

A Case on Comprehensive Software Development Framework Driven by Requirements: ReDSeeDS

Özgür TÜFEKÇİ, İlker ÇOKKEÇECİ , Semih ÇETİN
Cybersoft C/S Information Technologies Ltd. Co.,
ODTÜ Teknokent Silikon Blok No:18, Ankara, 06531, Turkey
Tel: +90 312 210 19 40, Fax: + 903122101752,

Email: ozgur.tufekci@cs.com.tr, ilker.cokkececi@cs.com.tr, semih.cetin@cs.com.tr

Abstract: This paper is based on the ideas of the research work within the FP6 EU-funded project ReDSeeDS (Requirements-driven Software Development). The project proposes an open framework consisting of a scenario-driven development method, a repository for reuse and tool support. The basic reuse approach is case-based where requirements are mapped to architecture, architecture to design and design to code. After making an introduction to the project, the paper focuses on Cybersoft's validation work which forms a crucial step for building a practical reuse platform. Discussion of the gained experience will follow.

1. Introduction

With rapid advances in hardware industry, software development industry has become a bottleneck in meeting the needs of customers getting even more challenging with the increased complexity and reduced time to market. The underlying problem may be defined as inability to reuse knowledge due to lack of practical mechanisms for expressing and reusing already solved software problems.

ReDSeeDS aims to fulfill the promise of software reuse by linking the requirements with architecture, design, and code. In order to develop ReDSeeDS framework, state of the art in requirements engineering, meta-modeling, model transformations, querying and inference techniques have been enhanced to enable a completely new approach to software development based on case-based reuse.

Cybersoft acts as one of the three senior users who are primarily responsible for the requirements specification for the project. The main purpose is testing the validity of ReDSeeDS framework in industrial cases for building a practical reuse platform. For pilot projects, the requirements are elicited, applicability of the transformation language to transformation rules and developed models are tested and the architectural design is created by using the tools developed in ReDSeeDS. This paper briefly introduces the project and focuses on the work accomplished by Cybersoft and concluding with the lessons learned.

2. Objectives

Complexity of modern software systems is linked with the complexity and changeability of problems specified through requirements specifications. Complex problems often lead to even more complex solutions implemented in the technological space that changes even faster than the problem domain. Despite this complexity (or maybe – due to it), vast majority of software development projects seem to ignore past knowledge about solving specific problems. This might be explained by significant difficulties to reuse knowledge in

such a complex domain as software engineering is. The main obstacle here is the lack of systemic ways to capture knowledge about complete cases leading from the problem (requirements) to its solution (architecture, design and code). There also seem to be no effective means to find and reuse past solutions to problems similar to existing ones [5].

A software development (SD) project generates specific artefacts (requirements documents, design documents, code, etc.). Additionally, certain tacit knowledge is gained by the SD project participants. Some of the SD project results can be generalised to form design or analytical patterns. Unfortunately, the above mentioned artefacts are usually very hard to reuse, even when the new problem is very similar to the previous one (or is simply a next version of an existing system). This is caused by the fact that this knowledge is not structured in a way that would allow for easy comparison and retrieval [1]. The inability to reuse knowledge about already solved software problems illustrated in Figure 1.

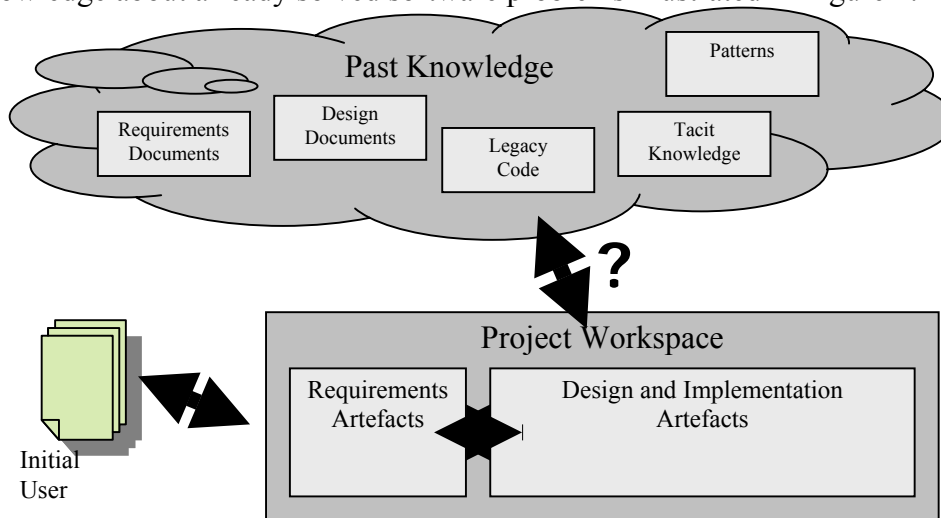


Figure 1: Reusing Unstructured Knowledge in Software Development Projects

Existing approaches to software reuse such as design patterns or object orientation have limited benefits and require significant effort for introducing the reuse mechanisms since they depend on “development for reuse” rather than “development with reuse”. To increase the effectiveness and efficiency of reuse, new approaches are needed to ease both construction and use of the reusable assets. Existing modelling and transformation languages do not have the potential to define fully reusable “software cases”. They mostly concentrate on the design models and transformations between various levels of design [1].

A “software case” consists of a requirements model, an architectural model, a detailed design and code [2]. The requirements model represents the problem whereas the architectural model, detailed design and code together represent the solution. Such a Software Case contains a precisely expressed problem statement in the form of a Requirements Model. All elements of this problem statement are then mapped onto appropriate elements of the solution, which is formed of (again) precisely expressed design models (including Architectural Model, Design Model, etc.) and the final code. Software Cases can be reused on the basis of their similarity to the currently developed system. This similarity can be determined by comparing the current requirements model with the requirements models of past Software Cases. The past solution can then be easily «reused» by modifying it in those places that are precisely marked as “needing rework” [1].

All software cases of a specific domain are collected and summarized in a software knowledge model. The variability in the software knowledge model is exploited during software development as follows. Based on a (partial) requirements specification, similar software cases are identified, i.e. software cases with similar requirements. The retrieved software cases also contain solutions (architecture, detailed design and code). Based on the

traceability links between requirements, architecture, detailed design and code, it is possible to reused adapted (partial) solutions [3]. With this vision in mind, ReDSeeDS project proposes a reuse-oriented, requirements-based development framework given in Figure 2.

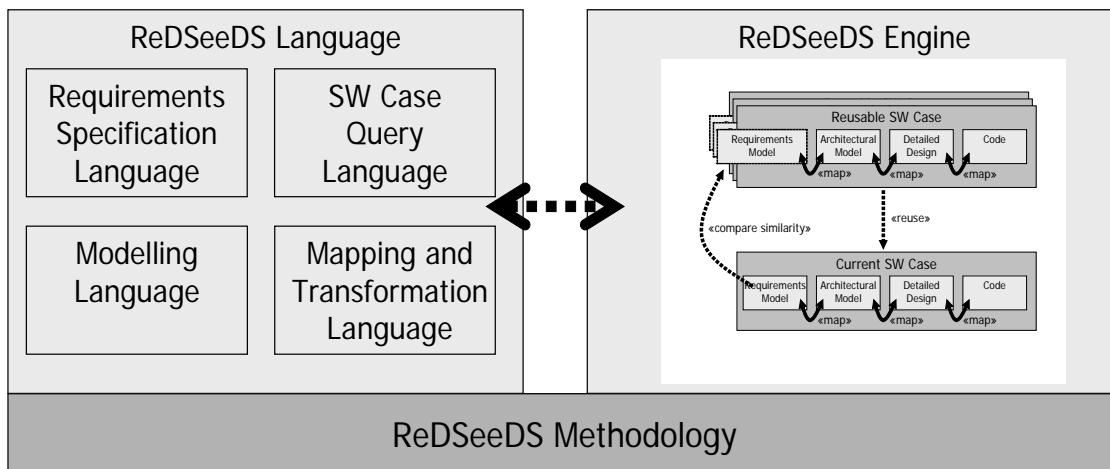


Figure 2. Elements of the ReDSeeDS Framework

The framework is composed of “ReDSeeDS Language”, “ReDSeeDS Engine” and “ReDSeeDS Methodology”. ReDSeeDS Language, as the combination of a Requirements Specification Language (RSL), a Modelling Language (ML), a Mapping and Transformation Language (MTL), and a Software Case Query Language (SCQL), is a precise representation of reusable software cases. The ReDSeeDS Methodology describes the software lifecycle based on creating and reusing software cases performed through the ReDSeeDS Engine. ReDSeeDS Methodology avoids the needs of preparing for reuse by altering the way software development is accomplished.

Cybersoft’s main role in the project is testing the validity of ReDSeeDS framework in industrial practice for building a usable reuse platform. The selected projects are “stock control system”, “material management system”, “procurement system”, and “purchase order system” as the major modules of an ERP system developed by Cybersoft for TEDAS (Turkish Electricity Distribution Corporation) built on AURORA e-business framework. AURORA is an application development and execution platform supporting today’s well known “Rich Internet Applications (RIA)” and “Enterprise Internet Applications (EIA)” concepts [7]. Industrial practice is crucial in achieving a convenient and reusable platform. In order to achieve this goal, a comparative study within this Web-based real-life project based on the 4-tier architecture was carried out for providing the solution to 3000 users.

3. Methodology

Validation activities used the ReDSeeDS engine to support development of software in proper problem domains. Empirical results were gathered to compare software development with and without the newly defined and developed framework. Validation activities were also closely related to the definition of the ReDSeeDS methodology. The methodology was originally designed based on the input from other activities, which has started when the first version of the methodology was ready. The second version of the methodology is then based on the experience from these first-level validation activities.

Experience from the validation was also used to prepare the final version of the ReDSeeDS framework. The second iteration of research and development starts after the first-level validation experiences had been formulated. This second iteration of R&D activities is a very important for assuring quality of the final ReDSeeDS framework. It accommodates for highly probable flaws and drawbacks of the language and the supporting

engine prototype identified during validation of the framework. It also gives the possibility to accommodate the methodology, which is certain to change after the first iteration.

3.1 *Creating Cases in Various Problem Domains*

Selected real-life projects with the following attributes were used for creating the cases:

- Stock Control System has 12 requirements, 12 use cases, 34 notions, 1 actor, and 1 system element.
- Material Management System has 12 requirements, 12 use cases, 19 notions, 1 actor, and 1 system element.
- Procurement System has 16 requirements, 16 use cases, 40 notions, 1 actor, and 1 system element.
- Purchase Order System has 16 requirements, 16 use cases, 39 notions, 1 actor and 1 system element.

Requirements and scenario sentences were specified using Requirements Specification Language (RSL). System objectives, vision and scope were written in requirements by using the natural language. Afterwards, links from the objectives, vision and scope to notions were created. Here, the basic ReDSeeDS methodology was used. Cybersoft put the “analyst” role in action for using the ReDSeeDS engine at this stage.

3.2 *Reusing Cases in Real-Life Projects*

Real-life projects were performed for creating new case and reusing old ones. The selected new case is “Material Requirement and Management System” as the real-life project. System objectives, vision and scope were written in requirements by using the natural language. Appropriate notions were created. Links from objectives, vision and scope to notions were created. Three significant use cases were specified. Then, similarities were manually compared between “Material Requirement and Management System” and other real-life projects which have been already created. Afterwards, the similarities were this time compared using the query manager of ReDSeeDS Engine.

These similarity results were compared at the end of the reuse activities. At least two mostly similar use cases were selected from each of the cases. Stock Control System was mostly similar (70%) with “Material Requirement and Management System”. Slices were prepared and imported into the “Material Requirement and Management System” using the “maximal slice” method. Imported slices were merged into the new case. Cybersoft put the “business analyst (domain expert)”, “system analyst (technical expert)”, “designer”, “developer”, “tester”, “reuse manager”, and “project manager” roles in action for using the ReDSeeDS engine at this stage.

4. **Technology Description**

Construction of the ReDSeeDS prototype is based on the knowledge of construction of existing tools like GraLab, MOLA and Scenario Writer. GraLab is an Application Programming Interface (API) created by the University of Koblenz-Landau for handling large graph-based repositories. There is a Java version “JGraLab” as well. Using GraLab, structured information is stored in typed, attributed, and ordered directed graphs. GraLab repositories reside in main storage and allow the handling of graphs of several million elements without loss of efficiency. Storage of TGraphs in GraLab is optimized for traversal, which is essential for querying [9].

The MOLA system is the result of research that Institute of Mathematics and Computer Science, University of Latvia (UL) has been doing in the area of model transformations since 2003. The result is the model transformation language MOLA, which has already

gained some popularity and serves as a prototype for the pattern matching based part of the hybrid modelling and transformation language to be developed in ReDSeeDS [10].

The Scenario Writer system is the result of research work performed at the Warsaw University of Technology, which is oriented to building precise yet comprehensible requirements specifications. The tool allows for formulation of use case scenarios in a uniform way. The tool is based on a precise MOF-based meta-model which facilitates its enhancement in the direction of transformations into design models. The tool is integrated with a UML modelling tool and as such forms an extension of the UML language in the area of scenario-oriented requirements [5].

4.1 Eclipse-based ReDSeeDS Engine

ReDSeeDS Engine is the main ReDSeeDS tool based on Eclipse Platform. ReDSeeDS tool components, during architectural consideration for the ReDSeeDS Engine, were identified as Eclipse platform plug-ins: some of them contributing to the UI-tier (and built using Eclipse toolkits), others contributing to business and application logic of the application. By using the Eclipse platform, all ReDSeeDS tools are integrated into one complete solution (and a stand-alone client application) [4]. The use of ReDSeeDS Engine in real life Stock Control project can be seen in Figure 3.

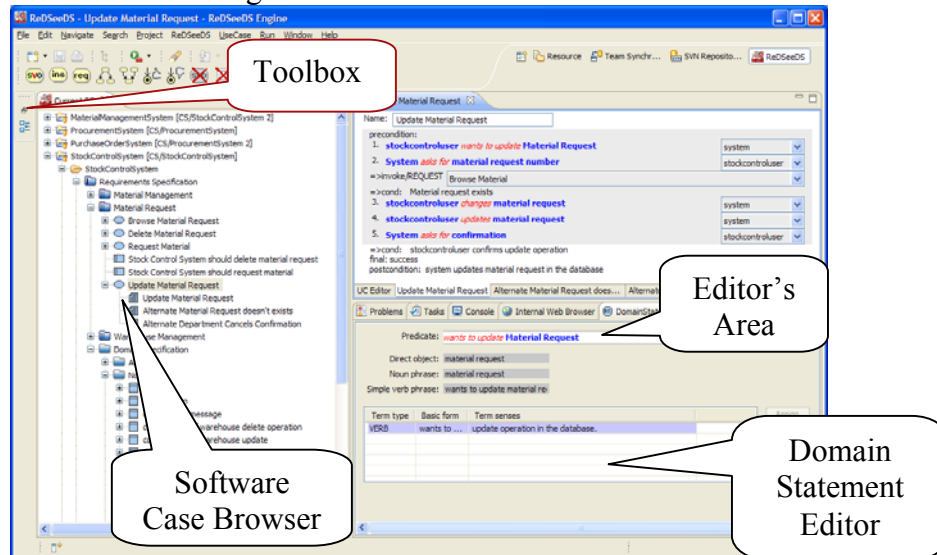


Figure 3. Eclipse-based ReDSeeDS Engine

Main parts of the Engine are:

- Eclipse-based integration platform.
- RSL Editor for describing Requirements; Projects, requirements and use cases are created in the Current SC tab. There is software case browser where projects are listed. Engine functions are listed in the toolbox, which are SVO, Insert, Requirement, Invoke etc. Scenario sentences are specified in the editor's area using RSL and functions for use cases.
- SDSL Editor for describing Architecture and Detailed Design; Architecture is generated in the SDSL Editor. Detailed Design is generated based on created Architecture. Architecture and Detailed Design are modified at Enterprise Architect.
- Transformations making development process much faster and allow easier reuse.
- Common data model for all parts of the engine.
- Reuse engine for past Software Cases query, retrieval and reuse.

4.2 Data Model

ReDSeeDS has an only one interchange data store. It bases on TGraphs and is implemented with JGraLab. We have one common data model for all different parts. It allows for defining traceability links between different layers of a Software Case. Traceability links play crucial role in the reuse. Another impact of this solution is the possibility of giving a total view of the whole Software Case [4].

4.3 RSL Editor

ReDSeeDS Engine user can view and edit all elements of the requirements specification expressed in the RSL in one of 5 editors. These editors are for Requirements, Use Cases, Notions, Actors, and System Elements. Each editor operates directly on the RSL model. User can edit element's description, name and relations to other requirements elements. Use Case Editor is the most sophisticated multipage editor, where the first page is similar to other editors and other pages contain scenarios for the particular use case. User may tag words as a proper part of the sentence in writing the sentences [4].

4.4 SDSL Editor

SDSL Editor is based on Sparx Enterprise Architect. SDSL was implemented as a subset of UML and it is fully supported by Enterprise Architect. Models created in the modelling tool can be converted to TGraphs and stored in the common data store. And vice versa, UML models created as part of a software case can be converted to and stored as Enterprise Architect files [4]. Among the four software cases developed by Cybersoft, one of them is the Stock Control System. SDSL diagram for Stock Control System in Enterprise Architect with customised UML meta-model can be seen in Figure 4.

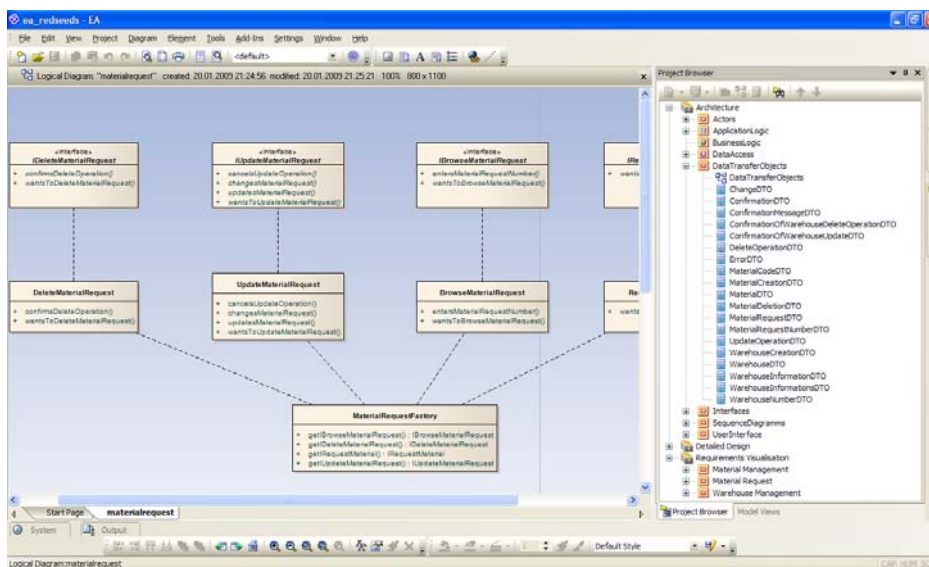


Figure 4. SDSL Editor

4.5 Transformations

Model-to-model transformation is implemented in the prototype. All transformations are implemented using the model transformation language MOLA. The transformation-ready model architecture style is used for both “architecture” and “detailed design” models. A compiler for the MOLA language that fits into the ReDSeeDS engine has been used for

transformations. This compiler generates Java code, which in turn is based on the same JGraLab repository used by other components of the ReDSeeDS engine. Additionally, final versions of the meta-models for RSL and SDSL (selected subset of UML) have been used during the validation process [4].

4.6 Reuse Engine

Reuse engine is based on the common terminology that is shared online by all users on a common server. Software Cases are stored on local computers and may be shared like other files (e.g. using SVN version control system). Software Cases are readable and editable in offline mode, but no new notions may be added. This terminology is a TGraph generated from the Princeton University's WordNet 3.0 [11] model and containing the mostly used words of the English language together with their different senses and relations [4].

5. Results

During the implementation of these pilot projects, the requirements are elicited via RSL editor, applicability of the transformation language to transformation rules and developed models are tested, and the architectural design is created by using MOLA transformations.

Definition of software cases in a formal or semi-formal way facilitates since software case contents allow the similarity measures. ReDSeeDS methodology provides guidance and a set of good practices for users of ReDSeeDS languages and the ReDSeeDS engine. On the other hand, the main ReDSeeDS tool is based on the Eclipse Platform. The layout and function of the application are under fine-grained control of its developer, but also a subject to easy customisation by the end-user. Complete ReDSeeDS tool components, during architectural consideration for the ReSeeDS engine, were identified as Eclipse platform plug-ins. By the use of Eclipse platform, all ReDSeeDS tools are integrated into one complete solution [6]. All these facilitate the integrated development. But, in the pilot projects, extra time was spent for learning the tools that can be reduced over time.

From the current viewpoint shaped by the Cybersoft's evaluation, it has been clearly seen that ReDSeeDS approach to the problem provides important improvements compared to solution-limited approaches neglecting the elicited requirement definitions. For instance, requirements are used to define the similarities and differences between Stock Control System within the extended software case definition "Material Requirement and Management System" and the similarity turned out to be 69%, which is quite significant. Furthermore, identification of the domain keywords eases domain analysis and design. Upon this prospect, it will be possible to achieve reusability in a high level both in development and maintenance projects.

6. Business Benefits

The ReDSeeDS Project will be finalized by December, 2009. ReDSeeDS tools on Eclipse platform will be made available to the use of software development community. In addition to this, Cybersoft is strongly interested in integrating the ReDSeeDS engine with its state-of-the-art software product line system – AURORA by 2010 [7], [8]. This integration will significantly enhance the capabilities for reuse and add more semantic capabilities to AURORA. With ReDSeeDS, Cybersoft has the following plans:

- 1) Internal use for improving software development lifecycle and reducing costs: Cybersoft has developed and been operating two major tax automation solutions: GIB-Turkey and AVIS-Azerbaijan. These systems have similar requirements that can benefit from ReDSeeDS. In this respect, Cybersoft plans to achieve reusable and complete assets based on requirements rather than programs running on each

project that needs to be customized. Another application domain may be the large-scale ERPs. All these projects are built on AURORA platform. Assuming a 10% estimate of improvement, Cybersoft could spare 70.000 Euros per year.

- 2) Offering maintenance to customers: After the warranty period, usually the IT divisions of the customer are responsible from the maintenance of the developed systems. However, the frequent changes in requirements and the regulations to put the changes into effect in a short time is quite a challenging task. Offering them the ReDSeeDS engine and methodology, training as well as the relevant assets may be quite beneficial in their maintenance effort. Overall return from the above package may be estimated to be 30.000 Euros per year assuming one project per year.

7. Conclusions and Future Work

The ReDSeeDS approach to software reuse enhances the current reuse strategies which are limited to the solution and not taking the requirements specification into account. Based on the enhanced definition of a software case, requirements are used to define similarities and differences between software systems. High levels of reuse are then possible both for completely new projects as well as for efforts in maintenance and enhancement projects.

Next steps will include working on the ReDSeeDS methodology and its integration in process management models. It is important that this integration covers the recently rising architectural models such as Service-Oriented Architecture (SOA) and Business Process Management (BPM). Thus, the user roles within ReDSeeDS methodology, as mentioned earlier, will contribute to this development process to the extent possible.

Acknowledgement

ReDSeeDs project is coordinated by Infovide, Poland with technical lead of Warsaw University of Technology and with University of Koblenz-Landau, Vienna University of Technology, Fraunhofer IESE, University of Latvia, HITeC e.V. c/o University of Hamburg, Heriot-Watt University, PRO DV, Cybersoft and Algoritmu Sistemas.

References

- [1] Smialek, M., "Software Development with Reusable Requirements-Based Cases, Warsaw University of Technology", 2007.
- [2] Smialek, M., "Towards a Requirements driven Software Development System." In: Models 2006, 2006.
- [3] Wolter, K., Hotz, L., Krebs, T., "Towards Integration of Modelling and Reusing Software Cases", Proc. Workshop on Software and Services Variability Management - Concepts, Models and Tools, Helsinki, Finland, April 2007.
- [4] Rein, M. et al, "ReDSeeDS Prototype Description, Implementing the ReDSeeDS Engine prototype-1st iteration", 2008.
- [5] Lisek, S. et al, "ReDSeeDS Annex I Description of Work", 2006.
- [6] Ambroziewicz, A., "Results Summary and Impact Report Year 2", 2008.
- [7] Altintas, N. I., Surav, M., Keskin, O., and Cetin, S., "Aurora Software Product Line", 2nd National Software Engineering Conference, Ankara, 2005.
- [8] Cetin, S., "Impact of Architectural Concerns on Continual Achievement of Enterprise Applications", First National Software Architecture Conference, Istanbul, 2006.
- [9] JGraLab, <http://userpages.uni-koblenz.de/~ist/JGraLab>, last accessed in June 2009.
- [10] MOLA, <http://mola.mii.lu.lv/>, last accessed in June 2009.
- [11] WordNet, <http://wordnet.princeton.edu/>, last accessed in June 2009.