Jornadas de Seguimiento de Proyectos, 2007
Programa Nacional de Tecnologías Informáticas

SELF: Software Engineering and Lightweight Formalisms TIN2004-07943-C04

Ernesto Pimentel [*] U. Málaga
María Alpuente [†] T. U. Valencia
Pascual Julián [‡] U. Castilla–La Mancha
Ricardo Peña [§] U. Complutense de Madrid

Abstract

One of the current problems in software systems development is the increasing complexity of analysing and guaranteeing the reliable behaviour of these systems. This project is oriented towards the development of the methods, tools and techniques necessary for supporting quality software construction, with emphasis on practical application to the industrial processes of software companies. This proposal is based on the use of lightweight formal methods in Software Engineering, i.e., the partial application of formalisms at different levels: language, modelling, analysis and composition. The basic idea is to subordinate general methods that support the entire development process and to enhance the real application of formal methods at certain phases of the software life cycle. In order to illustrate the feasibility of this approach, most of the project activities are within the field of component-based software development.

The project is a coordinated proposal of four university teams with previous R+D experience and skills in the area of formal methods in Software Engineering and multiparadigm declarative programming.

1 Project overview

Such as it was planned in the project proposal, all the activities to be developed in this context may be included in one of the following fields: a) Semantic description of software components. b) Software verification and debugging, and c) Optimization of multiparadigm programs. Different contributions in each of these areas were proposed, by using an approach based on lightweight formalisms, i.e. considering the application of formal methods but only when its transfer to an industrial context may be considered as feasible.

In the first research area, the ideas was to explore different possibilities provided by specification languages (e.g. MAUDE) or functional languages (e.g. HASKELL) to describe software systems where each element (subsystem or component) may present different complementary views. In order to guarantee the applicability to industrial environments, a number of goals were included among the project's objectives: the development of techniques to analyse the termination of executable specifications, the study of its efficiency, the proposal of automated optimization techniques, etc. Also related with the semantic description of software components, the project exposed the

[*]Email: `ernesto@lcc.uma.es`
[†]Email: `alpuente@dsic.upv.es`
[‡]Email: `Pascual.Julian@uclm.es`
[§]Email: `ricardo@ucm.es`

need of complement/extend current interface description languages (used in current component-based platforms) to ensure the correctness of systems constructed by combination of components, by means of automated tools.

In the second research line, the main project's goal was the improvement of *model checking* and *proof carrying code* techniques to make the automated validation of software systems feasible. Thus, on the one hand, the experience of two of the research groups on the application of abstract interpretation techniques to model checking could help to the adaptation of notations, algorithms and tools based on model-checking to the analysis of functional and extra-functional properties. On the other hand, an alternative approach to validate software correctness are the *proof carrying code* techniques, and different open issues were included as goals of the project; in particular, the project included several tasks oriented to apply proof carrying code techniques to declarative programming environments.

As it has been already mentioned, many of the project's goals are related with the exploitation of the high expressive power provided by multiparadigm declarative languages (logic, functions, concurrency, constraints, etc.). For this reason, the proposal was also interested in improving the efficiency of these languages, making their performance comparable to that in imperative languages.

Concerning applications, and maintaining the agile (lightweight) approach, the proposal stressed on specific application domains.

2 Project's achievements

In order to describe the achievements of the coordinated project, we will present the current situation of each subproject. The reader must take into account that we still have one year to finish the project. In any case, the achieved goals are already substantial enough to claim that the project will be successful.

2.1 University of Malaga UMA

The UMA subproject was redefined just before starting, because one of his members, who was responsible of one of the work packages, changed his dedication to a different project. In order to compensate this movement, a new member was added to the UMA's group, who already was cooperating in the work package 1.2. These changes were applied for to the *Subdirección General de Proyectos de Investigación* (Ministry of Education and Science), and they were approved.

Module 1.2: Coordination of Software components

The basic goal of this work package was the development of a coordination methodology to synchronize software components, by extending already well-known interface description languages to deal with behavioural issues. Different task were defined in the project, and the corresponding results are shown below.

Modeling and analysis of software components. We have proposed a process algebra based formalism for describing the interactions between software components [42, 44]. With this same purpose we have explored different coordination mechanisms, such as Linda or Reo [26], comparing their expressive power [25], and UML sequence diagrams [129, 130]. Other different technologies have also been used with this goal, as aspect-orientation [131, 132].

All this work is based on the idea of providing an extension of interface description languages, in such a way that behavioural information is also included in a component interface description. However, making useful this additional information is also an objective. Thus, we have also established techniques of analysis for detecting incompatibility and interoperability problems at the behavioral level [27, 47, 34].

Component adaptation. We have developed a theory of component adaptation. In this sense, we have defined a process algebra based notation for specifying the adaptation required [39, 45, 46], and we have defined its semantics [41]. We have implemented a prototype for an adaptation algorithm for components with interfaces described according to a behavioral type system.

Incremental and dynamic adaptation. We have extended our model and algorithm for component adaptation to take into account dynamic and incremental adaptation [51, 43]. Dynamic adaptation refers to situations in which the requited adaptation between the component may change over time, depending on the context. For that purpose, we have extended the aforementioned notation for considering contextual information. Incremental adaptation refers to situations in which software maintenance, in particular when new functionality is added to components may cause interoperability problems. For that purpose we are exploring how reflection [132] and Aspect-Oriented [50, 49] techniques may help, both defining maintenance in an incremental way, and allowing to get information on components functionality automatically.

Application to Web Services. We have applied the models, notation and tools mentioned above to the adaptation of Web Services, in particular for the adaptation of Web Service choreography and orchestrations [48]. We have also applied Web services-based technologies (e.g. subservicing) as an approach to achieve component adaptation [40].

Module 1.3: Analysis of component-based systems

Research in this module has been mainly focussed on two interrelated lines which have produced a number of results as described below.

Analysis of non-functional properties The goal in this task has been the extension of well-known techniques (typically, model checking techniques) to analyze non-functional properties of systems, such as performance and real time properties. Thus, on the one hand, we have studied the integration of abstraction techniques and model checking in the context of tccp (timed concurrent constraint programming language) to improve the applicability of model checking. We have developed a complete methodology for the abstraction of tccp models by source to source transformation, which allows the complete reuse of tccp model checkers [21, 16]. The proposal is interesting because tccp is a synchronous language and it contains sentences to model typical real time behaviors, such as timeouts, that have not a direct abstraction. In this same context, we have extended the tccp semantics to allow standard model checkers, that analyze temporal properties, to verify real time properties [22, 19].

The other approach in this task is related with the seamless integration of tools that analyze different system properties (performance and functional properties). In particular, making use of XML technologies, we have developed the language PiXL which behaves as a kernel language to interchange the input formats of different tools. PiXL has been used to analyze Enterprise Information Systems (EIS) and it is currently playing a key role in the task of analysis of programming languages described below [121, 89, 88, 87].

Analysis of implementation languages In this task, we have dealt with the problem of analyzing final applications described in a typical implementation language as C. We have explored different approaches to manage the complexity of analyzing programming languages. First, we have concentrated on the verification of distributed applications written in C, making use of the API Socket. In this context, we have followed the model extraction approach in order to construct mixed models for the distributed systems to be analyzed. The new models, that may be verified with SPIN [91], contain both instructions in the modelling language Promela giving the behaviour of the API, and instructions in the original programming language C.

We prove the correctness of the transformation by means of an operational semantics of the API. Tool SocketMC implements this proposal [52, 54].

A parallel research in this area is the integration of abstraction methods and classic static analysis in order to drastically improve the size of the state space stored when realizing model checking. To this end, we have developed the so-called influence analysis which may be used to guide the construction of the so-called abstract matching functions which are used to compare states during the execution of on-the-fly model checking algorithms [53, 85].

We have also worked on the extension of the verification toolbox CADP [1] integrating the model extraction technique above described for concurrent and distributed systems that make use of APIs with well defined semantics. In particular, we have designed and implemented several CADP modules to translate C programs into labelled transition systems which may be handled by the tool [86]. Language PiXL mentioned above has been extensively utilized in this transformation.

Other activities

On the other hand, in a different context, some members of the team have been working on the development of implementation techniques of linear logic-based languages [102, 101]. Although we intended to apply this kind of languages to the description of component behavioural interfaces, we have not explored this research line. Another fruitful research area developed by members of the group is related with the integration of constraint systems in declarative languages [80, 79, 78, 77, 81], in particular to functional-logic languages [64, 63, 66]. The application of these ideas to a specific functional-logic language (Toy) has also been made [28, 65, 82, 83, 30, 29].

2.2 Technical University of Valencia UPV

Module 2.1: Analysis, Optimization and Coordination of Software Components.

In this module we have developed new techniques for proving program termination and other related properties of programs. This includes research on basic techniques, which are central in all termination provers (e.g., polynomial interpretations), basic computational mechanisms which can be useful to model computations in sophisticated programming languages (like context-sensitive rewriting), and transformations of termination problems.

With regard to tasks T.2.1.1, T.2.1.4, and T.2.1.7, the following results have been achieved:

1. We have developed a new framework for using polynomial interpretations over the reals in proofs of termination [104]. This provides the basis for the generation of polynomial orderings in mu-term, see [105]. In [106], we have shown that the polynomials with *rational* coefficients are more powerful than (the more usual) polynomial interpretations with integer coefficients (this was a 25-years-old open problem in termination); we have also shown that the use of more general coefficients (e.g., irrational numbers) can also improve over the use of rational coefficients.

2. We have developed a number of new techniques for proving termination of context-sensitive rewriting (*csr*). We have implemented them as part of the tool mu-term, whose last version is described in [4]: 1) We have extended Arts and Giesl's dependency pairs approach to *csr* [3, 5]; 2) We have investigated the relative power of the different transformations for proving termination of *csr* [107]; 3) We have developed an expert for automatically proving termination of *csr* by combining the different techniques implemented in the tool [4]. In [90] we have investigated the use of proofs of termination of *CSR* for proving *confluence* of term rewriting systems.

3. We have developed software components (via a COM DLL) with the basic capabilities of mu-term, and a graphic interface for using mu-term within the .NET framework through

this COM DLL [6, 7]. This provides an interesting means to interconnect software tools, compilers, interpreters, etc., in complex software systems [24].

4. We have developed new techniques for proving termination of concurrent programs with fairness assumptions that can be modeled as term rewriting systems e.g. in Maude [110]. We have also introduced a new notion of termination (called *operational termination*) which is useful to reason about termination of programs whose operational behavior is modeled by means of inference systems [109]. This is interesting to deal with sophisticated programming languages with powerful expressive features which do not fit simple operational models like pure term rewriting. On this basis we have developed new techniques for proving (operational) termination of Maude programs. In [56], we describe a correct transformation of Maude programs into *csr* systems, which allows is to prove termination of these systems by using MU-TERM

Concerning tasks T2.1.2 and T2.1.3., we have investigated the completeness of conditional *csr* systems [103]. We have given conditions ensuring that restricted computations are still able to obtain canonical forms like head-normal forms, constructor terms or even normal forms. Finally, regarding tasks T.2.1.5 and T.2.1.6. the theoretical basis for the extension of Maude with functional-logic capabilities has been discussed in an invited talk given at *15th International Workshop on Functional and (Constraint) Logic Programming, WFLP'06* [58], and also in [13]. As a first step, a preliminary mechanism of unification modulo associativity and commutativity in Maude has been formulated. In 2007, we plan to extend this mechanism to consider more sophisticated narrowing strategies such as natural narrowing. We also plan to implement context-sensitive narrowing in Maude. As a new line emerging from the results achieved in this module, we are starting to apply rule-based techniques to the modeling and analysis of web sites and the semantic web [108].

Module 2.2: Declarative techniques for rule extraction

This module investigates new rule-based techniques for knowledge discovery. We have focused on the definition of a new generalisation framework which is based on the notions of pattern and distance. We have also developed new techniques for extracting or adapting comprehensible models, and specifically, for adapting them to a new context (misclassification costs and class distributions).

In Task 2.2.1, the concept of distance-based binary generalisation operators was introduced [68]. A generalization of this framework which allows us to work with n-ary operators was proposed in [72]. We have applied this framework to first-order objects (atoms and Horn clauses) in [73, 76] and to graphs in [71, 75]. Another related problem investigated in Task 2.2.4 is the definition of similarity functions over structured data, which allows us to apply distance-based methods to complex domains. In [67, 74] we described and analised several distance and pseudo-distance functions for structured datatypes: sets, multisets, terms, atoms, lists and graphs. Some of these functions were derived from kernel functions by exploiting the formal relationship between distances and kernels. In Task 2.2.5 we implemented some of these functions in the Distance-Based Decision Tree (DBDT) learning system defined in [74], which combines decision tree learning and distance-based learning. The key ingredient of our learning method is a new splitting criterion which is based on centroids of only one attribute, while other methods compute centroids by using the whole examples. Hence our method is able to induce models, i.e. decision trees, by using only the distance that corresponds to the datatype of the attribute selected by the splitting criterion. Consequently, the algorithm is able to deal with any datatype which is endowed with a metric distance. Since each partition involves just one attribute, the resulting model can be easily expressed by using declarative rules. The DBDT system has been experimentally evaluated over propositional problems and on datasets that contain structured attributes. It has been also applied to the classification of web pages in

[69, 70]. In Task 2.2.5 we have also applied data mining techniques for hospital management ([2]). The obtained results show a satisfactory performance in all the considered scenarios.

One of the approaches for extracting comprehensible models from other less intelligible models is the mimetic technique. Basically, it consists in using an accurate but generally incomprehensible model as an oracle for generating and labeling an invented dataset. Together with the original training set, this dataset can be then used for training a comprehensible model (for instance, a decision tree) which mimics the oracle. In Task 2.2.2 we have investigated the application of the mimetic technique when the original training data are not available (this is a typical situation in many areas such as expert systems, engineering, diagnosis, medicine, manufacturing, business, etc.). In [36], we have defined a theoretical framework based on the MML (Minimum Message Length) principle for estimating the optimal size of the invented dataset. In Task 2.2.2, we have also investigated the application of the mimetic technique to model revision. Our methodology formalized in [37, 38] applies to the scenario in which a change in the problem definition occurs, e.g. when new data belonging to an area initially not covered by the original model are considered. The key idea is to use the old model as an oracle in order to generate a wide and large dataset representing the initial situation. The empirical evaluation of the method shows that, by considering this dataset (where we have many data) together with the new one (where data are less detailed), the mimetic model performs better than the the classical approach.

In Task 2.2.3, we have defined new evaluation measures which use both estimated probabilities and example ordering [84]. In [35], we analyse three different techniques to establish an optimal-cost class threshold when training data are not available: one technique is directly derived from the definition of cost, a second one is derived from a ranking of estimated probabilities, and the third one is based on ROC analysis.

Module 2.3: Software Verification and Debugging

The main goal of this module is to develop methods, tools and techniques for (automatic) software verification and debugging. The module was organized in 5 tasks. Tasks 2.3.1 and 2.3.2 focused on a main problem of automated program verification and analysis: the state explosion problem. We have investigated three different approaches that improve previous techniques for verifiying reactive systems which are specified in ccp. In [15], we developed a symbolic method which is based on an extension of binary decision diagrams for handling constraints symbolically. In [18, 16, 17] we defined a method based on abstract interpretation which uses both over- and under-approximations in order to improve the accuracy of the approximation while correctly models synchronization. In [20, 22], we have defined a refinement of the logic which is used for the specification of properties in t ccp that also improves the precision and quality of the verification. We proposed a new model for timed ccp which includes an explicit notion of time and allows the user to verify more sophisticated properties. in [23] we have integrated a functional engine within the ccp paradigm. This framework has been implemented in the tccp-func prototype which is written in curry and support the specification of reactive systems that need some kind of sophisticated, arithmetic computations, .

The increased complexity of Web sites and the explosive growth of Web-based applications has turned their design and construction into a challenging problem. In Tasks 2.3.3 and 2.3.4, we have developed a framework for the automated verification of Web sites which can be used to specify integrity conditions for a given Web site, and then automatically check whether these conditions are fulfilled [8, 9, 31]. Our methodology is based on a novel rewriting-based technique, called *partial rewriting*, in which the traditional pattern matching mechanism is replaced by tree *simulation*, a suitable technique for recognizing patterns inside semistructured documents. The framework was in Java and endowed with a powerful rewriting engine written in Maude. We have also developed a novel, semi-automatic rewriting-based methodology for repairing faulty Web sites, as well as its application to the filtering of XML documents based on partial rewriting [10, 11, 33, 32]. With regard to program debugging, starting from the previous system Debussy, we have built an abstract debugger and a program corrector for Maude which are based on a highly compressed semantics that we developed for this language [12].

Concerning program optimization, we have defined a framework for the analysis, detection and removal of redundant arguments from functions. Our removal mechanism preserves the original program semantics. The proposed framework includes analysis techniques which are based on different formalisms including tree automata, and abstract interpretation. We have implemented a tool that includes some of the proposed techniques in [14].

Finally, Task 2.3.5 focused on the analysis and verification of communication protocols. We have reformulated the "NRL Protocol Analyzer" by exploiting Maude capabilities. We formulated a declarative framework based on rewriting logic in which security properties can be specified and analyzed [57, 62, 59, 60, 61].

2.3 University of Castilla–La Mancha UCLM

In general, the activities of the different tasks have progressed according to the workplan. In the following subsections we summarize the subproject's achievements.

Module 3.1: Multi-paradigm declarative languages: implementation advanced techniques

The main goal of this module is to develop implementation techniques that may contribute to the optimization and increasing efficiency of multi-paradigm declarative languages. We think that the optimization of these languages may help to use them as rapid prototype tools, as well as tools for introducing formal techniques in the field of Software Engineering. Also the improvement of multi-paradigm declarative languages may facilitate their introduction in industrial areas.

We investigate different ways of improving these languages: first, introducing optimizations in high level implementations of multi-paradigm declarative languages; second, examining how to adapt program transformation techniques developed in the functional programming area to the context of multi-paradigm declarative languages. Also we want to explore how to introduce fuzzy unification into the core of these languages. The first goal is dealt with in the task 3.1.1: Improvement of high level implementations.

In [55] we define an algorithm, based on an analysis of definitional trees, that transforms a constructor based, weakly orthogonal program into an inductively sequential one with a deterministic behavior (and therefore, more efficient). We describe the criteria that make this transformation effective and we prove its correctness if non-terminating or undefined expressions are not considered. When non-terminating or undefined expressions are permitted, we provide a negative result: it is impossible to construct a complete transformer to inductively sequential programs.

In [99] and [100] we address the problem of adapting the implementation of a Warren Abstracta Machine (WAM) to incorporate similarity-based fuzzy unification. The clue is to transform the Prolog source program into an intermediate code compiling the information provided by a set of similarity equations, which is proved to be semantically equivalent to the original one. As a result, we obtain a Prolog implementation based on similarity relations that we call S-Prolog. To the best of our knowledge this is the first WAM implementation that supports similarity-based SLD resolution.

Module 3.2: Multi-Paradigm Declarative Languages: transformations and extensions based on fuzzy logic

During the last years, we have had conscience of the important role played by fuzzy logic in the development of sophisticated software tools in many different settings, such as expert systems, medicine, industrial control, and so on. In order to help the development of such applications, it has been observed a growing interest for designing fuzzy declarative (in particular, logic) languages.

Among their expressive resources, these languages incorporate the ability for dealing with imprecise information in a natural way. On the other hand, (the so-called) multi-paradigm declarative programming integrates, together with concurrency, the (two) most important declarative programming paradigms: the functional and the logic ones. Having into account these antecedents, in the present module we focus our attention in two ambitious goals (to be developed in two task too) which can be summarized as follows: the complete integration into the multi-paradigm declarative framework of functional, logic, fuzzy and concurrent programming styles (task 3.2.1), and the subsequent design of transformation, optimization and specialization techniques to be applied to programs written with this kind of languages (task 3.2.2).

Concerning Task 3.2.1, we have developed an intensive work. Thus, in [115] and [118] we propose our first approach to the integration of functional–logic and fuzzy–logic programming paradigms, in order to obtain a richer programming framework where mathematical functions cohabit with fuzzy logic features. We have proposed an hybrid dialect where a set of rewriting rules associated to the functional logic dimension of the language, are accompanied with a set of similarity equations between symbols of the same nature and arity, which represents the fuzzy counterpart of the new framework. We have extended the operational mechanism of needed narrowing to deal safely with similarity relations. On the other hand, in [116] and [120] we have proved that the fuzzy variant of needed narrowing verifies the following properties: (crispness) it computes at least the same elements of the crisp case; (fuzziness) all similar terms of a given goal are completely treated by exploiting the similarities collected in a given program as much as possible; (termination) it avoids the risk of infinite loops associated to the intrinsic (reflexive, symmetric and transitive) properties of similarity relations. Finally, in [117] and [119] we complete the previous operational principle by accompanying it with a powerful notion of equality, called strict-similar equality, which is specially well-suited for declarative languages amalgamating functional-fuzzy-logic features. We have showed that, extending the notion of “strict equality” (used in languages such as Curry) with the more flexible of similar equality (used in the language Likelog), similarity relations can be successfully treated while mathematical functions are lazily evaluated in a given program. Our method can be implemented at a very high abstraction level by simply performing a static pre-process at compilation time which only manipulates the program at a syntactic level.

With regard to Task 3.2.2 (Transformation, optimization and specialization of fuzzy declarative programs), in [94] we introduce an unfolding rule for an extension of the fuzzy logic language defined by Vojtas and Paulík (1996). We demonstrate the strong soundness and completeness of the transformation. In order to prove the strong correctness property we need to adapt the classical concept of a *computation rule* to our setting, and we prove the independence of the new computation rule. Also, we show that our transformation rule always produces an improvement in the efficiency of the residual program, by reducing the length of successful fuzzy SLD-derivations. Moreover, such benefits can be reinforced with the combined use of unfolding with the preliminary version of a folding described in [114]. In [92] we adapt the unfolding rule introduced in [94] for the framework of multi–adjoint logic programming —developed by Medina et. alt. (2004)—. Also for this new framework we prove: i) the independence of the computation rule for multi–adjoint admissible computations. ii) the strong correctness and efficiency properties of the fuzzy unfolding rule.

In [93] and [97], following the clean separation between the operational and interpretative phases in a multi-adjoint logic computation, we distinguish between an operational and interpretive phase during the unfolding process. We introduce a novel *interpretive* unfolding transformation rule, which is proved strong sound and complete. In order to formalize the notion of interpretive unfolding we provide a procedural characterization of the interpretive phase of a computation (in terms of a state transition system) what constitute a distinguished feature with regard the approach considered in the original multi-adjoint logic programming framework.

In [96] we have introduced a definition of the concept of Partial Evaluation (PE) for multi-adjoint logic programs and goals. Using PE techniques we are able to define a refinement to the notion of classical reductant, that we call $PE\!-\!reductant$. Reductants are crucial to cope with a problem of incompleteness in the context of multi-adjoint logic programming. In [95] and [98] we provide a concrete algorithm for the construction of PE-reductants which is based on unfolding with a set of dynamic thresholds. This provide an effective and efficient method for computing reductants, based on PE techniques, which constitutes a novel application in the multi-adjoint logic programming framework.

2.4 Universidad Complutense de Madrid UCM

This subproject is placed in the Proof Carrying Code research area. In this framework, a code consumer verifies that the code provided by a non-reliable producer fulfils a set of security rules, which constitutes its *security policy*. This is done be mecanically checking a formal proof which the producer attaches to its code. The security policy which constitutes the main objective of the subproject is to certificate that programs have a *safe* memory management. In our context, safety means two things: (a) absence of dangling pointers; (b) programs have a proved upper bound on its run-time memory needs.

We have approached these elements from the program analysis perspective, analyses based both on abstract interpretation and on types and effects systems. We have decided the source language to be a functional one, building on our previous knowledge on this area. The detailed objectives can be enumerated as follows:

1. Definition of the language facilities from the programmer's point of view.
2. Definition of the different analyses. These should focus on placement of data structures, sharing properties, program termination and memory consumption. Usage of types guarantees certificates compositionality, so that a function that uses an already certificated function may incorporate that certificate.
3. Semantics definition, abstract machine definition and compilation process definition. From the source language we intend to obtain a Java-like byte-code.
4. Generation of certificates about the properties satisfied by programs.
5. Implementation of all the above.

We still have not achieved the objectives related to the termination analysis, the memory cost analyisis, and the certificate generation. A cooperation is running with the subgroup at UPV in order to use their termination tools to approach the first objective. Once this is solved, we hope that it will the basis to solve the second one. The third one —certificate generation— is being approached with an ongoing cooperation with Julio Rubio's group at the *Mathematics and Computation* Department of *Universidad de La Rioja*. This group has expertise in mecanically proving theorems with tools such as ACL2, Isabelle, and Coq. A help from them is possible in order to translate our correctness theorems into a formal proof that these tools could validate

¿From the results achieved in the subproject, we have presented and published the language syntax, examples of language usage and a preliminary version of the type system in [122, 123]. The sharing analysis and the big-step operational semantics have been presented in [124, 127] and will be formally published in [128]. The abstract machine derivation and the code generation phase have been presented in [125, 126] and they have been submitted for formal publication. The safety-types inference algorithm will be shortly presented at The Trends on Functional Programming Workshop TFP07, next April [113]. The correctness theorems have not been published yet. We are currently preparing a journal version of the whole language which will include these proofs.

Below, we give detailed information concerning work-packages such as they were defined in the project proposal.

Module 4.1: Design of the high-level language and the compiler–certifier

This first module was scheduled around the following activities:

Source language definition A first version of the language, called SAFE, has been thoroughly defined. It is eager, first order, polymorhic and with user-defined data constructors. There is a full-SAFE version, in which programmers write programs, and a Core-SAFE version in which the different analyses are defined. Full-SAFE has a Haskell-like syntax and programs scarcely differ from the ones a conventional functional programmer would write. The programmer has a novel facility for *destructive pattern matching* which allows to explicitly dispose the cells occupied by a particular data structure. A higher-order extension of the language is foreseen.

Big-step operational semantics Core-SAFE has explicit *regions* in which data structures are placed. No garbage collector is needed. The big-step operational semantics detailing the evaluation of expressions and programs, as well as region and closure creation/disposal has been defined.

Small-step operational semantics A second small-step operational semantics has also been defined. This is needed in order to prove the correctness of the type system and for the formal derivation of the abstract machine. We have formally proved the equivalence of the big-step and the small-step operational semantics.

Sharing analysis At Core-SAFE level, we have defined an abstract interpretation-like sharing analysis which annotates programs with four relations between the variables in scope at each particular expression. These relations tell which are the sharing properties of the data structures to which the variables refer to. For instance $x\ Sh\ y$ tells that structures pointed to by x and y will share at run-time a common substructure.

Sharing analysis correctness theorem We have defined an *abstraction function* between run-time pointers and compile-time program variables and formally proved that the sharing detected by our analysis is an upper bound of the actual sharing taking place at run-time. In this proof we have used the big-step operational semantics.

Safety type system The above relations between variables are extensively used by our type system ir order to give correct types to programs. The novelty of this type system is that it annotates types with information about the destruction of data structures. In this way, a data structure (e.g. a list) can be *condemned* (this means that it will be destroyed in the future), *in danger* (this means that it shares a condemned piece) or *safe* (it does not share or contain condemned pieces). Based on these annotated types, the type system prevents programs from performing dangerous actions such as reading an already destroyed data structure. The type system also detects in which regions data structures should be created in order that region disposal does not generate dangling pointers.

Safety type system correctness theorem We have formally proved that a well-typed program is guaranteed not to run into dangling pointers when executed. This has been the major achievement of the project up to now. The correctness proof is about ten pages long and uses the small-step semantics of the language, as well as the sharing analysis correctness theorem.

safety type inference algorithm We have defined an inference algorithm to compute SAFE types automatically, without relying on programmer's annotations. The algorithm has a Hindley-Milner phase with some additions in order to give polymorphic types to region identifiers, and a *destruction-propagation* phase which infers the condemned, in-danger or safe types for each of the variables and function arguments. The algorithm is modular in the sense that the inference of each function can be done by only using the already inferred types of the invoked functions.

Implementation All the above phases, except code generation, have been implemented. We have now a prototype of the certifying compiler to which new analyses and phases can be added. It is written in Haskell and has about 4000 lines up to now.

Module 4.2: Imperative abstract machine design and certify checking

The activities of this module have been made around the following tasks:

Imperative abstract machine From the small-step operational semantics we have formally derived several abstract machines, the last one of which is very close to the Java Virtual Machine. This process has been done by proving the correctness of each step so that it is guaranteed that the abstract machine correctly implements the semantics of the language.

Formal derivation of the code generation Translation schemes from Core-SAFE to abstract machine byte-code have been defined and proved correct. Now, we are in a position to generate imperative code from the full-SAFE language.

3 Results indicators

We include below some relevant indicators which provide information about the activities in which project members have been involved.

3.1 Summary of the most relevant publications:

In Section 2, different works were referenced showing the contributions in each subproject. Thus, the bibliography of this document may be considered a full list of publications derived from the project. To give an idea about the relevance of these contributions, we classify them by scope and type of publication:

International journals	30 (19 indexed in SCI)
Book chapters	2
Electronic journals	18 (ENTCS series)
Int. conferences and works.	48 (Springer LNCS, IEEE Press, ACM Press)
National conferences:	27
Others (PhD Thesis)	2

3.2 Software systems:

αSPIN	`http://gisum.lcc.uma.es/~gisum/fmse/tools`
MU-TERM	`http://www.dsic.upv.es/users/elp/slucas/muterm`
Natur	`http://www.dsic.upv.es/users/elp/natur`
GVERDI	`http://www.dsic.upv.es/users/elp/GVerdi/`
GVerdi-R	`http://www.dsic.upv.es/users/elp/GVerdiR/`
tccp-func	`http://www.dsic.upv.es/~villanue/tccp-func/`
DBDT	`http://www.dsic.upv.es/~flip/dbdt/`

3.3 Training activities:

The following PhD students are currently developing their PhD within the topics of the project:

Module 1.2: Javier Cubo (FPI-MEC), Javier Cámara (FPI-Junta Andalucía), Ana M. Roldán (Assistant Prof, Univ. Huelva), Silvia Amaro (Lecturer, Univ. Comahue, Argentina), José Antonio Martín (SELF contract).

Module 1.3: David Sanán (FPI-MEC).

Module 2.1: Beatriz Alarcón (FPU-MEC), Sonia Flores (ALFA LerNet grant), Raúl Gutiérrez (FPI-SELF grant), and Rafael Navarro (SELF contract; FPI-GV grant, pending).

Module 2.2: Antonio Bella (SELF contract; FPU-MEC grant, pending), Ricardo Blanco (SEIT-ANUIES), and Vicent Estruch (Ayudante).

Module 2.3: Mauricio Alba (ALFA LerNet grant), José Iborra (FPI-UPV grant), Alexey Lescaylle (SELF contract), Pedro Ojeda (MIUR grant, co-tutela), Daniel Romero (ALFA LerNet grant), and Matteo Zanella (Socrates co-tutela).

Module 4.2 Manuel Montenegro

By means of "Juan de la Cierva" contracts, Christoffe Joubert from INRIA-ALPES, who has has been working with Málaga in 2006, joins the UPV group in February 2007. With similar conditions, Gwen Salaün will (also from INRIA-ALPES) joined the UMA group since December 2006. Their activities will be developed basically in modules 2.3 and 1.2, respectively.

As part of the training activity of the project, we could also mention that some people involved in the project have moved to companies: Javier García (ex-FPI grant, with Hewlett-Packard now), Jose Daniel Llopis (ex-SELF contract, with Regional Industry Ministry now).

3.4 New national and international Collaboration:

Strongly related to the SELF objectives, the UMA team has participated in different CENIT (industrial-oriented projects) proposals, still pending of evaluation. UMA node is also actively participating in one of the work packages of the IST-033563 project (funded by the EU), and coordinated by Manuel Díaz (also from the University of Malaga). UMA node is currently maintaining some contacts with Sangiorgi and Zavattaro (Univ. of Bologna, Italy), Honda and Yoshida (Queen Mary College, UK), Pistore (ITC-irst Trento, Italy) and Margaria (Univ. of Dortmund, Germany), to apply for a project in the first call of FP7.

Also related to the SELF project, a new German-Spanish Acción Integrada has been recently approved which starts in January 2007.The bi-national project, lead by Jürgen Giesl (RWTH Aachen, Germany) and Salvador Lucas (UPV) aims to investigate the development of efficient tools for proving program termination. On the other side, a new national research network which focus on the development of the language Maude also starts in January 2007, led by Prof. Narciso Martí-Oliet from the Universidad Complutense de Madrid (UCM) (TIN2005-25854-E). Other partners of this project are UMA, UPV, UEX, UMU, and USA. Other two French-Spanish collaboration actions (Acción Integrada) were also applied for and scientifically well valuated, but they were not approved because of funding limitations.

The UPV node is member of the Coordination Action (CA) for Ubiquitous Knowledge Discovery, KDubiq, funded by the European Union under IST (Information Society Technology), FET Open (Future and Emerging Technologies) in the 6th Framework Programme. The research topics supported by this CA are related to the research activity developed in Module 2.2.

UCM node has started a new contact with researchers working on similar languages, such as the Munchen's group of Martin Hofmann and Hans-Wolfgang Loidl, and the Scottish groups that are currently developing the functional language Hume. UCM group has invited Hans-Wolfgang to visit Madrid next spring in order to interchange their respective experiences.

3.5 Other activities:

1. Participation in Program Committees:

   (a) International conferences: LPAR-12, FASE'05, COORDINATION'05, IJCAI'05, ICML'05, ECML'05/PKDD'05, LOPSTR'05, AISC'06, FMOODS'05, FACS'06, FASE'06, COORDINATION'06, ICLP'06, LPAR-13, ISOLA'06, FASE'06, ECAI'06, ECML'06/PKDD'06, IDEAL'06, RTA'07, PPDP'07, COORDINATION'07, FMICS'07.

   (b) International workshops: RULE'05, AAIP'05, ROCML'05, WCAT'05, IDEAS'05, RISE'05, FMICS'05, IDEAS'05, FOCLASA'05, IDEAS'06, WFLP'06, WRLA'06, WLPE'06, WRS'05, WRS'06, WRS'07, WWV'05, WWV'06, ROCML'06, WTDIA'06, WCAT'06, FOCLASA'06, RISE'06, FMICS'06, IDEAS'07.

   (c) National conferences: JIDI'05, PROLE'05, JISBD'05, TAMIDA'05, CAEPIA'05, PROLE'06, JISBD'06, WISBD'06, JIDI'06, JISBD¡07, PROLE'07, CEDI'07, CAEPIA'07, TTIA'07

2. Research Stays: Some members of the project have made different research stays:

Researcher	Centre	Period
María Alpuente	Birla Science Center (India)	Feb. 2005
María Alpuente	U. San Luis (Argentina)	Oct-Nov. 2006
Carlos Canal	Univ. of Pisa, Italy	Sept. 2005
Carlos Canal	Univ. d'Evry, France	July 2006
Carlos Canal	Univ. Extremadura	June 2006
Francisco Durán	Urbana-Champain (USA)	Aug 2005
Santiago Escobar	University of Illinois at Urbana-Champaign (USA)	Apr. - Aug. 2005 &Apr. - Sept. 2006
Santiago Escobar	U. Udine (Italia)	May 2006
Santiago Escobar	ENS Cachan (France)	March 2006
Santiago Escobar	ETH Zurich	March 2006
Santiago Escobar	SRI, CA (EEUU)	March 2006
Santiago Escobar	Naval Research Lab (USA)	Nov. 2005 & Sept. 2006
Santiago Escobar	Birla Science Center (India)	Feb. 2005
Vicent Estruch	Fraunhofer Inst. Bonn (Germany)	Oct. - Dec. 2006
Javier García-Vivó	University of Siena (Italia)	Sept. - Oct. 2005
Raul Gutierrez	Birla Science Center (India)	Dec. 2006
Raul Gutierrez	Cons. Nal. des Arts et Metiers CNAM	Dec 2006
José Iborra	U. Siena (Italy)	Nov.-Dec. 2006
Pablo López	Carnegie Mellon	Jul-Sept 2005
Salvador Lucas	University of Illinois	May 2005 & May 2006
Salvador Lucas	U. San Luis (Argentina)	Oct. 2005
Salvador Lucas	Tech. Universität, Wien (Austria)	Sept. 2005 & March-Apr. 2006
Pedro Ojeda	U. Siena (Italy)	Oct.-Dec. 2006
Ernesto Pimentel	U. Pisa (Italy)	Feb. 2005
Daniel Romero	U. Udine (Italy)	May 2006
Daniel Romero	Birla Science Center (India)	Sept. 2006
Alicia Villanueva	TU Wien	Nov. 2005 & March 2006
Alicia Villanueva	U. Málaga	Nov. 2006

And we have also enjoyed the visits of a number of researchers in 2005-2006:

- Researchers visiting UMA node: Carlos Areces (INRIA, Nice, France), Antonio Brogi (Univ. Pisa, Italia), Miguel Katrib (Univ. La Habana, Cuba), Tiziana Margaria (Univ. Dortmund, Germany),José Meseguer (U. Illinois in Urbana-Champaign, USA), Pascal Poizat (Univ. d'Èvry, France), Jeff Polakov (Nat. Inst. of Industrial Science and Tech., USA), Gwen Salaün (INRIA-ALPES, France).
- Researchers visiting UPV node: Marco Comini (U. di Udine, Italia), Evelyne Contejean (LRI, U. Paris-Sud), Pierre Courtieu (LRI, U. Paris-Sud), Nachum Dershowitz (Tel Aviv University, Israel), Yaniv Eytani (U. Illinois in Urbana-Champaign, EE.UU.), David Dowe (Monash University, Australia), Bernhard Gramlich (Technische Universität Wien, Austria), Christophe

Joubert (INRIA Grenoble, Francia), José Meseguer (U. Illinois in Urbana-Champaign, USA), Ugo Montanari (U. di Pisa, Italia).

3. Organization of International Conferences/Workshops:

   (a) 18th European Summer School on Language, Logic and Information. ESSLLI. August, 2006. Malaga.

   (b) 5th Workshop on the Foundations of Concurrent Languages and Software Architectures (FOCLASA'06), Bonn (Germany), August 31, 2006, affiliated to CONCUR 2006.

   (c) 2nd Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT'06). Nantes (France), July 4, 2006, associated to ECOOP 2006.

   (d) 4th Workshop on the Foundations of Concurrent Languages and Software Architectures San Francisco (USA), August 27, 2005, associated to CONCUR 2005.

   (e) 1st Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT'05). Glasgow (United Kingdom), July 25, 2005, associated to ECOOP 2005.

   (f) 1sA t Int'l Workshop on Automated Specification and Verification of Web Sites WWV 2005, March 14-15, 2005, Valencia, Spain (Proc. in ENTCS [112], Elsevier)

   (g) ROC Analysis in ML, ROCML'05, held within the 22nd Int'l Conf. on Machine Learning ICML 2005, Bonn, Germany, Aug 7-11, 2005.

   (h) 2nd Int'l Workshop on Automated Specification and Verification of Web Sites, co-located with ISOLA 2006, Paphos, Cyprus, November 19, 2006 (Proc. IEEE Computer Society Press [111])

   (i) Jornadas de Programación y Lenguajes. PROLE 2004. Malaga, November 10-12, 2004.

   (j) Jornadas de Ingeniería del Software y Bases de Datos. JISBD 2004. Malaga, November 10-12, 2004.

   (k) Annual Meeting of the European Consortium for Informatics and Mathamatics (ERCIM). Malaga, November 2-5, 2004.

References

[1] An overview of CADP. *EASST Newsletter*, 4:13–24, 2002.

[2] J. Alapont, A. Bella, C. Ferri, J. Hernández-Orallo, J. D. Llopis, and M. J. Ramírez-Quintana. Specialised Tools for Automating Data Mining for Hospital Management. In *Proc. First East European Conference on Health Care Modelling and Computation*, pages 7–19, 2005.

[3] B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. Proving termination of context-sensitive rewriting with mu-term. In *Actas de las VI Jornadas de Programación y Lenguajes, PROLE'06*, pages 57–66. Universidad Politécnica de Cataluña, 2006.

[4] B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. Proving termination of context-sensitive rewriting with mu-term. In *Actas de las VI Jornadas de Programación y Lenguajes, PROLE'06*, Electronic Notes in Theoretical Computer Science. Elsevier Sciences Publisher, 2007. To appear.

[5] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-sensitive dependency pairs. In *Proc. of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 4337 of *Lecture Notes in Computer Science*, pages 297–308. Springer-Verlag, Berlin, 2006.

[6] B. Alarcón and S. Lucas. Crossing the rubicon: from haskell to .net through com. *ERCIM News*, 63:51–52, 2005.

[7] B. Alarcón and S. Lucas. Building .net guis for haskell applications. In *Proc. of the 4th International Conference on .NET Technologies*, pages 57–66. University of West-Bohemia, 2006.

[8] M. Alpuente, D. Ballis, and M. Falaschi. A Rewriting-based Framework for Web Sites Verification. *Electronic Notes in Theoretical Computer Science*, 124(1):41–61, 2005.

[9] M. Alpuente, D. Ballis, and M. Falaschi. Rule-based Verification of Web sites. *Int'l Journal on Software Tools for Technology Transfer*, 8:565–585, 2006.

[10] M. Alpuente, D. Ballis, M. Falaschi, and D. Romero. A Semi-automatic Methodology for Reparing Faulty Web Sites. In *Proc. of the 4th IEEE Int'l Conference on Software Engineering and Formal Methods(SEFM'06). Pune, India*, pages 31–40. IEEE Computer Society Press, 2006.

[11] M. Alpuente, D. Ballis, M. Falaschi, and D. Romero. GVERDI-R A Tool for Repairing Faulty Web Sites. In *Proc. VI Jornadas sobre Programación y Lenguajes (PROLE'06). Sitges, Spain*, pages 221–230. CIMNE Barcelona, 2006.

[12] M. Alpuente, M. Comini, S. Escobar, and Falaschi. A Compact fixpoint semantics for Term Rewriting Systems. 2006. Submitted for publication.

[13] M. Alpuente, S. Escobar, and S. Lucas. A note on syntactic annotations for narrowing. In F.J. López-Fraguas, editor, *Proc. of the Quintas Jornadas sobre Programación y Lenguajes, PROLE'05, 2005*, pages 25–30. Thomson, 2005.

[14] M. Alpuente, S. Escobar, and S. Lucas. Removing Redundant Arguments Automatically. *Theory and Practice of Logic Programming*, 7(1):1–33, 2007.

[15] M. Alpuente, M. Falaschi, and A. Villanueva. A Symbolic Model Checker for tccp programs. In N. Guelfi, editor, *Selected papers of 1st International Workshop on Rapid Integration of Software Engineering Techniques*, volume 3475 of *Lecture Notes in Computer Science*, pages 45–56. Springer-Verlag, 2005.

[16] M. Alpuente, M. M. Gallardo, E. Pimentel, and A. Villanueva. A Semantic Framework for the Abstract Model Checking of tccp programs. In *Actas de las V Jornadas sobre Programación y Lenguajes, PROLE05*, pages 95–100, 2005.

[17] M. Alpuente, M. M. Gallardo, E. Pimentel, and A. Villanueva. A Semantic Framework for the Abstract Model Checking of tccp programs (extended abstract). In F.J. López-Fraguas, editor, *Proc. of the Quintas Jornadas sobre Programación y Lenguajes, PROLE'05, 2005*, pages 97–100, 2005.

[18] M. Alpuente, M. M. Gallardo, E. Pimentel, and A. Villanueva. Abstract Model Checking of tccp programs. In A. Cerone and A. di Pierro, editors, *Proceedings of the Second Workshop on Quantitative Aspects of Programming*, volume 112 of *Electronic Notes in Theoretical Computer Science*, pages 19–36. Elsevier Science, 2005.

[19] M. Alpuente, M. M. Gallardo, E. Pimentel, and A. Villanueva. Verifying Real-Time Properties of tccp Programs. In *Actas de las V Jornadas sobre Programación y Lenguajes, PROLE05*, pages 85–94, 2005.

[20] M. Alpuente, M. M. Gallardo, E. Pimentel, and A. Villanueva. Verifying Real-Time Properties of tccp Programs. In F.J. López-Fraguas, editor, *Proc. of the Quintas Jornadas sobre Programación y Lenguajes, PROLE'05, 2005*, pages 85–94, 2005.

[21] M. Alpuente, M.M. Gallardo, E. Pimentel, and A. Villanueva. A Semantic Framework for the Abstract Model Checking of tccp programs. *Theoretical Computer Science*, 346:58–95, 2005.

[22] M. Alpuente, M.M. Gallardo, E. Pimentel, and A. Villanueva. Verifying Real-Time Properties of tccp Programs. *The Journal of Universal Computer Science*, 12(11):1551–1573, 2006.

[23] M. Alpuente, B. Gramlich, and A. Villanueva. A Framework for Timed Concurrent Constraint Programming with Arithmetic Computations. In F. Lucio and F. Orejas, editors, *Actas de las VI Jornadas de Programación y Lenguajes (PROLE'06)*, Electronic Notes in Theoretical Computer Science. Elsevier Sciences Publisher, 2007. To appear.

[24] M. Alpuente and S. Lucas. Connecting remote tools: Do it by yourself! *ERCIM News*, 61:48–49, 2005.

[25] S. Amaro, E. Pimentel, and A.M. Roldán. A preliminary comparative study on the expressive power of reo and linda. In *Proc. of Foclasa'04*. FOCLASA'04, August 2004.

[26] S. Amaro, E. Pimentel, and A.M. Roldán. Coordinating behavioral descriptions of components. *Journal of Universal Computer Science*, 11(10):1676–1694, 2005.

[27] S. Amaro, E. Pimentel, and A.M. Roldán. Reo based interaction model. *Electr. Notes in Theor. Comput. Sci.*, 160:3–14, 2006.

[28] Fernando Sáenz-Pérez Antonio J. Fernández, Teresa Hortalá-González. Using the TOY(FD) Constraint Functional Logic System. In Francisco J. López-Fraguas, editor, *PROLE'05 - V Jornadas sobre Programación y Lenguajes (Simposio 10 del I Congreso Español de Informática (CEDI'05))*, pages 115–120, Granada, España, 2005. Thomson.

[29] Fernando Sáenz-Pérez Antonio J. Fernández, Teresa Hortalá-González and Rafael del Vado-Vírseda. Constraint functional logic programming over finite domains. *Theory and Practice of Logic Programming (TPLP)*, 2007. In press.

[30] Purificación Arenas, Antonio J. Fernández, Ana Gil, Francisco López-Fraguas, Mario Rodríguez-Artalejo, and Fernando Sáenz-Pérez. Toy, a multiparadigm declarative language. versión 2.2.3. Manual, June 2006.

[31] D. Ballis. *Rule-based Software Verification and Correction.* PhD thesis, University of Udine and Technical University of Valencia, 2005.

[32] D. Ballis and D. Romero. Filtering of XML documents. *Proc of 2nd Int'l Workshop on Automated Specification and Verification of Web Systems. Paphos, Cyprus*, pages 503–009, 2006.

[33] D. Ballis and D. Romero. Fixing web sites using correction strategies. *Proc of 2nd Int'l Workshop on Automated Specification and Verification of Web Systems. Paphos, Cyprus*, pages 495–502, 2006.

[34] S. Becker, C. Canal, N. Diakov, J. M. Murillo, and P. Poizat. Coordination and Adaptation Techniques: Bridging the Gap between Design and Implementation. In *Object-Oriented Technology. ECOOP 2006 Workshop Reader*, Lecture Notes in Computer Science. Springer, 2007. (in print).

[35] R. Blanco, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Estimating the class probability threshold without training data. In *3rd International Workshop on ROC Analysis in Machine Learning*, pages 9–16, 2006.

[36] R. Blanco, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Knowledge Acquisition through Machine Learning: minimising expert's effort. In *Proceedings of 4th International Conference on Machine Learning and Applications*, pages 49–54. IEEE Computer Society, 2005.

[37] R. Blanco, J. Hernández-Orallo, and M. J. Ramírez-Quintana. La técnica mimética en ausencia de datos originales: aprendizaje y revisión de modelos. In *III Taller de Minería de Datos y Aprendizaje (TAMIDA 2005)*, pages 213–222, 2005.

[38] R. Blanco, J. Hernández-Orallo, and M. J. Ramírez-Quintana. La técnica mimética en ausencia de datos originales: aprendizaje y revisión de modelos. *Inteligencia Artificial. Revista Iberoamericana de IA*, 29:59–68, 2006.

[39] A. Bracciali, A. Brogi, and C. Canal. A Formal Approach to Component Adaptation. 74(1):45–54, 2005.

[40] A. Brogi, C. Canal, and E. Pimentel. Component Adaptation Through Flexible Subservicing. *Science of Computer Programming*, 63:39–56.

[41] A. Brogi, C. Canal, and E. Pimentel. On the Semantics of Software Adaptation. *Science of Computer Programming*, 61(2).

[42] A Brogi, C. Canal, and E. Pimentel. Behavioural types for service integration: achievements and challenges. In *CONCUR'2004 Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA'04)*, Electronic Notes in Theoretical Computer Science. Elsevier, 2007. (in print).

[43] J. Cámara, C. Canal, J. Cubo, and E. Pimentel. Dynamic contextual adaptation. In *CONCUR'2006 Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA'06)*, Electronic Notes in Theoretical Computer Science. Elsevier, 2007. (in print).

[44] J. Cámara, C. Canal, J. Cubo, and A. Vallecillo. Formalizing wsbpel business processes using process algebra. In *CONCUR'2005 Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA'05)*, volume 154 of *ENTCS*, pages 159—173. Elsevier, 2006.

[45] C. Canal, J. M. Murillo, and P. Poizat. Software Adaptation: an Introduction. *L'Objet*, 12(1):9—31, 2006.

[46] C. Canal, P. Poizat, and G. Salaün. Adaptation de composants logiciels: une approche automatisée basée sur des expressions règuliéres de vecteurs de synchronisation. In *Premiére Confèrence Francophone sur les Architectures Logicielles (CAL'2006)*, pages 59–70. Editions Hermes Sciences / Lavoisier, 2006.

[47] C. Canal, P. Poizat, and G. Salaün. Synchronizing Behavioural Mismatch in Software Composition. In *Formal Methods for Open Object-Based Distributed Systems (FMOODS'06)*, volume 4037 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2006.

[48] J. Cámara, C. Canal, and J. Cubo. Issues in the formalization of Web Service Orchestrations. In *ECOOP'2005 Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT'05)*, pages 81–88. TR 119/2005 LaMI UMR 8042, CNRS / Université d'Évry, 2006.

[49] J. Cámara, C. Canal, J. Cubo, and J.M. Murillo. An Aspect-Oriented Adaptation Framework for Dynamic Component Evolution. In *ECOOP'2006 Workshop on Reflection, AOP and Meta-Data for Software Evolution (RAM-SE'06)*. TR 13-2006, Fakultät für Informatik, Otto-von-Guericke Universität Magdeburg, 2006.

[50] J. Cḿara, C. Canal, J. Cubo, and J.M. Murillo. AOP and Dynamic Software Adaptation. In *JISDB'2006. Taller de Desarrollo de Software Orientado a Aspectos (DSOA'06)*, pages 27–34. Informe Técnico TR 24/06. Escuela Politécnica, Universidad de Extremadura, 2006.

[51] J. Cámara, C. Canal, J. Cubo, and E. Pimentel. Dynamic Adaptation Using Contextual Environments. In *ECOOP'2006 Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT'06)*, pages 35–42. Technical Report IBISC RR 2006-03, IBISC - FRE 2873, CNRS / Université d'Évry, 2006.

[52] P. de-la Cá mara, M. M. Gallardo, P. Merino, and D. Sanán. Model Checking Software with well-defined APIs: The Socket case. In T. Margaria & M. Massink, editor, *Tenth International Workshop on Formal Methods for Industrial Critical Systems (FMICS05).*, pages 17–26, September 2005.

[53] P. de-la Cámara, M. M. Gallardo, and P. Merino. Abstract Matching for Software Model Checking. In *13th International Workshop on Model Checking of Software (SPIN06)*, volume 3925 of *Lecture Notes in Computer Science*, pages 182–200, Vienna, Austria, April 2006.

[54] P. de-la Cámara, M. M. Gallardo, P. Merino, and D. Sanán. SocketMC: A tool to verify C code. In *Actas de la XIII Jornadas de Concurrencia y Sistemas Distribuidos.*

[55] Alberto Domínguez and P. Julián. Incompleteness of program transformers to inductively sequential term rewriting systems. In F. López-Fraguas, editor, *Proc. of V Jornadas sobre Programación y Lenguajes, PROLE'2005, Granada, Spain, September 14-16*, pages 249–258. University of Granada, 2005.

[56] F. Durán, S. Lucas, C. Marché, J. Meseguer, and X. Urbain. Proving Operational Termination of Membership Equational Programs. *Higher-Order and Symbolic Computation*, 2007. to appear.

[57] S. Escobar, C. Meadows, and J. Meseguer. A Rewriting-Based Inference System for the NRL Protocol Analyzer: Grammar Generation. In *Proceedings of 3rd ACM Workshop on Formal Methods in Security Engineering: From Specifications to Code (FMSE'05)*, pages 1–12. ACM Press, 2005.

[58] S. Escobar, J. Meseguer, and P. Thati. Narrowing and rewriting logic: from foundations to applications. In F. López-Fraguas, editor, *Proc. of the 15th Int'l Workshop on Functional and (Constraint) Logic Programming WFLP'06*, volume to appear of *Electronic Notes in Theoretical Computer Science*. Elsevier Sciences Publisher, 2006.

[59] Santiago Escobar, Catherine Meadows, and José Meseguer. Equational cryptographic reasoning in the Maude-NRL Protocol Analyzer. In *Proc. of the First International Workshop on Security and Rewriting Techniques (SecReT 2006)*, Electronic Notes in Theoretical Computer Science. Elsevier Sciences Publisher, 2006. To appear.

[60] Santiago Escobar, Catherine Meadows, and José Meseguer. The maude-nrl protocol analyzer: A rewriting-based inference system for equational cryptographic reasoning. In *Proc. of Midwest Security Workshop (MSW'2006), Urbana, IL, USA*. University of Illinois at Urbana-Champaign, 2006.

[61] Santiago Escobar, Catherine Meadows, and José Meseguer. The maude-nrl protocol analyzer: A rewriting-based inference system for equational cryptographic reasoning. In *Actas de las VI Jornadas de Programación y Lenguajes, PROLE'06*. Universidad Politécnica de Cataluña, 2006.

[62] Santiago Escobar, Catherine Meadows, and José Meseguer. A rewriting-based inference system for the NRL Protocol Analyzer and its meta-logical properties. *Theoretical Computer Science*, 367(1-2):162–202, 2006.

[63] S. Estévez-Martín, A. Fernández, T. Hortalá-González, M. Rodríguez-Artalejo, and R. del Vado-Vírseda. A Fully Sound Goal Solving Calculus for the Cooperation of Solvers in the CFLP Scheme. *Electronics Notes in Theorical Computer Science*, 2007. In Press.

[64] S. Estévez-Martín, A. Fernández, T. Hortalá-González, M. Rodríguez-Artalejo, F. Sáenz-Pérez, and R. del Vado-Vírseda. A Proposal for the Cooperation of Solvers in Constraint Functional Logic Programming. *Electronics Notes in Theorical Computer Science*, 2007. In Press.

[65] S. Estévez-Martín, A. Fernández, and F. Sáenz-Pérez. Implementing TOY, a Constraint Functional Logic Programming with Solver Cooperation. Research Report LCC ITI 06-8, Universidad de Málaga, November 2006.

[66] S. Estévez-Martín, A. J. Fernández, T. Hortalá-González, M. Rodríguez-Artalejo, and R. del Vado-Vírseda. A Fully Sound Goal Solving Calculus for the Cooperation of Solvers in the CFLP Scheme. In Francisco J. López-Fraguas, editor, *WFLP'06 - 15th Workshop on Functional and (Constraint) Logic Programming*, pages 211–227, Madrid, Spain, 2006.

[67] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. A survey of (pseudo-distance) functions for structured-data. In *III Taller de Minería de Datos y Aprendizaje (TAMIDA 2005)*, pages 233–242, 2005.

[68] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Distance Based Generalisation. In *Proceedings of 15th International Conference on Inductive Logic Programming, ILP 2005*, volume 3625 of *Lecture Notes in Computer Science*. Springer Verlag, 2005. Best Student Paper ILP 2005.

[69] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Web Categorisation Using Distance-Based Decision Trees. In M. Alpuente, S. Escobar, and M. Falaschi, editors, *Proceedings of 1st International Workshop on Automated Specification and Verification of Web Sites, WWV'05*, pages 77–82, 2005.

[70] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Classifying Web Sites Using Distance-Based Decision Trees. *Electronic Notes in Theoretical Computer Science*, 157(2), 2006.

[71] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Distance-based generalisation operator for graphs. In *Proceedings of International Workshop on Mining and Learning with Graphs*, pages 133–140, 2006.

[72] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. On the relationship between distance and generalisation. Technical report, DSIC, UPV, 2006. http://www.dsic.upv.es/∼flip/#Papers.

[73] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. On the specificity of distance-based generalisation operators. In *Proceedings of International Workshop on Inductive Logic Programming*, 2006.

[74] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Similarity Functions for Structured Data. An application to decision trees. *Inteligencia Artificial. Revista Iberoamericana de IA*, 29:109–121, 2006.

[75] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Some results about generalisation of graphs embedded in metric spaces. Technical report, DSIC, UPV, 2006. http://www.dsic.upv.es/∼flip/#Papers.

[76] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Minimal distance-based generalisation operators for first-order objects. *Accepted for publication in Lecture Notes in Artificial Intelligence*, 2007.

[77] Antonio J. Fernández. Programación declarativa con restricciones. *Revista Iberoamericana de Inteligencia Artificial*, 9(27):1137–3601, 2005.

[78] Antonio J. Fernández. Un enfoque genérico y cooperativo para la resolución de restricciones de intervalo. *Revista Iberoamericana de Inteligencia Artificial*, 9(27):125–128, 2005.

[79] Antonio J. Fernández and Patricia M. Hill. A Chaotic Iteration View for Interval Constraint Solving over Lattices. In Francisco J. López-Fraguas, editor, *PROLE'05 - V Jornadas sobre Programación y Lenguajes (Simposio 10 del I Congreso Español de Informática (CEDI'05))*, pages 43–52, Granada, España, 2005. Thomson.

[80] Antonio J. Fernández and Patricia M. Hill. An Interval Constraint Branching Scheme for Lattice Domains . In Francisco J. López-Fraguas, editor, *PROLE'05 - V Jornadas sobre Programación y Lenguajes (Simposio 10 del I Congreso Español de Informática (CEDI'05))*, pages 33–42, Granada, España, 2005. Thomson.

[81] Antonio J. Fernández and Patricia M. Hill. An interval constraint branching scheme for lattice domains. *Journal of Universal Computer Science*, 12(11):1466–1499, 2006.

[82] Antonio J. Fernández, Maria Teresa Hortalá-González, and Fernando Sáenz-Pérez. Programming with toy(fd). In *CP*, pages 878–878, 2005.

[83] Antonio J. Fernández, Maria Teresa Hortalá-González, and Fernando Sáenz-Pérez. Solving fd constraints in toy(fd). In Carmen Gervet Francisco Azevedo and Enrico Pontelli, editors, *First International Workshop on Constraint Programming Beyond Finite Integer Domains (as part of CP'05)*, pages 1–15, Sitges, Spain, 2005.

[84] C. Ferri, P. Flach, J. Hernández-Orallo, and A. Senad. Modifying ROC Curves to Incorporate Predicted Probabilities. In *Proceedings of 2nd International Workshop on ROC Analysis in Machine Learning*, pages 33–40, 2005.

[85] M. M. Gallardo, C. Joubert, and P. Merino. Implementing Influence Analysis using Parameterised Boolean Equation Systems. In *Proc. of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, 2005.

[86] M. M. Gallardo, P. Merino, and D. Sanán. Towards Model Checking C Code with OPEN/CAESAR. In *Fourth International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems (MSVVEIS'06)*, Paphos, Cyprus, May 2006.

[87] M. M. Gallardo, J. Martínez, and P. Merino. Model Checking Active Network with SPIN. *Computer Communication*, 28:609–622, 2005.

[88] M. M. Gallardo, J. Martínez, P. Merino, P. Nú nez, and E. Pimentel. PiXL: Applying XML Standards to Support the Integration of Analysis Tools for Protocols. *Science of Computer Programming Journal (in press),doi:10.1016/j.scico.2006.08.006.*

[89] M.M. Gallardo, J. Martínez, P. Merino, P. Nú nez, and E. Pimentel. PiXL: Applying XML Standards to Support the Integration of Analysis Tools. In *Fourth International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems (MSVVEIS'06)*, Paphos, Cyprus, May 2006.

[90] B. Gramlich and S. Lucas. Generalizing newman's lemma for left-linear rewrite systems. In Frank Pfenning, editor, *Proc. of the 17th International Conference on Rewriting Techniques and Applications (RTA 2006)*, volume 4098 of *Lecture Notes in Computer Science*, pages 66–80. Springer-Verlag, Berlin, 2006.

[91] G.J. Holzmann. The model checker spin. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.

[92] P. Julián, G. Moreno, and J. Penabad. On Fuzzy Unfolding. A Multi-Adjoint Approach. *Fuzzy Sets and Systems, Elsevier*, 154:16–33, 2005.

[93] P. Julián, G. Moreno, and J. Penabad. Operational/Interpretive Unfolding of Multi-adjoint Logic Programs. In F. López-Fraguas, editor, *Proc. of V Jornadas sobre Programación y Lenguajes, PROLE'2005, Granada, Spain, September 14-16*, pages 239–248. University of Granada, 2005.

[94] P. Julián, G. Moreno, and J. Penabad. Unfolding-based Improvements on Fuzzy Logic Programs. In Salvador Lucas, editor, *Electronic Notes in Theoretical Computer Science*, volume 137, pages 69–103. Elsevier, 2005.

[95] P. Julián, G. Moreno, and J. Penabad. Efficient Reductants Calculi using Partial Evaluation Techniques with Thresholding. In P. Lucio, editor, *Actas de las VI Jornadas sobre Programación y Lenguajes, PROLE'2006, Sitges, Spain, Octubre 10-12*, pages 275–289. Universidad Politécnica de Barcelona, 2006.

[96] P. Julián, G. Moreno, and J. Penabad. Evaluación Parcial de Programas Lógicos Multi-adjuntos y Aplicaciones. In A. Fernández, editor, *Proc. of Campus Multidisciplinar en Percepción e Inteligencia, CMPI-2006, Albacete, Spain, July 10-14*, pages 712–724. Universidad de Castilla-La Mancha, 2006. An extended version has been submitted to the journal: *Fuzzy Sets and Systems.*

[97] P. Julián, G. Moreno, and J. Penabad. Operational/Interpretive Unfolding of Multi-adjoint Logic Programs. *Journal of Universal Computer Science*, 12:1679–1699, 2006. Extended version of [93].

[98] P. Julián, G. Moreno, and J. Penabad. Efficient reductants calculi using partial evaluation techniques with thresholding. In P. Lucio, editor, *Electronic Notes in Theoretical Computer Science*, pages 69–103. Elsevier, 2007. Extended version of [95].

[99] P. Julián-Iranzo and C. Rubio-Manzano. Introducing fuzzy unification into the warren abstract machine. In K. Sirlantzis, editor, *In Proc. of the 6th International Conference on Recent Advances in Soft Computing, RASC'2006. Canterbury, UK, July 10-12*, pages 36–41. University of Kent, 2006.

[100] P. Julián-Iranzo and C. Rubio-Manzano. A wam implementation for flexible query answering. In A.P. del Pobil, editor, *In Proc. of the 10th IASTED International Conference on Artificial Intelligence and Soft Computing (ASC 2006), August 28-30, 2006, Palma de Mallorca*, pages 262–267. ACTA Press, 2006.

[101] Pablo López, Frank Pfenning, Jeff Polakow, and Kevin Watkins. Monadic concurrent linear logic programming. In *PPDP*, pages 35–46, Lisbon, Portugal, jul 2005. ACM Press.

[102] Pablo López and Jeff Polakow. Implementing efficient resource management for linear logic programming. In *LPAR*, pages 528–543, 2005.

[103] S. Lucas. A note on completeness of conditional context-sensitive rewriting. In R. di Cosmo and Y. Toyama, editors, *5th International Workshop on Reduction Strategies in Rewriting and Programming, WRS'05*, pages 3–15, Nara, Japan, 2005. University of Tokyo.

[104] S. Lucas. Polynomials over the reals in proofs of termination: from theory to practice. *RAIRO Theoretical Informatics and Applications*, 39:547–586, 2005.

[105] S. Lucas. Report on the use of polynomials with real and rational coefficients in proofs of termination. In F.J. López-Fraguas, editor, *Proc. of the Quintas Jornadas sobre Programación y Lenguajes, PROLE'05, 2005*, pages 105–108, Granada, Spain, 2005. Thomson-Paraninfo, Madrid.

[106] S. Lucas. On the relative power of polynomials with real, rational, and integer coefficients in proofs of termination of rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 17:49–73, March 2006.

[107] S. Lucas. Proving termination of context-sensitive rewriting by transformation. *Information and Computation*, 204(12):1782–1846, 2006.

[108] S. Lucas. Rewriting-based navigation of web sites: Looking for models and logics. *Electronic Notes in Theoretical Computer Science*, 157:79–85, 2006.

[109] S. Lucas, C. Marché, and J. Meseguer. Operational termination of conditional term rewriting systems. *Information Processing Letters*, 95:446–453, 2005.

[110] S. Lucas and J. Meseguer. Termination of fair computations in term rewriting. In G. Sutcliff and A. Voronkov, editors, *12th International Conference on Logic for Programming, Artificial intelligence and Reasoning, LPAR'05*, volume 3835 of *Lecture Notes in Artificial Intelligence*, pages 184–198, Montego Bay, Jamaica, 2005. Springer-Verlag, Berlin.

[111] S. Escobar M. Alpuente and M. Falaschi (eds.). *Automated Specification and Verification of Web Systems.* IEEE Computer Society Press, 2005.

[112] S. Escobar M. Alpuente and M. Falaschi (eds.). *Automated Specification and Verification of Web Sites.* Elsevier Science, Electronic Notes in Theoretical Computer Science 157(2), 2006.

[113] M. Montenegro, R. Pe na, and C. Segura. An Inference Algorithm for Guaranteeing Safe Destruction. In *Proceedings of the Eighth Symposium on Trends in Functional Programming, TFP'07*, page to appear, 2007.

[114] G. Moreno. Building a Fuzzy Transformation System. In J. Wiedermann, G. Tel, J. Pokorný, M. Bieliková, and J. Stuller, editors, *Proc. of the 32nd Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM'2006. Merin, Czech Republic, January 21-27*, pages 409–418. Springer LNCS 3831, 2006.

[115] G. Moreno and V. Pascual. Functional Logic Programming with Similarity. In F. Lopez-Fraguas, editor, *Proc. of V Jornadas sobre Programación y Lenguajes, PROLE'2005. Granada, Spain, September 14-16*, pages 121–126. University of Granada, 2005.

[116] G. Moreno and V. Pascual. Formal Properties of Needed Narrowing with Similarity Relations. In P. Lucio, editor, *Actas de las VI Jornadas sobre Programación y Lenguajes, PROLE'2006, Sitges, Spain, Octubre 10-12*, pages 114–128. Universidad Politécnica de Barcelona, 2006.

[117] G. Moreno and V. Pascual. Programando con Igualdad Similar Estricta. In A. Fernández, editor, *Proc. of Campus Multidisciplinar en Percepción e Inteligencia, CMPI-2006, Albacete, Spain, July 10-14*, pages 712–724. Universidad de Castilla-La Mancha, 2006. An extended version has been submitted to the journal: *Applied Soft Computing.*

[118] G. Moreno and V. Pascual. Programming with Fuzzy Logic and Mathematical Functions. In I. Bloch, A. Petrosino, and A. Tettamanzi, editors, *Proc. of the 6th International Conference on Fuzzy Logic and Applications, WILF'2005. Crema, Italy, September 15-17, 2005*, pages 89–98. Springer LNAI 3849, 2006. An extended version has been submitted to the journal: *Fuzzy Sets and Systems.*

[119] G. Moreno and V. Pascual. Soft Computing with Strict Similar Equality. In K. Sirlantzis, editor, *6th International Conference on Recent Advances in Soft Computing, RASC'2006. Canterbury, UK, July 10-12*, pages 24–29. University of Kent, 2006.

[120] G. Moreno and V. Pascual. Formal Properties of Needed Narrowing with Similarity Relations. In P. Lucio, editor, *Electronic Notes in Theoretical Computer Science*, page 15. Elsevier, 2007. Extended version of [116].

[121] J. Martínez. *Un enfoque basado en estándares para la integración de técnicas y herramientas de Ingeniería de Protocolos.* PhD thesis, University of Málaga, 2005.

[122] R. Peña and C. Segura. A First-Order Functional Language for Reasoning about Heap Consumption. In *Actas de las IV Jornadas de Programación y Lenguajes, PROLE'04*, pages 153–162, 2004.

[123] R. Peña and C. Segura. A First-Order Functional Language for Reasoning about Heap Consumption. In *Proceedings of the 16th International Workshop on Implementation of Functional Languages, IFL'04. Technical Report 0408, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität zu Kiel*, pages 64–80, 2004.

[124] R. Peña and C. Segura. A Sharing Analysis Guaranteeing Safe Destruction in a First-Order Functional Language. In *Actas de las V Jornadas de Programación y Lenguajes, PROLE'05*, pages 3–12. Thomson-Paraninfo, 2005.

[125] R. Peña and C. Segura. Formally deriving a compiler for SAFE. In *Actas de las VI Jornadas de Programación y Lenguajes, PROLE'06*, pages 65–76. CIMNE, 2006.

[126] R. Peña and C. Segura. Formally Deriving a Compiler for SAFE. In *Proceedings of the 18th International Symposium on Implementation and Application of Functional Languages, IFL'06. Technical Report 2006-S01.Eotvos Loránd University*, pages 429–446, 2006.

[127] R. Peña, C. Segura, and M. Montenegro. A Sharing Analysis for SAFE. In *Proceedings of the Seventh Symposium on Trends in Functional Programming, TFP'06*, pages 205–221, 2006.

[128] R. Peña, C. Segura, and M. Montenegro. A Sharing Analysis for SAFE. In *Trends in Functional Programming (Volume 7) Selected Papers of the Seventh Symposium on Trends in Functional Programming, TFP'06. 20 pages. To appear.* Intellect, 2007.

[129] M.A. Pérez Toledano, A. Navasa, C. Canal, and J.M. Murillo. Desarrollo de Sistemas Basados en Componentes Utilizando Diagramas de Secuencia. In *VIII Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software (IDEAS'05)*, pages 229–240. En Actas, 2005.

[130] M.A. Pérez Toledano, A. Navasa, J.M. Murillo, and C. Canal. Síntesis de patrones de interacción a partir de diagramas de secuencia de UML. In *X Jornadas en Ingeniería del Software y Bases de Datos (JISBD'2005)*, pages 83–90. Thompson, 2005.

[131] M.A. Pérez Toledano, A. Navasa, J.M. Murillo, and C. Canal. Evolución de sistemas orientados a aspectos utilizando patrones de interacción. In *XI Jornadas en Ingeniería del Software y Bases de Datos (JISBD'2006)*, pages 514–519. CINME, 2006.

[132] M.A. Pérez Toledano, A. Navasa, J.M. Murillo, and C. Canal. Making Aspect-Oriented System Evolution Safer. In *ECOOP'2006 Workshop on Reflection, AOP and Meta-Data for Software Evolution (RAM-SE'06)*, pages 23–34. TR 13-2006, Fakultät für Informatik, Otto-von-Guericke Universität Magdeburg, 2006.