A Visual Tool for Supporting Developers in Ontology-based Application Integration

Tobias Wieschnowsky[1] and Heiko Paulheim[2]

[1] SAP Research
tobias.wieschnowsky@sap.com
[2] Technische Universität Darmstadt
Knowledge Engineering Group
paulheim@ke.tu-darmstadt.de

Abstract. Ontologies are increasingly used for a larger variety of tasks in software engineering. However, it turns out that many existing tools are directed rather at *ontology experts* than at *software engineering experts*, which often reduces their acceptance and hinders are larger breakthrough of semantic technologies in software engineering. Tools that speak the language of software engineers may help overcoming that gap. In this paper, we introduce an ontology and rule editor specifically tailored at the task of application integration, and demonstrate its usability for software engineers with a comparative user study.

1 Introduction

Ontologies are used in software engineering for various purposes, e.g., in integration tasks, for capturing requirements, for building abstraction layers between systems, for improving model based development, or for enabling better development tools, such as component repositories with semantic annotations [1]. While those approaches solve different problems, they most often demand that the developer learns to handle ontologies and understands the underlying logic foundations.

Experience has shown that the promises semantic technologies makes for software engineering are not always kept, and often, the reason is that software engineers lack the appropriate understanding for ontologies. While some concepts of ontology engineering, such as the definition of classes and sub classes, are rather intuitive for software engineers, working with more sophisticated concepts such as logic axioms or rules is not as trivial. Furthermore, existing ontology editors and tools most often demand a deeper insight into the underlying formalisms, and thus, lack good usability [2].

Therefore, in order to get semantic technologies accepted by software engineers, good tools are an essential requirement, which hide most of the logic complexity "under the hood", yet are versatile enough for the task at hand, and speak the developers' language. In this paper, we focus on the particular aspect of creating logic rules employed in an ontology-based application integration framework. We discuss a prototype for creating such rules, and compare it to other state-of-the-art tools in a comparative user study.

2 Application Integration on the User Interface Level

Ontologies have been successfully used in application integration during the past 20 years. The best-known approaches are ontology-based integration on the database level, where an ontology-based abstraction layer is placed on top of different, heterogeneous databases, and on the business logic level, e.g., by employing semantic web services.

In contrast to those well-researched approaches, ontology-based application integration on the user interface level is a comparatively new research direction. In [3], we have discussed a framework for employing ontologies and rules for integrating applications on the user interface level, i.e., facilitating *cross-application interactions* between existing applications, while, at the same time, *reusing existing user interfaces*. When using the framework, a developer providing an application to be integrated has to write an ontology describing the application, as well as a set of rules defining the possible interactions with other applications.

We have built a prototype implementation of that framework, using Java and OntoBroker [1], and F-Logic for describing both application ontologies and integration rules. The prototype of the framework was used in the large scale research project *SoKNOS* for developing a functional prototype for the domain of emergency management [4].

Each development team providing an application was asked to also develop the respective ontology and interaction rules. However, said task was not trivial for developers, even with a sophisticated tool such as OntoStudio. Especially the interaction rules in F-Logic proved difficult to create. Furthermore, it was hard for the developers to understand given rules, thus, the problem was not solved with providing a few example rules. In addition, the maintenance of existing rules was difficult due to that lack of understanding. These insights lead to the requirement of providing tool support to developers.

3 Prototype

Existing tools for developing ontologies and rules are typically *general-purpose* tools, which provide many capabilities, at the price of being very complex. Furthermore, they typically speak the language of formal logics, rather than the language of software engineering and/or the domain for which the software is developed.

The basic design rationales for developing a tool for supporting developers were therefore to reduce complexity at the cost of functionality, and to make the tool speak the developers' language. Thus, we have developed a tool which can be used to create only the specific types of ontologies and rules that were needed in the context of the integration framework.

The tool prototype is realized as an Eclipse plugin, since Eclipse is a well-known software engineering tool. Fig. 1 provides an overview of the architecture

[1] `http://www.ontoprise.de/en/products/ontobroker/`

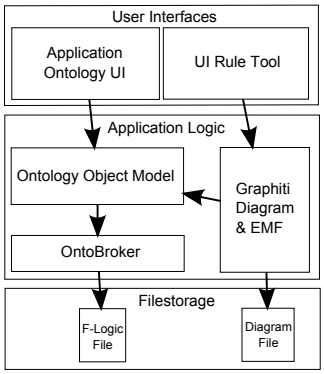

Fig. 1. Overview of the architecture of the UI Integration Tool.

of the prototype. On the top level it consists of two user interfaces that each consist of a number of eclipse views and editors. The first user interface is the application ontology editing user interface and the second is the integration rule user interface. The later is the main focus of this paper and is also called the UI Rule Tool. The main editor of the UI Rule Tool is a Graphitti/EMF based visual rule editor, where developers can construct their integration rules without any knowledge of F-Logic. Underneath the integration rule user interface lies the Graphiti/EMF model which is supported by a Java representation of the ontologies, called the Ontology Object Model. All application ontology user interfaces are based directly on the Ontology Object Model.

Any information in the Ontology Object Model is stored as F-Logic files through OntoBroker's Java API. Any existing integration rules are also saved in Graphiti's own Diagram format which includes additional information, like position of the parts in the diagram. This dual storage of the rule itself as well as the layout information allows for the exact reconstruction of the visual layout when a rule is reloaded, a feature which is not provided by editors that only store the logic rule itself.

Figure 2 shows an example integration rule in the UI Rule Tool. The user interface consists of several utility views, like for example the ontology viewer (on the left) and the main editor (in the middle). The main editor window in turn, contains the drawing area and the tool palette.

The two basic components of the visual integration rules are instances (displayed as boxes) and relations (displayed as the arrows connecting the boxes). This visual representation is very simple and should be intuitive. In addition there are more components like attributes (boxes on instances) and logic parts like OR and NOT.

The most visually dominant feature of the UI Rule Tool is the clear visible split into *cause* and *effect*. This reflects the two parts of a logic rule, *body* and *head*, but has been renamed to make it clearer which role the two parts of the rule play within the framework. The first part of the rule describes the conditions to

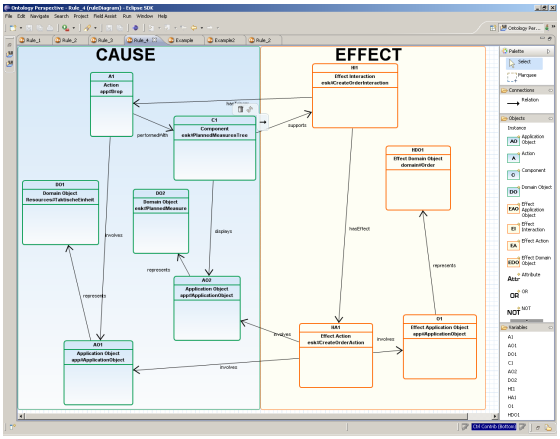

Fig. 2. An example rule in the UI Rule Tool.

the event that occurs within the framework and causes the reasoner to examine the rule (the cause). The second part describes what changes in the application that owns the rule if the cause part is found to be true (the effect). Thinking of the two parts of the rule in these terms should help developers to better understand how the rule works within the framework.

Another feature that attempts to incorporate the framework into the rule editing process are the instance roles, which are derived from reoccurring patterns of rules used in the integration framework. This pattern can also be seen in the example rule diagram in figure 2: The rule is "triggered" by an instance of type `Action` (or one of its subtypes), which may involve one or more `ApplicationObjects` representing objects from the domain, and which may be performed with a `Component`. The effect side of a rule typically consists of an `Interaction` supported by the `Component` and triggered by the `Action`, which in turn causes the performing of an `Action`. All those instances are especially marked in the tool with specific symbols, thus providing a domain-specific visualization of the rule. Furthermore, the palette contains a set of tools for creating that basic pattern or typical subpatterns of rules used in the framework.

4 Evaluation

In order to evaluate the rule editing interface, a user study was conducted, which compared it against those of three other ontology and/or rule editing tools. The goal of the user study was to determine if the UI Rule Tool is better suited for the creation of integration rule within the framework for UI integration.

4.1 Overview

The user study was comprised of a total of 16 participants, who were undergraduate students, PhD candidates, and researchers from SAP Research in Darmstadt

or the TU Darmstadt. Each participants was given the same 4 example rules, taken from the SoKNOS prototype of the UI Integration Framework. For each participant, each tool was randomly assigned to one of the four example rules. Overall, each tool was assigned to each rule an equal number of times. After completing all tasks for a tool/rule combination, the participants were asked to fill out a User Experience Questionnaire that captured their attitude towards the tool they had just worked with. After all 16 participants completed the study, the results were evaluated by three criteria: Time needed to complete the tasks, number of errors made, and the responses to he questionnaires.

4.2 Setup

All participants of the user study were volunteers from SAP Research and the TU Darmstadt and had a strong background in computer science. At the beginning of the study they were asked to self-asses their knowledge on ontologies and logic rules, as well as the three tools competing with the UI Rule tool. Depending on their response to the questions about their ontology and logic rule knowledge, the users were classified as either an expert, or a non-expert. This classification was later used for in-depth analysis of the results, but had no influence on the study design. Non of the participants indicated enough experience with any of the tools to warrant a separation into expert and non-expert groups.

The four tools compared in the user study were: the our prototype of the UI Rule Tool, *OntoStudio* [2], *Protégé* [3], and *Strelka* [4].

For each rule, participants had to complete a set of three tasks, two of them multiple choice and one modification task. The multiple choice questions were concerned with understanding the given rules: the first question asked users to identify the element in the rule that triggered the rule from the perspective of the framework, and the second question asked the users to identify the element in the rule that was the result of the rule and would lead to the framework modifying the application the rule belongs too. For both multiple choice tasks, error rate and completion time were recorded. The third task for each tool was a rather open task, where the user was asked to modify the rule and explore the capabilities of the tool. Participants were also allowed to ask questions to the interviewer when performing the third task. Error rate and completion time were not recorded for this task as it's focus was more to allow the user to work with the tools in order to increase the significance of the questionnaire results.

After completing all tasks assigned to a tool, the participants were asked to fill out a User Experience Questionnaire for the corresponding tool. The User Experience Questionnaire [5] (UEQ) asks participants to judge their experience with a tool based on 26 adjective pairs. Each pair contains two polar opposites like for example, *fast* and *slow*. Users then marked which of the two adjectives more closely described their interaction with the program. The 26 attribute

[2] `http://www.ontoprise.de/en/products/ontostudio/`
[3] `http://protege.stanford.edu/`
[4] `http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=Strelka`

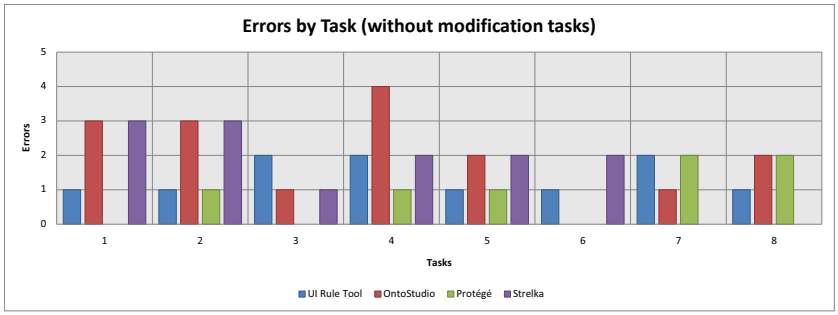

Fig. 3. Errors per Task and Tool

pairs are used to calculate six user experience factors: attractiveness, perspicuity, dependability, efficiency, novelty and stimulation.

As part of the evaluation of the results, the statistical significance of the recorded differences between the tools was computed. For time and UEQ results, the two sample t test (or short t test) was used to compute the statistical significance, with a threshold of 95% before a result was considered significant. For the differences in errors, the chi-square test (or chi-test) was applied to compute the significance, again using 95% as the threshold. These test were used to ensure that the results of the user study were not just due to random variations in the rather small data set of the 16 participants.

4.3 Results

Overall significantly more errors were made in all tools than expected, considering that all rules shared a common pattern and the correct responses were very similar for each rules tasks. Figure 3 shows a summary of errors made by task. The overall lowest number of errors were made in Protégé which was a surprising result, considering it is the only rule editor in the study that is text based. only the comparison between the errors made in Protégé (least amount of errors) and OntoStudio (most errors), was statistically significant (chi-test with threshold of 95%).

Figure 4 provides an overview of the average time recorded for the completion of each multiple choice task during the study. The two visual rule editing tools, UI Rule Tool and OntoStudio, are on average faster than the other two tools. The differences in the recorded times were significant in three cases. Participants solved the tasks significantly faster with the UI Rule Tool than with Protégé or with Strelka.

The final recorded metric of the user study were the participants responses to the UEQ for the individual tools. The UEQ yielded the most significant results of the three metrics. Figure 5 shows the computed results for all tools in the six attributes and the overall score.

The UI Rule Tool achieved the highest score in every single attribute of the UEQ. All score differences between the UEQ and the other tools were significant,

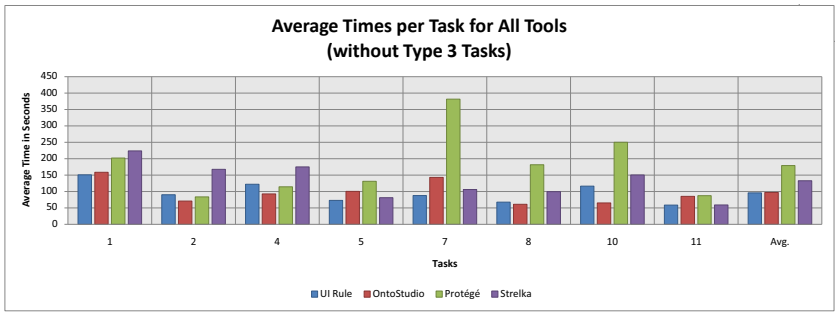

Fig. 4. Average time per task and tool for the multiple choice tasks.

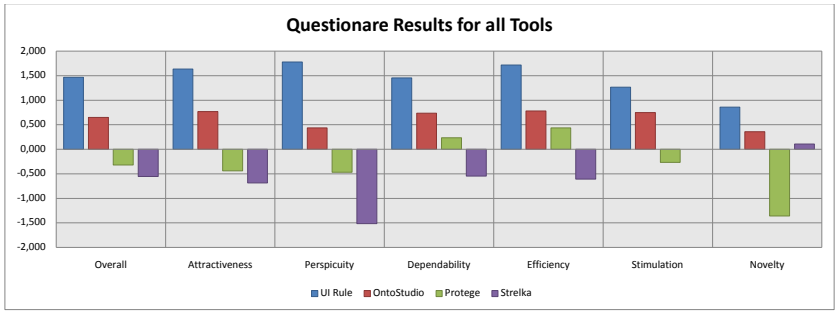

Fig. 5. Overview of the UEQ scores for all four tools.

with the exception of the novelty score in comparison to OntoStudio and Strelka. Especially the huge advantage in perspicuity was important for the tool, since perspicuity directly relates to how understandable and readable the integration rules are, which was one of the main goals of the tool. These overwhelmingly positive responses indicate that the participants of the user study clearly preferred the UI Rule Tool over the others for the given tasks.

Finally the UEQ score for the two participant groups, ontology experts and non-experts, were compared. Overall the UEQ scores were a lot closer for the expert group between tools. Especially Protégé and Strelka received significantly higher scores on average. A possible explanation for the large difference is that the expert group is more likely to be familiar with the notation of Protégé and Strelka.

5 Conclusion and Outlook

In this paper, we have introduced a tool which eases the creation and maintenance of ontologies and rules in a given software engineering setting, i.e., application integration on the user interface level. The key rationales were the reduction of complexity by moving from a general-purpose ontology and rule

editor to a tool specifically tailored to one particular task, i.e., the integration of applications.

We have compared our prototype tool to three state of the art tools in a user study. That study showed that our approach was better suited to the task of editing and understanding integration rules then the more generic competitors, especially in terms of task completion time and perceived user experience.

Some improvements are possible in the future. Especially the number of errors in all tools in the user study was higher then expected, suggesting that the rules are still complicated to understand and their role within the framework is at times unclear. Improvements that could be implemented include a split which divides the cause part of the rule into the two applications that are involved in the transaction. The first application is the one that triggers the event, the second the one is modified in response to the event. Since the rule can only modify the second one, the effect part of the rule should remain unsplit.

From the comments of the users, we have learned that the display of OR and NOT parts of rules should be changed, since it was misleading in some cases. Additionally due to the lower error rate in Protégé, a hybrid representation of visual and textual rule editing could be considered.

In summary, our prototype has shown that for specific software engineering tasks, it may be beneficial to step from general purpose ontology and rule editors to particularly tailored tools. As adequate tool support is an essential requirement for increasing the software engineers' acceptance of semantic technologies, and the results of our user study have shown significant improvements in terms of user experience, we are confident that this is a fruitful direction for future research.

References

1. Happel, H.J., Seedorf, S.: Applications of Ontologies in Software Engineering. In: Workshop on Semantic Web Enabled Software Engineering (SWESE) on the 5th International Semantic Web Conference (ISWC 2006), Athens, Georgia, November 5-9, 2006. (2006)
2. García-Barriocanal, E., Sicilia, M.A., Sánchez-Alonso, S.: Usability evaluation of ontology editors. Knowledge Organization **32**(1) (2005) 1–9
3. Paulheim, H., Probst, F.: Application Integration on the User Interface Level: an Ontology-Based Approach. Data & Knowledge Engineering Journal **69**(11) (2010) 1103–1116
4. Döweling, S., Probst, F., Ziegert, T., Manske, K.: SoKNOS - An Interactive Visual Emergency Management Framework. In Amicis, R.D., Stojanovic, R., Conti, G., eds.: GeoSpatial Visual Analytics. NATO Science for Peace and Security Series C: Environmental Security, Springer (2009) 251–262
5. Laugwitz, B., Held, T., Schrepp, M.: Construction and evaluation of a user experience questionnaire. In: Proceedings of the 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for Education and Work. USAB '08, Berlin, Heidelberg, Springer-Verlag (2008) 63–76