1 Bringing Complementary Models and People Together: A Semantic Wiki for Enterprise Process and Application Modelling

Enterprise modelling is the process of formally describing aspects of an enterprise, typically as process models, data models, resource models, competence profiles, etc. The availability of such enterprise models, expressed in computer-interpretable formal languages, is becoming an important factor for enterprises and organisations to make better and more flexible usage of the organisation’s knowledge capital and foster innovation. We describe here the design and applications of MoKi (the Modeling wiKi, [http://moki.fbk.eu](http://moki.fbk.eu)), and illustrate that complementary models can be created in the same modelling tool by people with complementary skills. MoKi is a semantic wiki for modelling enterprise processes and application domains, and intended to be used by knowledge engineers and domain experts, and everyone with skills in between.

State of the art modelling tools provide good support towards the creation of formal, computer interpretable models, but they suffer from two critical limitations: Different tools and different people are needed to model various enterprise aspects, and throughout all modelling stages.

- Different tools and modelling environments are provided for the specification of the different aspects of an enterprise. Most notably, the tools that are used to model business processes are usually separate and disconnected from the tools used to model the enterprise application model, which is more and more often encoded by means of ontology languages. This tool discontinuity produces two types of problems: Firstly, the modelling team needs to learn and interact with completely different tools. Especially in small environments like SMEs, where there may be scarce resources to allocate to modelling activities, this double-learning may lead to human resource problems. Secondly, the technical integration of the different parts (i.e., business processes and enterprise application domain) is not directly supported by the different tools and can therefore become unnecessarily complex and time-consuming.

- State of the art tools for ontology and process construction are often tailored to knowledge engineers, that is, to people who know how to create formal models. On the other hand, domain experts, that is, the people who know the domain to be modelled, often find such knowledge engineering tools scarcely usable. As a result the interaction between these different roles in the modelling team is regulated by a fairly rigid iterative waterfall paradigm in which domain experts produce or revise informal descriptions contained in textual documents and these informal descriptions need to be manually interpreted and transposed in a formal specification by a knowledge engineer with the obvious problems of duplication of efforts, wrong or misleading interpretations, and so on.

To overcome the limitations illustrated above, we have experimented with Semantic Web collaborative tools, and especially Semantic MediaWiki. The current
version of MoKi, an extension to Semantic MediaWiki, is the result of these experiments. We have used MoKi, in different development stages and customisations, in various application cases (see Section 1.2). MoKi addresses the above two limitations through (i) representing both tasks belonging to a business process and topics belonging to an enterprise application domain in a wiki based system, and (ii) representing both formal and informal content, and allowing different visual representations of the same content. A collage of views (tree view of concepts, process editor, a single concept) on MoKi content is depicted in Fig. 1.

Fig. 1 Collage of views on MoKi content.

1.1 A Conceptual System Description of MoKi

MoKi is an extension of Semantic MediaWiki, which itself is an extension of MediaWiki. Semantic MediaWiki adds labelled links and RDF interpretation of wiki content to MediaWiki, and MoKi adds OWL\textsuperscript{1} and BPMN\textsuperscript{2} interpretation of wiki content, as well as some knowledge engineering support, to Semantic MediaWiki. The decision to implement MoKi on top of an existing wiki was taken for several reasons. Firstly, a wiki environment was chosen since the wiki principles of (i) giving access to content (both with read and write permissions) to all, and (ii) making modifications of content as easy as possible for everybody, are well-aligned with the goal of MoKi to be a modelling environment not only for knowledge engineers.

\textsuperscript{1} Web Ontology Language, [http://www.w3.org/TR/owl2-overview](http://www.w3.org/TR/owl2-overview)

\textsuperscript{2} Business Process Modelling Notation, [http://www.bpmn.org](http://www.bpmn.org)
Secondly, most wiki environments support versioning and standard collaboration features like discussion threads or comments. Thirdly, most wikis are web-based, which enables geographically distributed modelling, and allows textual as well as multimedia content, i.e. everything which can be published on the web can be published in a wiki. Finally, most potential users of a system such as MoKi can be expected to know how a wiki looks and feels, and a large portion of these people also know how to actively contribute to wiki content. This is partly due to the large success of the online encyclopaedia Wikipedia, but as well to the arrival of wiki (and other semantic web) technology in the corporate world, as observed e.g. in [Gissing and Tochtermann, 2007, Schachner and Tochtermann, 2008]. Extending traditional wikis, semantic wikis [Schaffert et al., 2008] already provide the basic infrastructure to deal with structured data in addition to traditional human-readable content-types like text or multimedia. Thus, they are technically well suited to accommodate the results from informal as well as formal modelling activities. In choosing a particular semantic wiki upon which to build MoKi, we decided for Semantic MediaWiki, because MediaWiki has a large community of developers and users, it is easily extensible through plugins and because it is generic insofar as it is not adapted to any specific application scenario.

1.1.1 The “one page - one element” design principle

The basic design principle of MoKi is that every model element corresponds to a wiki page. For application domain modelling, the relevant model elements are concepts, individuals and properties/relations, while for business processes the relevant model elements are processes. Each model element is internally given a formal meaning. For instance, a concept is given the meaning of a description logic concept, i.e. it is a unary predicate and can be interpreted as a set of entities for which the unary predicate holds. It is essential to be clear about this internal interpretation of model elements, since this forms the basis of technically dealing with imports from various knowledge representation formalisms and exports to various knowledge representation formalisms. A MediaWiki category is used to distinguish different kinds of model element. Table 1 shows a synopsis of the model elements available in MoKi, their formal interpretation as well as in which kind of model the corresponding model element is expected to be used.

For every kind of model element supported by MoKi, a specific template is provided. Figure 2 shows an excerpt of an already filled-out concept description. The implementation of templates is based on the Semantic Forms extension to Semantic MediaWiki that allows users to define forms for editing pages and templates to store semantic data. From a user perspective, a template is displayed as a list of fields which have to be filled out in order to describe the model element. Fields can differ between model elements. Conceptually, a template asks the user for information which is typically needed for a specific kind of model element.

3 For the sake of simplicity, in MoKi we decided to not explicitly distinguish between atomic and not atomic processes (i.e. composed of two or more sub-processes).
Table 1 Category names in MoKi for designating different kinds of model elements in MoKi. "Type of Model" refers to the type of model in which such a model element is expected to occur.

<table>
<thead>
<tr>
<th>Category</th>
<th>Model Element</th>
<th>Interpretation</th>
<th>Type of Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Domain model&quot;</td>
<td>Concept</td>
<td>DL Concept</td>
<td>Domain ontology</td>
</tr>
<tr>
<td>&quot;MokiProperty&quot;</td>
<td>Property / Relation</td>
<td>DL Role</td>
<td>Domain ontology</td>
</tr>
<tr>
<td>&quot;Individuals&quot;</td>
<td>Individual</td>
<td>DL Nominal</td>
<td>Domain ontology</td>
</tr>
</tbody>
</table>

**Example:** When describing a domain concept, it is typical to ask "What is a superconcept?", i.e. what are more general notions than the currently described concept, in which categories does it fall?

Obviously, the users are not necessarily required to fill all the fields of a page when describing a specific model element. Additionally, some of the templates support the possibility for users to add custom defined fields.

**Example:** A user may want to describe the concept "Project", and then express that a project is typically managed by a person. In this case, the user can add a new field to the concept "Project" which is called "managed-by" and fill it with the concept "Person".

The use of templates allows for an easy customization of MoKi to hold additional kinds of models other than domain ontologies and business processes. Although such a customization can not yet be done solely at the user interface - some programming in PHP is required in order to define the formal meaning of the fields in a new template and to add import and export support for the new model elements - it requires some minimal software development effort.

1.1.2 Functionalities

In addition to the features offered by MediaWiki and Semantic MediaWiki, MoKi provides functionalities for importing/exporting models, navigating, editing, and validating models. In this section we briefly illustrate these functionalities. A more extensive description can be found at the MoKi web site and in [Ghidini et al., 2009].

The first group of functionalities concern the import/export of models. The application domain model can be exported into OWL 2, RDF/XML Rendering, and the business process model can be exported into BPMN, according to the Oryx eRDF (embedded RDF) serialization. Import of application domain models is possible from OWL 2 files. In addition MoKi also supports importing knowledge from less structured sources. Hierarchies of concepts can be imported by writing down the hierarchy as a simple ASCII list of terms, where indentation indicates the hierarchy. Knowledge can be imported from text documents by means of a term extraction functionality. Extracted terms can be added with one click as (candidate) concepts into MoKi domain ontology. This functionality uses at the backend the KnowMiner framework, a Java-based framework for knowledge discovery [Granitzer 2006, Klieber et al. 2009].
Fig. 2  Excerpt of a filled-out concept template in MoKi, shown in the figure for a concept called “Workshop”. The fields in the Annotation, Hierarchical structure and Notes boxes are available for all domain concepts. The fields in the Properties box are added by the ontology engineer specifically for each domain concept, as e.g. “hasParticipant” and “isOrganizedBy” for the concept “Workshop”.
The second group of functionalities concern the navigation of models. In any information or knowledge management system, navigation through content is vital to its success: the best content is useless if it cannot be easily accessed. MoKi content can be accessed through standard MediaWiki functionalities like search, or through typing in the URL of a single page in the address bar of the browser. Apart from this, there are the possibilities to get lists of model elements in a tabular style (where each element is shown alongside some relevant information characterizing it), and in graphical visualizations. For the ontology part, the graphical visualization concerns a tree-view rendering of the specialisation (Is-A) and mereological (Part-Of) hierarchies, and a tree-view rendering of the individuals concept membership role. Hierarchies are also editable by drag-and-drop features. For the process part, a graphical visualisation of the process workflow and of the different subprocesses it comprises is available in the process element page.

The third group of functionalities concern the editing of models. Editing activities relate to single model elements and concern the creation of model elements, their repeated editing (among this renaming the model element, or even changing its type), and the deletion of model elements. Depending on the type of element, the corresponding template is loaded when a model element is created or edited.

A fourth important group of functionalities concern the evaluation of models (described in detail in [Pammer, 2010, p93-121]). Whatever purpose models are created for, they need to be evaluated in order to ensure that they will serve their intended use. The current version of MoKi supports ontology evaluation through: (i) a models checklist, (ii) a quality indicator, (iii) the ontology questionnaire and (iv) through displaying assertional effects. The models checklist is a list of characteristics that typically point to oversights and modelling guidelines, and automatically retrieves elements that fit the characteristics.

Example: One point on the checklist is “Orphaned concepts”, i.e. concepts that have no super- or subconcepts, have no parts and are not part of anything. These are often concepts left-over from brainstorming or another earlier modelling iteration.

The quality indicator is displayed on the page of all elements and visualises the completeness and sharedness of the corresponding element as a bar that grows from “short and red” to “long and green”. Both completeness and sharedness are heuristic measures, where the first captures how much information (verbal, structural) about the element is available while the second captures how many people have contributed to the description of the element. The ontology questionnaire displays inferred knowledge, i.e. statements that can be derived from the models contained within MoKi, and provides explanations for them, as well as the possibility to remove them. In case explicitly made statements are deleted in order to remove an undesired inference, the ontology questionnaire also displays side-effects, i.e. all inferences that will be lost alongside. Assertional effects [Pammer et al., 2009] are displayed on concept and property pages directly after an ontology edit that causes one or more assertional effects. It is called questionnaire in order to point out that domain experts should go through inferred statements like going through a questionnaire and asking the question “Is this statement correct?”.
1.2 Application and Experiences with MoKi

MoKi has been applied in several scenarios, in varying stages of development and sometimes with small customisations.

Model tasks and topics for work-integrated learning

Two early versions of MoKi have been successfully applied [APOSDELE Deliverable 1.6 2009, p.21-25;32-47] to develop enterprise models in six different domains: Information and Consulting on Industrial Property Rights (94 domain concepts and 2 domain roles; 13 processes), Electromagnetism Simulation (115 domain concepts and 21 domain roles; 13 processes), Innovation and Knowledge Management (146 domain concepts and 5 domain roles; 31 processes), Requirements Engineering (the RESCUE methodology) (78 domain concepts and 2 domain roles; 77 processes), Statistical Data Analysis (69 domain concepts and 2 domain roles; 10 processes) and Information Technology Infrastructure Library (100 domain concepts and 2 domain roles; no processes). The enterprise models were created for the purpose of initialising and serving as the knowledge backend of a system for work-integrated learning [Lindstaedt et al. 2007, www.aposdle.org]. The modelling activities involved people with different modelling skills and levels of expertise of the application domains, and located in different places all over Europe.

The experiences of using the first version of MoKi, essentially Semantic MediaWiki without much additional convenience functionality for modelling, are described in [Christl et al. 2008]. These experiences provided the motivation for adding convenience, i.e. modelling support, functionality to Semantic MediaWiki in order to support enterprise modelling for processes and application domains. A qualitative evaluation on the entire modelling process including the usage of early versions of MoKi applied in APOSDELE is documented in [APOSDELE Deliverable 1.6 2009].

The evaluation of the second version of MoKi, much the same as the current open-source version in functionality if not in design and robustness, took the form of structured interviews. The interviews were composed both of open and closed questions. Interview partners were all involved domain experts, on-site knowledge engineers working at the application partners’ and external knowledge engineers providing additional modelling skills where necessary. Regarding MoKi, questions about its usability, the support for collaboration, the usefulness of its functionalities, the homogeneity of the modelling environment for modelling the different aspects (domain model and processes), etc, were asked. Note that before using MoKi, the users already tried modelling with Semantic MediaWiki for informal, integrated models (domain and business process) and Protégé and a YAWL editor for the formal models.

According to the results of the questionnaire, the users highly appreciated the form-based interface of MoKi, and the fact that they were able to participate in the creation of the models without having to know any particular syntax or deep knowledge engineering skills. Thus, MoKi was perceived as an adequate tool to actively involve
domain experts in the modelling process. From the questionnaire it further emerged that also people with some knowledge engineering skills found MoKi as comfortable to use as other state-of-the-art modelling tools. The answers show that MoKi helped the users in structuring and formalizing their knowledge in a simple, intuitive and efficient manner. Particularly appreciated have been the functionalities, in particular the graphical ones, which allow to navigate through the content defined in the models. Finally, the users have found MoKi, in its characteristic as web-based modelling environment, quite useful to produce a single model with a geographically distributed modelling team.

**Collaboratively build an ontology for annotating content**

The current version of MoKi has been used by a team of knowledge engineers and domain experts to collaboratively build an Organic Agriculture and Agroecology Ontology (61 domain concepts, 30 domain roles, and 222 individuals) within the FP7 EU-project Organic.Edunet (http://www.organic-edunet.eu/). This experience was perceived as positive enough to use MoKi as the central modelling tool in the follow-up EU project, Organic.Lingua.

**Maintain a glossary in a project**

The current version of MoKi is being used within the FP7 MIRROR project (www.mirror-project.eu/) to develop and maintain a common glossary.

**Model clinical protocols in Asbru**

Although MoKi as presented here is tailored to the development of ontologies and business processes, the applicability of MoKi goes beyond typical enterprise modelling. A preliminary and customized version of MoKi that supports the modelling of clinical protocols in the ASBRU modelling language is described in [Eccher et al., 2008]. This version of MoKi, called CliP-MoKi, provides support for modelling the key elements of an Asbru model (e.g. plans, parameters, http://www.asbrusoft.com/) as wiki pages, and for exploring the models created according to the mechanisms for structuring knowledge provided by the language (e.g. the plan/plan children decomposition).

**1.3 Discussion**

MoKi supports a variety of knowledge engineering activities, such as knowledge acquisition, informal modelling, formal modelling, and evaluation of models at var-
ious stages. Knowledge acquisition is supported through the term extraction functionality. The term extraction functionality is state-of-the-art, which unfortunately means that it supports only a limited number of languages (English and German currently) and that the quality of results depends a lot on the corpus it is given. Informal modelling is supported via the possibility to import simple hierarchies, via the prominently placed possibility to verbally describe and document (“Description”, “Synonym(s)”) all kinds of model elements, as well as via the possibility to richly document (“Free notes”) all kinds of model elements in all formats which can be held in a webpage. Evaluation of informal aspects of the models contained in MoKi is supported, in that for instance elements with no verbal description are explicitly pointed out to MoKi users (models checklist) and by giving direct feedback on completeness and sharedness on element pages (quality indicator). Formal modelling is supported from a user perspective by providing form fields with auto-fill functionality to ontology engineers. Fields are given a formal meaning, which is the basis of technically supporting formal modelling. Evaluation of the formal models is supported through listing and explaining inferences (ontology questionnaire) and through displaying the effects of editing formal axioms on data (assertional effects). However, on the formal modelling side, MoKi does not (yet) support the full expressivity of OWL 2. Most importantly, it does not yet support complex concepts. Additionally, MoKi versions that support both domain and process modelling, and a MoKi version that supports modelling clinical protocols, exist. Thus, MoKi does improve on the first limitation of many existing modelling tools and decreases the gap in tool support between tools for different enterprise aspects, and for different knowledge engineering activities.

Most requirements for collaborative knowledge construction tools, discussed for instance in [Noy et al. 2008] and in [Tudorache et al. 2008], are easily met by MoKi, merely by its being implemented on top of MediaWiki. The requirements on collaboration features satisfied by MoKi are distributed access to a shared ontology, version control, user identity management and tracking the provenance of information and discussion on model elements. Fine-grained access control is not possible in MoKi, and collaborative protocols that involve rating or voting are not supported in MoKi either. Coherence is ensured mostly through keeping the informal and formal model element descriptions in one place, i.e. in one wiki page. Inconsistencies between the natural language text or rich content on the one hand and the formal descriptions on the other hand are not detected. Indeed, this would exceed the state-of-the-art in natural language understanding, and even more so in multimedia understanding. However, coherence is supported in another slightly roundabout way, namely through the “watch” functionality of MediaWiki. Through this functionality, users can be notified if changes occur at a wiki page, which in MoKi means changes concerning a model element. Like this, both domain experts and ontology engineers can easily detect changes to parts of the ontology in which they hold an interest. Concerning the second limitation of most existing modelling tools, namely that users with different knowledge engineering skills (knowledge engineers, domain experts) are often not able to work within the same modelling environment, MoKi already now is able to hold rich, informal content as any MediaWiki can, as
well as formal content that can be exported into formal domain and business process modelling languages (OWL 2 and BPMN respectively).

References


