad Rusu - Inria Lille, France
- Ted Kaminsky - University of Minnesota, U.S.A.
- Anthony Cleve - FUNDP, Namur, Belgium
- Anthony Sloane - Macquarie University, Australia
- Elizabeth Scott - RHUL, London, England
- Peter Mosses - University of Swansea, Wales
- Adrian Johnstone - RHUL, London, England

6.3.1.1. Internships

- Douwe Kasemier
- Arnoud Roo
- Jasper Timmer
- Wietse Venema
- Ashim Shahi

- Jouke Stoel
- Dennis van Leeuwen
- Jeroen Lappenschaar
- Luuk Stevens
- Floris Looijesteijn
- Pieter Brantwijk

6.3.2. Visits to International Teams

- Tijs van der Storm visited University of Texas Austin for two weeks in November.
- Paul Klint visited University of London and University of Swansea
- Paul Klint visited University of Swansea
- Paul Klint visited FUNDP in Namur
- Jurgen Vinju and Tijs van der Storm visited RMOD at Inria Lille
- Jurgen Vinju visited VUB, Brussels, Belgium
- Vadim Zaytsev visited Universität Koblenz-Landau, Germany

7. Dissemination

7.1. Scientific Animation

7.1.1. Teaching

Master : Jurgen Vinju, Paul Klint, Software Evolution, 6 EC, Universiteit van Amsterdam, The Netherlands

Master : Tijs van der Storm, Jurgen Vinju, Software Construction, 6 EC, Universiteit van Amsterdam, The Netherlands

Master : Jan van Eijck, Bert Lisser, Software Testing, 6 EC, Universiteit van Amsterdam, The Netherlands

Master : Paul Klint, EASY Meta-Programming with Rascal, full-day course at Summer School on Language Frameworks, Sinaia, Romania

Master : Paul Klint, Rascal tutorial, one-day course, FUNDP, Namur, Belgium

Master : Tijs van der Storm, one-day course on Rascal, Universidade do Minho (Braga)

Graduate: Tijs van der Storm, one-day course on Rascal, Programming Language class, UT Austin, U.S.A.

7.1.2. Supervision

PhD : Floor Sietsma, Logics of Communication and Knowledge, Universiteit van Amsterdam, 13-12-2012, supervisors Jan van Eijck, Krzysztof Apt

PhD in progress : Paul Griffioen, Next Generation Computational Auditing, started 2011, supervisors Paul Klint, Philip Elsas

PhD in progress : Anastasia Izmaylova, A General Language Parametric Framework for Software Refactoring, started 2011, supervisors Paul Klint, Jurgen Vinju

PhD in progress : Jeroen van den Bos, Digital Forensics Software Engineering, started 2010, supervisors Paul Klint, Tijs van der Storm

PhD in progress : Atze van der Ploeg, Rapid Language Parametric Prototyping of Software Visualization and Exploration, started 2011, supervisor Paul Klint.

PhD in progress : Davy Landman, Recovery and Synthesis of Domain Specific Language Design, started 2011, supervisors Jurgen Vinju, Paul Klint, supervisors Jurgen Vinju, Paul Klintto .

PhD in progress : Riemer van Rozen, Software Engineering Principles for the Gaming Domain, started 2011, supervisor Paul Klint

PhD in progress : Ali Afroozeh, Ambiguity and Disambiguation for Context-free Grammars, started 2012, supervisor Jurgen Vinju

PhD in progress : Ashim Shahi, Principled Quality Assessment of Software, started 2012, supervisor Paul Klint, Jurgen Vinju

PhD in progress : Michael Steindorfer, Scaling meta-programming to data-programming, started 2012, supervisor Paul Klint, Jurgen Vinju

Paul Klint, Jurgen Vinju, Tijs van der Storm and Jan van Eijck together also supervised more than 30 master thesis projects for Universiteit van Amsterdam in 2012.

7.1.3. Juries

- Jurgen Vinju, PhD: Veronica Isabel Uquillas Gomez, Vrije Universiteit Brussel & Université Lille, Belgium & France.
- Paul Klint, PhD: J. Dormans, Universiteit van Amsterdam, The Netherlands.
- Paul Klint, PhD: R. Poss, Universiteit van Amsterdam, The Netherlands.
- Paul Klint, PhD: L. Kwiatkowski, Vrije Universiteit, The Netherlands.
- Jan van Eijck, PhD: Peter Gammie, University of Camberra, Australia.

7.1.4. Invited speakers

- Jurgen Vinju - IFIP WG 2.4, Vadstena, Sweden.
- Tijs van der Storm - IFIP TC2 2.16, London, England.
- Tijs van der Storm - IFIP TC2 2.16, Austin, U.S.A.
- Paul Klint - Invited speaker at PReCISE Conference, Namur, Belgium.
- Paul Klint - Keynote "The Spinoff Game" at Entrepreneurial Research Day, ICT Labs, Paris, France.
- Mark Hills - Invited speaker at International Workshop On Rewriting Logic and its Applications, Talinn, Estonia.

7.1.5. Positions, memberships and editorial work

- Jan van Eijck : Member of the European Network in Computational Logic
- Jan van Eijck : Member of the International Quality Assessment Committee
- Jan van Eijck: Member of the NWO committee “Vrije Competitie” for Computer Science.
- Jan van Eijck: Book editor “Games, Actions, and Social Software”, Springer.
- Jan van Eijck: Program committee IWCS
- Jan van Eijck: Program committee LORI-4
- Jan van Eijck: Reviewer ESSLLI
- Jan van Eijck: Reviewer Journal of Semantics
- Jan van Eijck: Reviewer Journal of Logic and Computation
- Jan van Eijck: Reviewer Fundamenta Informaticae
- Jan van Eijck: Reviewer Synthese
- Jan van Eijck: Reviewer Journal of Philosophical Logic
- Jan van Eijck: Reviewer Journal of Logic, Language and Information
- Jan van Eijck: Cambridge University Press, Australian National University (Canberra)

- Tijs van der Storm: Program co-chair IPA Lentedagen
- Tijs van der Storm: External reviewer OOPSLA
- Tijs van der Storm: External reviewer POPL 2013
- Tijs van der Storm: Reviewer Information and Software Technology
- Tijs van der Storm: Reviewer Software Practice & Experience
- Tijs van der Storm: Science of Computer Programming
- Mark Hills : Reviewing Software Practise & Experience
- Mark Hills : Reviewing IET Software
- Mark Hills : Program committee FMOODS
- Paul Klint : Editor for Science of Computer Programming
- Paul Klint : Editor for Springer Service Science Book Series
- Paul Klint : Visiting Professor University of London, Royal Holloway
- Paul Klint : Treasurer European Association for Programming Languages and Systems (EAPLS)
- Paul Klint : Steering committee member ETAPS
- Paul Klint : Board member Instituut voor Programmatuur en Architectuur (IPA)
- Paul Klint : External advisor PlanComps project (UK)
- Paul Klint : Full Professor at UvA, Software Engineering Chair
- Paul Klint : Director Master Software Engineering, UvA
- Paul Klint : Treasurer European Association for Programming Languages and Systems (EAPLS)
- Paul Klint : Steering committee member ETAPS
- Paul Klint : Board member Instituut voor Programmatuur en Architectuur (IPA)
- Paul Klint : Program committee SLE
- Jurgen Vinju : Head of CWI Research group Software Analysis and Transformation (SEN1)
- Jurgen Vinju : Program committee ICMT
- Jurgen Vinju : Program committee CC
- Jurgen Vinju : Program committee SCAM
- Jurgen Vinju : Steering committee SCAM
- Jurgen Vinju : Steering committee SLE
- Jurgen Vinju : Co-organizer workshop SL(E)BOKSLE
- Jurgen Vinju : Member Working Council (CWI)
- Jurgen Vinju : Workshop selection chair SLE
- Jurgen Vinju : Reviewing Journal of Software Maintenance and Evolution: Research and Practice
- Jurgen Vinju : Reviewing Transactions on Software Engineering (TSE)
- Jurgen Vinju : Reviewer Journal on Empirical Software Engineering (EMSE)
- Jurgen Vinju : Reviewer Science of Computer Programming (SCP)
- Jurgen Vinju : Member meetings between Inria & CWI management, Paris and Amsterdam.
- Vadim Zaytsev: Program committee LDTA
- Vadim Zaytsev: Program committee SCAM
- Vadim Zaytsev: Program committee SQM
- Vadim Zaytsev: Program committee Wikimedia Conference Netherlands
- Vadim Zaytsev: Reviewer Science of Computer Programming (SCP)
- Vadim Zaytsev: Reviewer Journal on Empirical Software Engineering (EMSE)

- Vadim Zaytsev: Reviewer IET Software
- Vadim Zaytsev: Program chair Wikimedia Conference Netherlands (WCN)
- Vadim Zaytsev: Social media chair SoTeSoLa
- Vadim Zaytsev: Hackathon coordinator SoTeSoLa
- Vadim Zaytsev: Colloquium organiser of Programming Environment Meetings (PEM)

7.2. Popularization

In 2012 our team has invested a large amount of time and energy in reaching out, in close collaboration with our support staff:

- The “Understanding Software” symposium with more than 250 participants from industry and academia. This was organized in honor of celibrating the 40th anniversary of Paul Klint at CWI. The symposium was a great success, with internationally renowned speakers and a large and diverse audience. The topic was entirely devoted to the research topics of our team.
- Turing centennial exposition, including Enigma machine and X1, hosted more than 1000 visitors.
- Lego Turing Machine, including IDE, was an online hit with millions of viewers. Articles about the machine and its meaning appeared in all major Dutch journals and we estimate a reach of more than 3 million readers.
- ICT Delta - we presented at the Dutch national symposium ICT Delta where industry meets academia in ICT.
- Floor Sietsma appeared on national television and radio, and in most newspapers to explain about her role as the youngest Phd candidate and her research.
- Paul Klint chaired public discussion on Government Data Policy, IPoort, The Hague, April 10, 2012
- Jurgen Vinju went to different companies to explain about Rascal and building your own software metrics, and software engineering in general (Sogyo, Software Factory)
- Tijs van der Storm, Jurgen Vinju and Vadim Zaytsev presented at “IPA Spring Days” about domain specific languages and related topics to all Dutch PhD students in Computer Science.
- Tijs van der Storm organized a Lisp course for Devnology
- Tijs van der Storm presented at Optiver about Domain-Specific Languages

8. Bibliography

Major publications by the team in recent years

[1] B. BASTEN. *Tracking Down the Origins of Ambiguity in Context-Free Grammars*, in "Seventh International Colloquium on Theoretical Aspects of Computing (ICTAC 2010)", A. CAVALCANTI, D. DEHARBE, M.-C. GAUDEL, J. WOODCOCK (editors), Springer, September 2010, vol. 6255, p. 76-90.

[2] B. BASTEN, J. VINJU. *Faster Ambiguity Detection by Grammar Filtering*, in "Proceedings of the tenth workshop on Language Descriptions Tools and Applications", C. BRABRAND, P.-E. MOREAU (editors), 2010.

[3] P. CHARLES, R. M. FUHRER, S. M. SUTTON JR, E. DUESTERWALD, J. VINJU. *Accelerating the Creation of Customized, Language-Specific IDEs in Eclipse*, in "Proceedings of the 24th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2009", S. ARORA, G. T. LEAVENS (editors), 2009.

[4] JAN VAN. EIJCK, C. UNGER. *Computational Semantics with Functional Programming*, Cambridge University Press, September 2010.

[5] M. HILLS, P. KLINT, J. VINJU. *Meta-Language Support For Type-Safe Access To External Resources*, in "Pre-Proceedings of the 5th International Conference on Software Language Engineering", Dresden, Netherlands, K. CZARNECKI, G. HEDIN (editors), Fakultät Informatik, Technische Universität Dresden, 2012, p. 370 - 389, http://hal.inria.fr/hal-00756878.

[6] P. KLINT, T. VAN DER STORM, J. VINJU. *RASCAL: A Domain Specific Language for Source Code Analysis and Manipulation*, in "Source Code Analysis and Manipulation, IEEE International Workshop on", Los Alamitos, CA, USA, 2009, p. 168-177, http://doi.ieeecomputersociety.org/10.1109/SCAM.2009.28.

[7] P. KLINT, T. VAN DER STORM, J. VINJU. *EASY Meta-programming with Rascal*, in "Generative and Transformational Techniques in Software Engineering III", J. FERNANDES, R. LÄMMEL, J. VISSER, J. SARAIVA (editors), Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2011, vol. 6491, p. 222-289, 10.1007/978-3-642-18023-1_6, http://dx.doi.org/10.1007/978-3-642-18023-1_6.

[8] T. VAN DER STORM, J. VAN DEN BOS. *Bringing Domain-Specific Languages to Digital Forensics*, in "Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, Waikiki, Honolulu , HI, USA, May 21-28, 2011", Honolulu, United States, ACM, 2011, p. 671-680, http://hal.inria.fr/hal-00644687/en.

[9] T. VAN DER STORM, J. W. COOK, A. LOH. *Object Grammars: Compositional & Bidirectional Mapping Between Text and Graphs*, in "Software Language Engineering", Dresden, Germany, K. CZARNECKI, G. HEDIN (editors), September 2012, http://hal.inria.fr/hal-00758627.

Publications of the year

Doctoral Dissertations and Habilitation Theses

[10] F. SIETSMA. *Logics of Communication and Knowledge*, Universiteit van Amsterdam, December 2012, http://hal.inria.fr/tel-00756861.

Articles in International Peer-Reviewed Journals

[11] M. HILLS, P. KLINT, T. VAN DER STORM, J. VINJU. *A One-Stop Shop For Software Evolution Tool Construction*, in "ERCIM News", 2012, n^{o} 88, p. 11 - 12, http://hal.inria.fr/hal-00756876.

[12] V. ZAYTSEV. *Language Evolution, Metasyntactically*, in "Electronic Communications of the European Association of Software Science and Technology", 2012, vol. 49, article-no = 3, p. 1 - 17, http://hal.inria.fr/hal-00756854.

[13] J. VAN EIJCK, J. RUAN, T. SADZIK. *Action Emulation*, in "Synthese", 2012, vol. 185, n^{o} 1, p. 131 - 151, http://hal.inria.fr/hal-00756870.

[14] J. VAN DEN BOS, T. VAN DER STORM. *Domain-Specific Languages For Better Forensic Software*, in "ERCIM News", 2012, vol. 2012, n^{o} 90, http://hal.inria.fr/hal-00756885.

Articles in National Peer-Reviewed Journals

[15] P. KLINT. *Ode Aan Turing*, in "I/O ICT Onderzoek", 2012, vol. 9, n^{o} 1, 19, http://hal.inria.fr/hal-00756857.

Articles in Non Peer-Reviewed Journals

[16] P. KLINT. *De Stille Kracht Van Informatietechnologie, We Kunnen Geen Dag Zonder!*, in "I/O ICT Onderzoek", 2012, vol. 9, n^{o} 2, 19, http://hal.inria.fr/hal-00756856.

[17] P. KLINT. *De Toekomst Van Het Hoorcollege*, in "I/O ICT Onderzoek", 2012, vol. 9, n^{o} 3, 19, http://hal.inria.fr/hal-00756855.

International Conferences with Proceedings

[18] K. APT, S. SIMON. *A Classification Of Weakly Acyclic Games*, in "Proceedings of the Symposium on Algorithmic Game Theory (SAGT, 2012)", Barcelona, Spain, Springer, 2012, vol. 7615, p. 1 - 12, http://hal.inria.fr/hal-00756884.

[19] K. APT, S. SIMON. *Choosing Products In Social Networks*, in "Proceedings of the International Workshop on Internet And Network Economics (WINE, 2012)", Liverpool, United Kingdom, Springer, 2012, vol. 7695, p. 100 - 113, http://hal.inria.fr/hal-00756883.

[20] B. FISCHER, R. LÄMMEL, V. ZAYTSEV. *Comparison Of Context-Free Grammars Based On Parsing Generated Test Data*, in "Post-proceedings of the Fourth International Conference on Software Language Engineering (SLE 2011)", Braga, Portugal, U. ASSMANN, A. SLOANE (editors), Springer, Heidelberg, 2012, p. 324 - 343, http://hal.inria.fr/hal-00756892.

[21] M. HILLS, P. KLINT, J. VINJU. *Meta-Language Support For Type-Safe Access To External Resources*, in "Pre-Proceedings of the 5th International Conference on Software Language Engineering", Dresden, Netherlands, K. CZARNECKI, G. HEDIN (editors), Fakultät Informatik, Technische Universität Dresden, 2012, p. 370 - 389, http://hal.inria.fr/hal-00756878.

[22] M. HILLS, P. KLINT, J. VINJU. *Program Analysis Scenarios In Rascal*, in "Proceedings of the International Workshop on Rewriting Logic and its Applications (WRLA, 2012)", Talinn, Estonia, F. DURÁN (editor), Springer, 2012, vol. 7571, p. 10 - 30, An invited paper for WRLA 2012, describing our work on program analysis and comparing our approach to approaches based on rewriting logic semantics., http://hal.inria.fr/hal-00756880.

[23] M. HILLS, P. KLINT, J. VINJU. *Scripting A Refactoring With Rascal And Eclipse*, in "Proceedings of the 5th Workshop on Refactoring Tools 2012", Rapperswil, Switzerland, P. SOMMERLAD (editor), ACM, 2012, p. 40 - 49, http://hal.inria.fr/hal-00756879.

[24] P. KLINT, L. ROOSENDAAL, R. VAN ROZEN. *Game Developers Need Lua AiR Static Analysis Of Lua Using Interface Models*, in "Proceedings of International Conference on Entertainment Computing 2012", Bremen, Germany, R. MALAKA, R. MASUCH (editors), Springer, 2012, p. 530 - 535, http://hal.inria.fr/hal-00758607.

[25] A. LOH, T. VAN DER STORM, J. W. COOK. *Managed Data: Modular Strategies For Data Abstraction*, in "Proceedings of the ACM international symposium on New ideas, new paradigms, and reflections on programming and software 2012", Tucson, United States, ACM, 2012, p. 179 - 194, http://hal.inria.fr/hal-00756886.

[26] V. ZAYTSEV. *BNF WAS HERE: What Have We Done About The Unnecessary Diversity Of Notation For Syntactic Definitions*, in "Programming Languages Track, Volume II of the Proceedings of the 27th ACM Symposium on Applied Computing (SAC 2012)", Riva del Garda, Italy, M. MERNIK, B. BRYANT (editors), ACM, 2012, p. 1910 - 1915, http://hal.inria.fr/hal-00756890.

[27] V. ZAYTSEV. *Language Evolution, Metasyntactically*, in "Pre-proceedings of the First International Workshop on Bidirectional Transformation (BX 2012)", Talinn, Estonia, F. HERMANN, J. VOIGTLÄNDER (editors), Institute of Cybernetics at Tallinn University of Technology, 2012, http://hal.inria.fr/hal-00756852.

[28] V. ZAYTSEV. *Notation-Parametric Grammar Recovery*, in "Pre-proceedings of the 12th International Workshop on Language Descriptions, Tools, and Applications (LDTA 2012)", Talinn, Estonia, A. SLOANE, S. ANDOVA (editors), Institute of Cybernetics at Tallinn University of Technology, 2012, p. 105 - 118, http://hal.inria.fr/hal-00756889.

[29] H. VAN DITMARSCH, J. VAN EIJCK, F. SIETSMA, S. SIMON. *Modelling Cryptographic Keys In Dynamic Epistemic Logic With DEMO*, in "Highlights on Practical Applications of Agents and Multi-Agent Systems", Salamanca, Spain, J. B. PEREZ (editor), Springer, 2012, vol. 156, p. 155 - 162, http://hal.inria.fr/hal-00756869.

[30] J. VAN EIJCK, F. SIETSMA. *Action Emulation Between Canonical Models*, in "Proceedings of Conference on Logic and the Foundations of Game and Decision Theory 2012", Sevilla, Spain, G. BONANNO, H. VAN DITMARSCH, W. VAN DER HOEK (editors), University of Sevilla, 2012, http://hal.inria.fr/hal-00756863.

[31] J. VAN EIJCK, F. SIETSMA. *Questions About Voting Rules, With Some Answers*, in "W16 Workshop on Logical Aspects of Multi-Agent Systems", Valencia, Spain, V. GORANKO, W. JAMROGA (editors), internet, 2012, http://hal.inria.fr/hal-00756866.

[32] J. VAN DEN BOS, T. VAN DER STORM. *Domain-Specific Optimization In Digital Forensics*, in "Proceedings of the International Conference on Model Transformation (ICMT, 2012)", Prague, Czech Republic, Z. HU, J. DE LARA (editors), Springer, 2012, vol. 7307, p. 121 - 136, http://hal.inria.fr/hal-00756891.

[33] T. VAN DER STORM, J. W. COOK, A. LOH. *Object Grammars: Compositional & Bidirectional Mapping Between Text and Graphs*, in "Software Language Engineering", Dresden, Germany, K. CZARNECKI, G. HEDIN (editors), September 2012, http://hal.inria.fr/hal-00758627.

Scientific Books (or Scientific Book chapters)

[34] N. DIMITRI, J. VAN EIJCK. *Time Discounting And Time Consistency*, in "Games, Actions and Social Software", J. VAN EIJCK (editor), Springer, 2012, vol. 7010, p. 29 - 38, http://hal.inria.fr/hal-00756874.

[35] M. HILLS. *Streamlining Policy Creation In Policy Frameworks*, in "WADT 2012 Preliminary Proceedings", N. MARTÌ-OLIET, M. PALOMINO (editors), Universidad Complutense de Madrid, Departamento de Sistemas Informáticos y Computación, 2012, p. 61 - 63, http://hal.inria.fr/hal-00756877.

[36] J. VAN BENTHEM, J. VAN EIJCK, J. JASPARS, H. VAN DITMARSCH. *Logic In Action*, Internet, 2012, 1, http://hal.inria.fr/hal-00756881.

[37] H. VAN DITMARSCH, J. VAN EIJCK, F. SIETSMA. *On The Logic Of Lying*, in "Games, Actions and Social Software", J. VAN EIJCK, R. VERBRUGGE (editors), Springer, 2012, vol. 7010, p. 41 - 72, http://hal.inria.fr/hal-00756875.

[38] J. VAN EIJCK, R. VERBRUGGE. *Games, Actions And Social Software*, Springer, 2012, vol. 7010, 1, http://hal.inria.fr/hal-00756872.

[39] J. VAN EIJCK, F. SIETSMA. *Questions About Voting Rules, With Some Answers*, in "Logic and Interactive Rationality: Yearbook 2011", A. BALTAG (editor), ILLC, Amsterdam, 2012, http://hal.inria.fr/hal-00756864.

[40] J. VAN EIJCK. *Perception And Change In Update Logic*, in "Perception and Change in Update Logic", J. VAN EIJCK, R. VERBRUGGE (editors), Springer, 2012, vol. 7010, p. 119 - 140, http://hal.inria.fr/hal-00756873.

Research Reports

[41] V. ZAYTSEV. *Guided Grammar Convergence. Full Case Study Report. Generated By Converge::Guided*, Centrum Wiskunde & Informatica, 2012, n^{o} arXiv-abs/1207.6541, p. 1 - 44, http://hal.inria.fr/hal-00756853.

[42] V. ZAYTSEV. *Negotiated Grammar Transformation*, Centrum Wiskunde & Informatica, 2012, n^{o} XM, p. 1 - 6, http://hal.inria.fr/hal-00756850.

[43] V. ZAYTSEV. *Renarrating Linguistic Architecture: A Case Study*, Centrum Wiskunde & Informatica, 2012, n^{o} MPM-6, p. 1 - 6, http://hal.inria.fr/hal-00756851.