Ontologies in domain specific languages : a systematic literature review

Sutii, A.M.; Verhoeff, T.; van den Brand, M.G.J.

Published: 01/01/2014

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Technische Universiteit Eindhoven
Department of Mathematics and Computer Science

Ontologies in domain specific languages – A systematic literature review

Ana-Maria Şutîi, Tom Verhoeff and M.G.J. van den Brand

14/09

ISSN 0926-4515

editors: prof.dr. P.M.E. De Bra
prof.dr.ir. J.J. van Wijk

Reports are available at:
http://library.tue.nl/catalog/TUEPublication.csp?Language=dut&Type=ComputerScienceReports&Sort=Author&level=1 and
http://library.tue.nl/catalog/TUEPublication.csp?Language=dut&Type=ComputerScienceReports&Sort=Year&Level=1

Computer Science Reports 14-09
Eindhoven, November 2014

Ontologies in domain specific languages - A systematic literature review

Ana-Maria Ștefănescu, Tom Verhoeff, M.G.J. van den Brand

October 21, 2014

Abstract

The systematic literature review conducted in this paper explores the current techniques employed to leverage the development of DSLs using ontologies. Similarities and differences between ontologies and DSLs, techniques to combine DSLs with ontologies, the rationale of these techniques and challenges in the DSL approaches addressed by the used techniques have been investigated. Details about these topics have been provided for each relevant research paper that we were able to investigate in the limited amount of time of one month. At the same time, a synthesis describing the main trends in all the topics mentioned above has been done.

1 Introduction

This work is motivated by the fact that ontologies, as knowledge representation systems, can be used in analysing the domain of a DSL [51]. From this, the question whether ontologies can be reused in other phases of building DSLs follows up. This further lead to a search through the literature that we reported in this paper.

The purpose of this paper is to conduct a systematic literature review on different techniques that exist on leveraging ontologies in domain-specific languages (DSL). The rationale for using these techniques and the challenges that the usage of the ontologies address in the DSL approach are also considered. At the same time, similarities and differences between ontologies and DSLs are investigated as a way of offering context to the other topics.

A paragraph in Richard's Hamming Turing award lecture refering to developing solutions from existing ones represents a valid argument for conducting literature reviews as well: "Indeed, one of my major complaints about the computer field is that whereas Newton could say, 'If I have seen a little farther than others, it is because I have stood on the shoulders of giants,' I am forced to say, 'Today we stand on each other's feet.' Perhaps the central problem we face in all of computer science is how we are to get to the situation where we build on top of the work of others rather than redoing so much of it in a trivially different way. Science is supposed to be cumulative, not almost endless duplication of the same kind of things." [25].

Context There are initiatives, like those of the TWOMDE workshop [39], to investigate how can ontologies and the reasoning capabilities supported by them be used in Model Driven Engineering. One of the areas in MDE where ontologies could be valuable is the area of domain specific languages. Our focus is on ontologies and domain specific languages.

The development of domain specific languages and ontologies has taken place separately. Domain specific languages have been the focus of designers of applications in engineering fields, while ontologies have been more the focus of Artificial Intelligence and Knowledge Engineering [35]. Our goal is to identify ways to benefit from ontologies in DSLs, in spite of their different origins.

Structure of the paper Section 2 introduces terms used in the explanation of the investigated papers and Section 3 gives an overview of the research questions, search process, inclusion and exclusion criteria for papers and quality assessment attributes of the papers. Then, Section 4 answers to the research questions in each paper individually. Section 5 gives

grades to the read papers based on the way they answer the research questions. Section 6 describes papers that did not make it into the final batch of accepted papers, but that still present interesting ideas on our subject. Then, Section 7 presents some statistics on the selected papers. Finally, Section 8 makes a synthesis of the information obtained from the research papers, Section 9 presents threats to the validity of the results obtained and Section 10 makes the final remarks.

2 Preliminary notes

In this section we concisely define ontologies and DSLs. Besides the definitions, there are multiple languages, frameworks and architectures related to the subject of ontologies or DSLs. We also give a short explanation to some notions that we encountered in the investigated papers. We have separated the notions as being part of two different technological spaces.

A technical (technological) space, as defined by Kurtev et al. [32], is a "working context with a set of associated concepts, body of knowledge, tools, required skills, and possibilities". A technological space has a user community around it that shares the knowledge, the literature and that even organizes conferences and workshops. The two technical spaces that we are using are the ontological technical space and the model driven architecture (MDA) technical space. Ontologies are part of the ontological technical space, while DSLs are part of the MDA technical space.

2.1 Ontologies

An ontology is, as defined by Gruber in 1993, an explicit specification of a shared conceptualization [21]. The definition was extended in 1998 by Struder et al. [50] into: "An ontology is a formal, explicit specification of a shared conceptualization". The 'shared' part in the definition refers to the fact that the knowledge represented by an ontology is understood and agreed upon by most of the experts in a domain. The 'conceptualization' part in the definition refers to the fact that the ontology represents an abstract, simplified view of the world described by the ontology. The formal, explicit part of the definition makes reference to the fact that a language is needed to describe the concepts in the domain.

There are two types of ontologies: domain ontologies and upper ontologies. Domain ontologies deal with real-world descriptors of business entities, while upper ontologies provide "meta-level" concepts for the domain ontologies.

Ontologies are expressed in formal languages based on logic, so semantic reasoners behind ontologies can also reason about the data besides representing it. Between the reasoning capabilities of a semantic reasoner, we mention consistency checking, transitive relations, value partitions, automated classification, inheritance or constraint checking [16].

Ontological technical space OWL [36] stands for Web Ontology Language and it is part of the ontological technical space. One can represent terms in an ontology and make interrelations between them. There are three increasingly expressive sublanguages within OWL: OWL Lite, OWL DL and OWL Full. OWL is used for information that not only needs to be presented to humans, but that also needs to be processed by applications. OWL is based on description logics, a formal language used to represent knowledge and reason about it. A knowledge representation system based on description logic is made of two components: the TBox and the ABox. The TBox introduces the terminology of an application domain and the ABox contains assertions about named individuals using terms of the vocabulary [7].

The Semantic Web Rule Language (SWRL) is a World Wide Web Consortium (W3C) proposed language that combines OWL DL or OWL Lite sublanguages with the Unary/Binary Datalog sublanguage of Rule Markup Language [27]. The combination is provided such that, besides logic, rules can also be expressed in the language.

Finally, SPARQL [43] is a query language for RDF [12] recommended by W3C.

2.2 DSLs

Domain specific languages do not have a clear definition in the literature. They are defined as "a computer programming language of limited expressiveness focused on a particular domain"

by Martin Fowler [19]. As examples of DSLs, we mention regular expression languages and SQL. To illustrate the difficulty in determining the fact that a certain programming language or not, we mention that there are SQL variants nowadays that are Turing complete [34]. That means that the limited expressiveness character does not apply to SQL anymore. But SQL still remains focused on a particular domain, that of relational databases, and we can write code in SQL in terms of database concepts and that makes it a DSL in out opinion.

There are multiple factors involved in the design of a good DSL: a syntax definition, a proper semantics, tooling and eventually methodology and documentation [4]. These factors have a different importance depending on the executability level of the DSL [37] and the type of the DSL (internal or external) [19].

Mernik et al. [37] give detailed information on when and how to develop domain-specific languages. They identified five stages in the DSL development process: decision, analysis, design, implementation and deployment. They also identified patterns for the first four stages of the DSL development process. For example, patterns that occur in the decision stage are the need for: a new notation, a domain-specific analysis, verification, optimization, parallelization and transformation, an automation task, a product line, a data structure representation, a data structure traversal, etc. The DSL development process is seen as a hard endeavour because of the domain and language development knowledge that it requires. The decision to develop a new domain specific language is thus not easy. On the other hand, the language workbenches are becoming more and more powerful and therefore, they decrease more and more the effort put into the creation of a new DSL. Language workbenches are tools that help developers in more efficiently defining and using DSLs [53]. They offer productive DSLs and APIs for the definition of languages and their IDEs.

There are open issues in the DSL community [20, 55], issues that make it difficult for the DSLs to be accepted and used in industry. The challenges that current DSL approaches face are:

- tooling related: debuggers, testing engines;
- interoperability with other languages;
- formal semantics;
- learning curve;
- domain analysis;
- integration of graphical and textual editing;
- scalability;
- DSL evolution.

There is, in general, a strong relation between DSLs and model-based engineering. Kurtev et al. [33] define DSLs as a set of coordinated models. That is why, even though some of the studies selected for the systematic literature review do not mention DSLs, but only discuss on the relation between DSLs and metamodeling, we took them into consideration.

MDA technical space In the MDA technical space, EMF (Eclipse Modelling Framework) is a modelling framework and code generation facility for building Java applications from model definitions [49]. EMF tries to bridge the gap between Java programmers and modellers. The model used to represent models in EMF is Ecore [49].

OCL (Object Constraint Language) is a formal language used to specify invariant conditions that must hold for the modelled system or queries over objects in the model [3]. OCL is aligned with UML [47] and MOF [1] (thus with Ecore).

The architecture in metamodeling proposed by the Object Management Group (OMG) has four layers. The M0 layer is the data layer, the M1 layer is the model layer, the M2 layer is the metamodel layer and the M3 layer is the metametamodeling layer (MOF). An object in M0 is an instance of a class in the M1 level, a class in M1 is said to be an instance of a meta-class in the M2 layer and so on [26]. In the document of MOF 2.0 [1], it is emphasized that the four-layered architecture is not rigid, one being able to have as many layers as it wishes as long as their number is greater or equal to two. The fundamental concept is to be able to navigate from an instance to its metaobject. The architectures with the leveled meta-layers are also named meta-pyramids.

One other notion, the Ontology Definition Metamodel (ODM) [2], is a specification for enabling the formal grounding for representation, management, interoperability and application of business semantics to the capabilities of MDA based software engineering. ODM permits the modelling of an ontology and the interoperability with other modeling languages (like UML). This is also part of the ontological technical space.

Finally, IQPL [8] is a graph query language for EMF models.

3 Method

The systematic literature review has been done according to the method proposed by Kitchenham et al. [31]. The article suggests guidelines on how to evaluate and interpret all available research related to particular research questions or topic areas. There are three phases involved in a systematic literature review: planning the review, conducting the review and reporting it.

3.1 Research questions

The research questions whose answers we look for in the read papers are listed next.

RQ1
What are the similarities and differences between ontologies and DSLs?

RQ2
What technique is used to apply ontology technologies in DSLs?

RQ3
Why did the authors choose to use ontologies in DSLs?

RQ4
What challenges faced by DSLs are solved/addressed by using ontologies?

3.2 Search process

We first searched for papers on ontologies and DSLs that answered all of the above questions on the SpringerLink digital libraries. The limited amount of time of one month that we had to perform the literature study was not enough to go through all the search results on SpringerLink. We have also examined all the references in the selected papers so that we have a greater chance of finding more relevant papers.

We have used the following strings to search the digital library: "ontology domain specific language", "ontology metamodeling" and "ontology model driven engineering". We did not use quotes in the search strings because we did not want the search to be too restrictive. Then, we have also looked at the references of the selected papers to find more relevant papers.

3.3 Inclusion and exclusion criteria

Articles that answer in detail research question number two were considered for the study. Research question number one is there to offer context to our investigation, while research questions number three and four are natural follow-up question for the second research question.

The articles' fitness was judged based on their title's, keywords', abstract's and conclusion's connection to research question number two. The first phase in selecting an article was to look at the title and keywords. The next step for the articles selected in phase one, was to read their abstract and their conclusions section. Then, finally, the entire article was read. Proceeding from one phase to the other was done by judging the content based on research question number two. The phases were conducted on one article per turn and we would proceed to a following article in the search results of SpringerLink only after completely finishing with the preceding article.

We considered papers that directly tackled the subject of using ontologies in DSLs approaches, but also those that tackle the subject of metamodeling and ontologies, metamodeling being strongly related to DSLs. SpringerLink includes both conference proceedings and journal papers.

We did not take into account papers that only tackle ontologies or only tackle DSLs. We also did not consider papers that tackle the subject of developing ontologies using model driven approaches.

3.4 Quality assessment

The quality assessment scores are giving the quality of the paper in what regards the level of details on which research questions are described in the paper. Because of the correlated research questions, the papers were expected to have high scores in general and that was indeed the case.

The criteria are based on six quality assessment questions:

QA1
Does the paper address RQ1 with sufficient level of detail?

QA2
Does the paper address RQ2 with sufficient level of detail?

QA3
Does the paper address RQ3 with sufficient level of detail?

QA4
Does the paper address RQ4 with sufficient level of detail?

The questions were scored as follows:

- QA1: Y(Yes), the authors explicitly give both similarities and differences between ontologies and DSLs (at least one of each); P(Partly), the authors explicitly give only similarities or only differences between ontologies and DSLs (at least one); N(No), the authors don't explicitly mention any of the similarities or differences between ontologies and DSLs.
- QA2: Y(Yes), the technique for applying ontology technologies in DSLs is clearly described; P(Partly), the technique is not described sufficiently; N(No), there is no mention of any technique of applying ontology technologies in DSLs.
- QA3: Y(Yes), the reason for using the ontologies is clearly stated; P(Partly), the reason is implicit; N(No), the reason cannot be deduced clearly.
- QA4: Y(Yes), the authors mention at least one addressed challenge; P(Partly), the challenge addressed is implicit; N(No), there is no challenge addressed.

  A "yes" scores one point, a "partly" scores half a point and a "no" scores a zero. This scoring model is taken from an example of Kitchenham et al. [31].

4 Results

At times, there were several papers published on the same technique and by the same authors (or partly the same authors). We have chosen a representative for that group and assessed that representative.

4.1 Search results

In this subsection we assign an id to the investigated papers and we also mention the papers that reside in the same group as the investigated papers. Papers in the same group describe the same subject at different level of details. The group of papers is not exhaustive because we did not do a systematic search for papers in the same group. Papers residing in a group were discovered during the normal search process of papers on ontologies used in DSLs. The results can be seen in Table 1.

4.2 Data collection

This subsection gives a small abstract of the studied research papers and answers to our research questions.

Paper	Author and citation	Papers describing the same technique
P1	Walter et al. [55]	[57], [42], [56], [48]
P2	Lortal et al. [35]	-
P3	Tairas et al. [51]	-
P4	Walter et al. [54]	[48]
P5	Bräuer et al. [11]	[10]
P6	Curé et al. [17]	-
P7	Guizzardi et el. [23]	[24], [22]
P8	Čeh et al. [13]	[15], [14]
P9	Roser et al. [46]	[45]
P10	Rahmani et al. [44]	-
P11	Izsó et al. [28]	-
P12	Parreiras et al. [40]	-
P13	Erofeev et al. [18]	-
P14	Kappel et al. [30]	-

Table 1: Groups of papers touching the same technique.

4.2.1 P1

Walter et al. [55] describe an ontology-based framework for domain-specific languages, framework that permits the definition of DSLs enriched with formal descriptions of classes. The main idea of the paper is that ontologies and the automated reasoning that they provide help in addressing major challenges faced by current DSL approaches.

RQ1 The authors point out that there is a mismatch on the underlying semantics of modelling between UML-based class modelling and OWL because the former one adopts the closed world assumption and OWL adopts the open world assumption by default.

RQ2 They integrate ontologies with DSLs at the metametalevel. They use KM3 [29] to define the general structure of the language, OWL2 [38] to define the semantics and OCL to define operations for calling the reasoning services. Reasoning services provide means to derive facts that are not explicitly stated in the model. To provide ontology reasoning, the DSL metamodel and domain model are transformed into a Description Logics knowledge base (TBox and ABox).

By using this technique, they provided a new technical space which allows implementing DSL metamodels with formal semantics, constraints and queries.

RQ3 Walter et al. [55] use ontologies in the development of a domain-specific languages' framework because some of the main challenges of developing DSLs were motivation for developing ontologies, like interoperability and formal semantics. This comes to the benefit of the DSL designers and DSL users. The DSL designers profit from constraint definition, formal representations and expressive languages. The DSL users profit from progressive verification (verification of incomplete models), reasoning explanation, assisted programming (suggesting concepts to the user and explaining inferences) and different ways of describing constructs.

RQ4 The challenges that are partially solved by Walter et al.'s approach are those related to formal semantics (by constraint definition), learning curve (by progressive verification, suggestions of suitable domain concepts to be used, reasoning explanation and syntactic sugar) and tooling (by progressive verfication and reasoning explanation).

4.2.2 P2

Lortal et al. [35] use robotic ontologies to develop robotic DSLs. The main idea of the paper is to reuse ready-made information from an ontology to ease the building of DSLs.

RQ1 The authors note that ontologies and DSLs have the same building phases (except for the fact that the implementation phase is not as emphasized for ontologies as much as

for DSLs). Moreover, their development faces the same problems. At the same time, ontologies and DSLs both structure data and information for application use.

On the other hand, models in ontologies are used for different applications than models in DSLs (former ones are used in artificial intelligence and web application mostly and the later ones are used in code generation, systems modelling, verification, simulation etc.). At the same time, different technologies and tools are used for each.

RQ2 Ontologies were used during the requirements specification phase of the DSL by gathering requirements when inspecting the ontologies and during the design of the domain models of the DSL by using specific mappings between ontologies (OWL) and domain models (UML class diagrams). For example, concepts in ontologies are mapped to classes in domain models.

Thus, by extracting concepts that are specific to the domain from an ontology, the DSL corresponds to domain concepts defined in the ontology.

RQ3 The rationale of using ontologies in DSLs consists in knowledge reuse. Lortal et al. [35] represent the domain by inferring information from a knowledge base and capturing the experts' knowledge.

RQ4 The challenges in the DSL development process solved by this technique are those related to easing domain analysis.

4.2.3 P3

Tairas et al. [51] use ontologies for the phase of domain analysis in a DSL.

RQ1 The authors note that both ontologies and DSLs contain the domain model vocabulary and the interdependencies between the concepts in the domain.

RQ2 Starting from an existing ontology and based on its structure, a conceptual class diagram can be designed manually in an informal way. Thus, the information in the ontology assists in designing the conceptual class diagram. Afterwards, the conceptual class diagram is being manually transformed into an initial context-free grammar following a predefined collection of transformation rules.

At the same time, the instances of the ontology can be used to capture the commonalities and variabilities in a DSL.

RQ3 Tairas et al. [51] used ontologies in the development of DSLs because the domain analysis part in a DSL is not researched enough and ontologies can provide a structured mechanism for domain analysis.

RQ4 The challenges in the DSL development process solved by this technique are those related to domain analysis.

4.2.4 P4

Walter et al. [54] combine ontologies with two other DSLs at the metamodel level in order to be able to express semantical constraints.

RQ1 The authors report on the equivalences that exist in the ontological technology space and metamodeling technology space. For example, 'cardinality' in the ontological technological space is equivalent to 'multiplicity' in the metamodeling technological space.

RQ2 The technique used by the authors in this study to combine ontologies and DSLs is to unify them at the metamodel level. In their case study, they do manual transformations to create an integrated metamodel consisting of two DSLs and OWL. The integration is done without any loss of information from any of the three metamodels. Then, they project the integrated domain metamodel to a complete ontology for reasoning. This step is also performed manually.

RQ3 They have done the integration at the metamodel level in order to be able to define domain models and semantics for model elements simultaneously. Ontologies are also attractive because they provides the means for reasoning, querying and constraint checking.

RQ4 The challenges in the DSL approaches addressed by this technique are the specifications of formal semantics for the DSLs.

4.2.5 P5

Bräuer et al. [11] create an upper ontology (see Section 1) software for software models that permits integrity and consistency checking across the boundaries of individual models. Integrity refers to conditions that need to hold in order for the software models to be in a valid state.

RQ1 The authors emphasize, as a difference, the closed-world assumption of models in MDA and the open-world assumption of ontologies. At the same time, the closed-world assumption in models is closely related to nonmonotonic reasoning, while ontologies are built on monotonic formalisms.

RQ2 The authors created an upper ontology so that they can integrate different domain-specific modelling languages based on the upper ontology. For this, one needs to establish a binding between the domain-specific modelling language and the concepts and relationships in the upper ontology. The upper ontology behaves like a semantic connector.

The presented method permits integrity and consistency checking for domain models.

RQ3 Ontologies were used in the development of DSLs because the model-driven engineering approach advocates the modelling of a system from different viewpoints, that raising the problem of interoperability between the DSLs used to express these views. That also implies consistency checking between individual models and automatic generation of model transformations. All these reasons lead to the decision to use ontologies in the development of DSLs.

RQ4 The technique addresses the challenge of semantic relationships and interoperability between DSLs.

4.2.6 P6

Curé et al. [17] focus on a domain specific language based on an ontology. The language is called Ocelet and it is used to model dynamic landscaping.

RQ1 The authors note that the steps needed to describe an ontology are the same ones involved in the development of a DSL: identify the domain problem, collect domain knowledge and establish domain vocabulary and semantics. This leads to the possibility of establishing relations between ontology concepts and DSL concepts. This point has been tackled also in paper $P2$.

RQ2 The usage of ontologies in the DSL development process takes place in the first step. The process starts with the development of OWL ontologies that are being verified for consistency with reasoners. Then, the ontologies are automatically transformed into Ocelet models. The transformation process can occur in the other direction too, as the transformations are bijections. The Ocelet models are then merged into a global Ocelet model in a manual fashion. This model is transformed into the structures of a reasoner and a consistency checking is done on the global model. The inconsistencies discovered need to be solved by hand as there is usually more than one solution possible.

RQ3 The ontologies were used in order to do local and global consistency checking on the Ocelet models. All started from the fact that models in the Ocelet framework would be developed by different persons. Reasoning is thus a prerequisite for the framework.

RQ4 The technique used in this study addresses the challenges of formal semantics in DSLs.

4.2.7 P7

Guizzardi et al. [23] present how can ontologies be used to evaluate and design a domain specific visual modelling language.

RQ1 This question has not been approached in the article.

RQ2 The quality of a domain-specific modelling language with respect to a domain ontology is guaranteed if an isomorphism between the ontology and the domain-specific modelling language can be established. This isomorphism is guaranteed if the mapping between a domain ontology and the domain language's metamodel has a number of properties:

soundness, completeness, laconicity and lucidity [24]. These properties are verified manually in order to establish the quality of the domain language's metamodel regarding the ontology.

Using the domain ontology, one can, besides evaluating the quality of the domain-specific modelling language with respect to the domain ontology, also see it as a starting point for the design of a new modeling language in the given domain that is isomorphic to the ontology.

RQ3 Using the domain ontology and keeping some mapping properties between the ontology and the domain modelling language, the quality of the domain specific modelling language can be guaranteed with respect to the ontology. The quality of the domain specific modeling language represents the degree to which it ensures the proper representation of the subject domain.

RQ4 The technique addresses the challenge of formal semantics in DSLs.

4.2.8 P8

Čeh et al. present a framework (Ontology2DSL) where a DSL grammar is automatically created from an ontology and some transformation patterns.

RQ1 This question has not been approached in the article.

RQ2 The technique described in this paper starts from an ontology that is transformed into an appropriate internal data structure of Ontology2DSL. On the data structure, a series of transformation patterns are applied and a grammar for a DSL is obtained. The irregularities found in the resulted grammar are solved either in the ontology, or in the transformation patterns. The transformation patterns applied on the data structure include basic concept transformations (class hierarchy transformed into production alternatives), generalization abstraction and so on.

In this technique, the ontology based domain analysis replaces classic domain analysis.

RQ3 The reason for choosing this technique is the fact that ontologies come with reasoning and querying, which allows the validation of the ontology. A valid ontology reduces errors during DSL development. The semantics in an ontology also help in establishing the semantics of the DSL.

RQ4 The challenges in the DSL development process that this technique addresses are domain analysis and formal semantics.

4.2.9 P9

Using an upper ontology, Roser et al. [46] describe a framework for the automatic generation and evolution of model transformations.

RQ1 This question has not been approached in the article.

RQ2 The technique starts from an upper ontology. Bindings to the required metamodels are established with the upper ontology. The binding represents a semantic mapping of the metamodels to their semantic concepts in the ontology. In order to perform model transformations, an initial model transformation needs to be provided (or it can also be automatically generated). The level of automation depends on the differences between the employed metamodels. The automated process of model transformation generation is based on the framework establishing several substitution proposals and choosing the one that scores best (based on some heuristics). The framework supports the evolution and reuse of existing mappings too.

RQ3 The reason for choosing to integrate ontologies in model transformations is because of their reasoning capabilities that can help in automating the process of model transformation. This is in relation to the need to exchange information between organizations, that boils down to interoperability support in modeling applications.

RQ4 The technique presented in this research paper addresses the challenges of formal semantics and language interoperability in DSLs.

4.2.10 P10

Rahmani et al. [44] describe a transformation from OWL to Ecore and OCL that can be adjusted depending on the level on which we want the ontology to be reflected in the Ecore model.

RQ1 The comparison is made between Ecore and OWL, so the comparison is more specific then the comparison between ontologies and DSLs in general, but it is still relevant. The differences between the two are the following:

- The open world assumption of OWL and the closed world assumption of Ecore. This point has been tackled in other papers too.
- The high web compliance of OWL and the low web compliance of Ecore. Web compliance refers to the degree to which a system is suitable to publish and exchange knowledge on the web. This leads to different identification mechanisms in OWL and Ecore.
- The unique name assumption in Ecore, that does not hold in OWL.
- Properties in OWL are first-class citizens, in contrast to their counterparts in Ecore, the references. This means that properties in OWL can be applied between several different pairs of classes, can be put in hierarchies and can be constrained, in contrast with references in Ecore.
- The expressiveness of OWL and Ecore does not overlap. This means that there are modeling constructs that can appear in OWL or Ecore, but not in the other.
- Ecore is built on four layers of modeling, while OWL is built only on two layers of modeling, the TBox and the ABox.
- The intuition of cardinality is different for OWL and Ecore: 0..* can implicitly mean 0..1 for ontologists.
- OWL benefits from an inference engine, while Ecore does not.

RQ2 The authors give transformations for every OWL modeling primitive to Ecore and OCL. Some transformations are straightforward, like the OWL classes that can be generally transformed directly to Ecore classes. Other transformations are more complex, like the transformation of the property hierarchy of OWL into Ecore. The materialization of all inferred implicit knowledge of the reasoner on property relations needs to be added in Ecore. This is done using OCL constraints on the corresponding classes.

The transformation to Ecore and OCL can be done in a way that preserves the entire ontology or only partly, depending on the types of transformations that the user chooses to perform and their number. For example, a user may choose to skip the transformation of property hierarchy. The transformations are adjustable at both meta and instance level.

RQ3 The rationale behind the technique was to leverage existing knowledge captured in ontologies to Ecore models. Thus, software engineers do not have to manually remodel models written in OWL.

RQ4 The implied challenges in DSL development that the technique addresses are domain analysis challenges.

4.2.11 P11

Izsó et al.[28] are describing the creation of domain specific modeling environments starting from ontologies. As a result, a validation of both metamodel-level and instance level models is done.

RQ1 The differences between ontologies and DSLs start from their different purposes. The ontologies are used to capture the knowledge and the requirements in a domain in very early phases of the design and the ontology reasoners are made for meta-level validation. On the other hand, domain-specific language tools are made for increasing the productivity of the developers and they use instance level validators like, for example, OCL.

RQ2 The technique consists in transforming OWL2 enriched with SWRL into EMF enriched with IQPL (see Section 2). The process starts with an ontology where formal textual requirements are captured and the meta-level consistency is checked. Then, a first transformation is done between the OWL2 ontology to the EMF metamodel, followed by a transformation of more complex OWL2 axioms into graph patterns. Finally, the SWRL rules are mapped into graph patterns. The transformations occur only at the meta-level (from TBox). EMF instances can be validated using EMF-IncQuery.

RQ3 The reason for the entire process was to be able to combine the benefits from both the ontology world and the domain specific modelling world. Domain requirements captured in the ontologies drive the development of domain specific modeling environments. At the same time, ontologies are used for the consistency checking at the meta-level, while EMF and IncQuery are used to validate instances.

RQ4 The challenges in DSL development that are addressed are those related to domain analysis, and implicitly those related to formal semantics.

4.2.12 P12

Parreiras et al. [40] describe a method to integrate OWL and UML at the metamodel level. The strengths and weaknesses of the two modelling approaches complement each other and they are appropriate for specifying different aspects of the software systems. The approach is implemented in a tool called TwoUse.

RQ1 The similarities and differences are not made in general for DSLs and ontologies, but they are still relevant.

OWL ontologies and UML class-based modelling are similar with respect to classes, associations, properties, packages, types, generalization and instances.

On the other hand, there are also differences between the two modelling approaches. UML class-based modelling is able to capture only static specification of specialization and generalization of classes and relationships, while OWL can do this dynamically as well. At the same time, UML provides mechanisms to define dynamic behaviour, while OWL does not.

RQ2 TwoUse uses UML profiles as concrete syntax and the profiles offer the possibility to design both UML models and OWL ontologies. These UML profiles are then transformed to TwoUse models, conforming to TwoUse metamodels, that represent the abstract syntax. The TwoUse metamodel contains the OWL metamodel and some packages from the UML2 metamodel. The OWL metamodel allows describing semantically expressive classes and the UML2 metamodel allows describing behavioral and structural features of classes. Further transformations take TwoUse models and produce OWL ontologies and Java code.

The advantage of the TwoUse metamodel is the fact that it offers SPARQL-like expressions for reasoning over OWL models.

RQ3 The reason for doing the integration between OWL and UML-class based modeling was the complementary benefits that the two approaches bring. The result provides more modeling power to the developers. The semantic of the models is also better expressed with ontologies.

RQ4 The challenges addressed are related to formal semantics in DSLs.

4.2.13 P13

Erofeev et al. [18] describe a method to achieve semantic interoperability between different technologies used in Ambient Intelligence.

RQ1 This question has not been approached in the article.

RQ2 They start with a core ontology of a reference technology. When an integration is needed with other technologies conforming to different metamodels, they transform the other metamodels into the core ontology metamodel. Subsequently, the models of the other technologies will be automatically transformed into models corresponding to the core ontology.

RQ3 They have chosen to use ontologies in order to obtain semantic interoperability between different technologies.

RQ4 The challenges addressed are those of interoperability.

4.2.14 P14

Kappel et al. [30] describe a method of lifting metamodels to ontologies as a way of integrating modeling languages. The work is part of a bigger project, ModelCVS, whose purpose is to create a framework for semi-automatic generation of transformation programs.

RQ1 Ontologies and metamodeling are created with different goals in mind, but they share common ground in conceptual modeling in general. At the same time, metamodeling is more implementation oriented, while ontologies are more knowledge representation oriented.

RQ2 The technique employed in this paper has been coined as lifting. The approach is divided into three steps. The first step is called conversion, during which, Ecore metamodels are transformed into ODM metamodels. This step introduces a change of formalism and it takes care of the subtle semantic nuances that occur between Ecore and ODM in the transformation. The result of this step is called a pseudo-ontology. In the next step, this pseudo-ontology is refactored, the result being a semantically richer view of the pseudo-ontology. Refactoring is needed to make explicit the concepts that are hidden in attributes or in association ends, as not all concepts are represented as first-class citizens in metamodels. The third step consists in semantically enriching the ontology with axioms with the purpose of integrating it with other ontologies.

The resulting ontologies are the main artifacts of semantic integration. The matching between ontologies and a code generation step are the ingredients for obtaining model transformations between the original metamodels.

RQ3 The starting point is again the need to use tools in combination. The approach was chosen in order to do a conceptual integration between the metamodels via the creation of ontologies. The ontologies created from the metamodels are able to capture more concisely the mapping between them and the mapping between ontologies can further be used to give rise to a bridging between the initial metamodels.

RQ4 The implicit challenge that the technique addresses is that of language/tool interoperability.

5 Quality evaluation

In this section we present the scores of each of the investigated papers. As it was expected, there is no "N" answer to question two in Table 2, because an answer of "Y" or "P" was the inclusion criteria of the papers to be investigated. As can be seen from Table 2, all of the papers explain the rationale of using the chosen technique (answer to question three) and almost all explain explicitly what challenge in the development of DSLs they tackle. This is to be expected, as questions three and four come as natural follow-ups of question two. At the same time, there is only one paper discussing both differences and similarities between ontologies and DSLs. This question being a question offering context to the subject, no answer to this question was not an exclusion criteria. Given these explanations, it was to be expected that the total scores of the papers would be similar.

6 Other relevant papers

The papers we are briefly discussing next offer small overviews on the usage of ontologies in metamodeling or DSLs, or present an idea (without sufficient level of detail and experimentation) on how the integration could take place. These papers are papers that were discarded only in the last phase, after reading their content. So, as with the papers selected for the systematic literature review, we do not claim that this list is exhaustive.

Bézivin et al. [9] propose building bridges between the software engineering and ontology engineering technical spaces at the M3 level. To illustrate the concept, the authors suggested

ID	QA1	QA2	QA3	QA4	Total score
P1	P	Y	Y	Y	3.5
P2	Y	P	Y	Y	3.5
P3	P	Y	Y	Y	3.5
P4	P	Y	Y	Y	3.5
P5	P	Y	Y	Y	3.5
P6	P	P	Y	P	2.5
P7	N	Y	Y	Y	3
P8	N	Y	Y	Y	3
P9	N	Y	Y	Y	3
P10	P	Y	Y	P	3
P11	P	Y	Y	Y	3.5
P12	P	Y	Y	Y	3.5
P13	N	P	Y	Y	2.5
P14	P	Y	Y	Y	3.5

Table 2: Scores of investigated papers.

bridging the MOF-based ontology language ODM with OWL. The two of them being conceptual technical spaces, they are bridged through a concrete technical space (spaces that have techniques with more material representations of conceptual elements), the EBNF technical space. The transformations occurring at the M3 level between different technical spaces (MOF to EBNF, and Metametaontology to EBNF, and vice versa in both cases) are called projectors. The transformations occurring at the M3 level can then be pushed down to the M2 level. This method then saves one from doing 2N different mappings for N metamodels at the M2 level instead of one mapping at the M3 level. The problem is that the method is only mentioned in the paper and not detailed and exemplified.

Atkinson [6] argues the case for core level unification for MDA and ontology representation languages. This comes from the observation that MDA and ontology description languages are not inherently distinct technologies, UML being able to capture the knowledge captured in ontology representation languages by "programming around" features that are not directly supported. The conclusion he draws is that MDA should not be extended to add ontology features to the infrastructure through ODM, but a unified language should be developed as the core of MDA.

Henderson-Sellers [26] is identifying a couple of similarities and differences between ontologies and metamodels in order to provide a bridge between the two. Henderson-Sellers also emphasizes two kinds of ontologies: the domain ontologies and the meta-ontologies or foundational ontologies. The author discusses on the different meta-levels where ontology concepts are used in literature. In some publications cited in the paper, the authors regard an ontology as a M1 model, while others regard it as a M2 model.

Aßmann et al. [5] try to clarify the role of ontologies in MDE. They start from the observation that models in MDA are mostly prescriptive, while ontologies are descriptive models. They then describe an ontology-aware meta-pyramid, where upper-ontologies live at the M2 metamodel level, and domain ontologies live at the M1 model level. This meta-pyramid brings a series of conceptual benefits, like a common vocabulary for the software architect, customer and domain expert or a more concrete model-driven software development with ontologies as analysis models.

Parreiras et al. [41] conduct a domain analysis on the combination of metamodeling technical space and ontology technical space. The result of the domain analysis is a feature model of the existing approaches in literature. The discussed features consist of language, formalism, data model, reasoning, querying, rules, transformation, mediation and modeling level. The mediation process (reconciling different models) is classified in mapping, integration and composition. Another feature we are more interested in is the transformation feature with its three aspects: semantical, syntactical and directionality. Furthermore, the authors classify the approaches that transform the metamodeling technical space into the ontological technical space. These transformations take place for model checking (the ontology resulted

from the transformation is checked for consistency, class hierarchy etc.), model enrichment (transform model to ontology, derive new facts and transform back to model), ontology modeling (from model to ontology via transformation rules) and hybrid approaches (the TwoUse approach with composition between source metamodel and target ontology and bidirectional transformation with querying).

Staab et al. [48] classify methods of model driven engineering with ontology technologies. They distinguish between language bridges and model bridges among software languages and ontologies. The language bridges occur at the M3 level in the form of integrations or transformations. The model bridges occur at the M2 level also in the form of integrations and transformations. These methods appear also in our literature research.

7 Observations

In this section we present some statistics on the 14 investigated papers. The oldest paper we have found is from 2002, while the newest ones are from 2012. Most of the papers were published in 2006 (5 papers), 2009 (4 papers), 2010 (8 papers) and 2011 (4 papers). This shows quite some interest in the subject in the last years. In what regards the journals these papers were published in, there is no major trend. The journals where these papers were published include "Software and Systems Modelling", "Computer Science and Information Systems", "Data and Knowledge Engineering" and "Journal of Systems and Software". There was also a workshop organized between 2008 and 2010 on the subject (workshop on transforming and weaving ontologies in model driven engineering). Then, the conferences were the papers were published include conferences on Semantic Web subjects ("Reasoning Web Semantic Technologies for Software Engineering", "Workshop on Semantic Web Enabled Software Engineering" etc.) and Model Driven Engineering subjects ("Ontologies for Software Engineering and Software Technology", "Models in Software Engineering", etc.).

Most of the research done on ontologies used in the development of DSLs is conducted in Europe in countries such as Germany, Slovenia, France, Austria, the Netherlands, Hungary and Spain. On the other hand, Springer is more Europe-based, so it might partially explain this outcome.

8 Discussion

In this section we emphasize the main trends in the methods of utilizing ontologies in DSLs.

As an introduction to our main research question, the techniques of introducing ontologies to DSLs, we first look at the identified similarities and differences between ontologies and DSLs. This comparison can give us an impression on what the benefits of a combined scheme could be and how hard could be to combine the two.

The *similarities* between ontologies and DSLs consist of the fact that they both have the same building phases (with different focuses) (P2, P6), they both structure data for application use (P2), they present a domain model vocabulary and the relations between the concepts in the domain (P3, P14) and they exhibit equivalence relationships between their main concepts (P4, P6, P12).

The *differences* that exist between the ontologies and the DSLs consist of different application domains (P2, P10, P11, P14), the closed world assumption and nonmonotonic reasoning associated with models and the open-world assumption and monotonic reasoning associated to ontologies (P1, P5, P10) and the different technologies and tools used by each (P2, P10, P12).

The *techniques* employed to leverage DSLs by the usage of ontologies are:

- integration at the M3 model (P1)
- integration at the M2 model (P4)
- mapping from ontology to M2 model (P2, P3, P6, P8, P10, P11)
- mapping from M2 model to ontology (P5, P7, P9, P12, P13, P14)
- ontology inspection to gather requirements for DSLs (P2)

The *reasons* for which ontologies were considered to be used in DSLs in the first place are the need of interoperability between tools / DSL views / technologies, the knowledge reuse, the reasoning and querying capabilities behind ontologies, the complementary benefits of the two approaches and the need for consistency checking at the metamodel level.

The *challenges* in the DSL approaches that appear to be addressed and partially solved by using ontologies in DSLs are those related to formal semantics, interoperability between tools, domain analysis, learning curve of the DSL and tooling (through progressive verification and reasoning explanations).

As a possible direction to explore, we suggest that an in-depth study of the productivity gains of using ontologies in the development of DSLs should be made. None of the papers considered treats this subject. As a result of this sort of studies, engineers could decide whether it is profitable to start with an ontology when building a DSL.

One point that was not clear was how easily can interoperability between tools / DSLs be achieved using ontologies. There were no examples of considerable sized tools that would be made interoperable using ontologies. Such an example and a report on the amount of work to make it work would make clear the feasibility of such an approach.

9 Threats to validity

There are also some threats to the validity of the conclusions drawn. The first threat is the fact that not all search results on SpringerLink were examined. This was due to time limitation. On the other hand, taking into account that all the references of the selected studies found during the one month period were covered, we can conclude that we covered a good part of the literature. Other threat to the validity could be the fact that we only looked at search results on SpringerLink. This should not be a problem, because a quick search and glance on the first results given on the ACM and IEEE websites did not bring anything new.

10 Conclusions

The complementary benefits of DSLs and ontology technologies seem to make them suitable for combination. DSLs' development could mostly profit from the reasoning capabilities supported by the ontologies and the concepts that are captured and related in an ontology. On the other hand, the question is whether the benefits brought justify the effort put in the combination of ontologies and DSLs. That is because transformations or integrations between ontologies and DSL metamodels/models seems not to be trivial in most of the techniques described. Although some techniques involve a certain amount of automation, manual work cannot be removed completely from these processes in most of the cases. That is also due to the semantic gap between DSL metamodels and ontologies [52].

Although the research of this subject only took one month, we consider that we managed to cover a good part of the literature. That is because at the end of this month we did not have any paper that seemed to be suitable to our investigation (from the references of the selected studies and the other papers in their group).

Acknowledgments

This work was supported in part by the European Union's ARTEMIS Joint Undertaking for CRYSTAL - Critical System Engineering Acceleration - under grant agreement No. 332830.

References

[1] Meta Object Facility (MOF) Core Specification, 2006. version 2.0.

[2] Ontology Definition Metamodel, 2009. version 1.0.

[3] Object Constraint Language, 2010. version 2.2.

[4] S. Andova, M. G. J. van den Brand, L. J. P. Engelen, and T. Verhoeff. MDE Basics with a DSL Focus. In *Formal Methods for Model-Driven Engineering*, pages 21–57. Springer, 2012.

[5] U. Aßmann, S. Zschaler, and G. Wagner. Ontologies, meta-models, and the model-driven paradigm. In *Ontologies for Software Engineering and Software Technology*, pages 249–273. Springer, 2006.

[6] C. Atkinson. Unifying MDA and knowledge representation technologies. In *Proceedings of the International Workshop on the Model-Driven Semantic Web (At the 8th International Conference on Enterprise Distributed Object Computing), Monterey, CA*. Citeseer, 2004.

[7] F. Baader and W. Nutt. Basic description logics. In *Description logic handbook*, pages 43–95, 2003.

[8] G. Bergmann, Z. Ujhelyi, I. Ráth, and D. Varró. A graph query language for EMF models. In *Theory and Practice of Model Transformations*, pages 167–182. Springer, 2011.

[9] J. Bézivin, V. Devedzic, D. Djuric, J.-M. Favreau, D. Gasevic, and F. Jouault. An M3-Neutral infrastructure for bridging model engineering and ontology engineering. In *Interoperability of enterprise software and applications*, pages 159–171. Springer, 2006.

[10] M. Bräuer and H. Lochmann. Towards semantic integration of multiple domain-specific languages using ontological foundations. In *Proceedings of 4th International Workshop on (Software) Language Engineering (ATEM 2007) co-located with MoDELS*, 2007.

[11] M. Bräuer and H. Lochmann. An ontology for software models and its practical implications for semantic web reasoning. In *The Semantic Web: Research and Applications*, pages 34–48. Springer, 2008.

[12] D. Brickley and R. V. Guha. Resource Description Framework (RDF) Schema Specification 1.0: W3C Candidate Recommendation 27 March 2000. 2000.

[13] I. Čeh, M. Črepinšek, T. Kosar, and M. Mernik. Using ontology in the development of domain-specific languages. In *INForum*, pages 185–196. 2010.

[14] I. Ceh, M. Crepinšek, T. Kosar, and M. Mernik. Ontology driven development of domain-specific languages. *Computer Science and Information Systems*, 8(2):317–342, 2011.

[15] I. Ceh, M. Crepinsek, T. Kosar, M. Mernik, P. Henriques, M. J. Pereira, D. Cruz, and N. Oliveira. Tool-supported building of DSLs from OWL ontologies. In *INForum*. 2011.

[16] O. Corcho and A. Gomez-Perez. Evaluating knowledge representation and reasoning capabilites of ontology specification languages. In *ECAI'00 Workshop on Applications of Ontologies and Problem Solving Methods*. Informatica, 2000.

[17] O. Curé, R. Forax, P. Degenne, D. L. Seen, D. Parigot, and A. A. Lahcen. Ocelet: An ontology-based domain specific language to model complex domains. In *Communication Theory, Reliability, and Quality of Service (CTRQ), 2010 Third International Conference on*, pages 255–260. IEEE, 2010.

[18] S. Erofeev and X. Larrucea. Ontology-based transformations for achieving interoperability in AmI. In *Enterprise Interoperability*, pages 297–306. Springer, 2007.

[19] M. Fowler. *Domain-specific languages*. Pearson Education, 2010.

[20] J. Gray, K. Fisher, C. Consel, G. Karsai, M. Mernik, and J.-P. Tolvanen. DSLs: the good, the bad, and the ugly. In *Companion to the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications*, pages 791–794. ACM, 2008.

[21] T. R. Gruber et al. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[22] G. Guizzardi, H. Herre, and G. Wagner. Towards ontological foundations for UML conceptual models. In *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, pages 1100–1117. Springer, 2002.

[23] G. Guizzardi, L. Pires, and M. van Sinderen. Ontology-based evaluation and design of domain-specific visual modeling languages. In *Advances in Information Systems Development*, pages 217–228. Springer, 2006.

[24] G. Guizzardi, L. F. Pires, and M. van Sinderen. An ontology-based approach for evaluating the domain appropriateness and comprehensibility appropriateness of modeling languages. In *MoDELS*, pages 691–705. Springer, 2005.

[25] Richard Wesley Hamming. One man's view of computer science. *Journal of the ACM (JACM)*, 16(1):3–12, 1969.

[26] B. Henderson-Sellers. Bridging metamodels and ontologies in software engineering. *Journal of Systems and Software*, 84(2):301–313, 2011.

[27] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, et al. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, 21:79, 2004.

[28] B. Izsó, Z. Szatmári, G. Bergmann, Á. Horváth, I. Ráth, and D. Varró. Ontology driven design of EMF metamodels and well-formedness constraints. In *Proceedings of the 12th Workshop on OCL and Textual Modelling*, pages 37–42. ACM, 2012.

[29] F. Jouault and J. Bézivin. KM3: a DSL for Metamodel Specification. In *Formal Methods for Open Object-Based Distributed Systems*, pages 171–185. Springer, 2006.

[30] G. Kappel, E. Kapsammer, H. Kargl, G. Kramler, T. Reiter, W. Retschitzegger, W. Schwinger, and M. Wimmer. Lifting metamodels to ontologies: A step to the semantic integration of modeling languages. In *Model Driven Engineering Languages and Systems*, pages 528–542. Springer, 2006.

[31] B. A. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. 2007.

[32] I. Kurtev, J. Bézivin, and M. Akşit. Technological spaces: An initial appraisal. In *CoopIS, DOA'2002 Federated Conferences, Industrial track*, 2002.

[33] I. Kurtev, J. Bézivin, F. Jouault, and P. Valduriez. Model-based DSL frameworks. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, pages 602–616. ACM, 2006.

[34] Y.-N. Law, H. Wang, and C. Zaniolo. Query languages and data models for database sequences and data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, pages 492–503. VLDB Endowment, 2004.

[35] G. Lortal, S. Dhouib, and S. Gérard. Integrating ontological domain knowledge into a robotic DSL. In *Models in Software Engineering*, pages 401–414. Springer, 2011.

[36] D. L. McGuinness, F. Van Harmelen, et al. OWL web ontology language overview. *W3C recommendation*, 10(2004-03):10, 2004.

[37] M. Mernik, J. Heering, and A. M. Sloane. When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, 37(4):316–344, 2005.

[38] B. Motik et al. OWL2 web ontology language: Structural specification and functional-style syntax. *W3C recommendation*, 27:17, 2009.

[39] F. S. Parreiras, J. Z. Pan, U. Assmann, and J. Herinksson. First workshop on transforming and weaving ontologies in model driven engineering (TWOMDE 2008). *ACM SIGMOD Record*, 37(4):126–126, 2009.

[40] F. S. Parreiras and S. Staab. Using ontologies with UML class-based modeling: The TwoUse approach. *Data & Knowledge Engineering*, 69(11):1194–1207, 2010.

[41] F. S. Parreiras, S. Staab, and A. Winter. On marrying ontological and metamodeling technical spaces. In *Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 439–448. ACM, 2007.

[42] F. S. Parreiras, T. Walter, C. Wende, and E. Thomas. Bridging software languages and ontology technologies: tutorial summary. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, pages 311–315. ACM, 2010.

[43] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and Complexity of SPARQL. In *The Semantic Web-ISWC 2006*, pages 30–43. Springer, 2006.

[44] T. Rahmani, D. Oberle, and M. Dahms. An adjustable transformation from OWL to Ecore. In *Model Driven Engineering Languages and Systems*, pages 243–257. Springer, 2010.

[45] S. Roser and B. Bauer. An approach to automatically generated model transformations using ontology engineering space. In *Proceedings of Workshop on Semantic Web Enabled Software Engineering (SWESE)*, 2006.

[46] S. Roser and B. Bauer. Automatic generation and evolution of model transformations using ontology engineering space. In *Journal on Data Semantics XI*, pages 32–64. Springer, 2008.

[47] J. Rumbaugh, I. Jacobson, and G. Booch. The unified modeling language reference manual. 1999.

[48] S. Staab, T. Walter, G. Gröner, and F. S. Parreiras. Model driven engineering with ontology technologies. In *Reasoning Web. Semantic Technologies for Software Engineering*, pages 62–98. Springer, 2010.

[49] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro. *EMF: eclipse modeling framework*. Pearson Education, 2008.

[50] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: principles and methods. *Data & knowledge engineering*, 25(1):161–197, 1998.

[51] R. Tairas, M. Mernik, and J. Gray. Using ontologies in the domain analysis of domain-specific languages. In *Models in Software Engineering*, pages 332–342. Springer, 2009.

[52] M.F. van Amstel, M.G.J. van den Brand, Z. Protić, and T. Verhoeff. Transforming Process Algebra Models into UML State Machines: Bridging a Semantic Gap? In *Theory and Practice of Model Transformations*, pages 61–75. Springer, 2008.

[53] M. Volter. *Generic Tools, Specific Languages*. Phd, Delft University of Technology, 2014.

[54] T. Walter and J. Ebert. Combining DSLs and ontologies using metamodel integration. In *Domain-Specific Languages*, pages 148–169. Springer, 2009.

[55] T. Walter, F. Parreiras, and S. Staab. OntoDSL: An ontology-based framework for domain-specific languages. In *Model Driven Engineering Languages and Systems*, pages 408–422. Springer, 2009.

[56] T. Walter, F. S. Parreiras, and S. Staab. An ontology-based framework for domain-specific modeling. *Software & Systems Modeling*, pages 1–26, 2012.

[57] T. Walter, F. S. Parreiras, S. Staab, and J. Ebert. Joint language and domain engineering. In *Modelling Foundations and Applications*, pages 321–336. Springer, 2010.

Science Reports **Department of Mathematics and Computer Science**
Technische Universiteit Eindhoven

If you want to receive reports, send an email to: wsinsan@tue.nl (we cannot guarantee the availability of the requested reports).

In this series appeared (from 2012):

12/01	S. Cranen	Model checking the FlexRay startup phase
12/02	U. Khadim and P.J.L. Cuijpers	Appendix C / G of the paper: Repairing Time-Determinism in the Process Algebra for Hybrid Systems ACP
12/03	M.M.H.P. van den Heuvel, P.J.L. Cuijpers, J.J. Lukkien and N.W. Fisher	Revised budget allocations for fixed-priority-scheduled periodic resources
12/04	Ammar Osaiweran, Tom Fransen, Jan Friso Groote and Bart van Rijnsoever	Experience Report on Designing and Developing Control Components using Formal Methods
12/05	Sjoerd Cranen, Jeroen J.A. Keiren and Tim A.C. Willemse	A cure for stuttering parity games
12/06	A.P. van der Meer	CIF MSOS type system
12/07	Dirk Fahland and Robert Prüfer	Data and Abstraction for Scenario-Based Modeling with Petri Nets
12/08	Luc Engelen and Anton Wijs	Checking Property Preservation of Refining Transformations for Model-Driven Development
12/09	M.M.H.P. van den Heuvel, M. Behnam, R.J. Bril, J.J. Lukkien and T. Nolte	Opaque analysis for resource-sharing components in hierarchical real-time systems - extended version –
12/10	Milosh Stolikj, Pieter J. L. Cuijpers and Johan J. Lukkien	Efficient reprogramming of sensor networks using incremental updates and data compression
12/11	John Businge, Alexander Serebrenik and Mark van den Brand	Survival of Eclipse Third-party Plug-ins
12/12	Jeroen J.A. Keiren and Martijn D. Klabbers	Modelling and verifying IEEE Std 11073-20601 session setup using mCRL2
12/13	Ammar Osaiweran, Jan Friso Groote, Mathijs Schuts, Jozef Hooman and Bart van Rijnsoever	Evaluating the Effect of Formal Techniques in Industry
12/14	Ammar Osaiweran, Mathijs Schuts, and Jozef Hooman	Incorporating Formal Techniques into Industrial Practice
13/01	S. Cranen, M.W. Gazda, J.W. Wesselink and T.A.C. Willemse	Abstraction in Parameterised Boolean Equation Systems
13/02	Neda Noroozi, Mohammad Reza Mousavi and Tim A.C. Willemse	Decomposability in Formal Conformance Testing
13/03	D. Bera, K.M. van Hee and N. Sidorova	Discrete Timed Petri nets
13/04	A. Kota Gopalakrishna, T. Ozcelebi, A. Liotta and J.J. Lukkien	Relevance as a Metric for Evaluating Machine Learning Algorithms
13/05	T. Ozcelebi, A. Weffers-Albu and J.J. Lukkien	Proceedings of the 2012 Workshop on Ambient Intelligence Infrastructures (WAmIi)
13/06	Lotfi ben Othmane, Pelin Angin, Harold Weffers and Bharat Bhargava	Extending the Agile Development Process to Develop Acceptably Secure Software
13/07	R.H. Mak	Resource-aware Life Cycle Models for Service-oriented Applications managed by a Component Framework
13/08	Mark van den Brand and Jan Friso Groote	Software Engineering: Redundancy is Key
13/09	P.J.L. Cuijpers	Prefix Orders as a General Model of Dynamics

14/01	Jan Friso Groote, Remco van der Hofstad and Matthias Raffelsieper	On the Random Structure of Behavioural Transition Systems
14/02	Maurice H. ter Beek and Erik P. de Vink	Using mCRL2 for the analysis of software product lines
14/03	Frank Peeters, Ion Barosan, Tao Yue and Alexander Serebrenik	A Modeling Environment Supporting the Co-evolution of User Requirements and Design
14/04	Jan Friso Groote and Hans Zantema	A probabilistic analysis of the Game of the Goose
14/05	Hrishikesh Salunkhe, Orlando Moreira and Kees van Berkel	Buffer Allocation for Real-Time Streaming on a Multi-Processor without Back-Pressure
14/06	D. Bera, K.M. van Hee and H. Nijmeijer	Relationship between Simulink and Petri nets
14/07	Reinder J. Bril and Jinkyu Lee	CRTS 2014 - Proceedings of the 7th International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems
14/08	Fatih Turkmen, Jerry den Hartog, Silvio Ranise and Nicola Zannone	Analysis of XACML Policies with SMT
14/09	Ana-Maria Şutîi, Tom Verhoeff and M.G.J. van den Brand	Ontologies in domain specific languages – A systematic literature review