8th Workshop on

Knowledge Engineering and Software Engineering (KESE8)

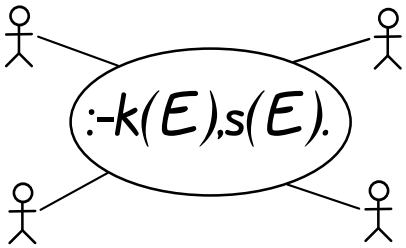
at the

20th biennial European Conference on Artificial Intelligence (ECAI 2012)

Montpellier, France, August 28, 2012

Grzegorz J. Nalepa, Joaquín Cañadas, Joachim Baumeister

(Editors)



Technical Report TR-2012/1,

Department of Languages and Computation. University of Almería,

Almería, Spain, 2012

The KESE Workshop Series is available online: http://kese.ia.agh.edu.pl

Preface

Grzegorz J. Nalepa, Joaquín Cañadas and Joachim Baumeister

AGH University of Science and Technology
Kraków, Poland
gjn@agh.edu.pl
—
Dept. of Languages and Computation. University of Almeria.
Agrifood Campus of International Excellence, ceiA3. Almeria, Spain
jjcanada@ual.es
—
denkbares GmbH, Friedrich-Bergius-Ring 15, 97076 Würzburg, Germany
joachim.baumeister@denkbares.com

Intelligent knowledge-based systems have been successfully developed in many domains. They employ techniques and tools from the fields of knowledge engineering and software engineering. Thus, declarative software engineering techniques have been established in many areas, such as knowledge systems, logic programming, constraint programming, and lately in the context of the Semantic Web and business rules.

The eight workshop on Knowledge Engineering and Software Engineering (KESE8) was held at the ECAI 2012 (The European Conference on Artificial Intelligence) organized by European Coordinating Committee for Artificial Intelligence in Montpellier, France, and wants to bring together researchers and practitioners from both fields of software engineering and knowledge engineering, as well as the Semantic Web community. The intention was to give ample space for exchanging latest research results as well as knowledge about practical experience. Moreover the workshop endeavors to promote the use of KE techniques in SE problems, where significant benefits can be derived from their use. The general goal of the workshop was to show how the KE techniques can provide practical solutions in SE issues. On the other hand, the influence of SE methods and tools on the practical design of KBS within KE.

The principal focus of the Workshop was on methods of KE rooted in the symbolic logic-based representations and their novel applications in Software Engineering. Moreover, a synergistic use and development of these KE methods together with recent formalized and declarative SE methods, including Model-Driven Architecture and Development, ontological modelling as well as Business Process modelling was emphasized. Finally, the studies of the impact of these SE methods on the classic KE development processes were welcomed.

Topics of the workshop are generally related to the applications of symbolic KE techniques in SE as well as the use of KE in the SE practice. Specific topic the areas include but are not limited to:

- Knowledge and software engineering for the Semantic Web
- Ontologies in practical knowledge and software engineering
- Business Rules design, management and analysis

- Business Processes modelling in KE and SE
- Practical knowledge representation and discovery techniques in software engineering
- Agent-oriented software engineering
- Knowledge base management in KE systems
- Evaluation and verification of KBS
- Practical tools for KBS engineering
- Process models in KE applications
- Software requirements and design for KBS applications
- Declarative, logic-based, including constraint programming approaches in SE

This year, we received contributions focussing on different aspects of knowledge engineering and software engineering, promoting the influence and benefits of their joined use

Hatko et al. present a set of coverage metrics to assess the thoroughness of testing efforts for clinical guidelines in Diaflux language, providing some novel metrics (coverage metrics) and suggests the use of a graphical method (city metaphor) to visualize the coverage levels.

Águila and Sagrado introduce a metamodel and an UML profile for modeling of Bayesian Networks, enabling integration with UML diagrams and introducing such probabilistic graphical models in the MDA context.

The contribution of Giurca et al. elaborates a preference logic framework for conjoint analysis that can cope with the non-transitivity and inconsistency in preference data, useful when capturing phycological phenomena such as change or irrationality (inconsistency) as well as when formal explanations of decisions need to be computed.

A proposal for classifying errors in ontologies, with the aim of using such framework to map errors identified in automatic ontology building processes, is defined by Gherasim et al. providing a taxonomy of problems impacting the quality of automatically built ontologies and a classification with possible anomalies.

Kluza and Kaczor emboss the issue of a normalized Business Process Model and Notation (BPMN) modelling technique, presenting a survey on BPMN models' equivalences and several approaches to simplify BPMN models.

Kaczor and Nalepa compare two rule approaches, logically well defined rule processing systems like XTT2 and application driven intuitive popular rule-based tool such as CLIPS, in order to gain insights with respect to their applicability for business rule interchange.

Template-based Extensible Prototyping approach is introduced by Freiberg and Puppe, to perform usability evaluations of user interfaces in knowledge-based systems, useful for the validation of the collected knowledge.

Ligęza et al. describe a social platform called Social Threat Monitor (STM) aimed at improving safety of local communities in urban environment, managing and monitoring social threats through the collaborative knowledge engineering.

This year we also encouraged to submit tool presentations, i.e., system descriptions that clearly show the interaction between knowledge engineering and

software engineering research and practice. At the workshop, one presentation about current tools was given: Baumeister et al. present KnowWE, a semantic wiki providing collaborative platform for knowledge acquisition and testing. It uses different types of knowledge ranging from semantically annotated text to strong problem-solving knowledge, adapting continuous integration approach of software engineering for knowledge engineering.

The organizers would like to thank all who contributed to the success of the workshop. We thank all authors for submitting papers to the workshop, and we thank the members of the program committee as well as the external reviewers for reviewing and collaboratively discussing the submissions. For the submission and reviewing process we used the EasyChair system, for which the organizers would like to thank Andrei Voronkov, the developer of the system. Last but not least, we would like to thank the organizers of the ECAI 2012 conference for hosting the KESE8 workshop.

Grzegorz J. Nalepa
Joaquín Cañadas
Joachim Baumeister

Workshop Organization

The 8th Workshop on Knowledge Engineering and Software Engineering
(KESE8)
was held as a one-day event at the
20th biennial European Conference on Artificial Intelligence
(ECAI 2012)
on August 28, 2012, Montpellier, France.

Workshop Chairs and Organizers

Grzegorz J. Nalepa, AGH UST, Kraków, Poland
Joaquín Cañadas, University of Almeria, Spain
Joachim Baumeister, denkbares GmbH, Germany

Programme Committee

Isabel María del Águila, University of Almeria, Spain
Klaus-Dieter Althoff, University Hildesheim, Germany
Joachim Baumeister, denkbares GmbH, Germany
Joaquín Cañadas, University of Almeria, Spain
Jesualdo Tomás Fernández-Breis, University of Murcia, Spain
Adrian Giurca, BTU Cottbus, Germany
José M. Juarez, University of Murcia, Spain
Jason Jung, Yeungnam University, Korea
Rainer Knauf, TU Ilmenau, Germany
Pascal Molli, University of Nantes - LINA, France
Grzegorz J. Nalepa, AGH UST, Kraków, Poland
José Palma, University of Murcia, Spain
Alvaro E. Prieto, Univesity of Extremadura, Spain
José del Sagrado, University of Almeria, Spain
Dietmar Seipel, University Würzburg, Germany
Rafael Valencia-García, University of Murcia, Spain

External reviewers
Hala Skaf-Molli, University of Nantes - LINA, France
Ludwig Ostermayer, University Würzburg, Germany

Table of Contents

CoverageCity: Test Coverage for Clinical Guidelines

Reinhard Hatko[1] and **Joachim Baumeister**[2] and **Frank Puppe**[3]

Abstract. In this paper, we introduce various metrics for test coverage of clinical guidelines, modeled in the graphical language DiaFlux. Additionally, an intuitive visualization method supports the process of test creation and communicating the reached coverage levels to medical experts involved in the authoring of the guideline. The goal is to reach a sufficiently high test coverage to assure patient safety under all circumstances.

1 Introduction

Testing is an important step in the development of any software artifact, be it a program, a knowledge-based system, or a clinical guideline. The two most prevalent testing strategies in Software Engineering are *black-box* and *white-box* testing. The former approach is unconcerned with the actual implementation and derives the test cases solely from the underlying specification. The latter one, in contrast, allows to create tests based on the implementation and to examine it during execution of the tests. This introspection enables to capture which basic elements of the tested artifact were executed - and thus *covered* - by a test suite. Different metrics of *Test Coverage* were developed to objectively measure and assess the thoroughness of such testing efforts. In classic Software Engineering, metrics have been defined to assess the coverage of, e.g., methods of a program, statements of a method, taken decisions of control-flow, and so on [16].

Hence, the benefit of coverage metrics - and their proper visualization - is two-fold, increasing the effectiveness and efficiency of the test creation process: First, they help to avoid the creation of redundant tests. Second, they can be used to identify untested elements.

Both are also important aspects in the area of knowledge-based system in general and computerized clinical guidelines in particular. The creation of test cases for a clinical guideline will most likely involve both parties, the medical expert and the knowledge engineer. It is thus an expensive task, which should be completed efficiently. Though, the overall goal is to create a guideline, that assures patient safety under all circumstances, which can best be guaranteed by a thorough test suite. This is especially important in the area of automated guidelines, which are applied by closed-loop devices. They act autonomously on a patient to improve her state, without requiring constant supervision by a clinical user.

For the interpretation of coverage metrics a visualization is helpful, especially to find untested elements. Test coverage of software programs most usually is visualized by some kind of syntax highlighting, by coloring, e.g., the executed statements. Though also graphical representations were developed, e.g., [11]. We adopted a visualization method from a related area in (object-oriented) Software Engineering, namely *Software Metrics*. They are used to assess code quality with respect to structural properties of classes, e.g., number of methods, number of members, lines of code, and so forth. Those metrics are purely static, not involving the execution of the program itself. They can graphically be visualized as a *CodeCity* [22] to determine design flaws.

In this paper, we introduce various metrics to determine the test coverage of clinical guidelines, modeled in the graphical language DiaFlux. Furthermore, we adapted the city metaphor by creating a *CoverageCity* for communicating the reached coverage levels to the involved medical experts in an accessible manner.

The rest of this paper is structured as follows: In the next section we give a short introduction into the graphical language DiaFlux for clinical guidelines. Section 3 presents coverage metrics for DiaFlux models. Following, in Section 4, we present the results of a case study. Finally, we conclude the paper with a summary and an outlook.

2 The DiaFlux Guideline Language

Clinical guidelines are an accepted means to improve patient outcome. Therefore, they offer a standardized treatment, based on evidence-based medicine. They are developed for several decades. In their beginnings, they were solely text-based documents that relied on the proper application by clinicians. The ongoing computerization and data availability, also in domains with high-frequency data as, e.g., Intensive Care Units (ICUs), allows for an automation of guideline application by medical devices.

Several formalisms for Computer-Interpretable Guidelines were developed, every one with its own focus, like shareability between institutions [4] or decision support. Most of them are graphical approaches, that employ a kind of *Task-Network-Model* to express the guideline steps [17]. However, in the area of (semi-)closed-loop devices, rule-based approaches are predominant, e.g., [12, 15].

A downside of rule-based representations is their lower comprehensibility compared to graphical ones. This especially holds true, as medical experts are most usually involved in the creation of guidelines. Therefore, we have developed a graphical guideline formalism called DiaFlux [7]. Its main focus lies on the direct applicability and understandability by domain specialists.

2.1 Application Scenario

The main application area of DiaFlux are mixed-initiative devices that continuously monitor, diagnose and treat a patient in the setting of an ICU. Such closed-loop systems interact with the clinical user during the process of care. Both, the clinician and the device, are able to initiate actions on the patient. Data is continuously available as a

[1] University of Wuerzburg, Germany, email: hatko@informatik.uni-wuerzburg.de
[2] denkbares GmbH, Germany, email: joachim.baumeister@denkbares.com
[3] University of Wuerzburg, Germany, email: puppe@informatik.uni-wuerzburg.de

result of the monitoring task. It allows for repeated reasoning about the patient state, and to carry out appropriate actions to improve her condition, if necessary.

2.2 Language Description

To specify a clinical guideline, two different types of knowledge have to be effectively combined, namely declarative and procedural knowledge [6]. The declarative part contains the facts of a given domain, i.e., findings, diagnoses, treatments and their interrelation. The knowledge of how to perform a task, i.e., the correct sequence of actions, is expressed in the procedural knowledge. It is responsible for the decision which action to perform next, e.g., asking a question or carrying out a test, in a given situation. Each of these actions has a cost (monetary or associated risk) and a benefit (like information gain or therapeutic effect) associated with it. Therefore, the choice of an appropriate sequence of actions is mandatory for efficient diagnosis and treatment.

In DiaFlux models, the declarative knowledge is represented by a domain-specific ontology, which contains the definition of findings and solutions. This application ontology is an extension of the task ontology of diagnostic problem solving [3]. The ontology is strongly formalized and provides the necessary semantics for executing the guideline. Like most graphical guideline languages, DiaFlux employs flowcharts as the Task-Network-Model. They describe decisions, actions and constraints about their ordering in a guideline plan. These flowcharts consist of nodes and connecting edges. Nodes represent different kinds of actions. Edges connect nodes to create paths of possible actions. Edges can be guarded by conditions, that evaluate the state of the current patient session, and thus guide the course of the care process. Figure 1 shows a module of an exemplary guideline for diagnosing weight problems.

In the following, we informally describe the most important language elements:

- **Test node:** Test nodes represent an action for the acquisition of data during runtime. This may trigger a question, the user has to answer, or data to be automatically obtained by sensors or from a database.
- **Solution node:** Solution nodes are used to set the rating of a solution based on the given inputs. Established solutions generate messages for the clinical user and can, e.g., advice him to conduct some action.
- **Wait node:** Upon reaching a wait node, the execution of the protocol is suspended until the given period of time has elapsed.
- **Composed node:** DiaFlux models can be hierarchically structured. Defined models can be reused as modules, represented by a composed node in the flowchart using it.
- **Abstraction node:** Abstraction nodes offer the possibility to create abstractions from available data. These values can then be used for therapeutic actions by influencing the settings of the host device.

2.3 Guideline Execution

The execution engine for DiaFlux models is intended for, but not limited to, closed-loop devices, that provide data from sensors or manually entered by the clinical user and carry out therapeutic actions on the patient, i.e., changing device settings in certain ranges. The architecture of the DiaFlux guideline execution engine consists of three components. First, a knowledge base, that contains the application ontology and the flowcharts. Second, a blackboard, that stores all findings about the current patient session. Third, a reasoning engine, that executes the guideline and carries out its steps, depending on the current state as given by the contents of the blackboard. Therefore, the reasoning engine is notified about all findings, that enter the blackboard. A more thorough introduction to the DiaFlux language and its execution engine is given in [7].

The execution of the guideline is time-driven. The reasoning starts by acquiring data and by interpreting this data. The results are written to the blackboard. Then, the guideline is executed. This involves making decisions and possibly the generation of hints to the user and therapeutic actions by the device. Finally, the time of the next execution is scheduled, pausing the execution until that instance in time, waiting for the effects of the therapeutic interventions to take place.

3 Test Coverage of Clinical Guidelines

Verification and validation of a clinical guideline are important steps in its development. That way, patient safety shall be assured under all circumstances. Verification usually consists of formal methods, proofing, that a given guideline is free of internal inconsistencies and incompleteness. Normally, these kinds of checks can be performed without executing the guideline. An overview of verification methods applied to clinical guidelines is given in [10]. Anomaly detection for DiaFlux models is described in [8]. In contrast, the validation of a guideline usually involves its execution by a set of test cases (i.e. a test suite) and comparing the actual results against the expected ones [2]. The thoroughness of such empirical testings can be determined by different metrics of test coverage.

In conventional software engineering (SE), test coverage (also known as *code coverage*) is a well-established technique to measure how well a piece of software is exercised by a test suite. Often, the reached level of coverage is also used as an indicator for the quality of the tested program, as tested elements are less likely to contain errors than untested ones. Coverage criteria have been defined on different levels of granularity, from the method-level down to single statements, or even parts of them (so called *condition coverage*) [16]. In the field of AI research, coverage measures have been proposed for rule-based systems. In this case, the results of such a coverage analysis can, e.g., be used to prune the rule base [1]. For graphical model representations, coverage measures have been proposed, e.g., for business processes [14], taking their specifics into account, for example, the coverage of *fault handlers*.

In general, employing coverage metrics during the creation of a test suite may help improving it in terms of minimality and completeness. While a high coverage of the object under test is worthwhile, this should be accomplished with the possibly minimal set of test cases, as test creation is a difficult and costly task. This especially holds true for the test creation of clinical guidelines, as it may involve knowledge engineers as well as domain specialists like medical experts. Therefore, adopting the mentioned techniques for clinical guidelines and their graphical representations, offers the possibility to improve this process.

In the following, we introduce coverage metrics on different levels of granularity to assess the test coverage of a DiaFlux guideline. Such a guideline usually is modularized in self-contained modules, i.e. flowcharts. To represent this modularization also by the coverage metrics, we define most of them over the elements of individual flowcharts, hence the restriction of nodes and edges to those of a single one. This focusing alleviates the increase of coverage by additional tests, as deficiencies are easier to spot.

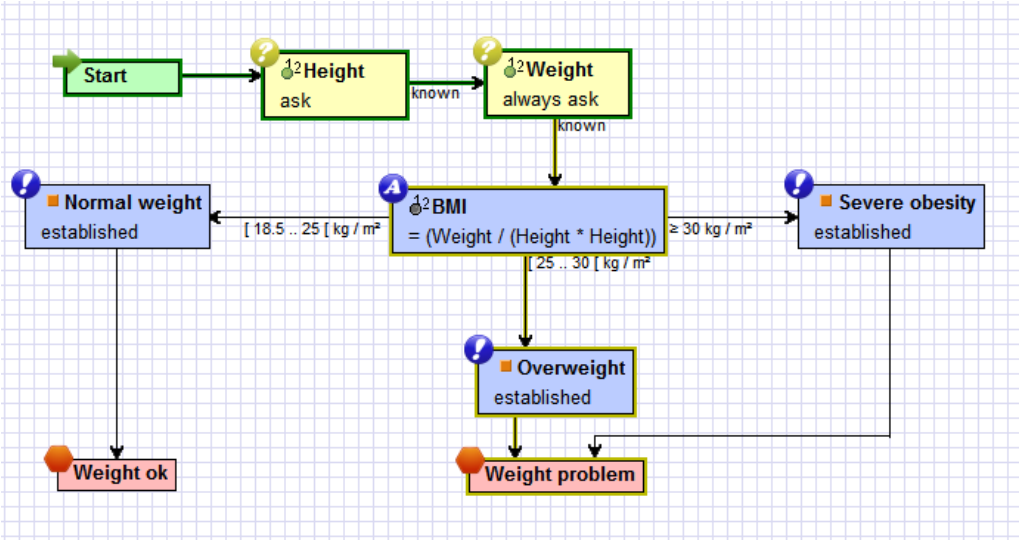


Figure 1. A module of a DiaFlux guideline for diagnosing weight problems.

3.1 Flowchart Coverage

Flowchart Coverage is defined as the ratio of the number of flowcharts that are executed by a test suite and the overall number of flowcharts in the guideline.

Let F be the set of DiaFlux models in guideline G, and F_T^{ex} be the set of flowcharts executed by test suite T. Then, the Flowchart Coverage FC_T of G is given by:

$$FC_T(G) = \frac{|F_T^{ex}|}{|F|}$$

This metric only gives a bird's eye view of the testing situation. It can be used to guarantee at least some testing in all areas of the guideline during the starting phase of test creation. As it is a very coarse-grained measurement, a FC_T value of 100% should be aimed at. Otherwise, major parts of the guideline remain untested. In SE, the equivalent metric is *function coverage*, which reports, if every function of a program has been tested.

3.2 Node Coverage

Nodes represent the elementary steps of a guideline. A node being covered by a test suite, means, that its associated guideline step has been carried out at least once during the execution of the test suite.

Let $N_f, f \in F$ be the set of nodes of the flowchart f, and $N_{f,T}^{ex}, f \in F$ be the set of nodes of the flowchart f, that are executed by test suite T. The according *Node Coverage* metric NC_T of a flowchart f for a given test suite T can then be calculated as:

$$NC_T(f) = \frac{|N_{f,T}^{ex}|}{|N_f|}, f \in F$$

Similar to *Flow Coverage*, a NC_T level of 100% should be reached for every flowchart in the guideline, as untested nodes can enact actions with unforeseen effects. This metric is equivalent to *Statement Coverage* in classic SE, which reports, if every statement of a tested function has been executed.

3.3 Edge Coverage

Edges are used to create the control flow of a flowchart, by defining paths of possible sequences of guideline steps. Every node can have several outgoing edges. Each of these edges can be guarded by a condition, to select the appropriate successor node, depending on the outcome of each guideline step. Therefore, the *Edge Coverage* metric reports, if all possible outcomes of the guideline steps - in terms of the equivalence classes that are defined by the edge guards - have been considered within the test suite.

Let $E_f, f \in F$ be the set of edges of flowchart f, and $E_{f,T}^{ex}, f \in F$ be the set of edges of the flowchart f, that are executed by test suite T. Then, EC_T is defined as the number of activated edges of a flowchart f to their overall number, executing a given test suite T:

$$EC_T(f) = \frac{|E_{f,T}^{ex}|}{|E_f|}, f \in F$$

As edges connect nodes, an *Edge Coverage* subsumes the *Node Coverage* of the according flowchart. This metric can be compared to Decision Coverage in SE, that keeps track, if each decision in a program under test (e.g., `if`- and `switch`-statements) has at least once been taken and once not.

3.4 Condition Coverage

An edge guard may not be an atomic condition, but consist of several sub-conditions, connected by boolean operators. For such non-atomic guards, *Edge Coverage* gives no detailed information about which of its sub-conditions were satisfied and which were not. This is of special interest, if the sub-conditions are joined by an *OR*-operator. In this case, every possible combination of atomic conditions, that can fulfill the overall condition, have to be tested.

A more detailed view about this issue is given by *Condition Coverage*. It checks, if every atomic condition has once been satisfied and once not. In classical SE, several different metrics for this issue have been developed, e.g. *Modified Condition / Decision Coverage*

[5]. As those can directly be applied to the guarding conditions of edges, we will not further elaborate on this issue.

3.5 Path Coverage

A path through the guideline consists of consecutive nodes and edges. Such a path can be seen as the execution of decisions and actions for a given clinical scenario. In Software Engineering, it usually is not possible to reach a full path coverage, as soon as loops are involved, as each number of iterations results in an additional path. In clinical guidelines, there are no loops of an unlimited number of iterations, as, for instance, some time has to pass, until an action can be repeated. Nevertheless, the number of paths through the complete guideline throughout multiple nested DiaFlux models most likely exceeds the possibilities of test creation. Given a proper modularization, each flowchart is responsible for a specific aspect of a guideline. Each path through such a single flowchart can be seen as one specific scenario concerning this aspect. Therefore, we assess each flowchart independently, and define an according *Path Coverage* metric over the paths of each individual flowchart.

Let $P_f, f \in F$ be the set of paths through flowchart f, and $P^{ex}_{f,T}, f \in F$ be the set of paths through flowchart f, that are executed by test suite T. Then, PC_T is calculated as the number of paths taken through flowchart f, by the execution of test suite T, divided by the total number of paths:

$$PC_T(f) = \frac{|P^{ex}_{f,T}|}{|P_f|}, f \in F$$

Even with a proper modularization given, not every modeled path may be enactable, due to implicit dependencies between the guideline steps. If certain combinations of decisions and actions on a single path exist, that can not occur, the targeted value of Path Coverage has to be decreased accordingly.

As a path consists of consecutive nodes and edges, Path Coverage satisfies Node Coverage as well as Edge Coverage.

Path Coverage, as defined above, is a rather aggregated measurement and thus gives little advice of how to improve coverage with further tests. Therefore, Path Coverage can also be restricted to the paths through a specific node $n \in N$:

Let P_n be the set of paths containing n ($\forall p \in P_n : n \in p$), and $P^{ex}_{n,T}$ be the set of paths containing n exercised by test suite T. Accordingly, the Path Coverage of node n is defined as:

$$PC_T(n) = \frac{|P^{ex}_{n,T}|}{|P_n|}$$

A node with a low Path Coverage is only tested under a small fraction of the contexts in which it is contained. Again, further tests should be created for those scenarios, unless they expose dependencies that makes not every path feasible.

3.6 Value Coverage

The metrics presented so far assess the test coverage with respect to structural properties, each considering some kind of modeled element, like nodes and edges. However, the actual input data, that directs the execution of the guideline, is not assessed by these metrics in any way.

Beside the mentioned control-flow-based metrics, a second perspective on coverage in classic Software Engineering is given by data-flow-based metrics. Those measure the coverage of *definition-use (du)* sequences of variables, i.e., a block of instructions in which a variable is defined and subsequently used without a redefinition of the variable, e.g., [19]. Black-box testing strategies concerned with data usage are *Equivalence Partitioning* and *Boundary Value Analysis* [16]. As an exhaustive testing with all valid input data is most likely not tractable even for a small program, its specification can be used to partition the input space into equivalence classes. Under the assumption that each value of a partition is treated equally by the program, an arbitrary representative of each class can be chosen for a test case. As errors are more probable at the boundaries of an equivalence class, Boundary Value Analysis is often used to derive additional test cases at these spots [20], for example to find "off-by-one" errors (e.g., resulting from the use of the operator "$\leq$" rather than "$<$" in a numerical comparison).

DiaFlux models do not contain variables as they are common in procedural programming languages, and the input data is not modified during guideline execution. Therefore, a *definition-use* analysis is not applicable. Equivalence Partitioning and Boundary Value Analysis can also not be used as they are. Explicit equivalence classes usually can not be stated for inputs. Even thresholds for less determined assessments (like "low", "normal", "high") are often hard to specify for a medical expert. Those can furthermore vary between different types of patients, which, e.g., share an underlying disease. However, for each numerical input, a contiguous interval of possible values can typically be given, according to the human body's physiological system and/or the preconditions of guideline applicability. To assure a proper coverage of allowed input data regardless of concepts like equivalence classes, we define the metric *Input Coverage*: Let the interval $[min_i; max_i]$ be the domain D_i for numerical input i, and $n \in \mathbb{N}, n > 0$ be the number of equally-sized partitions of D_i. The function $cover(i, j)$ returns 1, iff the $j-th$ partition of D_i contains at least one input value in test suite T, and 0 otherwise. Then, the Input Coverage of i is given by:

$$IC_{n,T}(i) = \frac{\sum_{j=1}^{n} cover(i, j)}{n}$$

Clearly, the significance of Input Coverage depends on the actual value of n. It should be chosen to appropriately represent the sensitivity of the outcome of the guideline to changes in values of i. At later stages of test creation, the value can be increased stepwise to test more fine-granular in terms of i's input values.

Similarly, the output of the guideline (which mainly consists of numerical values of the host device's settings in predefined ranges) can be assessed by the analog defined metric *Output Coverage* $OC_{n,T}$.

3.7 Measuring Test Coverage

Commonly, there are two strategies to gather the data needed for calculating test coverage metrics. The first one is called *instrumentation*, which modifies the tested piece of software by including new code that collects the necessary information. The second strategy is *tracing*, which traces the executed elements by using some sort of debugging API (Application Programming Interface) of the execution environment. Clearly, both approaches have an effect on the execution time of the tests, as additional data has to be gathered. An advantage of tracing is, that it does not alter the executed artifact. Under certain circumstances this may also influence its behavior. Under this aspect, "tracing" seems preferable, if the necessary API is available.

3.8 Visualization of Test Coverage

The calculated metrics result in a numerical value representing the test coverage of the exercised artifact. This may very well be comprehensible for software and knowledge engineers, though for non-technical domain specialists, like medical experts, these sole numbers may not be accessible enough. Furthermore, only a proper composition of metrics yields a meaningful overall picture, as each metric represents a different aspect of coverage. Thus, an intuitive visualization as a means for communicating the reached coverage levels to domain specialists seems preferable. One approach to this need are so called "Polymetric Views" [13]. They allow to display various metrics in an aggregated view.

4 Test Coverage for DiaFlux Guidelines

This section describes an implementation of coverage metrics for DiaFlux guidelines and a small case study.

4.1 Implementation

The development environment for DiaFlux guidelines is integrated into the Semantic Wiki KnowWE. We created a plugin to calculate the test coverage of DiaFlux models, when executing a test suite. It employs the tracing approach to collect the information about exercised elements of the guideline. For each execution of the guideline, the chosen path according to the input data is recorded. After finishing the test suite, the metrics are calculated and can subsequently be visualized as CoverageCity, which can freely be scaled and rotated (cf. Figure 2). For creating the visualization, we used the WebGL[4]-based JavaScript library SceneJS[5]. It is hence accessible with every (modern) web browser from within KnowWE, not requiring any proprietary software.

4.2 The CoverageCity Visualization

For an accessible visualization of the reached coverage levels, we use the metaphor of a city. It has been introduced as graphical representation of static code metrics (e.g. number of methods,...) in the context of reverse-engineering of software systems ("CodeCity") [21]. Such a city consists of districts that represent the nesting structure of packages and buildings representing classes. Each building is located in the district corresponding to its package. The actual values of the metrics of each class determine the visual appearance of the matching building. A building can represent up to four different metrics, influencing its length, width, height, and color. Besides the package structure, districts can depict one metric by their color.

We adapted the city metaphor for the visual representation of test coverage of DiaFlux guidelines. Districts stand for individual flowcharts. They are nested as given by their call hierarchy. Buildings correspond to the nodes of the district's corresponding flowchart. Edges are not explicitly included but mapped to one of the buildings' dimensions. In particular, we used the following assignment of metrics to visual properties of buildings:

- *Length:* The number of outgoing edges of a node determines the length of the building.
- *Width:* The width of a building relates to the number of outgoing edges that are covered by the test suite.
- *Height:* The height of a building shows how often it was covered by the test suite.
- *Color:* The color represents how well the paths, that go through a particular node, are covered by a test suite. In case a building is "red", only few paths are executed. "Green" stands for a high Path Coverage.

[4] http://webgl.org
[5] http://scenejs.org

4.3 Interpretation of the Visualization

The complexity that a node introduces into a guideline, mostly comes from its number of outgoing edges. In case the associated action has numerous possible outcomes (each represented by an outgoing edge), it is more likely to contain errors. Thus, one dimension of the base area of a building is influenced by the number of outgoing edges of the corresponding node. The increased size makes the building easier to spot. The other dimension of the base area grows with the number of covered outgoing edges. As a result, a non-quadratic building stands for a node, whose outgoing edges are not completely covered. The lower the aspect ratio, the more uncovered edges exist. This deficiency in test coverage can easily be observed.

The color of a building depicts its Path Coverage. A red one, represents a node, that is contained in much more paths than were covered. With increasing Path Coverage, the color becomes green. The height of a building corresponds to the number of times, it has been exercised by the test suite. Clearly, those two properties are not independent. On the one hand, a small building is more likely to be contained only in a small number of paths, and thus be red. On the other hand, a building that is tall and red, implies that the corresponding node is often tested under similar circumstances. This can give hints for creating new test cases, that are not contained so far. Tall, green buildings have been thoroughly exercised and do not require additional test cases.

The nesting level of the districts helps in estimating the necessary effort for testing a specific flowchart or node. Deeply nested module are probably harder to reach by a test case, as each module may only be called under certain preconditions.

The aggregated view of CoverageCity makes it easy to spot deficiencies in test coverage quickly. Though, a more detailed view is necessary to identify the specific nodes and their test deficiencies. Hence, each building be selected by the user. Then, the corresponding flowchart is shown below the CoverageCity, and the node and its coverage is highlighted visually.

4.4 Case Study

Currently, we are involved in the implementation of a computerized guideline for automated mechanical ventilation [9]. The guideline is intended to run on a mechanical ventilator, and is able to derive new ventilatory settings in order to improve the ventilation of the patient. First testing efforts of the guideline were conducted using a physiological simulation. The guideline was run against a software tool that simulates a mechanically ventilated patient. It employs a physiological lung model to determine the effects of the current ventilation settings to the patient. The simulation is able to deliver the necessary data (ventilation settings and measured patient response) to the guideline execution engine. Based on this data, the guideline derives optimized settings and returns them to the simulation environment, that uses them for the further simulation. The simulation tool was used by medical experts to generate the test cases and review the derived ventilation settings. The generated test cases are saved to a file,

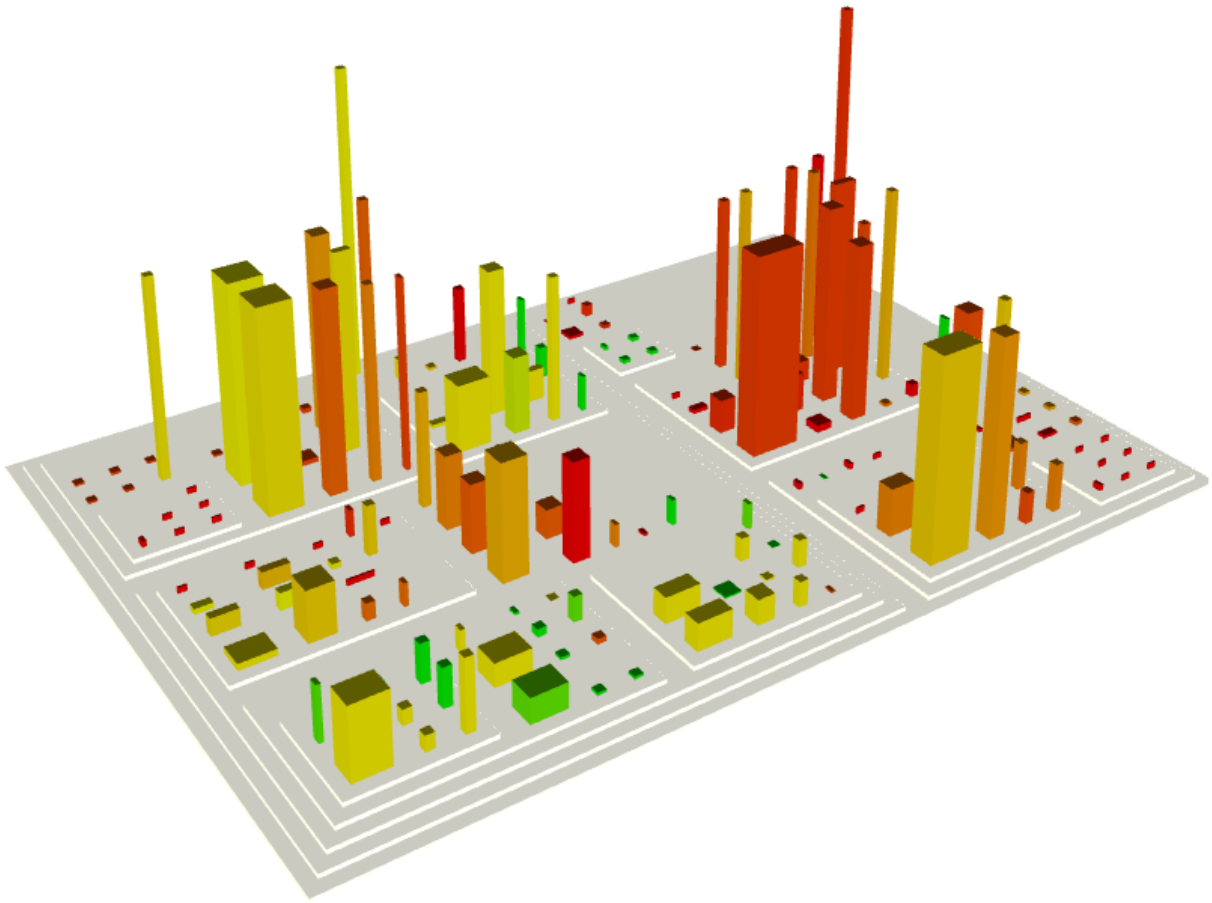


Figure 2. The test coverage visualization using the city metaphor. Nodes are represented by buildings, flowcharts by districts. The visual properties of building are determined by the coverage metrics of the corresponding node.

and are then uploaded to KnowWE for the introspection of guideline execution and coverage analysis.

We selected a sample of ten generated test cases. The visualization of the acquired coverage levels is shown in Figure 2. Currently, we are in the process of evaluating our visualization with medical experts. Furthermore, we identify other meaningful assignments of metrics to visual properties of the buildings.

5 Conclusion

In this paper, we formally defined different coverage metrics to assess the thoroughness of testing efforts for clinical guidelines. They can be used to identify insufficiently tested elements, and to improve the process of test creation in terms of efficiency, as this may involve domain specialists as well as knowledge engineers. An intuitive visualization method helps in communicating the acquired coverage levels to domain specialists, for which numerically expressed metrics probably are less helpful than for knowledge engineers.

Additional metrics could be defined over more dynamic aspects of a guideline. First, the distribution of values could be tracked for each activated flowchart element. As there clearly are dependencies between the actual values and the possible ones - given by the context (i.e. path) of the element - proper preprocessing would be necessary. Ultimately, this could give insight, if parts of the guideline were only tested, e.g., for a certain patient type. Second, it would be helpful to define scenarios with respect to the occurrence of a certain sequence of input data or therapeutic actions over time and trace their coverage by the test suite. In terms of the CoverageCity-visualization, we will evaluate different mappings of the coverage metrics to the visual properties of the city, to create new perspectives on test coverage.

One shortcoming of white-box testing in general is, that it is unable to detect errors by omission, i.e. some requirement may not have been included in the implementation under test. An approach to find this type of errors is *Requirements-based Test Coverage* [18]. It defines coverage with respect to implementation-independently defined requirements that are exercised by a test suite. Having formally defined requirements, this approach should be transferable to testing clinical guidelines.

Acknowledgements

University of Würzburg is funded by the German Federal Ministry for Education and Research under the project “WiM-Vent” (Knowledge- and Model-based Ventilation), grant number 01IB10002E.

REFERENCES

[1] Valerie Barr, 'Applications of rule-base coverage measures to expert system evaluation', *Knowledge-Based Systems*, **12**, 27–35, (1999).

[2] Joachim Baumeister, 'Advanced empirical testing', *Knowledge-Based Systems*, **24**(1), 83–94, (2011).

[3] Joachim Baumeister, Jochen Reutelshoefer, and Frank Puppe, 'KnowWE: A semantic wiki for knowledge engineering', *Applied Intelligence*, **35**(3), 323–344, (2011).

[4] Aziz A. Boxwala, Mor Peleg, Samson Tu, Omolola Ogunyemi, Qing T. Zeng, Dongwen Wang, Vimla L. Patel, Robert A. Greenes, and Edward H. Shortliffe, 'GLIF3: a representation format for sharable computer-interpretable clinical practice guidelines', *J. of Biomedical Informatics*, **37**(3), 147–161, (2004).

[5] J.J. Chilenski and S.P. Miller, 'Applicability of modified condition/decision coverage to software testing', *Software Engineering Journal*, **9**(5), 193–200, (1994).

[6] Paul de Clercq, Katharina Kaiser, and Arie Hasman, 'Computer-interpretable guideline formalisms', in *Computer-based Medical Guidelines and Protocols: A Primer and Current Trends*, eds., Annette ten Teije, Silvia Miksch, and Peter Lucas, 22–43, IOS Press, Amsterdam, The Netherlands, (2008).

[7] Reinhard Hatko, Joachim Baumeister, Volker Belli, and Frank Puppe, 'DiaFlux: A graphical language for computer-interpretable guidelines', in *Knowledge Representation for Health-Care*, eds., David Riaño, Annette ten Teije, and Silvia Miksch, volume 6924 of *Lecture Notes in Computer Science*, 94–107, Springer, Berlin / Heidelberg, (2012).

[8] Reinhard Hatko, Joachim Baumeister, Gritje Meinke, Stefan Mersmann, and Frank Puppe, 'Anomaly detection in DiaFlux models', in *KESE7: 7th Workshop on Knowledge Engineering and Software Engineering, San Cristobal de La Laguna, Spain, November 10, 2011*, volume 805 of *CEUR Workshop Proceedings*, Tenerife, Spain, (2011). CEUR-WS.org.

[9] Reinhard Hatko, Dirk Schädler, Stefan Mersmann, Joachim Baumeister, Norbert Weiler, and Frank Puppe, 'Implementing an automated ventilation guideline using the semantic wiki knowwe', in *EKAW 2012: 18th International Conference on Knowledge Engineering and Knowledge Management*, eds., Heiner Stuckenschmidt, Annette ten Teije, and Johanna Voelker, (2012).

[10] Arjen Hommersom, Perry Groot, Michael Balser, and Peter Lucas, 'Formal methods for verification of clinical practice guidelines', in *Computer-based Medical Guidelines and Protocols: A Primer and Current Trends*, eds., Annette ten Teije, Silvia Miksch, and Peter Lucas, 63–80, IOS Press, Amsterdam, The Netherlands, (2008).

[11] J.A. Jones, M.J. Harrold, and J. Stasko, 'Visualization of test information to assist fault localization', in *Proceedings of the 24th international conference on Software engineering*, pp. 467–477. ACM, (2002).

[12] H.F Kwok, D.A Linkens, M Mahfouf, and G.H Mills, 'Rule-base derivation for intensive care ventilator control using ANFIS', *Artificial Intelligence in Medicine*, **29**(3), 185 – 201, (2003).

[13] Michele Lanza and Stéphane Ducasse, 'Polymetric views - a lightweight visual approach to reverse engineering', *IEEE Trans. Software Eng.*, **29**(9), 782–795, (2003).

[14] Daniel Lübke, Leif Singer, and Alex Salnikow, 'Calculating bpel test coverage through instrumentation', in *AST*, eds., Dimitris Dranidis, Stephen P. Masticola, and Paul A. Strooper, pp. 115–122. IEEE, (2009).

[15] Stefan Mersmann and Michel Dojat, 'SmartCare^{tm} - automated clinical guidelines in critical care', in *ECAI'04/PAIS'04: Proceedings of the 16th European Conference on Artificial Intelligence, including Prestigious Applications of Intelligent Systems*, pp. 745–749, Valencia, Spain, (2004). IOS Press.

[16] Glenford J. Myers, Corey Sandler, and Tom Badgett, *The art of software testing*, John Wiley & Sons, Hoboken, N.J., 3 edn., 2011.

[17] Mor Peleg, Samson Tu, Jonathan Bury, Paolo Ciccarese, John Fox, Robert A Greenes, Silvia Miksch, Silvana Quaglini, Andreas Seyfang, Edward H Shortliffe, Mario Stefanelli, and et al., 'Comparing computer-interpretable guideline models: A case-study approach', *JAMIA*, **10**, (2003).

[18] A. Rajan, 'Coverage metrics to measure adequacy of black-box test suites', in *Automated Software Engineering, 2006. ASE '06. 21st IEEE/ACM International Conference on*, pp. 335 –338, (sept. 2006).

[19] Sandra Rapps and Elaine J. Weyuker, 'Selecting software test data using data flow information', *IEEE Trans. Softw. Eng.*, **11**(4), 367–375, (April 1985).

[20] S.C. Reid, 'An empirical analysis of equivalence partitioning, boundary value analysis and random testing', in *Software Metrics Symposium, 1997. Proceedings., Fourth International*, pp. 64 –73, (November 1997).

[21] Richard Wettel and Michele Lanza, 'Visualizing software systems as cities', in *VISSOFT*, eds., Jonathan I. Maletic, Alexandru Telea, and Andrian Marcus, pp. 92–99. IEEE Computer Society, (2007).

[22] Richard Wettel and Michele Lanza, 'CodeCity: 3D visualization of large-scale software', in *ICSE Companion*, eds., Wilhelm Schäfer, Matthew B. Dwyer, and Volker Gruhn, pp. 921–922. ACM, (2008).

Metamodeling of Bayesian networks for decision-support systems development

Isabel M. del Águila and **José del Sagrado** [1]

Abstract. The knowledge modeling and software modeling phases in Knowledge-Based System development are not integrable, in terms of representation, due to the different languages needed at the steps of the development. This paper focuses on bring closer these languages. By one hand, we define a meta model which contains the key concepts used in the definition of a knowledge model as a Bayesian network. On the other hand, we define an extension of UML using profiles that can bridge the gap in representation and facilitate the seamless incorporation of a knowledge model, as Bayesian network, in the context of a knowledge-based software development.

1 Introduction

Knowledge-based systems (KBSs) are characterized by their high risk, loose definition, poor structure and subjective requirements. These software systems were introduced in the early 1970s as expert systems from the field of artificial intelligence (AI) research. Originally their goal was to transfer expertise from domain experts to a some kind of knowledge base that may be integrable in a software system. However, nowadays knowledge engineering is changing as it turns towards the modeling approach. A KBS can be defined as "software that has some knowledge or expertise about specific, narrow domain, and is implemented such that Knowledge base and the control architecture are separated. Knowledge-Based Systems have capabilities that often include inferential processing (as opposed to algorithmic processing), explaining rationale to users an generating non-unique results" [19]. From this definition is easy to see the many roles played by the knowledge model (knowledge bases). Models provide an abstraction about reality and through this knowledge models the human experts problem solving approaches in order to be used in the development of software solutions. Knowledge models usually are described in a specific purpose language. There is not a standard, because it depends heavily on knowledge representation mechanisms (i.e. rules, semantic networks, frames). However, from a commercial point of view, the development of a software system focuses on customers, that is, in order to develop any software solution we need to gather requirements from customers and translate it into software functionalities. All the desired functionalities do not have to apply artificial intelligence techniques.Thus, software systems usually integrate a KBS with other needed software enterprise components.

These not knowledge based components are described using modeling languages from the software engineering domain (UML is the most used standard). This lead us to combine several modeling languages in the same project. Figure 1 shows the vision of a software development project from the points of view of a customer, a software engineer and a knowledge engineer. The knowledge engineer (Figure 1A) uses knowledge engineering methods to define the project's task, relegating to the background the tasks defined by software engineering. The software product that results from this process is a KBS. From another perspective, the software engineer (Figure 1B) systematically applies its skills, tools and software engineering methods to develop a software product (system), where knowledge is only another element. Finally, the customer's view of the project (Figure 1C) focuses on quality and the need of cooperation between the two engineerings and their own modeling languages [2, 3], so that the final software product properly covers all her/his needs. In other words, software components based and not based on knowledge must be integrated in shaping the software system for the end user.

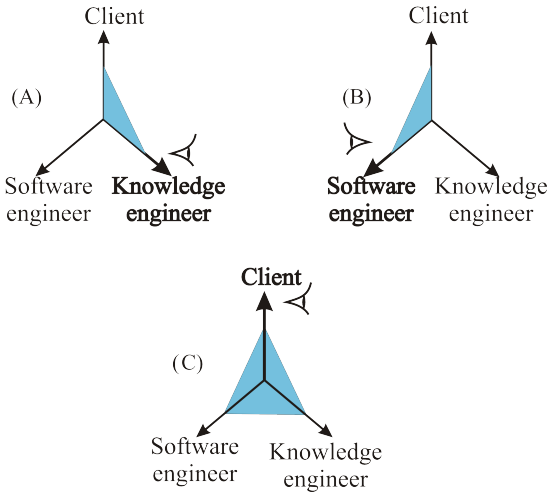


Figure 1. Different views of a software development project

The use of different modeling languages limits the applicability of one of the software development schemes more widespread in our day, Model-Driven Architecture (MDA) [13]. This approach, to information software systems development, separates specification of the system functions from implementation of these functions in a given platform, focusing on models as a higher level of abstraction during systems construction [5]. This fact leads developers to a significant decoupling between platform-independent models (PIMs) and platform-specific models (PSMs). Separation between specification and implementation is also a basic feature in KBS (see

[1] Department of Languages and Computation, University of Almería, Spain, email: imaguila@ual.es

[19]). Our problem is putting together different modeling language notations into a single platform independent model, so that the knowledge model and the UML model are expressed in a compatible format. Therefore, we need a extended PIM that include algorithmic and inference functionalities. A single language allows a great level of abstraction and makes easy the implementation process. The same PIM model can be translated to different platforms: the core goal of MDA.

This work propose a extension of UML when Bayesian Networks (BN) [21, 15, 16] are the representation selected by knowledge engineers to model the knowledge. We focus on BN because among all the knowledge modeling languages, Bayesian networks can be successfully used to represent expert knowledge on an uncertain domain. Today, BNs are expressed by applying its own algebraic notation, but if we want a BNs to be part of a software solution, we must be able to express themselves in the same language that is commonly used to model general purpose software (i.e. UML). In this paper we propose an approach which aims to improve the development of decision support system, reinforcing the aspects of BNs modeling. In order to achieve this goal, we introduce a BN metamodel and a UML profile as means to build the PIM for a KBS that represents knowledge based on a BN.

The rest of this paper is structured as follows. Section 2 introduces the basic fundamentals of Bayesian networks. The general MDA translation schema for BNs is explained in Section 3. Section 4 describe a basic metamodel for BNs (BayNet) that provides a specific and intuitive notation for modeling BN-based KBS. The extension of UML taking as basis the BN metamodel BayNet to create an UML profile for BNs (UBN) is studied in Section 5. Then, Section 6 shows how to apply UBN profile to the development of a simple pest control BN-based KBS. Finally, the conclusions and future works are exposed in Section 7.

2 Bayesian Networks Basics

A Bayesian network is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph. Because a Bayesian network is a complete model for the variables and their relationships, it can be used to answer probabilistic queries about them.

Formally, a BN [21, 15, 16] is a pair $(G, \mathbf{P})$ where

- $G = (\mathbf{V}, \mathbf{E})$ is a directed acyclic graph whose set of nodes $\mathbf{V} = \{\mathbf{X_1}, \mathbf{X_2}, \cdots, \mathbf{X_n}\}$ represents the system variables and whose set of arcs $\mathbf{E}$ represents direct dependence relationships among the variables, and
- $\mathbf{P}$ is a set of conditional probability distributions containing a conditional probability distribution $P(X_i|pa(X_i))$ for each variable X_i given its set of parents $pa(X_i)$ in the graph.

The joint probability distribution over $\mathbf{V}$ can be recovered from this set $\mathbf{P}$ of conditional probability distributions applying the chain rule as:

$$P(X_1, \cdots, X_n) = \prod_{i=1}^{n} P(X_i \mid pa(X_i)). \quad (1)$$

The process of obtaining the graph and the probabilities of a BN can be done either *manually*, from experts' knowledge on the domain, or *automatically*, from databases. In the first case, the elicitation of probabilities constitutes a bottleneck in the development of BNs [9].

A BN-based KBS needs to translate this algebraic notation to environments, such as Elvira [10], that consists of software packages for the edition, use and evaluation of BN. These environments provide class libraries that can be integrated as any other component into a software application (i.e. BN-based KBS). This application will be released to the end user and will contain an implementation of the network itself and network capabilities, such that inference from observed values of some variables.

In particular, the Elvira system [10] is tool to construct probabilistic decision support systems. Elvira works with Bayesian networks and influence diagrams and can operate with discrete, continuous and temporal variables. It has an easy to use Graphical User Interface (GUI) (see Figure 2). In addition you can edit, easily, the ASCII format of the Elvira systems to introduce your models. It has methods for inference, learning, abduction, fusion of knowledge and making decisions, and some tools to check the efficiency of the algorithms.

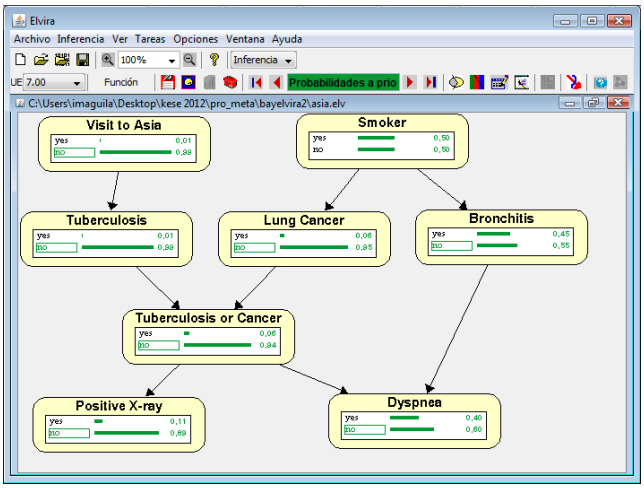


Figure 2. Elvira's main interface

The program Elvira has its own format for storing models, a parser, exact and approximate (stochastic and deterministic) algorithms for inference on both discrete and continuous variables, a graphical interface for building and evaluating Bayesian networks and influence diagrams, with specific options for canonical models (OR, AND, MAX, etc.), explanation of reasoning, decision making algorithms, learning (model building) from databases, fusion of networks, etc. Elvira is written and compiled in Java, which allows the program to run on different platforms and operating systems: Linux, MS-DOS/Windows, Solaris, etc.

2.1 Reasoning with Bayesian Networks

Probabilistic reasoning in BNs consists in computing the posterior probability distributions of some variables of interest $v_I \in \mathbf{V_I}$ given some observed variables $\mathbf{V_E}$ (this sets of findings is called *evidence*), $P(v_I|\mathbf{V_E})$. This process is performed via a flow of information through the network in any direction. If we give a *causal interpretation* to the links in the network (i.e. for an arc $X_i \rightarrow X_j$ we say that X_i is a *cause* of X_j and X_j is the *effect* of X_i), we can perform several types of reasoning [17]:

- *Diagnostic* reasoning, the evidence flows in the opposite direction to the arcs, from effects to cause (i.e. some effects receive evidence and some of their causes are the variables of interest).

- *Predictive* reasoning, the evidence flows in the direction of the arcs, from cause to effect (i.e. some causes receive evidence and their effects are the variable of interests).
- *Intercausal* reasoning, the evidence flows in all directions. This type of reasoning involves reasoning about mutual causes of a common effect (i.e. the effect and some of the causes receive evidence and some of the causes are the variables of interests).

Sometimes reasoning does not match into these three types, because any variable can be an interest variable and may be an evidence variable and information flows in either direction (i.e. the effects of a causes and the causes of the second receive evidence and the central cause is the variable of interest). This situation occurs when diagnostic and predictive reasoning are used simultaneously.

3 Bayesian Network as PIM

In terms of MDA vocabulary, a BN is a PIM, Elvira is a platform model into which a BN has to be translated in order to define the final PSM. Elvira software also offers Java support for the BN, that is to say, it provides the last level of the MDA approach (i.e. the code).

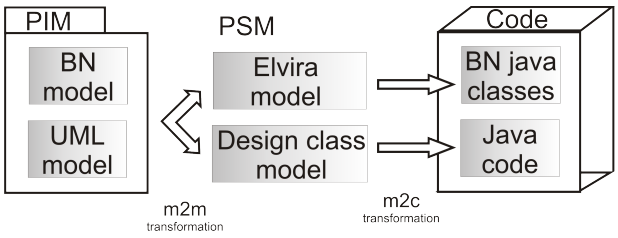


Figure 3. MDA for BN-based KBS

Figure 3 shows the translation model proposed. Both, the transactional (or interaction) PSM and the knowledge (or BN) PSM are expresed in terms of object-oriented design languages coming from Software Engineering (i.e. UML) and their translation into code is solved by means of a m2c translation. Many CASE tools, like Visual Paradigm, Microsoft Visual Studio or Enterprise Architect, already incorporate, at least to some degree, this kind of translation. But to use the power of MDA, is also necessary to express the PIM of these two parts of the software solution. There are modeling languages in the Software Engineering area that allow to express the PIM, but what about the BN? What is its language model? Is it UML compatible?

The MDA approach has been applied in other forms of knowledge representation as rules [7, 1]. From the metamodel of the rule-based languages have been defined UML profiles and automated tools, that translate the rules of PIM models into rules-based web-systems by combining Java Server Faces with Jess rule engine [6]. But, BNs lack of a modeling language compatible with UML that allows the application of MDA. Model transformation consists on the process of converting one model to another model of the same system in a different abstraction level: from PIM to PSM and from PSM to code. MDA tools allow these transformations to be automated and executed automatically.

Our goal is to create an UML-compatible modeling language for BNs. The Object Management Group (OMG) has defined two extension mechanisms for UML: metamodel model and profile extensions [14]. First extension mechanism involves the process of defining a new metamodel on which to build an entirely new language defined through the Meta-Object Facility (MOF) specification. But if we do not want to change UML semantics and only particularize some concepts, we can extend UML using a series of mechanism offered by the language itself: the profiles. [12].

We know that all the knowledge representation mechanisms are in themselves languages. So we can choose to build a totally new MOF language, but it may not respect the standard UML metamodel. This fact will prevent existing UML tools to manage the new language concepts in a natural way. To offer a proposal that also gives support to UML, leads to a greater number of users and reduce the learning curve in the new language. For all these reasons and agreeing with the proposals of several authors [18, 22, 4], we also propose the use of UML profiles. In any case, according to several studies [11, 12], the starting point for developing a UML profile is the metamodel for the platform or of the domain of the application that is going to be modeled. In our case, the starting point is a BN metamodel (called BayNet) that will allow us to specify how BN concepts are related to and represented in standard UML. Visual Paradigm is the CASE tool used to define this metamodel and its associated profile.

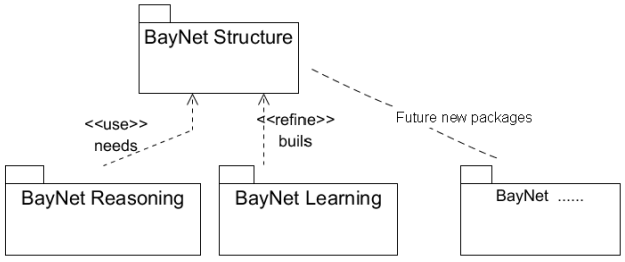


Figure 4. BayNet metamodel's basic structure

4 Metamodel for Bayesian Network

A metamodel includes domain entities, retrictions between them and limitations on the use of entities and relationships. BNs are complex in nature, beside its structure we have to face with complex concepts, as inference and learning, that have to be approached by successive approximations. This is the reason why we have split the metamodel in several packages as it is shown in Figure 4. The BayNet metamodel is the basis for providing a specific and intuitive notation for modeling BN-based KBS.

In a first approximation to BayNet, we only focus on BayNet structure (as we need to define the BN structure to define a model) and BayNet reasoning (as we need to carry out inference in order to reasoning with a BN) packages (see Figure 5). The BayNet structure package represents the basic components of a BN (BNet class): its qualitative (i.e. directed acyclic graph) and quantitative (i.e. set of probability distributions) parts. The qualitative part is represented by the class *Variable* and its self-association. An *Assignment* consists in assigning a *State* to a *Variable* modifying, accordingly, its marginal probability. The quantitative part is represented by means of the classes *Configuration* and *Relation*. For each given child-father association (*Configuration*) in the directed acyclic graph is assigned a conditional probability value (*Relation*). That is, we assign a

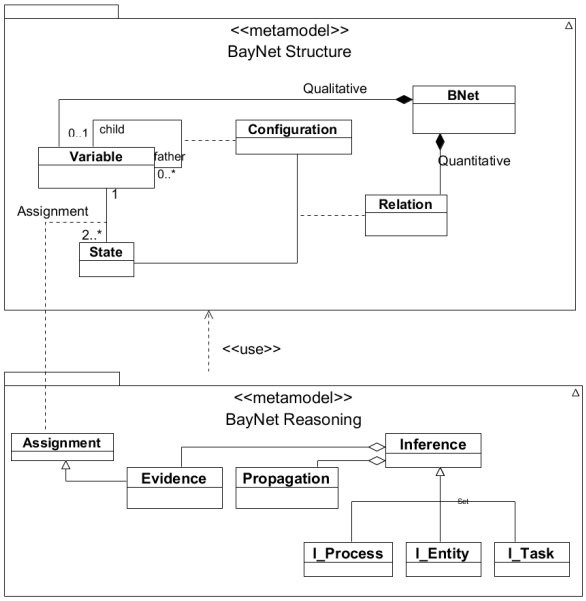


Figure 5. BayNet metamodel

probability value to each combination of values of a variable X_i and its parents $pa(X_i)$ in the graph, to define the conditional probability distribution $P(X_i|pa(X_i))$.

In BayNet reasoning, an inference can be view as a process (*I_Process*), a class able to carry out inferences (*I_entity*) or an operation inside a class (*I_task*). These three views allows to model different levels of abstraction in the decision tasks associated to a BN-based KBS. An *Inference* is an aggregation of the observed variables (*Evidence*) together with the execution of the operations needed to make evidence flow on the network and to compute the posterior probability distributions of the variables of interest (*Propagation*).

5 UBN profile

UML offers the possibility to extend and adapt its metamodel to a specific area of application through the creation of profiles. The BayNet metamodel is the basis that will provide a specific and intuitive notation for modeling BN-based KBS and including it in an UML project. UML profiles are UML packages with the stereotype $<< profile >>$. A profile can extend a metamodel or another profile while preserving the syntax and semantic of existing UML elements. It adds elements which extend existing classes. UML profiles consist of *Stereotypes*, *Constraints* and *Tagged Values*.

- A *stereotype* is a model element defined by its name and by the base class(es) to which it is assigned. Base classes are usually metaclasses from the UML metamodel, for instance the metaclass $<< Class >>$, but can also be stereotypes from another profile.
- *Constraints* are applied to stereotypes in order to indicate restrictions. They specify pre- or post conditions, invariants, etc., and must comply with the restrictions of the base class. Constraints can be expressed in any language, such as programming languages, natural language or Object Constraint Language (OCL).
- *Tagged values* are additional meta-attributes assigned to a stereotype, specified as name-value pairs. They have a name and a type and can be used to attach arbitrary information to model elements.

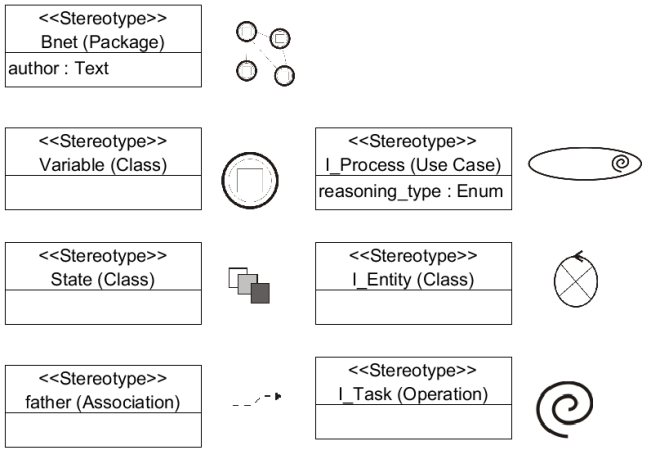


Figure 6. UBN stereotypes and icons

We use UML profile to define a UML Bayesian network profile (UBN). The aim of UBN is to define a language for designing, visualizing, specifying, analyzing, constructing and documenting the artifacts of knowledge-based systems, that represents it knowledge as a BN. The next step is to map the BayNet metamodel, described in the above section, to UML metaclases and make the necessary extensions. The mapping is a non-trivial task, because we need to know in deep how to apply the UML language. Most of concepts will map to stereotypes on a selected UML metaclass. Also we can define icons for most of the stereotypes, that allows the modeler to use intuitive symbols instead of UML shapes. Figure 6 shows the actual mapping with UML metaclasses.

Once the UBN is defined, it can be used in the software development of a particular application by defining a stereotyped dependency ($<< applyProfile >>$) between the UBN package an the package that is being under development for the application, as Figure 7 shows. A partial view of the class model of Elvira is included as it is needed in order to define the m2m translation between PIM and PSM (see Figure 3).

6 Case Study: A pest control BN-based KBS

This section shows how to apply UBN in an specific KBS development project. The project follows a development methodology described in [3], the process model proposed allows the seamless inclusion of Bayesian networks into the final software solution for an organizational environment. Let us begin with a brief description of the project to assist decision making in an agricultural domain. Our problem is related to pest control in a given crop under the regulation of Integrated Production. The

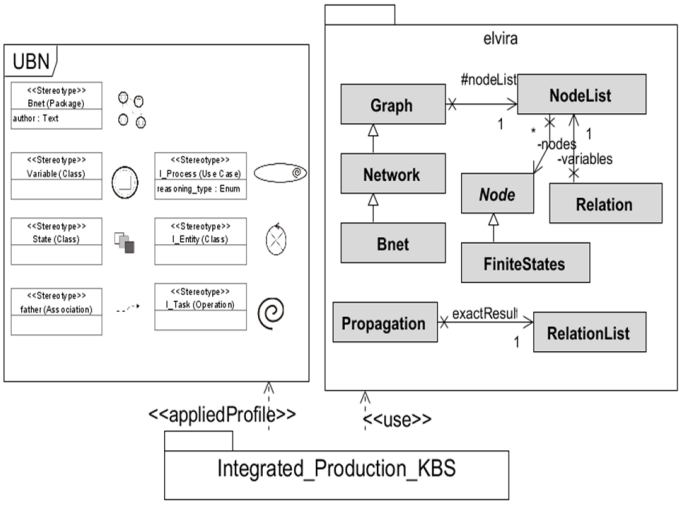


Figure 7. Bayesian network KBS modeling packages: pest control application

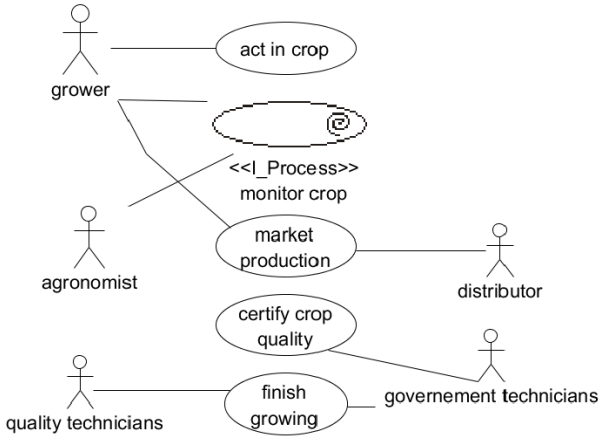


Figure 8. Use cases

Integrated Production Quality standard is adopted by a group of growers in order to achieve a quality production certification. The adoption of this standard forces growers to be disciplined in growing which involves intervention by technicians, marketing controls, and periodical inspection by the standard certification agencies.

The three main steps in the development of software systems that embed functionalities based and not based on knowledge, concerning the decision support process and the information management processes, are: *Requirement modeling* (RM), *Expert modeling* (EM) and *Specification of the software solution* (SSS)[3]. The first two are in charge of the definition on the PIM model according MDA (see Figure 3), and here is where UBN gets its value, because we can use only UML in order to execute RM and EM.

Software project development starts with business and RM modeling. The first activity consists of collecting, structuring and organizing all the relevant data for a concise and accurate definition of the problem to be solved by the software system. Integrated production involves handling and storing a huge amount of information about crops, and making decisions about all the tasks that have to be performed to fulfill the quality regulations.

We model the processes that are represented as use cases. The typical processes in an integrated production problem are shown in Figure 8. All tasks related to information required for quality management standards, without needing any knowledge based approach, are: Market Production, Act in Crop, Certify Crop Quality, and Finish Growing. All tasks related to pest control are performed by growers and agronomists in the Monitor crop process and represents the inference tasks that we need to model by means of a Bayesian network. The decision process when monitoring a crop is made at two levels. First, a decision is made on whether crop control action is necessary by sampling pests and estimating risk of attack. Then if it is decided that crop control action is required, the product (chemical or biological) to be applied has to be selected. The treatment advised has to respect natural enemies and other biological products previously used.

The next steep is to finish the PIM, using UML and UBN in order to define the system without considering platform level details. That is, from use cases we need to define the conceptual models. In this section, as specific case, we focus on the Monitor crop use case that can be described as the following informal scenario: Each week, the agricultural expert samples the crops condition and makes an estimate of the risk of pest attack. Crop sampling consists of direct observation and count of harmful agents in randomly selected plants. Where imbalance is detected, the expert advises treatment meeting the integrated production standard.

A crop is a complex system consisting of a plot of land, plants, a set of diseases and pests, and natural enemies that may be able to control them. The problem is to decide what treatment to apply, in order to maintain a balanced system. Figure 9, shows the UML class diagram obtained. Some of the classes in the model are variables of the Bayesian network (EM).

Within the scope of integrated production systems, when an agricultural expert visits a greenhouse, he writes down the date of the visit and samples the crop, including information about fauna, weather (wind, rain, etc.) and environment (weeds). The general schema for a crop-harmful agent pair consists of observing the crops condition and fauna. Crop condition is measured in terms of its phenology. The presence of fauna is important to estimate the intensity of the attack. The crop condition, along with the intensity of pest attack, determines the need for applying a plant health treatment or not. Periodical inspections of this kind are performed weekly. Figure 10 shows the BN structure elicited from the knowledge of the domain expert. Once the BN structure has been established, the probabilities are estimated completing the construction of the BN model. This expert modelling process has been successfully applied to determine the need of applying a treatment for olives' fly [8].

In order to select the set of relevant variables, we start from the initial conceptual model of the project that has to be refined. A first version of the PIM is shown in figure 9. Some of the modeling elements are stereotyped, using UBN, as nodes in a BN, Crop condition is measured in terms of its phenology. The presence of fauna is important to estimate the intensity of the attack. The crop condition, along with the intensity of pest attack, determines the need for applying a plant health treatment or not. Periodical inspections of this kind are performed weekly. Relationship that complete the qualitative part of the BN are shown as stereotyped associations of

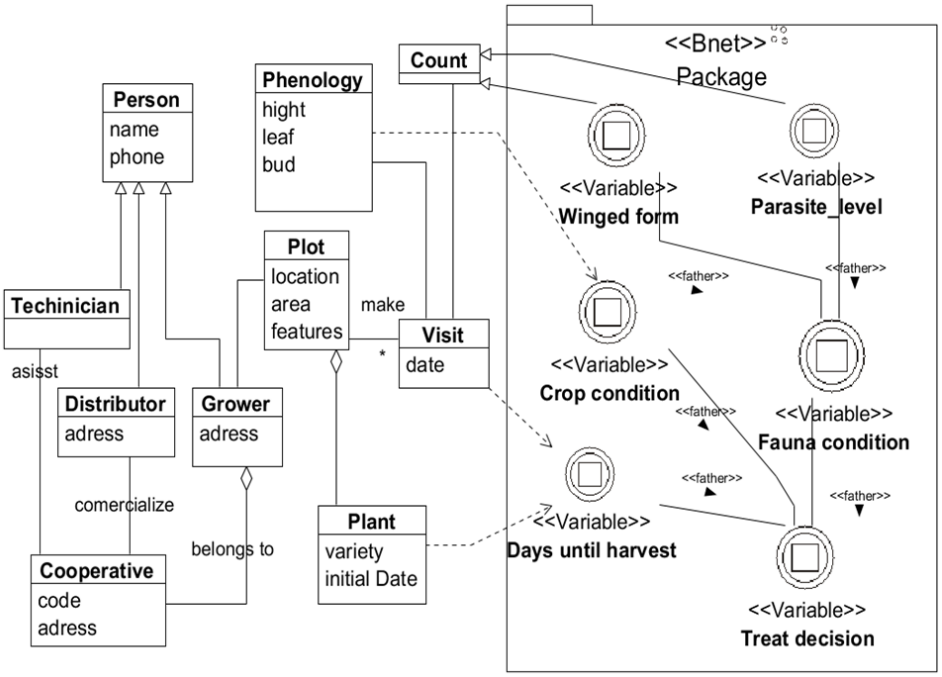


Figure 9. Partial view of the PIM

type $<< father >>$. Once the BN structure has been established, the probabilities are estimated (quantitative elicitation activity) based on a local government database of cases, completing the construction of the BN model. This expert modeling process has been successfully applied to determine the need of applying a treatment for the olives' fly (dacus olae) [20].

Finally, the specification of the software solution (SSS) represent a m2m translation that produces the PSM. Based on the PSM obtained, a m2c translation can be used to obtain a BN-based KBS in order to assists grower and technicians in pest control decision support tasks.

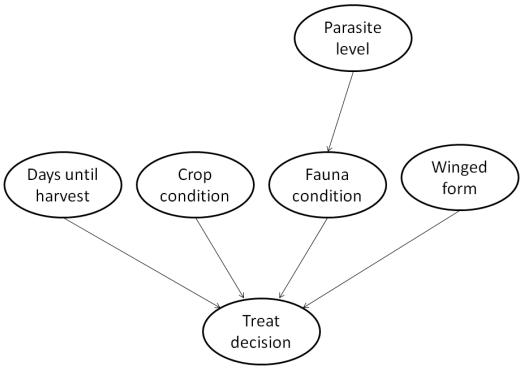


Figure 10. A BN structure for a crop-harmful agent pair

7 Conclusions

In this work we have presented a metamodel (BayNet) and an UML profile (UBN) for BN-based KBS modeling. This metamodel covers several important aspects for achieving the seamless inclusion of BN models into a final software solution for an organizational environment. The applicability of our solution has been tested in a simplified version of a real world problem: integrated production in agriculture.

Our proposal allows to manage a domain-specific language for BN without changing UML semantics. This can be view as a general framework to apply Model Driven Development, extending it to the BN-based KBS case. Developing a profile is a difficult task that implies to perform many steps. The next steps of this research will consist in defining an specification of constraints and operations using OCL, validate the profile using a CASE tool as Visual Paradigm and test it in a real-life development project that includes knowledge-base features.

ACKNOWLEDGEMENTS

This research has been funded by the Control crop Project (PIO-TEP-6174) from the Counseling of Economy, Innovation and Science, Government of Andalusia (Spain) and the Spanish Ministry of Education, Culture and Sport under project TIN2010-20900-C04-02.

REFERENCES

[1] Mohd Abdullah, Ian Benest, Richard Paige, and Chris Kimble, 'Using unified modeling language for conceptual modelling of knowledge-based systems', in *Conceptual Modeling - ER 2007*, eds., Christine Parent, Klaus-Dieter Schewe, Veda Storey, and Bernhard Thalheim, volume 4801 of *Lecture Notes in Computer Science*, 438–453, Springer Berlin-Heidelberg, (2007).

[2] Isabel María del Águila, Joaquín Cañadas, José Palma, and Samuel Túnez, 'Towards a methodology for hybrid systems software development', in *SEKE*, eds., Kang Zhang, George Spanoudakis, and Giuseppe Visaggio, pp. 188–193, (2006).
[3] Isabel María del Águila, José del Sagrado, Samuel Túnez, and Francisco Javier Orellana, 'Seamless software development for systems based on bayesian networks - an agricultural pest control system example', in *ICSOFT (2)*, eds., José A. Moinhos Cordeiro, Maria Virvou, and Boris Shishkov, pp. 456–461. SciTePress, (2010).
[4] Saartje Brockmans, Robert Colomb, Peter Haase, Elisa Kendall, Evan Wallace, Chris Welty, and Guo Xie, 'A model driven approach for building owl dl and owl full ontologies', in *The Semantic Web - ISWC 2006*, eds., Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora Aroyo, volume 4273 of *Lecture Notes in Computer Science*, 187–200, Springer Berlin / Heidelberg, (2006).
[5] Alan W. Brown, 'Model driven architecture: Principles and practice', *Software and System Modeling*, **3**(4), 314–327, (2004).
[6] Joaquín Cañadas, José Palma, and Samuel Túnez, 'A tool for mdd of rule-based web applications based on owl and swrl', in *KESE*, eds., Grzegorz J. Nalepa and Joachim Baumeister, volume 636 of *CEUR Workshop Proceedings*. CEUR-WS.org, (2010).
[7] Joaquín Cañadas, José Palma, and Samuel Túnez, 'Defining the semantics of rule-based web applications through model-driven development', *Applied Mathematics and Computer Science*, **21**(1), 41–55, (2011).
[8] José del Sagrado and Isabel del Águila, 'Olive fly infestation prediction using machine learning techniques', in *Current Topics in Artificial Intelligence*, eds., Daniel Borrajo, Luis Castillo, and Juan Corchado, volume 4788 of *Lecture Notes in Computer Science*, 229–238, Springer Berlin / Heidelberg, (2007).
[9] Marek J. Druzdzel and Roger R. Flynn, 'Decision support systems', in *Encyclopedia of Library and Information Science*, ed., Allen Kent, volume 67, 120–133, Marcel Dekker, Inc., New York, NY, (2000).
[10] T. Elvira Consortium, 'Elvira: An environment for creating and using probabilistic graphical models', in *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM-02*, eds., J. Gómez and A. Salmerón, pp. 1–11, (2002).
[11] Lidia Fuentes and Antonio Vallecillo, 'An Introduction to UML Profiles', *The European Journal for the Informatics Professional*, **5**(2), (April 2004).
[12] Giovanni Giachetti, Francisco Valverde, and Oscar Pastor, 'Improving automatic uml2 profile generation for mda industrial development', in *ER Workshops*, eds., Il-Yeol Song, Mario Piattini, Yi-Ping Phoebe Chen, Sven Hartmann, Fabio Grandi, Juan Trujillo, Andreas L. Opdahl, Fernando Ferri, Patrizia Grifoni, Maria Chiara Caschera, Colette Rolland, Carson Woo, Camille Salinesi, Esteban Zimányi, Christophe Claramunt, Flavius Frasincar, Geert-Jan Houben, and Philippe Thiran, volume 5232 of *Lecture Notes in Computer Science*, pp. 113–122. Springer, (2008).
[13] O. M. G. Group, *MDA Guide Version 1.0.1*, document omg/03-06-01 edn., june 2003. http://www.omg.org/cgi-bin/doc?omg/03-06-01.
[14] O. M. G. Group, *UML Specification, Version 2.0*, 2005. http://www.omg.org/spec/UML/.
[15] Finn V. Jensen and Thomas D. Nielsen, *Bayesian Networks and Decision Graphs*, Springer Publishing Company, Incorporated, 2nd edn., 2007.
[16] Uffe B Kjrulff and Anders L Madsen, *Bayesian Networks and Influence Diagrams*, Springer New York, 2008.
[17] K. Korb and A. Nicholson, *Bayesian Artificial Intelligence*, Chapman and Hall, 2nd edn., 2010.
[18] François Lagarde, Huáscar Espinoza, François Terrier, and Sébastien Gérard, 'Improving uml profile design practices by leveraging conceptual domain models', in *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, ASE '07, pp. 445–448, New York, NY, USA, (2007). ACM.
[19] Mary Lou Maher and R. H. Allen, *Expert System Components*, American Society of Civil Engineering, 1987.
[20] Francisco Javier Orellana, José del Sagrado, and Isabel María del Águila, 'Saifa: A web-based system for integrated production of olive cultivation', *Comput. Electron. Agric.*, **78**(2), 231–237, (September 2011).
[21] Judea Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
[22] Bran Selic, 'A Systematic Approach to Domain-Specific Language Design Using UML', *, IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, 2–9, (May 2007).

Can Adaptive Conjoint Analysis perform in a Preference Logic Framework?[1]

Adrian Giurca, Ingo Schmitt and Daniel Baier
{giurca, schmitt, daniel.baier }@tu-cottbus.de

Abstract. Research on conjoint analysis/preference aggregation/social choice aggregation is performed by more than forty years by various communities. However, many proposed mathematical models understand preferences as irreflexive, transitive and statical relations while there is human psychology research work questioning these properties as being not enough motivated. This works propose to position the conjoint analysis inside a logical framework allowing for non-transitive and globally inconsistent preferences. Using a preference logics one can define a logic-based utility allowing to obtain an aggregate semantics of the collective choice.

1 Introduction and Motivation

Conjoint Analysis (CA) in marketing research was introduced forty years ago [26] being influenced by economics ([36], [35]) and mathematical psychology ([39], [40], [7]). While the beginning was devoted mostly to understand how individuals evaluate products/services and form preferences (see, [26], [34], [43] and possibly others), in the last thirty years the CA literature focused more on predicting behavioral outcomes by using statistical methods and techniques ([8]) and this resulted in a widespread variation in CA practice. Recently, applications in innovation market were developed ([9]).

The traditional conjoint task is related to the rational economy model where agents tend to action towards maximizing their utilities.

While traditional models obtain significant results when processing *complete*, *transitive* and *acyclic* (consistent) preferences, many communities mention that such models are quite far from the real life. When asking people about thing they like, then they may not answer (*incompleteness*), or they may change their initial preferences due to reception of new information (*preference change*). In addition, while it seems that the preference system of one respondent must be non contradictory, when processing preferences from many respondents this assumption does not remain valid. Some of our previous work argued towards a logic-based model for conjoint analysis.

[1] This research is supported by (1)DFG Project SQ-System: Entwicklung von Konzepten für ein quantenlogikbasiertes Retrieval-Datenbank-Anfragesystem: Anfragesprache, interaktive Suchformulierungsowie effiziente Anfrageauswertung and (2) German Federal Ministry of Education and Research, ForMaT project (Forschung für den Markt im Team), Phase II, Innovationslabor: Multimediale Ähnlichkeitssuche zum Matchen, Typologisieren und Segmentieren

The research reported by [46] proposed a mathematical optimization approach by translating ratings into algebraic constraints, but such solution requires acyclicity and transitivity and not changing preferences. New debates on solution proposed by [46] were reported by [31] in the context of non-additive utility aggregations such as Choquet integral. However, none of these approaches consider non-transitive and/or cyclic preferences, [48].

[23] introduced a logic-based utility but the approach was limited by a number of assumptions such as *consistency* (acyclic preferences) *ignorance* (of neutral rated questions), *transitivity* and the restriction of using only 2 stimuli choice pair comparisons. Moreover, while it argued on the logical nature of the users ratings and rankings, it does not consider *preference change* and interview adaptation. Many of these restrictions were introduced by the method of computing the logic-based utility, basically adaptation of the weighted majority learning algorithm allowing only binary preference as input.

As discussed by [24], computing beliefs from ratings and rankings is much close to the mental expectations of respondents and identified three kinds of beliefs that can be obtained from question answers. The proposed framework considers consistent respondent belief sets but on belief sets aggregation there is no need to require consistency: moreover this is inline with the Arrow's impossibility theorem (see [5] and [6]).

Although traditional non-adaptive conjoint solutions require static, non-changing, preferences, when data collection is interactive one may experience *preference change*. Moreover, the actual online solutions on data collection show many cases when the data is collected over days and not by a standard survey in a contiguous manner. As such, respondents may remake-up their mind therefore change is frequently expected. Also, [24] pointed that may be useful to use weighted beliefs due to the imprecise nature of the user ratings. In addition, among other distinctions it was emphasized that while *individual beliefs are consistent* (no assumption of user irrationality), *collective beliefs may not be consistent*. In addition, while the AGM model [4] considered *consolidation* as a maintenance operation of removing some dispensable beliefs resulting in a consistent knowledge base, we would like to avoid such approach due to missing of motivated criteria with respect of belief elimination.

The goal of this paper is to argue on the opportunity to use a preference logics framework allowing non-transitivity and inconsistency in preference data.

2 Related Work

The classical model of computing an utility function is the additive linear model (see [8] for details). Basically, the overall utility is an additive linear combination on value scores adjusted with attribute scores and compensated with a constant depending on interview i.e.,

$$U(o_j) = \mu + \sum_{k=1}^{N} \sum_{l=1}^{n_i} \beta_{kl} \cdot x_{jkl}$$

where
$U(o_j)$ – is the total score on product profile o_j,
$\beta_{kl} = U_k(a_{kl})$ – is the user preference on value a_{kl} of attribute A_k, and
$x_{jkl} = \begin{cases} 1, \ if \ o_j.A_k = a_{kl} \\ 0, \ otherwise \end{cases}$,
μ is a calibration constant (mean preference value across all objects). Usually $U_k()$ is called part-utility function or part-worth function and its specification depends of the attribute type (categorical and quantitative).

In practice a conjoint study may contain both types of attributes. Significant examples of categorical attributes are brand names or verbal descriptions containing levels such as "high", "medium", "low" while quantitative attributes are the ones which are measurable on either an interval scale or a ratio scale (e.g., speed of a processor, size of a screen). While there were proposed many models to encode the part-worth functions, two models are representative:

1. the *vector model*, $U_k(a_{kl}) = w_k\theta_{kl}$, where w_k is the weight of attribute A_k, and θ_{kl} is the weight of the value $a_{kl} \in dom(A_k)$) and
2. the *ideal point model*, $U_k(a_{kl}) = w_k(\theta_{kl} - \theta_{k0})^2$, where θ_{k0} is the weight of the ideal value a_{k0} of attribute A_k.

In overall the standard conjoint problem reduces to find all β_{kl} and μ by using training data of user-rated utilities for a training object dataset.

2.1 Machine Learning Approaches

During the last thirty years, Machine Learning research developed very similar problems, offering either statistically-based or logic-based solutions. As in traditional conjoint analysis, the difficulty relates to the fact that the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed examples (training data). Hence the learner must generalize from the training data. Learning from examples towards forecasting the future behavior is one large field of research.

2.1.1 Support Vector Machines

Support Vector Machines, [10], [47] was proposed as a classification methodology by machine learning community. Basically, the standard model takes a set of input data and, classify each given input as being part of one of two possible categories (such as "like" and "unlike"). There is research proposing to use this model on conjoint analysis too (e.g., [16]).

The main assumptions of this method are: (a) there is preference data for a set of objects $\mathcal{O}$ and (b) the utility function is linear. Each preference data (e.g., $o_1 \preceq o_2$) is translated into an inequality between corresponding utilities of the corresponding objects ($u(o_1) \leq u(o_2)$). The method then involves minimizing the sum of errors for the inequalities and the sum of of the squares of the weights in the utility function.

As usual, each attribute value $a_{ij} \in dom(A_i)$, $i = 1, ...n$ has weight θ_{ij}. We denote $\overline{\theta}_j^{(k)}$ the weights vector corresponding to the k-th object $o_j^{(k)}$. The goal is to estimate the individual partworths $w = (w_1, ..., w_n)$ considering a linear utility function (e.g., the vector model) $U(o) = \overline{w} \cdot \overline{\theta}$ for each $\overline{\theta}$ corresponding to an object $o \in \mathcal{O}$.

We encode preference data by respondent interviews: at the k-th question we show a subset $\mathcal{O}_k = \{o_1^{(k)}, ..., o_{n_k}^{(k)}\} \subset \mathcal{O}$ asking the respondent to choose one object as "the most liked". Without loosing the generality (via reordering) we can assume that the respondent choose first object as the preferred one. This choice is encoded as the set of constraints, $\overline{w}(\theta_1^{(k)} - \theta_i^{(k)}) \leq 0, i = 2, ..., n_k$, and reduce the conjoint problem to a classification problem. [16] proposes to train a L_2-soft margin classifier only with positive examples obtained from respondent ratings, using a with a hyperplane through the origin and modeling the answering noise with dummy variables $\varepsilon_i^{(k)}$. It trains one algorithm per respondent to get individual vector weights $\overline{w}^{(p)}$ for each respondent p and then to compute individual partworths by calibration with the aggregated partworths i.e. $\widetilde{w} = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \overline{w}^{(p)}$ and then $\overline{w}_*^{(p)} = \frac{\overline{w}^{(p)} + \widetilde{w}}{2}$. The training conditions are:

$$\begin{cases} Minimize : \overline{w}^2 + C \sum_{p_k \in \mathcal{P}} \sum_{i=2}^{n_k} (\varepsilon_i^{(k)})^2 \\ suchthat : \overline{w}(\theta_1^{(k)} - \theta_i^{(k)}) \leq 1 - \varepsilon_i^{(k)} \end{cases}$$

where C is a constant depending on the respondents set.

2.1.2 Learning from Preferences

Recall the learning problem similar with most of conjoint analysis tasks:

> Given a (very large) set of objects (each object represented as a set of attribute-value pairs), and a set of evaluation instances (each object is evaluated by experts obtaining a score, typically a real number) find a learning algorithm being able to evaluate any subset of the initial set of objects being compliant with expert evaluations.

As learning algorithms use evaluated training data it looks straightforward to input the learner with a database of examples in which the human expert has entered scores for each possible choice. However, similar with traditional conjoint analysis, there are two critical issues of this approach: (a) many domains have very large set of possible objects therefore is would be a tremendously time consuming for the expert to create the complete evaluation rank. Moreover, the training dataset must also contain enough "bad" alternatives otherwise the expert will be tempted to produce only high scores for everything and as such, to obtain a rank which is not useful; (b) in many cases experts do not think in terms of absolute scoring functions therefore will be very difficult,

sometimes impossible, to create training data containing absolute scores. These reasons yields many researchers to consider pair comparisons rather than scoring individual alternatives(there is a large literature concerning the way users create preferences. The reader may consider [37], [12], [40], [17] and probably many other). Preference learning was pioneered by [53] and continued by [55], [33], [20] and possibly others. Basically, given a set of (partial) profiles and a preference function of these profiles we want be able to train a computer program to classify new (so far unseen) profiles by assigning a correct rank to each profile. The ratio of correctly classified data points is called the accuracy of the system.

As such conjoint analysis is similar with a learning task: *learning utility functions from respondent preferences.* The conjoint problem can be seen as learning to rank a set of objects by combining a given collection of initial rankings or preference functions. In machine learning community this problem of combining preferences arises in several applications, such as that of combining the results of different search engines, or the collaborative filtering problem. During the last 20 years a number of algorithm were developed: a pioneering algorithm is described in [14] and [15] as an extension of the early work reported by [38]. Advances in learning from preferences were reported by [19], [20], and [30]. As described by [20], the task of learning object preferences is:

> Let $\mathcal{O} = \{(a_1, ..., a_n) | a_i \in dom(A_i)\}$ be the set of all possible product representations and let $\mathcal{S} = \{o_1, ..., o_n\} \subseteq \mathcal{O}$ be a set of training objects (aka full profiles, product representations). Let $\mathcal{P}$ be a set of respondents and $\{P_{\mathcal{S},p} : \mathcal{S} \times \mathcal{S} \longrightarrow \{0, 1\} | p \in \mathcal{P}\}$ the set of pairwise preferences on training data. Learn a utility function $U : \mathcal{O} \longrightarrow \mathbb{R}$ that ranks any subset of $\mathcal{O}$.

Notable, while conjoint analysis typically assume a linear utility function (see details by [8]), learning from preferences does not require utility linearity but many strategies on learning from preferences still assume linear combinations as potential ranking functions. A significant solution introduced by [14] and improved in [15] considers learning a global preference as a weighted linear combination of all respondent preferences, and then derive a final ordering which is maximal consistent with this preference. Other research ([53], [30]) uses a different strategy, specifically direct learning of the utility function directly from the respondent preferences. [53] introduces a two-state symmetric neural network architecture that can be trained with representations of states and a training signal (corresponding to the user preferences) indicated the preferred state. Subsequent works on this solution were reported by [55], [29], [33], and [27].

2.1.3 Logic-based Approaches

A logic-based approach was proposed by [46]by replacing the utility function with a logical formula best fulfilling a set of algebraic constraints derived from preference processing. They use Commuting Quantum Query Language (CQQL, [45]) a logical language based on combinations between Boolean conditions and proximity/similarity conditions over specialized variants of logical operators producing weighted formulas. The problem is formulated as below:

> Let $\mathcal{O} = \{(a_1, ..., a_n) | a_i \in dom(A_i)\}$ be the set of all possible object representations and $\mathcal{S} \subseteq \mathcal{O}$ a set of training objects. $\preceq$ denotes the preference relation on training data $\mathcal{S}$. Find a weighted full DNF CQQL formula $U = \bigvee_j w_j m_j$ (m_j is the j-th minterm and $w_j \in [0, 1]$ its weight) such that U best fulfills the user preferences i.e. when CQQL evaluation is performed over objects in $\mathcal{O}$ then the obtained rank is consistent with user initial preferences.

If $o_{i_2} \preceq o_{i_1}$ then the following constraint is considered

$$eval_{CQQL}(U, o_{i_1}) - eval_{CQQL}(U, o_{i_2}) \geq 0$$

Because CQQL evaluation has simple arithmetic rules for formula evaluation, from the computational point of view the problem reduces to a linear optimization: $Maximize : \sum_{o_{i_2} \preceq o_{i_1}} (eval_{CQQL}(U, o_{i_1}) - eval_{CQQL}(U, o_{i_2}))$ under the above described constraints. The readers may consider [46] for details on problem solving strategies (such as simplex computations, feasible and unfeasible states, solutions to avoid overfitting and more.)

Automated extraction of rules from evidences was largely discussed by connectionist learning community (early work by [41], pioneered by [21] and subsequently discussed by [51], [25], [52], [11], [49], and possibly others) under the umbrella of a much general task:

> How can we extract models from the training data in an automated manner and use these models as the basis of an autonomous rational agent in the given domain.

One of the most important features of such an approach is that it combines the computational advantages of connectionist models with the qualitative knowledge representation proposed by the AI community.

It is obvious that a solution of this problem must consider two stages: (1) Learning the model and (2)Performing inference using this model. This work follows only the first stage of the problem – if there is a learned ruleset then there are many opportunities to perform inference according with various semantics (crisp, probabilistic, fuzzy and so on) and a discussion of appropriateness of each of them should be large.

Inside a rule framework the conjoint problem is to find out a set of rules that best model the respondent preferences. One can consider learning of various kinds of rules (possibly weighted), each of them supporting various semantics including probabilistic models [42], incomplete/imprecise information, [54], plausibility-based models [18], [22] or quantum logic semantics [45]:

1. Simple rules (propositional rules):

   $$[(\neg)A_{i_1} \wedge ..., \wedge (\neg)A_{i_k} \rightsquigarrow A_{i_{k+1}}]$$

   where $(\neg)A$ denotes a possibly negated attribute;
2. Positive attribute-value rules:

   $$[A_{i_1} \simeq v_{i_1} \wedge ..., \wedge A_{i_k} \simeq v_{i_k} \rightsquigarrow A_{i_{k+1}} \simeq v_{i_{k+1}}]$$

   where $v_{i_j} \in dom(A_{i_j})$, $A_{i_j} \simeq v_{i_j}$ means that A_{i_j} takes a value around v_{i_j} (The reader should notice that $\simeq$ includes ordinal values, e.g., $A_{i_j} = v_{i_j}$);

3. Attribute-value rules with negation:

$$[(\neg)A_{i_1} \simeq v_{i_1} \wedge ..., \wedge(\neg)A_{i_k} \simeq v_{i_k} \rightsquigarrow A_{i_{k+1}} \simeq v_{i_{k+1}}]$$

where $\neg A_{i_j} = v_{i_j}$ means $A_{i_j} \neq v_{i_j}$;

4. General attribute-value rules:

$$[(\neg)A_{i_1} \simeq v_{i_1} \wedge ..., \wedge(\neg)A_{i_k} \simeq v_{i_k} \rightsquigarrow (\neg)A_{i_{k+1}} \simeq v_{i_{k+1}}]$$

The first three kinds of rules were largely addressed by data mining community when learning association rules. Researchers developed different kinds of association rules: Boolean (crisp) association rules, quantitative association rules, fuzzy association rules. Association rules were pioneered by [44] and then established by [2], and [3]). Standard association rules consider two measures of interestingness: *support* and *confidence* although other models may add two more: *lift* and *conviction* or adopt non-standard ones, [32]. Learning association rules is usually performed under both a user-specified minimum support and a user-specified minimum confidence requirements.

There were developed many algorithms starting with the most known one, Apriori ([3]) and continuing with many others (Eclat, FP-growth and so on.) A significant step is the Assoc algorithm [28] which enables mining for generalized association rules (including negation i.e. attribute-value rules with negation) and does not restrict for minimum support and confidence.

However, on our knowledge, none of this research considering the conjoint analysis task: basically the training data set for learning association rules does not distinguish various users. All the data is uniform (mostly, it comes from e-commerce transactions) and it may refer to one user (such as in recommender systems, [1]) or to many but not considering distinct training data for each of them, therefore the conjoint task is somehow hidden. In addition the conjoint analysis problem in the context of learning association rules does not directly performs from preferences: using transactional data as input, there should be some algorithm computing binary preferences.

The first kind of rules were considered, in context of adaptive conjoint analysis, by [23] in conjunction with weighted CQQL (see [45] for language description), an extension of the relational calculus using quantum logic paradigm which defines *metric*(or *similarity*) predicates, weighted conjunction ($\wedge_{\theta_1,\theta_2}$), weighted disjunction ($\vee_{\theta_1,\theta_2}$) and quantum negation. Clearly (as explained by [25] and [52]) there is a need for both a preference measure to rank the rules and a learning algorithm which uses the preference measure to find the best k rules. The work reported by [23] describes a heuristic and learning approach to use the respondent preferences on stimuli to compute a rule preference relation (called minterm preference because the rules were learned as weighted minterms of the CQQL full disjunctive normal form) and then use a learning algorithm to compute a ranking on the minterms set.

3 Conjoint Analysis using Preference Logics

This section introduces a logical framework allowing (a) encoding of preferences as choice formulas, (b) defining a logic-based utility inside a preference logic to allow creation of collective beliefs and (c) performing rule extraction and explanation and formal interpretation.

3.1 Preference Logics

We follow the approach defined by [50] on preference logic introduced as a special case of logic by defining a preference relation between the interpretations of the underlining logic as we consider this approach being simple and powerful. Below we recall some of the [50] results.

Let $\mathcal{L}$ be a standard logic and $\sqsubset$ a strict partial order on interpretations (we say $\mathcal{I}_2$ is preferred to $\mathcal{I}_1$ and denote $\mathcal{I}_1 \sqsubset \mathcal{I}_2$). Then, $\mathcal{L}_\sqsubset = (\mathcal{L}, \sqsubset)$ is a new logic, a preference logic. The basic artifacts such as satisfaction, validity and entailment are defined by [50]. Recall that while the standard logics are monotonic[2]. Recall the definitions of satisfiability, validity and entailment:

Definition 1 ([50])
Let $F, G \in \mathcal{L}$. Let $\mathcal{I}$ be an interpretation.
$\mathcal{I}$ preferentially satisfies F *(denoted $\mathcal{I} \models_\sqsubset F$) if $\mathcal{I} \models F$ and there is no $\mathcal{I}'$ such that $\mathcal{I} \sqsubset \mathcal{I}'$ and $\mathcal{I}' \models F$. As usual, $\mathcal{I}$ is called the* model *of F.*
F preferentially entails G *(denoted $F \models_\sqsubset G$) if*

$$\forall \mathcal{I}, \mathcal{I} \models_\sqsubset F \Rightarrow \mathcal{I} \models_\sqsubset G$$

That is the preferred models of G are also preferred models of F.

As described by [50], $\mathcal{L}_\sqsubset$ is a non-monotonic logic because there may be formulas $F, G \in \mathcal{L}_\sqsubset$ such that both $F \models_\sqsubset G$ and $F \models_\sqsubset \neg G$. Moreover, it is not necessary that F is inconsistent, it is just sufficient that F do not have preferred models.

A significant case of preference logics was introduced by [13] under the name of choice logic. Basically, choice logic defines the ordered disjunction (denoted $\times$) as a special kind of standard disjunction ($\vee$) as such introducing a preference relation between the interpretations and models. The ordered disjunction has the same models as regular disjunction but there is a preference relation between these models. For example, if $A \times B$ is a disjunction between two atoms. Then $\mathcal{I}_1 = \{A\}$, $\mathcal{I}_2 = \{A, B\}$ and $\mathcal{I}_3 = \{B\}$ are its models. Then $\mathcal{I}_3 \sqsubset \mathcal{I}_2$ and $\mathcal{I}_3 \sqsubset \mathcal{I}_1$ meaning that $\mathcal{I}_1$ and $\mathcal{I}_2$ are preferred models.

Intuitively, as [13] reports, the ordered disjunction means that when $F_1 \times ...F_n$ we prefer models that first satisfies F_1 and if this is not possible then we prefer models satisfying F_2, and so on. Choice logic defines the degree of satisfaction for all logic formulas

Definition 2 ([13])
The optionality *of a formula (the number of choices to satisfy a formula) is $opt(A) = 1$ if A is an atom.*
$opt(\neg F) = 1$
$opt(F_1 \vee F_2) = \max(opt(F_1), opt(F_2))$
$opt(F_1 \wedge F_2) = \max(opt(F_1), opt(F_2))$
$opt(F_1 \times F_2) = opt(F_1) + opt(F_2)$

[13] defines the preference relation ($\sqsubset$) between models of logic formulas and consequently the entailment. It is shown that

[2] In the sense that if $F_1, F_2, F_3 \in \mathcal{L}$, if $F_1 \models F_3$ then $F_1 \wedge F_2 \models F_3$.

the entailment satisfies cautious monotony and cumulative transitivity:

Proposition 1 ([13])
Let S be a set of choice logic formulas and A, B be classical formulas.
$S \models_{\sqsubset} A$ *and* $S \models_{\sqsubset} B \Rightarrow S \cup \{A\} \models_{\sqsubset} B$
$S \models_{\sqsubset} A$ *and* $S \cup \{A\} \models_{\sqsubset} B \Rightarrow S \models_{\sqsubset} B$

From the computational point of view, choice logic can be translated to stratified knowledge bases.

4 Modeling Conjoint Analysis

Conjoint analysis collects preferences from user interviews using a variety of question types but the most used ones are trade-off matrices and pair-comparisons. A trade-off matrix ([34]) asks a respondent to consider a pair of attributes. It displays all combinations of values for those attributes, asking the respondents to provide a ranking for the combinations. The Table 1 show an example of a trade-off matrix related to attributes *OperatingSystem* and *Battery life*. While trade-off

	12 hours	6 hours	4 hours	2 hours
Android	1	2	7	5
WinPhone	3	4	6	11
other OS	8	9	10	12

Table 1. A trade-off matrix with respondent ranking

matrix are quite efficient on ranking binary stimuli, trade-off matrices cannot be used if we consider stimuli with more than two attributes. A solution to these limitations is to use pair comparisons. Pair comparisons are seen as choice questions

Left side	OR	Right side
Android AND $\geq$ 500EUR, ...	Left	Windows Phone,... AND $\leq$ 3.5" screen, ...
$\geq$ 4" screen,... AND Battery life 6h	Neutral	And,... AND WIFI, ...
$\geq$ 4" screen,... AND other OS	Left	Battery life 10h,... AND no WIFI, ...

Table 2. Pair Comparisons and Ratings

evaluated by favoring either "the left side" or "the right side" or "neutral".

4.1 Preferences as Choice Formulas

Let $A_1, ..., A_n$ be a set of attributes (unary predicates) with $dom(A_i)$ the domain of values. Let $\mathcal{O} = \{(a_1, ..., a_n) | a_i \in dom(A_i)\}$ be the set of all possible product representations. The choice logic ordered disjunction operator makes this logic suitable candidate to encode user ratings as choice formulas. The trade-off matrices introduces a rank between choices e.g., the matrix from Table 1 say that $OS(\text{"}Android\text{"}) \wedge Battery(\text{"}12h\text{"})$ is preferred to $OS(\text{"}Android) \wedge Battery(\text{"}6h\text{"})$ as well as $OS(\text{"}WinPhone\text{"}) \wedge Battery(\text{"}12h\text{"})$ is preferred to $OS(\text{"}Android) \wedge Battery(\text{"}4h\text{"})$ and so on.

Definition 3 (Mapping trade-off matrices)
Let a trade-off matrix based on predicates A_1 and A_2.
If $A_1(u) \wedge A_2(v)$ is preferred to $A_1(u') \wedge A_2(v')$ then this preference is encoded into the choice formula:

$$A_1(u) \wedge A_2(v) \times A_1(u') \wedge A_2(v')$$

that is preferring models that, if possible first satisfy $A_1(u) \wedge A_2(v)$[3].

Definition 4 (Mapping pair comparisons)
Let q be the pair comparison
$q = A(a)$ and $B(b)$ OR $C(c)$ and $D(d)$.
If the left side is preferred then this preference is encoded into the choice formula:

$$A(a) \wedge B(b) \times C(c) \wedge D(d)$$

If q is rated neutral then this preference is encoded into the formula:

$$A(a) \wedge B(b) \vee C(c) \wedge D(d)$$

Similarly, if the right side is preferred then this preference is encoded into the choice formula:

$$C(c) \wedge D(d) \times A(a) \wedge B(b)$$

4.2 Towards Logic-based Conjoint Analysis

Let $\mathcal{A} = \{A_1, ..., A_n\}$ be a set of unary predicates with $dom(A_i)$ the domain of values. Let $\mathcal{O} = \{(a_1, ..., a_n) | a_i \in dom(A_i)\}$ be the set of all possible product representations.

Definition 5 (Normal Form)
A full ordered disjunctive normal form *(ODNF) over choice logic defined by the language $\mathcal{A}$ is a formula*

$$U = \times_j (L_1(l_1^j) \wedge ... \wedge L_n(l_n^j))$$

where $L_k^(l_k^j)$ is a literal corresponding to the predicate A_k (either $A_k(l_k^j)$ or $\neg A_k(l_k^j)$) and $l_k^j \in dom(A_k)$.

Let $\mathcal{C}$ the set of all choice formulas derived from user preferences. Then, the generic conjoint analysis task is described as below:

> Find $U = \times_j (L_1(l_1^j) \wedge ... \wedge L_n(l_n^j))$ such that U best fulfills the user preferences i.e. there is a maximal set of constraints $\mathcal{C}' \subseteq \mathcal{C}$ such that $U \models_{\sqsubset} C$ for all $C \in \mathcal{C}'$.

Of course, the economics community does not really need the complete DNF but, most of the cases only a subset of the ODNF (the most important clauses). In addition, sometimes the constraints may come weighted (using some weight $w \in (0, 1]$) and then the concept of maximal set can be replaced by a subset of constraints with a sum of weights greater than a specified threshold.

[3] This corresponds completely to the psychological meaning of trade-off matrices where the respondent *does not reject* any of the alternatives

Rule extraction from a computed ODNF (or a subset) is straightforward as the experts like to understand the dependencies of a specific predicate value with respect of the remaining predicates. As such rules are obtained by transforming U to conjunctive normal form (CNF) and then deriving rules from each clause according with specific predicates as conclusions.

Let $\mathcal{R}$ be a the derived ruleset as described above. Then, all preferred models of $\mathcal{R}$ corresponds to preferred objects in $\mathcal{O}$.

As such we propose an updated process chain of adaptive logic-based conjoint analysis as depicted by Figure 1.

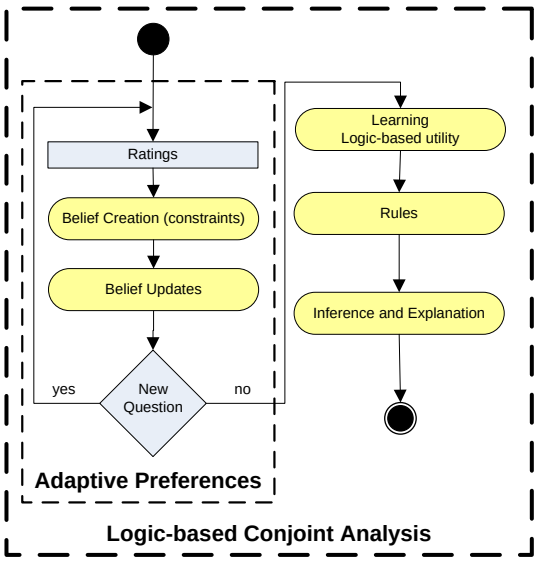


Figure 1. Logic-Based Adaptive Conjoint Analysis Chain

5 Conclusion

We proposed a model of logic-based conjoint analysis by considering encoding respondent preferences as beliefs (as such allowing belief change) and encoding this beliefs to choice formulas. While the individual beliefs translates into consistent constraints set the collective beliefs (all constraints collected from all respondents) may not be a consistent set. The Table 3 describes the kind of preferences used by the analyzed models. As seen the proposed approach is useful when the model intends to capture phycological phenomena such as change or irrationality (inconsistency) as well as when formal explanations of decisions need to be computed. This work is at its beginnings: beside fine tuning and debugging, obtaining feasible algorithms to compute the logic-based utility is a mandatory next step. Analyzing such algorithms may open discussion on improvements of the preference logic too as traditional processing of pair comparisons also consider Likert scales as rating methods. In addition, a close look on the necessary belief framework (a discussion was started by [24]) is necessary.

Aggregation Models	Require Irreflexive	Require Transitive	Allow Indifference	Static Preference
CA (econ.)	yes	yes	yes	yes
SVM	yes	yes	no	yes
Preference Learning	yes	yes	no	yes
Rule Learning	yes	yes	no	yes
Preference Logic	yes	**no**	yes	**no** (belief rev)

Table 3. Conjoint Analysis Preference Requirements

REFERENCES

[1] G. Adomavicius, and A. Tuzhilin, Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions, IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 6, June 2005, pp. 734-749.

[2] R. Agrawal, T. Imielinski, and A.N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia (Eds.), Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., pp. 207-216, May 26-28, 1993.

[3] R. Agrawal, and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo (Eds.), Proc. 20th Int. Conf. Very Large Data Bases, (VLDB), pp. 487-499, Morgan Kaufmann, 1994.

[4] C.E. Alchourron, P. Gärdenfors and D. Makinson. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. Journal of Symbolic Logic, 50: 510-530, 1985.

[5] K.J. Arrow. A Difficulty in the Concept of Social Welfare. Journal of Political Economy 58(4) (August, 1950), pp. 328-346.

[6] K. J. Arrow. Social Choice and Individual Values. 2nd ed., 1963.

[7] N.H. Anderson. Foundations of information integration theory. Academic Press, 1981.

[8] D. Baier and M. Brusch (Eds.) Conjointanalyse, Methoden - Anwendungen - Praxisbeispiele, Springer, Berlin, 2009.

[9] D. Baier. Conjoint Measurement in der Innovationsmarktforschung, in: Baaken, Thomas; Höft, Uwe; Kesting, Tobias (Hrsg.), Marketing für Innovationen - Wie innovative Unternehmen die Bedürfnisse ihrer Kunden erfüllen, Harland Media, Münster, ISBN-13 978-3-938363-42-3.

[10] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In Proc. 5th Annu. Workshop on Comput. Learning Theory, 1992.

[11] O. Boz. Knowledge Integration and Rule Extraction. Neural Networks, University of Leigh, 1995.

[12] R.A. Bradley and M.E. Terry. Rank analysis of incomplete block designs: the method of paired comparisons. Biometrika, 39 (3-4), 1952, pp.324-345.

[13] G. Brewka, S. Benferhat and D. Le Berre. Qualitative Choice Logic. Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002, pp.158-169.

[14] W. Cohen, R.E. Schapire and Y. Singer. Learning to Order Things. Advances in Neural Information Processing Systems 10, Morgan Kaufmann, 1998.

[15] W. Cohen, R.E. Schapire and Y. Singer. Learning to Order Things. Journal of Artificial Intelligence Research 10, pp. 213-270, 1999.

[16] T. Evgeniou, C. Boussios, and G. Zacharia. Generalized robust conjoint estimation. Marketing Science, 25, 2005.

[17] J. Eliashberg. Consumer Preference Judgments: An Exposition with Empirical Applications. Management Science, 26, 1, (January), 1980, pp.60-77.

[18] N. Friedman, and J.Y. Halpern. Plausibility measures and default reasoning. Journal of the ACM, 48, 2001, pp.648-685.

[19] J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. Procs. of the 14th European Conference on Machine Learning (ECML-03), LNAI 2837, 2003, pp.145-156, Springer Verlag, 2003.
[20] J. Fürnkranz and E. Hüllermeier. Preference learning. Künstliche Intelligenz, 19(1), pp. 60-61, 2005.
[21] S.I. Galant. Connectionist Expert Systems. Communications of ACM, 31, 1988, pp.152-169.
[22] A. Giurca. A Logic with Plausibility. Annales of Craiova University, Mathematics and Computer Science Series, XXVII, pp.105-115, 2000.
[23] A. Giurca, I. Schmitt, and D. Baier. Performing Conjoint Analysis within a Logic-based Framework. Proc of IEEE Federated Conference on Computer Science and Information Systems, (FedCSIS2011), Szczecin, Poland, 18-21 September, 2011.
[24] A. Giurca, I. Schmitt, and D. Baier. Adaptive Conjoint Analysis. Training Data: Knowledge or Beliefs? A Logical Perspective of Preferences as Beliefs, KAM'2012 - 18th Conference on Knowledge Acquisition and Management, at FEDCSIS 2012, Wroclaw, Poland.
[25] R. M. Goodman, C. M. Higgins, J. W. Miller, and P. Smyth. Rule-Based Neural Networks for Classification and Probability Estimation. Neural Computation 4(6), pp.781-804, 1992.
[26] P. E. Green and V. Rao. Conjoint measurement for quantifying judgmental data. Journal of Marketing Research, 8, 1971, pp.355-363.
[27] P. Haddawy, V. Ha, A. Restificar, B. Geisler, and J. Miyamoto. Preference elicitation via theory refinement. Journal of Machine Learning Research, 4, pp. 317-337, 2003.
[28] P. Hájek. The new version of the GUHA procedure ASSOC, COMPSTAT 1984, pp.360-365.
[29] Ralf Herbrich, Thore Graepel, Peter Bollmann-Sdorra, and Klaus Obermayer. Supervised learning of preference relations. Procs. des Fachgruppentreffens Maschinelles Lernen (FGML-98), 1998, pp. 43-47.
[30] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences, Artificial Intelligence, Volume 172, Issues 16-17, pp. 1897-1916, 2008.
[31] E. Hüllermeier and I. Schmitt. Non-Additive Utility Functions: Choquet Integral versus Weighted DNF Formulas, The 4th Japanese-German Symposium on Classification (JGSC2012),March 9-10, 2012, Kyoto, Japan.
[32] I. Iancu, M. Gabroveanu and A. Giurca. A Pair of Confidence Measures for Association Rules, 30th Annual Conference of the German Classification Society, GfKl2006, March 8-10, 2006, Berlin, Germany.
[33] T. Joachims. Optimizing search engines using clickthrough data. Procs. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-02), pp. 133-142. ACM Press, 2002.
[34] R. M. Johnson. Tradeoff Analysis of Consumer Values. Journal of Marketing Research, 1974, pp. 121-127.
[35] R.L. Keeney and H. Raiffa. Decisions with multiple objectives: Preferences and value tradeoffs. Wiley Series in Probability and Mathematical Statistics. NY: John Wiley & Sons, 1976.
[36] K. Lancaster. A new approach to consumer theory. Journal of Political Economy, 74, 1966, pp.132-157.
[37] R. Likert. A Technique for the Measurement of Attitudes. Archives of Psychology 140, 1932, pp.1-55.
[38] N. Littlestone and M. Warmuth. The weighted majority algorithm. Information and Computation, 108 (2), 1994, pp. 212-261.
[39] R.D. Luce and J. W. Tukey. Simultaneous Conjoint Measurement: A New Type of Fundamental Measurement. Journal of Mathematical Psychology, 1, 1964, pp.1-27.
[40] R.D. Luce and P. Suppes. Preference, utility and subjective probability. in Luce, R.D., Bush, R.R., and Galanter, E. (Eds.), Handbook of Mathematical Psychology, III, New York: Wiley, 1965, pp.235-406.
[41] M. C. Mozer. RAMBOT: A Connectionist Expert System That Learns by Example. Tech. Report. California Univ., San Diego, La Jolla. Inst. for Cognitive Science, 1986.
[42] N. J. Nilsson. Probabilistic logic. Artificial Intelligence 28(1), pp.71-87, 1986.
[43] K.L. Norman and J.J. Louviere. Integration of attributes in public bus transportation: two modeling approaches. Journal of Applied Psychology, 59, 6, 1974, pp.753-758.
[44] G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules, in G. Piatetsky-Shapiro and W. J. Frawley (eds): Knowledge Discovery in Databases. AAAI/MIT Press, Cambridge, MA, 1991.
[45] I. Schmitt. QQL: A DB&IR Query Language. VLDB Journal, 17(1), pp.39-56, 2008.
[46] I. Schmitt, and D. Baier. Logic Based Conjoint Analysis using the Commuting Quantum Query Language, Proc. of Conference of the German Classification Society (GfKl2011), August 31 to September 2, 2011, Frankfurt am Main, Germany.
[47] B. Schölkopf and A. Smola. Learning with kernels. MIT Press, 2002.
[48] G.F. Schumm. Transitivity, Preference and Indifference. Philosophical Studies, 52: 435-437, 1987.
[49] J. Sima. Neural Expert System. Journal of Neural Networks, vol. 8, no. 2, pp. 261-271, 1995.
[50] Y. Shoham. Nonmonotonic Logics: meaning and utility. Proceedings of 10th IJCAI, pp.388-393, Milan, 1987.
[51] P. Smyth, and R. M. Goodman. An Information Theoretic Approach to Rule Induction from Databases. IEEE Trans. Knowl. Data Eng. 4(4), pp.301-316, 1992.
[52] R. Sun. Integrating rules and connectionism for robust commonsense reasoning. Hoboken, N.J: Wiley & Sons, 1994, ISBN 0-471-59324-9.
[53] G. Tesario. Connectionist learning of expert preferences by comparison training. Advances in Neural Information Processing Systems, 1, pp. 99-106, Morgan Kaufmann, 1989.
[54] G. Wagner. Logic Programming with Strong Negation and Inexact Predicates. Journal of Logic and Computation 1(6), pp. 835-859, 1991.
[55] J. Wang. Artificial neural networks versus natural neural networks:A connectionist paradigm for preference assessment. Decision Support Systems, 11, pp. 415-429, 1994.

Problems impacting the quality of automatically built ontologies

Toader GHERASIM[1] and **Giuseppe BERIO**[2] and **Mounira HARZALLAH**[3] and **Pascale KUNTZ**[4]

Abstract. Building ontologies and debugging them is a time-consuming task. Over the recent years, several approaches and tools for the automatic construction of ontologies from textual resources have been proposed. But, due to the limitations highlighted by experimentations in real-life applications, different researches focused on the identification and classification of the errors that affect the ontology quality. However, these classifications are incomplete and the error description is not yet standardized. In this paper we introduce a new framework providing standardized definitions which leads to a new error classification that removes ambiguities of the previous ones. Then, we focus on the quality of automatically built ontologies and we present experimental results of our analysis on an ontology automatically built by Text2Onto for the domain of composite materials manufacturing.

1 Introduction

Since the pioneering works of Gruber [15], ontologies play a major role in knowledge engineering whose importance is growing with the rise of the semantic Web. Today they are an essential component in numerous applications in various fields: e.g. information retrieval [22, 20], knowledge management [26], analysis of social semantic networks [8] and business intelligence [27]. However, despite the maturity level reached in ontology engineering, important problems remain open and are still widely discussed in the literature. The most challenging issues concern the automation of ontology construction and their evaluation.

The increasing popularity of ontologies and the scaling changes of this last decade have motivated the development of ontology learning techniques. Promising results have been obtained [6, 5]. And, although these techniques have been often experimentally proved to be not sufficient enough for constructing ready-to-use ontology [5], their interest is not questioned in particular in technical domains [17]. Few recent works recommend an integration between ontology learning techniques and manual intervention [27].

Whatever their use, it is essential to assess their quality throughout their development. Several ontology quality criteria and different evaluation methods have been proposed in the literature [19, 4, 11, 21, 1]. However, as mentioned by [28], defining "a good ontology" remains a difficult problem and the different approaches only permit to "recognize problematic parts of an ontology". From an operational point of view, error identification is a very important step for the ontology integration in real-life complex systems. And, different researches recently focused on that issue [13, 2, 24]. However, as far as we know, a generic standardized description of these errors does not still exist. It seems however a preliminary step for the development of assisted construction method.

In this paper, we focus on the most important errors that affect the quality of semi-automatically built ontologies. To get closer the operational concerns we propose a detailed typology of the different types of problems that can be identified when evaluating an ontology. Our typology is inspired from a generic standardized description of the notion of quality in conceptual modeling [18]. And, our analysis is applied on a real-life situation concerning the manufacturing of pieces in composite materials for the aerospace industry.

The rest of this paper is organized as follows. Section 2 is a state-of-the art of the ontology errors. Section 3 describes a framework which provides a standardized description of the errors and draws correspondences between our new classification and the main errors previously identified in the literature. Section 4 presents our experimental results in the domain of composite materials manufacturing. More precisely, we analyze errors affecting an ontology produced by an automatic construction tool (here Text2Onto) from a set of technical textual resources.

2 State-of-the art on ontological errors

In the literature, the notion of "ontological error" is often used in a broad sense covering a wide variety of problems which affect the ontology quality. But, from several studies published this last decade, we have identified four major denominations associated to complementary definitions: (1) "taxonomic errors" [14, 13, 9, 2], (2) "design anomalies" or "deficiencies" [2, 3], (3) "anti-patterns" [7, 25, 23], and (4) "pitfalls" or "worst practices [23, 24].

2.1 Taxonomic errors

From the pioneering works of Gomez-Perrez [14], the denomination "taxonomic error" is used to refer to three types of errors that affect the taxonomic structure of ontologies: inconsistency, incompleteness and redundancy. Recently, extensions have been proposed to non-taxonomic properties [3], but in this synthesis we focus on taxonomic errors.

Inconsistencies in the ontology may be logical or semantic. More precisely, three classes of inconsistencies in the taxonomic structure have been detailed: circularity errors (e.g. a concept that is a specialization or a generalization of itself), partitioning errors which produce logical inconsistencies (e.g. a concept defined as a specialization of two disjoint concepts), and semantic errors (e.g. a taxonomic relationship between two concepts that is not consistent with the semantics of the latter).

[1] LINA, UMR 6241 CNRS, e-mail: toader.gherasim@univ-nantes.fr
[2] LABSTICC, UMR 6285 CNRS, email: giuseppe.berio@univ-ubs.fr
[3] LINA, UMR 6241 CNRS, e-mail: mounira.harzallah@univ-nantes.fr
[4] LINA, UMR 6241 CNRS, e-mail: pascale.kuntz@polytech.univ-nantes.fr

Incompleteness is met when concepts or relations of specialization are missing, or when some distributions of the instances of a concept between its sons are not stated as exhaustive and/or disjoint.

In the opposite way, *redundancy* errors are met when a taxonomic relationship can be directly deduced by logical inference from the other relationships of the ontology, or when concepts with the same father in the taxonomy do not share any common information (no instances, no children, no axioms, etc.) and can be only differentiated by their names.

2.2 Design anomalies

Roughly speaking, design anomalies mainly focus on ontology understanding and maintainability. They are not necessarily errors but undesirable situations. Five classes of design anomalies have been described: (1) "lazy concepts" (leaf concepts in the taxonomy not implied in any axiom and without any instances); (2) "chains of inheritance" (long chains composed of intermediate concepts with a single child); (3) "lonely disjoint" concepts (superfluous disjunction axiom between distant concepts in the taxonomy which may disrupt inference reasoning); (4) "over-specific property range" (too specific property range which should be replaced by a coarser range which fits the considered domain better); (5) "property clumps" (duplication of the same properties for a large set of concepts instead of the inheritance of these properties from a more general concept).

2.3 Anti-patterns

Ontology design patterns (ODP) are formal models of solutions commonly used by domain experts to solve recurrent modeling problems. Anti-patterns are ODP that are *a priori* known to produce inconsistencies or unsuitable behaviors. [23] also called anti-patterns *ad-hoc* solutions specifically designed for a problem even if well-known ODP are available. Three classes of anti-patterns have been described [7, 25, 23]: (1) "logical anti-patterns" that can be detected by logical reasoning; (2) "cognitive anti-patterns" (possible modeling errors due to misunderstanding of the logical consequences of the used expression); (3) "guidelines" (complex expressions valid from a logical and a cognitive point of view but for which simpler or more accurate alternatives exist).

2.4 Pitfalls

Pitfalls are complementary to ODPs. Their broad definition covers problems affecting the ontology quality for which ODPs are not available. Poveda et al. [24] described 24 types of experimentally identified pitfalls as, for instance, forgetting the declaration of an inverse relation when this latter exists or of the attribute range. And they proposed a pitfall classification which follows the three evaluable dimensions of an ontology proposed by Gangemi et al. [11]: (1) structural dimension (aspects related to syntax and logical properties), (2) functional dimension (how well the ontology fits a predefined function), (3) the usability dimension (to which extent the ontology is easy to be understood and used). Four pitfall classes correspond to the structural dimension: "modeling decisions" (*MD*, situations where OWL primitives are not used properly), "wrong inference" (*WI*, e.g. relationships or axioms that allow false reasoning), "no inference" (*NI*, gaps in the ontology which do not allow inferences required to produce new desirable knowledge), "real world modeling" (*RWM*, when commonsense knowledge is missing in the ontology). One class corresponds to the functional dimension: "requirement completeness" (*RC*, when the ontology does not cover its specifications). And, two classes correspond to the usability dimension: "ontology understanding" (*OU*, information that makes understandability more difficult e.g. concept label polysemy or label synonymy for distinct concepts, non explicit declaration of inverse relations or equivalent properties) and "ontology clarity" (*OC*, e.g. variations of writing-rule and typography for the labels).

It is easy to deduce from this classification that some pitfalls should belong to different classes associated to different dimensions (e.g. the fact that two inverse relations are not stated as inverse is both a "no inference" (*NI*) pitfall and an "ontology understanding" (*OU*) pitfall). Another attempt [24] proposed a classification of the 24 identified pitfalls in the three error classes (inconsistency, incompleteness and redundancy) given by Gomez-Perrez et al. [14]. But, these classes are concerned by the ontology structure and content, and consequently four pitfalls associated with the ontology context do not fit with this classification.

In order to highlight the links between the different classifications, Poveda et al. tried to define a mapping between the classification in 7 classes deduced from the dimensions defined by Gangemi et al. [11] and the 3 error classes proposed by Gomez-Perrez et al. [14]. However, this task turned out to be very complex, and only four pitfall classes exactly fit with one of the error classes. For the other, there is overlapping or no possible fitting.

3 The framework

The state of the art briefly presented in the previous section shows that the terminology used for describing the different problems impacting on the quality of ontologies is not yet standardized and that existing classifications do not cover the whole diversity of problems described in the literature.

In this section we present a framework providing standardized definitions for quality problems of ontologies and leading to a new classification of these problems. The framework comprises two distinct and orthogonal dimensions: errors vs. unsuitable situations (first dimension) and logical facet vs. social facet of problems (second dimension).

Unsuitable situations identify problems which do not prevent the usage of an ontology (within specific targeted domain and applications). On the contrary, errors identify problems preventing the usage of an ontology.

It is well known that one ontology has two distinct facets: an ontology can be processed by machines (according to its logical specification) and can be used by humans (including an implicit reference to a social sharing).

The remainder of the section is organized alongside the second dimension (i.e. logic vs. social facet) and within each facet, errors and unsuitable situations are defined. The framework is based on "natural" analogies between respectively social and logical errors and social and logical unsuitable situations.

3.1 Problem classification

3.1.1 Logical ground problems

The logical ground problems can be formally defined by considering notions defined by Guarino et al. [16]: e.g. Interpretation (Extensional first order structure), Intended Model, Language, Ontology and the two usual relations $\models$, $\vdash$ provided in any logical language. The relation $\models$ is used to express both that one interpretation I is a

model of a logical theory T, written as $I \models T$ (i.e. all the formulas in T are true in I, written for each formula $\varphi \in T$, $I \models \varphi$), and also for expressing the logical consequence (i.e. that any model of a logical theory T is also a model of a formula, written as $T \models \varphi$). The relation $\vdash$ is used to express the logical calculus i.e. the set of rules used to prove a theorem (i.e. any formula) φ starting from a theory T, written as $T \vdash \varphi$.

Examples and formalizations hereinafter are provided by using a typical Description Logics notation (but easily transformable in first order or other logics).

The usual **logical ground errors** are listed below.

1. *Logical inconsistency* corresponding to ontologies containing logical contradictions for which a model does not exist (because the set of intended models is never empty, an ontology without models does not make sense anyway; formally, given an ontology O and the logical consequence relation $\models$ according to the logical language L used for building O, there is no interpretation I of O such that $I \models O$). For example, if an ontology contains the following axioms $B \subseteq A$ ($B\ is\_a\ A$), $A \cap B \subseteq \top$ (A and B are $disjoint$), $c \subseteq B$ (c is $instance\_of\ B$), then $c \subseteq A$ and $c \subseteq A \cap B$, so there is a logical contradiction in the definition of this ontology;
2. *Unadapted*[5] *ontologies* wrt to intended models[6] i.e. an ontology for which something that is false in all (some of) the intended models of L is true in the ontology; formally, there exists a formula φ such that for each (for some) intended model(s) of L, φ is false and $O \models \varphi$. For example, if we have in the ontology two concepts A and B that are declared as disjoint ($O \models A \cap B \subseteq \bot$) and in each intended model there exists an instance c that is common between A and B (i.e. $c \subseteq A \cap B$), then the ontology is unadapted;
3. *Incomplete ontologies* wrt to intended models i.e. an ontology for which something that is true in all the intended models of L, is not necessarily true in all the models of O; formally, there exists a formula φ such that for each intended model of L, φ is true and $O \not\models \varphi$. As an example, if in all the intended models $C \cup B = A$, and the ontology O defines $B \subseteq A$ and $C \subseteq A$, it is not possible to prove that $C \cup B = A$;
4. *Incorrect (or unsound) reasoning* wrt the logical consequence i.e. when some specific conclusions are derived by using suitable reasoning systems for targeted ontology applications even if these conclusions are not true in the intended models and must not be derived by any reasoning according to the targeted ontology applications (formally, when a specific formula φ, false in the intended models $O \not\models \varphi$, can be derived $O \vdash \varphi$ within any of those suitable reasoning systems);
5. *Incomplete reasoning* wrt the logical consequence i.e. when some specific conclusions cannot be derived by using suitable reasoning systems for targeted ontology applications even if these conclusions are true in intended models and must be derived by some reasoning according to the targeted ontology applications (formally, for some specific formula φ, true in the intended models $O \models \varphi$, cannot be derived $O \nvdash \varphi$ within those suitable reasoning systems);

The most common **logical ground unsuitable situations** are listed below. These situations impact negatively on the "non functional qualities" of ontologies such as reusability, maintainability, efficiency as defined in the ISO 9126 standard for software quality.

6. *Logical equivalence of distinct artifacts* (concepts / relationships / instances) i.e. whenever two distinct artifacts are proved to be logically equivalent; for example, A and B are two concepts in O and $O \models A = B$;
7. *Symmetrically, logically indistinguishable artifacts* i.e. whenever it is not possible to prove that two distinct artifacts are not equivalent from a logical point of view; in other words, if not possible to prove anyone of the following statements: ($O \models A = B$), ($O \models A \cap B \subseteq \bot$) and ($O \models c \subseteq A and O \models c \subseteq B$); this case (7) can be partially covered in the case (3) above whenever intended models provide precise information on the equivalence or the difference between A and B;
8. *OR artifacts* i.e. an artifact A equivalent to a disjunction like $C \cup S$, $A \neq C, S$ but for which, if applicable, it does not exist at least a common (non optional) role / property for C and S or because C and S have common instances; in the first case, a simple formalization can be expressed by saying that it does not exist a (non optional) role R such that $O \models (C \cup S) \subseteq \exists R.\top$; in the second case, an even simpler formalization is $O \models c \subseteq C$ and $O \models c \subseteq S$, being c one constant not part of O; the first case targets **potentially heterogeneous artifacts** such as $Car \cup Person$, with probably no counterpart in the intended models, thus possibly leading to unadapted ontologies according to case (2) above; the second case targets **potential ambiguities** as, for instance, one role (property) R logically equivalent to a disjunction ($R_1 \cup R_2$) being ($R_1 \cap R_2$) satisfiable;
9. *AND artifacts* i.e. one artifact A equivalent to a conjunction like $C \cap S$, $A \neq C, S$ but for which, if applicable, it does not exist at least a common (non optional) role / property for C and S; this case is relevant to limit as much as possible some **potentially heterogeneous artifacts** such as $Car \cap Person$, possibly leading to artifact unsatisfiability;
10. While some case of *unsatisfiability* of ontology artifacts (concepts, roles, properties etc.) can be covered by (2) because intended models may not contain void concepts, unsatisfiability tout-court is not necessarily an error but a situation which is not suitable for ontology artifacts (i.e. given an ontology artifact A, $O \models A \subseteq \bot$); even if in ontologies it might be possible to define what must not be true (instead of what must be true), this practice is not encouraged;
11. *High complexity of the reasoning task* i.e. whenever something is expressed in a way that complicates the reasoning, while there exist more simple ways to express the same thing;
12. *Ontology not minimal* i.e. whenever the ontology contains unnecessary information:
    - Unnecessary because it can be derived or built[7]. An example of such unsuitable situation is the redundancy of taxonomic relations such as whenever $A \subseteq B$, $B \subseteq C$, and $A \subseteq C$ are all ontology axioms, the last axiom can be derived from the first two ones;

[5] We use the term "unadapted" instead of "incorrect" ontologies because it remains unclear if intended models are defined for building the ontology or may also be defined independently. However, if intended models are defined for building the ontology, the term "incorrect" may be more appropriate.

[6] Intended models should have been defined fully and independently as in the case of models representing abstract structures or concepts such as numbers, processes, events, time and other "upper concepts", often defined according to their own properties. If intended models are not available, some specific entailments can be defined as facts that should necessarily be true in the targeted domain (or for targeted applications); specific counterexamples can also be defined instead of building entire intended models.

[7] Built means that the artifact can be defined by using other artifacts.

- Unnecessary because it is not part of the intended models. For instance, a concept A being part of the ontology (language) but not defined by intended models.

3.1.2 Social ground problems

Social ground problems are related to the **perception (interpretation)** and the **targeted usage** of ontologies by social actors (humans, applications based on social artifacts like WordNet, etc.). Perception (interpretation) and usage may not be formalized at all. In some sense, a further distinction between social facet and logical facet is as the distinction between respectively tacit and explicit knowledge.

There are four **social ground errors**:

1. *Social contradiction* i.e. the perception (interpretation) that the social actor gives to the ontology or to the ontology artifacts is in contradiction with the ontology axioms and their consequences; a natural analogy is with *unadapted ontologies*;
2. *Perception of design errors* i.e. the social actor perception accounts for some design errors such as modeling instances as concepts; a natural analogy is with *unadapted ontologies*;
3. *Socially meaningless* i.e. the social actor is unable to give any interpretation to the ontology or to ontology artifacts as in the case of artificial labels such as "XYHG45"; a natural analogy is with *unadapted ontologies*;
4. *Social incompleteness* i.e. the social actor perception is that one or several artifacts (axioms and/or their consequences) are missing in the ontology; a natural analogy is with *incomplete ontologies*;

The **social ground unsuitable situations** are mostly related to the difficulties that a social actor has to overcome for using the ontology especially due to limited understandability, learnability and compliance (as defined in ISO 9126). As for the logical ground unsuitable situations, it is difficult to dress an exhaustive list; the most common and important are listed below.

5. *Lack of or poor textual explanations* i.e. when there are few, no or poor annotations; prevents understanding by social actors; there are no natural analogies;
6. *Potentially equivalent artifacts* i.e. the social actors may identify as equivalent (similar) distinct artifacts as in the case of artifacts with synonymous or exactly the same labels assigned to distinct artifacts; a natural analogy is with *logically equivalent artifacts*;
7. *Socially indistinguishable artifacts* i.e. the social actors would not be able to distinguish two distinct artifacts as, for instance, in the case of artifacts with polysemic labels assigned to distinct artifacts; a natural analogy is with *logically indistinguishable artifacts*;
8. *Artifacts with polysemic labels* may be interpreted as union or intersection of their several rather distinct meanings associated to labels; a natural analogy is therefore with *OR* and *AND artifacts*.
9. *Flatness of the ontology* (or non modularity), i.e. ontology presented as a set of artifacts without any additional structure, especially if coupled with a important number of artifacts; a natural analogy is with *high complexity of the reasoning task* but also preventing effective learning and understanding by social actors;
10. *Non-standard formalization of the ontology*, using a very specific logics or theory, requires a specific effort by social actors for understanding and learning the ontology but also to use the ontology in standard contexts (reduced compliance); there are no natural analogies;
11. *Lack of adapted and certified versions of the ontology in various languages* requires specific efforts by social actors for understanding and learning the ontology but also to use the ontology in specific standard contexts (limited compliance); there are no natural analogies;
12. *Socially useless artifacts* included in the ontology; a natural analogy is with *ontology not minimal*.

3.2 Positioning state of the art relevant problem classes in to the proposed framework

The precise definitions of the proposed framework allow us to classify most of the ontology quality problems described in literature. Table 1 presents our classification of the different problems mentioned in Section 2. Some of the problems described in literature may correspond to more than one class of problems from our framework, as the definitions of these problems are often very large and sometimes ambiguous.

Table 1 reveals, at a first view, that the proposed framework provides additional problems that are not directly pointed out, to our knowledge, in the current literature about ontology quality and evaluation (but may be mentioned elsewhere). These problems are *No adapted and certified ontology version*, *Indistinguishable artifacts*, *Socially meaningless*, *High complexity of the reasoning task* and *Incorrect reasoning*. However, while covered, other problems are, in our opinion, too much narrowly defined in existing literature about ontology quality and evaluation. For instance, *No standard formalization* is specific to very simple situations while we refer to complete non standard theories.

A deeper analysis of Table 1 reveals that the "logical anti-patterns" presented in [7, 25] belong to the logical ground category and are focusing on *unadapted ontologies* error and *unsatisfability* unsuitable situation. The "non-logical anti patterns" presented in [7, 25] partially cover the logical ground unsuitable situations. The "guidelines" presented in [7, 25] span only over unsuitable situations from both logical and social ground category.

What is qualified as "inconsistency" in [14] span over errors and unsuitable situations and also (as in the case of "semantic inconsistency") over the two dimensions (logical and social), making, in our opinion, the terminology a little bit confusing. According to our framework, we perceive "circularity in taxonomies", as defined in [14], as an unsuitable situation (*logical equivalence of distinct artifacts*) because, from a logical point of veiw, this only means that artifacts are equivalent (not requiring a fixpoint semantics). However, "circularity in taxonomies" can be seen also within a *social contradiction* if actors assign distinct meanings to the various involved artifacts. The problems presented as "incompleteness errors" in [13] belong to the *incomplete ontologies* class of logical errors. The "redundancy errors" fits, in our classification, within the *ontology not minimal* class of logical unsuitable situations.

None of the "design anomalies" presented in [2] is perceived as a logical error. Two of them correspond to a logical unsuitable situation (*logically undistinguishable artifacts*), one to a social error (*perception of design errors*) and the last one to a social unsuitable situation (*no standard formalization*).

Concerning "pitfalls" [24], the most remarkable fact concerns what we call *incomplete reasoning*. Indeed, introducing *ad-hoc* relations such as is_a, $instance\_of$, etc., replacing the "standard" relations such as $subsumption$, $member\_of$, etc., should not be considered as a case of *incomplete ontologies* but as a case of *incomplete reasoning*. This is because accepting a specific ontological commit-

Table 1. Positioning state of the art relevant problem classes in to the proposed framework.

Framework				State of the art problems
Logical ground	**Errors**	1	Logical inconsistency	➢ inconsistency error: ”partition errors - common instances in disjoint decomposition”
		2	Unadapted ontologies	➢ inconsistency errors: ”partition errors - common classes in disjoint decomposition”, ”semantic inconsistency” ➢ logical anti-patterns: ”OnlynessIsLoneliness”, ”UniversalExistence”, ”AndIsOR”, ”EquivalenceIsDifference” ➢ pitfalls: P5 (wrong inverse relationship, *WI*), P14 (misusing ”allValuesFrom”, *MD*), P15 (misusing ”not some”/”some not”, *WI*), P18 (specifying too much the domain / range, *WI*), P19 (swapping ∩ and ∪, *WI*)
		3	Incomplete ontologies	➢ incompleteness errors: ”incomplete concept classification”, ”disjoint / exhaustive knowledge omission” ➢ pitfalls: P3 (”is a” instead of ”subclass-of”, *MD*), P9 (missing basic information, *RC & RWM*), P10 (missing disjointness, *RWM*), P11 (missing domain / range in prop., *NI & OU*), P12 (missing equiv. prop., *NI & OU*), P13 (missing inv. rel., *NI & OU*), P16 (misusing primitive and defined classes, *NI*)
		4	Incorrect reasoning	
		5	Incomplete reasoning	➢ pitfalls: P3 (using ”is a” instead of ”subclass-of”, *MD*), P24 - using recursive def., *MD*)
	Unsuitable situations	6	Logical equivalence of distinct artifacts	➢ inconsistency error: ”circularity” ➢ pitfall: P6 (cycles in the hierarchy, *WI*) ➢ non logical anti-pattern: ”SynonymeOfEquivalence”
		7	Logically indistinguishable artifacts	➢ pitfall: P4 (unconnected ontology elements, *RC*) ➢ design anomalies: ”lazy concepts” and ”chains of inheritance”
		8	OR artifacts	➢ pitfall: P7 (merging concepts to form a class, *MD & OU*)
		9	AND artifacts	➢ pitfall: P7 (merging concepts to form a class, *MD & OU*)
		10	Unsatisfiability	inconsistency error: ”partition errors - common classes in disjoint decomposition” ➢ logical anti-patterns: ”OnlynessIsLoneliness”, ”UniversalExistence”, ”AndIsOR”, ”EquivalenceIsDifference”
		11	High complexity of the reasoning task	
		12	Ontology not minimal	➢ redundancy error: ”redundancy of taxonomic relations” ➢ pitfalls: P3 (using ”is a” instead of ”subclass-of”, *MD*), P7 (merging concepts to form a class, *MD & OU*), P21 (miscellaneous class, *MD*) ➢ non logical anti-pattern: ”SomeMeansAtLeastOne” ➢ guidelines: ”Domain&CardinalityConstraints”, ”MinIsZero”
Social ground	**Errors**	1	Social contradiction	➢ inconsistency error: ”semantic inconsistency” ➢ logical anti-pattern: ”AndIsOR” ➢ pitfalls: P1 (polysemic elements, *MD*), P5 (wrong inv. rel., *WI*), P14 (misusing ”allValuesFrom”, *MD*), P15 (misusing ”not some”/”some not”, *WI*), P19 (swapping ∩ and ∪, *WI*)
		2	Perception of design errors	➢ pitfalls: P17 (specializing too much the hierarchy, *MD*), P18 (specifying too much the domain / range, *WI*), P23 (using incorrectly ontology elements, *MD*) ➢ non logical anti-pattern: ”SumOfSome” ➢ design anomaly: ”lonely disjoints”
		3	Socially meaningless	
		4	Social incompleteness	➢ pitfalls: P12 (missing equiv. prop., *NI & OU*), P13 (missing inv. rel., *NI & OU*), P16 (misusing primitive and defined classes, *NI*)
	Unsuitable situations	5	Lack/poor textual explanations	➢ pitfalls: P8 (missing annotation, *OC & OU*)
		6	Potentially equiv. artifacts	➢ pitfalls: P2 (synonym as classes, *MD & OU*)
		7	Indistinguishable artifacts	
		8	Polysemic labels	➢ pitfalls: P1 (polysemic elements, *MD & OU*)
		9	Flatness of the ontology	
		10	No standard formalization	➢ pitfalls: P20 (swapping label and comment, *OU*), P22 (using different naming criteria in the ontology, *OC*) ➢ guidelines: ”GroupAxioms”, ”DisjointnessOfComplement” and ”Domain&CardinalityConstraints” ➢ design anomaly: ”property clumps”
		11	No adapted and certified ontology version	
		12	Useless artifacts	➢ pitfall: P21 (using a miscellaneous class, *MD & OU*)

ment for building intended models, *ad-hoc* relations can be defined in the same way as standard relations. However, using standard reasoning it is expected (and even proved once fixing the logics) that reasoning algorithms are incomplete. However, adding artifacts may also solve some incompleteness and may also be useful for speeding up reasoning.

Only one of the seven classes of "pitfalls" [24] perfectly fits in one class of our typology: the "real world modeling" pitfalls belong to the *incomplete ontologies* logical errors. All the "ontology clarity" pitfalls are social unsuitable situations. All the "requirement completeness" pitfalls are logical problems. The "no inference" pitfalls are logical or social *incomplete ontologies* errors. Most (6/9 and 4/5) of the "modeling decisions" and "wrong inference" pitfalls are considered as errors. The class of "ontology understanding" pitfalls spans over 10 classes of problems, covering logical and social errors and unsuitable situations.

Most (16/20) of the pitfalls concerning the "structural dimension" of the ontology [11] are perceived as errors. All (2/2) the pitfalls concerning the "functional dimension" of the ontology are logical problems.

4 Problems that affect the quality of automatically built ontologies

Although the proposed framework is general, we are especially concerned by ontologies automatically built from textual resources. We therefore aim at pointing the problems that are expected in automatically constructed ontologies (i.e. there is evidence of their presence or they will appear in future enrichments[8] of the ontology). We are also interested by the opposite case, i.e. if there are unexpected problems in automatically constructed ontologies: it should be noted that unexpected problems are problems that even if the ontology may suffer of them, there is no evidence of their presence/absence for the ontology as it is (however, these problems may appear in future enrichments of the ontology). Our analysis is performed in two steps. In the first step (Section 4.1), we point out expected/unexpected problems due to inherent limitations of the tools for automatic ontology construction. In the second step (Section 4.2), we assess the results obtained in the first step by discussing our experience with the tool Text2Onto.

4.1 Expected and unexpected problems in an automatically built ontology

In a previous work [12] we have deeply studied four approaches (and associated tools) for the automatic construction of ontologies form texts and we compared them with a classical methodology for manual ontology construction (Methontology). This analysis highlighted that none of the automated approaches (and associated tools) covers all the tasks and subtasks associated to each step of the classical manual method. The ignored tasks/subtasks are:

1. The explicit formation of artifacts (concepts, instances and relationships) from terms[9]; usually, the automatic tools consider that each term represents a distinct artifact: they do not group synonymous terms and do not choose a single sense for polysemic terms
2. The identification of axioms (e.g. the disjunction axioms)
3. The identification of attributes for concepts
4. The identification of natural language definitions for concepts

[8] Enrichment should be understood as adding artifacts to the existing ones.
[9] A term corresponds to one or several words found in one text.

Table 2. What problems are expected in automatically built ontologies.

Types of problems	Expected (Yes/No) and Why
1. Logical inconsistency	N (no axiom is defined ⇒ contradictions are unexpected; but they remain possible in the case of future enrichments)
2. Unadapted ontologies	Y (taxonomic relationships extraction algorithms are syntax based ≠ from the intended models)
3. Incomplete ontologies	Y (automatically extracted knowledge is limited to concepts and taxonomies ≠ from the intended models)
4. Incorrect reasoning	N (they might appear for complete formalization of concepts and relationships)
5. Incomplete reasoning	
6. Logical equivalence of distinct artifacts	Y (automatic tools consider that each term defines a different artifact ⇒ the ontology may contain logically equivalent & logically indistinguishable artifacts)
7. Logically indistinguishable artifacts	
8. OR artifacts	Y (polysemy of terms directly affects concepts / relationships: OR / AND concepts / relationships may appear)
9. AND artifacts	
10. Unsatisfiability	Y (polysemy of terms directly affects concepts / relationships: these latter may become unsatisfiable if their polysemic senses are combined)
11. High complexity of the reasoning task	N (few or no axioms are defined ⇒ reasoning remains very basic; but, it can be more complex if the ontology is further enriched)
12. Ontology not minimal	Y (automatic tools introduce redundancies in taxonomies)

1. Social contradiction	Y (ontologies are built from limited textual resources which may introduce contradiction in taxonomies)
2. Perception of design errors	Y (the built ontology may contain concepts that are considered more close to instances by the social actor.)
3. Social meaningless	Y (several meaningless concepts with obscure labels are often introduced)
4. Social incompleteness	Y (probably due to limited textual corpus)
5. Lack of or poor textual explanations	Y (usually automatic tools do not provide textual explanations)
6. Potentially equivalent artifacts	Y (automatic tools consider that each term defines a different artifact ⇒ distinct concepts can have synonymous labels ⇒ these latter are perceived as potentially equivalent)
7. Indistinguishable artifacts	Y (the ontology is incomplete ⇒ it contains concepts that can be distinguished only by their labels; if such concepts have synonymous labels, they are indistinguishable)
8. Artifacts with polysemic labels	Y (automatic tools consider that each term defines a different artifact ⇒ it is possible to have concepts with polysemic labels)
9. Flatness of the ontology	Y (the ontology is poorly structured and has no design constraints - e.g. no disjunction axiom, lazy concepts)
10. No standard formalization	N (automatic tools usually can export their results in different formalization)
11. No adapted and certified ontology version	Y (automatically obtained results closely depend on the input texts language (often English) and certifying them is difficult)
12. Useless artifacts	Y (automatic tools often generate useless artifacts from additional external resources)

Table 2 provides a complete view of expected and unexpected problems according to our experience and suggest why each problem is expected or not.

4.2 Experience with Text2Onto

4.2.1 The experimental setup

During the last two years we were implied in a project called ISTA3 that proposed an ontology based solution for problems related to the integration of heterogeneous sources of information. The application domain was the management of the production of composite components for the aerospace industry. In this context, we tried to simplify the process of deploying the interoperability solution in new domains by using automatic solution for constructing the required ontologies.

The analysis presented in [12] conducted us to choose Text2Onto [6] for the automatic construction of our ontologies. Text2Onto takes as input textual resources from which it extracts different ontological artifacts (concepts, instances, taxonomic relationships, etc.) that are structured together to construct an ontology. Text2Onto performances for extracting concepts and taxonomical relationships are better than its performances for extracting other types of ontological artifacts; consequently, in our tests we used Text2Onto for constructing ontologies containing concepts and taxonomical relationships only.

The textual resource used in the experiment presented in this paper is a technical glossary composed of 376 definitions of the most important terms of the domain of composite materials and how are they used for manufacturing pieces. The glossary contains 9500 words. For constructing the ontology we resort to the standard configuration for the different parameters of Text2Onto: all the proposed algorithms for concepts (and respectively for taxonomic relations) extractions have been used and their results have been combined with the default strategy.

The constructed ontology is an automatically built domain ontology that contains 965 concepts and 408 taxonomic relationships. Some of the central concepts of this ontology are: ”technique”, ”step”, ”compound”, ”fiber”, ”resin”, ”polymerization”, ”laminate”, ”substance”, ”form”.

4.2.2 Identified problems

Table 3 summarizes which types of problems have been identified in the automatically constructed ontology in our experience with Text2Onto. It also indicates, when possible, how many problems have been identified. Most of problems are relatively easy to identify and to quantify (e.g. the number of cycles in the taxonomical structure), but there are exceptions (e.g. the number of concepts or taxonomic relationships that are missing from the ontology).

4.2.3 Discussion

No intended model or use case scenario was available when the expert analyzed the automatically constructed ontology. Consequently, it was able only to make a supposition concerning the logical completeness of the ontology and no logical error (*unadapted ontology*, *incomplete* or *incorrect reasoning*) was identified.

Few logical unsuitable situations are identified, but it is remarkable that they were identified automatically.

Unsurprisingly, most of the identified problems are social problems.

Table 3. Types of problems identified in the automatically constructed ontology.

Types of problems	**Identyfied (Yes/No) and How**
1. Logical inconsistency	No
2. Unadapted ontologies	No
3. Incomplete ontologies	Yes: Some relationships are missing to connect the 389 lazy concepts; some of them are explicitly indicated in the textual corpus
4. Incorrect reasoning	No
5. Incomplete reasoning	No
6. Logical equivalence of distinct artifacts	Yes: 3 cycles in the hierarchy; (automatically detected by reasoners)
7. Logically indistinguishable artifacts	Yes: * 389 lazy concepts (automatically identified by an ad-hoc algorithm) *73 groups of ”leaf” concepts; each group is composed of concepts that are indistinguishable; (automatically identified by an ad-hoc algorithm)
8. OR artifacts	No
9. AND artifacts	No
10. Unsatisfiability	No
11. High complexity of the reasoning task	No
12. Ontology not minimal	Yes: one taxonomical relationship can be deduced from two taxonomical relationships already present in the ontology (automatically identified by an ad-hoc algorithm)

1. Social contradiction	Yes: 15 taxonomic relationships are jugged semantically inconsistent by the expert
2. Perception of design errors	Yes: 5 concepts that are interpreted as instances by the expert (units of measure and proper names)
3. Social meaningless	Yes: 21 concepts that have meaningless labels, for the expert
4. Social incompleteness	Yes
5. Lack of or poor textual explanations	Yes: no annotation associated to the ontology or to its artifacts
6. Potentially equivalent artifacts	Yes: 6 pairs of concepts have synonym labels, for the expert
7. Indistinguishable artifacts	No
8. Artifacts with polysemic labels	Yes: 69 concepts with polysemic labels, for the expert
9. Flatness of the ontology	Yes: 389 lazy concepts lead to a poorly structured ontology
10. No standard formalization	No
11. No adapted and certified ontology version	No
12. Useless artifacts	Yes: 28 concepts are not necessary (3 are too generic, 25 are out of the domain)

The analysis in Section 4.1 suggest that most of the problems that are expected in the automatically constructed ontologies are due to the fact that the automatic tool do not take into account the synonymy and the polysemy of terms when constructing concepts. However, even if Text2Onto, as configured for our test, do not group synonym terms when forming concepts, and allows polysemic terms to be labels for concepts, our test-case reveals that only two types of problems (*socially indistinguishable artifacts* and artifacts with *polysemic labels*) may be imputed to this limitation.

Most of the identified problems are related to the fact that the automatically constructed ontology seems to be incomplete.

5 Conclusion

In this paper, we have introduced a framework providing standardized definitions for different errors that have some impact on the quality of the ontologies. This framework aims at both unifying various error descriptions presented in the recent literature and completing them. It also leads to a new error classification that removes ambiguities of the previous ones. During ontology evaluation this framework may be used as a support for verifying in a systematic way if the ontology contains errors or unsuitable situations.

In the second part of the paper we focused on the quality of automatically built ontologies and we present experimental results of our analysis on an ontology automatically built by Text2Onto. The results show that a large part of the identified errors are linked to the ontology incompleteness. Moreover, it confirms that the identification of logical errors other than inconsistency requires intended models (or at least a set of positive and negative examples) and use case scenarii.

Due to the increasing complexity of the software, the identification of the origin of each error in the ontology building process remains an open question. And a further works consists in associating the identified errors with the different tasks of an ontology construction (e.g. the Methontology tasks [10]). This work could help to improve the quality results of the software by a retro-engineering process and/or to design assistant to detect and to solve major errors.

REFERENCES

[1] M. Almeida, 'A proposal to evaluate ontology content', *Journal of Applied Ontology*, **4(3-4)**, 245–265, (2009).

[2] J. Baumeister and D. Seipel, 'Smelly owls design anomalies in ontologies', in *Proc. of 18th Int. Florida Artificial Intelligence Research Society Conf. (FLAIRS)*, pp. 215–220, (2005).

[3] J. Baumeister and D. Seipel, 'Anomalies in ontologies with rules', *Web Semantics: Science, Services and Agents on the World Wide Web*, **8**(1), 55–68, (2010).

[4] A. Burton-Jones, V. Storey, and V. Sugumaran, 'A semiotic metrics suite for assessing the quality of ontologies', *Data Knowl. Eng.*, **55(1)**, 84–102, (2005).

[5] P. Cimiano, A. Madche, S. Staab, and J. Volker, 'Ontology learning', in *Handbook on Ontologies*, eds., R. Studer and S. Staab, Int. Handbook on Inf. Syst., 245–267, Springer, 2 edn., (2009).

[6] P. Cimiano and J. Volker, 'Text2onto - a framework for ontology learning and data-driven change discovery', in *2nd Eur. Semantic Web Conf.*, eds., A. Montoyo, R. Munoz, and E. Metais, volume 3513, pp. 227–238, (2005).

[7] O. Corcho, C. Roussey, and L. M. V. Blazquez, 'Catalogue of anti-patterns for formal ontology debugging', in *Atelier Construction d'ontologies: vers un guide des bonnes pratiques, AFIA 2009*, (2009).

[8] G. Ereteo, M. Buffa, O. Corby, and F. Gandon, 'Semantic social network analysis: A concrete case', in *Handbook of Research on Methods and Techniques for Studying Virtual Communities: Paradigms and Phenomena*, 122–156, IGI Global, (2010).

[9] M. Fahad and M. Qadir, 'A framework for ontology evaluation', in *Proc. of the 16th Int. Conf. on Conceptual Struct. (ICCS2008)*, volume 354, pp. 149–158, (2008).

[10] M. Fernandez, A. Gomez-Prez, and N. Juristo, 'Methontology: From ontological art towards ontological engineering', in *Proc. of the AAAI97 Spring Symposium Series on Ontological Engineering*, pp. 33–40, (1997).

[11] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann, 'Modelling ontology evaluation and validation', in *Proc. of Eur. Sem. Web Conf. (ESWC2006)*, number 4011 in LNCS, (2006).

[12] T. Gherasim, M. Harzallah, G. Berio, and P. Kuntz, 'Analyse comparative de methodologies et d'outils de construction automatique d'ontologies a partir de ressources textuelles', in *Proc. of EGC'2011*, pp. 377–388, (2011).

[13] A. Gomez-Perez, 'Ontology evaluation', in *Handbook on Ontologies*, eds., S. Staab and R. Studer, Int. Handbook on Inf. Syst., pp. 251–274, Springer, 1 edn., (2004).

[14] A. Gomez-Perez, M.F. Lopez, and O.C. Garcia, *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web*, chapter Chap 3.8.2 Taxonomy evaluation, 180–184, Advanced Information and Knowledge Processing, Springer, 2001.

[15] T. R. Gruber, 'A translation approach to portable ontology specifications', *Knowl. Acquisition*, **5(2)**, 199–220, (1993).

[16] N. Guarino, D. Oberle, and S. Staab, 'What is an ontology?', in *Handbook on Ontologies*, 1–17, Springer, 2 edn., (2009).

[17] G. Hirst, 'Ontology and the lexicon', in *Handbook on Ontologies*, eds., R. Studer and S. Staab, Int. Handbook on Inf. Syst., 269–292, Springer, 2 edn., (2009).

[18] J. Krogstie, O.I. Lindland, and G. Sindre, 'Defining quality aspects for conceptual models', in *Proc. of the IFIP8.1 Working Conference on Information Systems Concepts: Towards a Consolidation of Views (ISCO3)*, (1995).

[19] A. Lozano-Tello and A. Gomez-Perez, 'Ontometric: A method to choose the appropriate ontology', *Journal of Database Management*, **15(2)**, 1–18, (2004).

[20] N. Ben Mustapha, H. Baazaoui Zghal, M.A. Aufaure, and H. Ben Ghezala, 'Enhancing semantic search using case-based modular ontology', in *Proc. of the 2010 ACM Symposium on Applied Computing*, pp. 1438–1439, (2010).

[21] L. Obrst, B. Ashpole, W. Ceusters, I. Mani, S. Ray, and B. Smith, 'The evaluation of ontologies: toward improved semantic interoperability', in *SemanticWeb: Revolutionizing Knowledge Discovery in the Life Sciences*, ed., K.-H. Cheung C. J. O. Baker, 139–158, Springer, (2007).

[22] J.D. Osborne, J. Flatow, M. Holko, S.M. Lin, W.A. Kibbe, L. Zhu, M.I. Danila, G. Feng, and R. L. Chisholm, 'Annotating the human genome with disease ontology', *BMC Genomics*, **10**, 63–68, (2009).

[23] M. Poveda, M. C. Suarez-Figueroa, and A. Gomez-Perez, 'Common pitfalls in ontology development', in *Proc. of the Current topics in artificial intelligence (CAEPIA09), and 13th conference on Spanish association for artificial intelligence*, (2009).

[24] M. Poveda, M. C. Suarez-Figueroa, and A. Gomez-Perez, 'A double classification of common pitfalls in ontologies', in *Proc. of Workshop on Ontology Quality (OntoQual 2010), Co-located with EKAW 2010*, (2010).

[25] C. Roussey, O. Corcho, and L. M. V. Blzquez, 'A catalogue of owl ontology antipatterns', in *Proc. of the Fifth Int. Conf. on Know. Capture KCAP*, pp. 205–206, (2009).

[26] N. H. Shah and M. A. Musen, 'Ontologies for formal representation of biological systems', in *Handbook on Ontologies*, eds., R. Studer and S. Staab, Int. Handbook on Inf. Syst., 445–462, Springer, 2 edn., (2009).

[27] E. Simperl and Tempich C., 'Exploring the economical aspects of ontology engineering', in *Handbook on Ontologies*, eds., R. Studer and S. Staab, Int. Handbook on Inf. Syst., 445–462, Springer, 2 edn., (2009).

[28] D. Vrandecic, 'Ontology evaluation', in *Handbook on Ontologies*, eds., R. Studer and S. Staab, Int. Handbook on Inf. Syst., 293–314, Springer, 2 edn., (2009).

KnowWE – A Wiki for Knowledge Base Development

Joachim Baumeister[1], **Jochen Reutelshoefer**[1], **Volker Belli**[1], **Albrecht Striffler**[1], **Reinhard Hatko**[2] and **Markus Friedrich**[1]

Abstract. The development of knowledge systems has been driven by changing approaches, starting with special purpose languages in the 1960s that evolved later to dedicated editors and environments. Nowadays, tools for the collaborative creation and maintenance of knowledge became attractive. Such tools allow for the work on knowledge even for distributed panels of experts and for knowledge at different formalization levels. The paper (tool presentation) introduces the semantic wiki KnowWE, a collaborative platform for the acquisition and use of different types of knowledge, ranging from semantically annotated text to strong problem-solving knowledge. We also report on some current use cases of KnowWE.

1 Introduction

The utility of decision-support systems proved in numerous examples over the past years. The actual progression of knowledge-based systems goes back to the early years of expert systems. Starting with dedicated AI languages, such as LISP [22] and Prolog [15], task-driven tools have been developed to construct intelligent systems more efficiently, e.g., see [6, 7]. Recently, a number of development tools promoted the creation of knowledge on different formalization levels. That way, explicit process knowledge (e.g., rules, decision trees, fault models) can be linked with ontological relations or even text and multimedia content. Semantic wikis [21] are a prominent example for supporting such a *knowledge formalization continuum* [3], e.g., see the systems Semantic MediaWiki[14], PlWiki [16], and MoKi [8].

In this paper, we introduce the semantic wiki KnowWE that emphasizes the development of strong problem-solving knowledge within the knowledge formalization continuum. The system is the latest successor of a 30-years list of ancestors of diagnostic expert shell kits. Starting with the system MED1 [19] and MED2 [17] (initially implemented in INTERLISP, then ported to FRANZLISP) the knowledge engineers needed to use an internal knowledge representation syntax to built the knowledge bases. The successor D3 [18]—an implementation in Allegro Common Lisp—offered a graphical user interface based on forms, tables, and trees to simplify the knowledge acquisition and to enable domain specialists to define the knowledge by themselves. The full reimplementation d3web (started in 2000 and implemented in Java) brought multi-user and multi-session capabilities to the reasoning engines and also offered a web-based user interface for developed knowledge bases for the first time. As well, the knowledge modeling environment KnowME (Knowledge Modeling Environment) was implemented in Java and copied the graphical editors of the shell-kit D3, but also added sophisticated tools for testing and refactoring the developed knowledge bases [5].

However, all aforementioned systems only support the work of one knowledge engineer at the same time, thus hindering a collaborative and distributed development process with many participants. Furthermore, the graphical editors restricted the structuring possibilities of the knowledge bases by the system-defined structure and expressiveness. In consequence, the engineers often needed to fit their knowledge structure into the possibilities of the tool. More importantly, the mix of different formalization levels was not possible, e.g., by relating ontological knowledge with solutions of a decision tree.

As the successor of KnowME the system KnowWE (Knowledge Wiki Environment) [4] offered a web-based wiki front-end for the knowledge acquisition and supported the collaborative engineering of knowledge at different formalization levels. Strong problem-solving knowledge is mixed with corresponding text and multimedia in a natural manner. The knowledge base can be flexibly structured by distributing the particular knowledge modules over a collection of linked wiki articles, each covering a particular aspect of the domain.

In the following sections, we describe notable features and developments of the system KnowWE and we briefly discuss some current applications.

2 Applications and Usage of KnowWE

In this section, we first sketch the typical application domains of KnowWE and then we describe typical practices for knowledge development with the system.

2.1 Application Domains

Historically, the typical use of the system was the development of diagnostic knowledge bases, since this problem category was the core domain of d3web and its predecessors. Nowadays, KnowWE is still used to develop decision-support systems for diagnosis, classification, or recommendation tasks. As KnowWE can be also used for ontology engineering and clinical guideline engineering, however, the application areas are broadened today. For example, we see applications for the definition of clinical guidelines [9], the configuration of HCI devices [13], and the ontological formalization of ancient history [20].

In summary, almost all applications combine formal knowledge with informal content of the wiki, thus improving the development and the use of the knowledge system. In the following section we describe basic practices for developing knowledge bases with KnowWE.

[1] denkbares GmbH, Friedrich-Bergius-Ring 15, D-97076 Würzburg, Germany, email: name.surname@denkbares.com

[2] Department of Intelligent Systems, University of Würzburg, Germany, email: name.surname@uni-wuerzburg.de

2.2 Practices for Knowledge Development

Distribution of Knowledge In form-based tools the knowledge is typically entered in predefined editor fields. That way, the knowledge engineer is bound to the given organization strategy of the particular tool. In a semantic wiki the engineer is free to partition and distribute the knowledge across the wiki articles. Thus, specific articles can be created to define the particular aspects of the knowledge base. In many cases, this freedom is a significant advantage when compared to form-based tools, since the distribution strategy can be adapted to the current project requirements and the characteristics of the knowledge. However, in any way the knowledge engineer has the burden to formulate a distribution strategy for the knowledge in the wiki before starting with the knowledge engineering task.

In the past, a number of useful distribution patterns have been identified. It is important to notice that the patterns can/should be modified according to the project requirements, and that they can be combined with other patterns.

- **Solution-oriented distribution**: For each possible system output (or coherent group of outputs), an article is created in the wiki. The article contains the definitions of the output and formal knowledge to derive this particular output. For larger systems, sub-articles can be defined that are linked from the main article.
- **Problem area-oriented distribution**: For each problem area (coherent and named groups of inputs to the system), an article is created in the wiki. Each article contains the definitions of the problem area (e.g., symptoms concerning the problem area) and links to articles, where derivation knowledge is defined relevant to the particular problem areas.
- **Concept-oriented distribution**: For each concept of the application domain an article is created. Attributes and relations of this concept are also defined on this article. Also links to related concepts are included.

Namespaces and Compilation of Knowledge In the past, tools only allowed the creation of one knowledge base at the same time. Current environments enable the development of a collection of knowledge bases within one workspace. Here, coherent parts of knowledge need to be clustered and labeled by namespaces. For smaller knowledge bases, namespaces are often used to tag the knowledge relevant for this knowledge base.

A dedicated article is used as a sink for the definition of a knowledge base, i.e., to collect the knowledge packages for the specified namespaces. That way, a wiki can be used to create different variants of a knowledge base, i.e., by having an article compiling all knowledge labeled with namespaces *n1*, and by having another article compiling all knowledge labeled with namespaces *n1* and *n2*. The namespaces and corresponding compilation of knowledge is depicted in Figure 1.

As a historical remark, the current mechanism of namespaces and their compilation is different from the original ideas of KnowWE described in [4]: Back then, every article was compiled into a single knowledge base and therefore had to include all relevant concepts and derivations. In a distributed problem-solving process the different wiki articles and knowledge bases, respectively, communicated with each other exchanging input and output concepts. The outputs of the problem-solving process were displayed to the user in an aggregated view. The concept of distributed problem-solving uncovered two critical issues in real-world knowledge base development: First, the reasoning process was not intuitive for domain specialists who

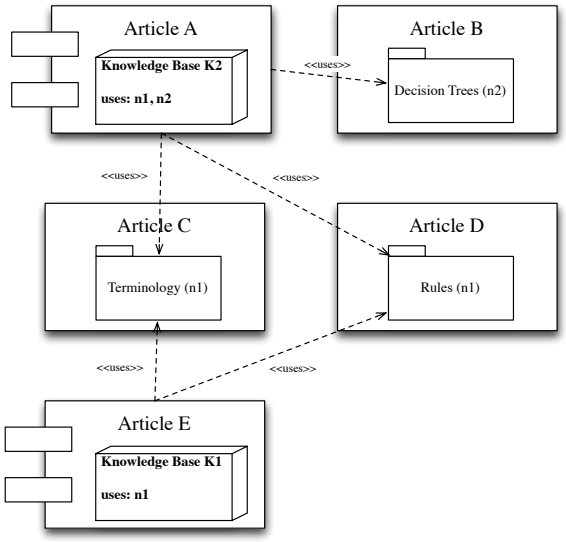


Figure 1. Distribution and namespaces of the knowledge across a set of wiki articles. Knowledge bases are flexibly compiled be defining a number of namespaces.

were usually not familiar with distributed reasoning algorithms. To help the users, very sophisticated explanations for derived solutions needed to be presented in order to allow for effective debugging when problems appeared. Second, the wiki often was used only as the *development environment* of the knowledge base. The target platform of the knowledge system typically differed from the wiki system, so the knowledge base needed to be joined and exported from the wiki into a *single knowledge base* to be applicable for the later use. In consequence, the exported knowledge base needed further quality management, since the reasoning results of the distributed reasoning may differed from the reasoning results of the monolithic knowledge base. Therefore, the test and development of a monolithic knowledge base (the setting of the target platform) within the wiki appeared to be more efficient for developers.

Endpoints for Testing the Knowledge During knowledge base development it is important to have powerful interfaces to test the current state of the knowledge base. In KnowWE, we offer a dialog interface for testing strong problem-solving knowledge, i.e., by presenting a form to enter values for input concepts. Derived solutions are presented in a configurable output panel. The test dialog and output panel can be placed in an arbitrary wiki article in order to give the user the required flexibility to test the knowledge base where it is currently developed.

For ontology engineering we offer a markup to formulate SPARQL [24] queries for RDF ontologies [23]. For OWL ontologies we are able to formulate specific class expression queries in Manchester OWL syntax [12].

Simple Support for Authoring Administration Within a collaborative development process not all involved engineers are working on the knowledge base at the same time. Moreover, the engineers are often not located at the same place. Therefore, the tool needs to offer support for administrative authoring tasks. Typical examples are as follows:

- Label unfinished areas of the knowledge base, i.e., todo tasks.

- Mark identified issues in knowledge definitions, i.e., problems.
- Specify urgent tasks for the development phase.

For all these tasks a specific user or group of users needs to be attachable in order to personalize them.

KnowWE offers a simple todo markup, that can be used to label content or formal knowledge in the wiki article with the action requests as described above. Furthermore, a tagging plugin allows for the annotation of entire pages. Tag clouds with instant access to tagged pages can be inserted into the wiki; most often at the bottom of the left navigation panel.

Use of Standard Wiki Features KnowWE benefits from a set of useful features, that usually comes with a standard wiki distribution:

- A user and group management allows for the fine-grained definition of user rights (view and edit) for single articles.
- All wiki articles are under version control. That way, older versions of an article (and its contained knowledge definitions) can be compared with the current version of the article. When necessary an older version of an article can be restored.
- A *recent changes* view displays a list of recently modified articles and knowledge definitions. With this feature, it is easy to keep track of the current development process.

3 Notable Features

KnowWE is a development environment that supports the knowledge engineer on all aspects of the development process, such as authoring assistance, error handling, refactoring, manual testing, and quality management. In this section we present a selection of the most relevant features of KnowWE.

3.1 Knowledge Acquisition

In KnowWE, knowledge is formalized by using (knowledge) markup languages. A markup language is a formal syntax provided with an internal mapping to the target knowledge representation which is performed instantly after page save by a compilation script. The markup languages can be used at any place in the wiki articles to create elements of the knowledge base allowing for interweaving formal and informal knowledge. Figure 2 shows an article taken from an exemplary car fault diagnosis wiki describing the concept *Clogged air filter*. The article contains informal content such as plain text and images (e.g., in the top half of the article) as well as formalized knowledge (rules at the bottom part of the article). KnowWE provides markup languages for creating knowledge bases in the d3web[3] format and for creating ontologies in OWL. For the d3web reasoner, markups for decision trees, set-covering models, decision tables, and rules are provided as introduced in [2]. Additionally, executable flowcharts can be designed in the DiaFlux language by using a graphical editor available the wiki [10]. For the development of ontologies KnowWE provides markups based on well-known languages such as the Manchester Syntax for OWL [12] and the Turtle Syntax for RDF [4].

3.2 Authoring Support

In addition to the basic wiki editing interface, KnowWE provides different kinds of editing support. The system provides *instant edit* functionality that allows to edit a section, i.e. a coherent part of an article, within the view of the wiki page as shown in Figure 3.

Typically, the editing of tables is difficult when using the standard text markup for tables. Therefore, KnowWE provides instant editing capabilities for tables in a WYSIWYG style allowing each cell to be edited by one click as shown in Figure 5. The table content is stored within the wiki page source in standard wiki markup.

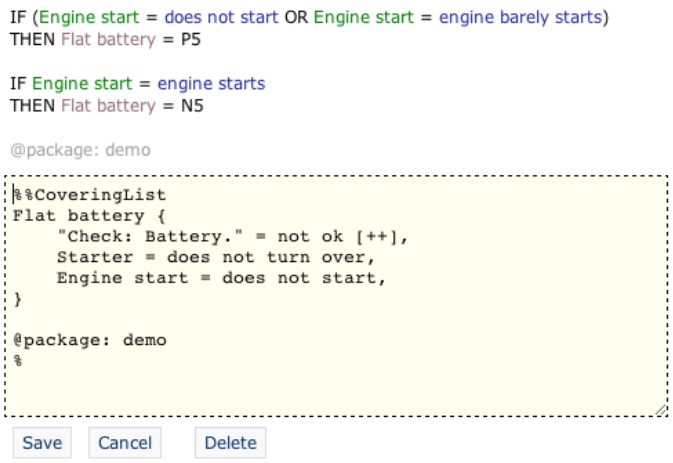


Figure 3. Authoring parts of an article using the instant edit feature.

Additionally, a code completion mechanism supports the user to create markup sections in the text editing panel.

Often, it becomes necessary to obtain an overview of the occurrences and uses of a particular domain concept. Figure 4 shows an overview page for the concept *Leaking air intake system*, that is dynamically generated when requested by clicking on the concept name in the wiki. Besides the pure information about the concept, also small refactoring capabilities are available: At the top, a renaming tool is presented that allows the wiki-wide renaming of the concept, thus ensuring a working and consistent knowledge base. In the bottom part of the info page, the user can see an overview of the wiki articles, where the concept is used (links yield to the particular occurrences in the wiki).

3.3 Testing

As a modern knowledge engineering environment, KnowWE supports an agile knowledge engineering approach. Here, knowledge bases are developed in an evolutionary manner, always maintaining an executable and correct version at a certain level of competency. In this context, (automated) testing is very important to ensure successful evolutionary development cycles. Test cases are either developed manually by defining expected solutions for a given set of inputs or are imported from external testing suites. We adopted the continuous integration practice known from software engineering into the knowledge engineering tool KnowWE. A continuous integration dashboard in the wiki is used to define a collection of quality tests (for validation and verification). As a special knowledge markup, the dashboard can be configured easily to support tailored quality management for the respective project. Registered automated tests are performed on the current version of the wiki knowledge base and

[3] http://d3web.sourceforge.net
[4] http://www.w3.org/TeamSubmission/turtle

KnowWE

Demo - CloggedAirFilter

G'day (anonymous guest) Log in My Prefs

Quick Navigation

Your trail: Demo - Master, Demo - CloggedAirFilter, Demo - Continuous Integration, Main, Demo - Main - Car Diagnosis

View Attach (2) Info Edit More...

Home

Documentation / FAQ

Demos

- Car Fault Diagnosis
- Body-Mass-Index
- Temperature Progression

Administration

- All pages
- Recent changes
- Plugins
- Recent changes
- Left menu

Continuous Integration

Demo DemoBMI DemoTemperature Documentation article attachment basicMarkup battery compile continuousIntegration coveringList expressions formulas imageMap interview knowledgebase package properties question quickl resource rule setcovering solution tables testcase timedb todo variables wikiMarkup xcl

Tags (edit): Demo

KnowWE 20120604_02:29

JSPWiki v2.8.3-svn-19

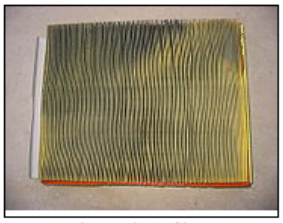
Clogged air filter

General

The (combustion) air filter prevents abrasive particulate matter from entering the engine's cylinders, where it would cause mechanical wear and oil contamination.

Most fuel injected vehicles use a pleated paper filter element in the form of a flat panel. This filter is usually placed inside a plastic box connected to the throttle body with an intake tube.

Older vehicles that use carburetors or throttle body fuel injection typically use a cylindrical air filter, usually a few inches high and between 6 and 16 inches in diameter. This is positioned above the carburetor or throttle body, usually in a metal or plastic container which may incorporate ducting to provide cool and/or warm inlet air, and secured with a metal or plastic lid.



clogged air filter

Typical Symptoms

Typical symptoms for a clogged air filter are for example: Driving, unsteady idle speed and weak acceleration, but also problems when starting the car starting problems and an increased fuel consumption (based on average mileage) and the currently measured mileage or abnormal exhaust fumes.

A typical starting problem which is connected to this problem is a barely or not starting engine in combination with a starter that turns over.

A clogged air filter can cause black exhaust fumes which will turn the color of the exhaust pipe to sooty black.

```
IF Driving =  unsteady idle speed
THEN Clogged air filter = P4

IF NOT (Driving = unsteady idle speed
   OR Driving = weak acceleration)
THEN Clogged air filter = N5

IF (Exhaust fumes = black AND Fuel = unleaded gasoline)
THEN Clogged air filter = P5

IF ((Engine start = does not start
      OR Engine start = engine barely starts)
    AND Starter = turns over)
THEN Clogged air filter = P4

@package: demo
```

Repair Instructions

A clogged air filter needs to be replaced by a new one. Therefore, the air filter housing have to be found. It will be either square (on fuel-injected engines) or round (on older carbureted engines) and about 12 inches (30 cm) in diameter.

After locating the housing the screws or clambs on the top of it have to be

Figure 2. A wiki page from a car-fault diagnosis knowledge base in KnowWE.

Figure 6. The continuous integration dashboard of KnowWE showing messages of the current test runs and the history of the previous development stages.

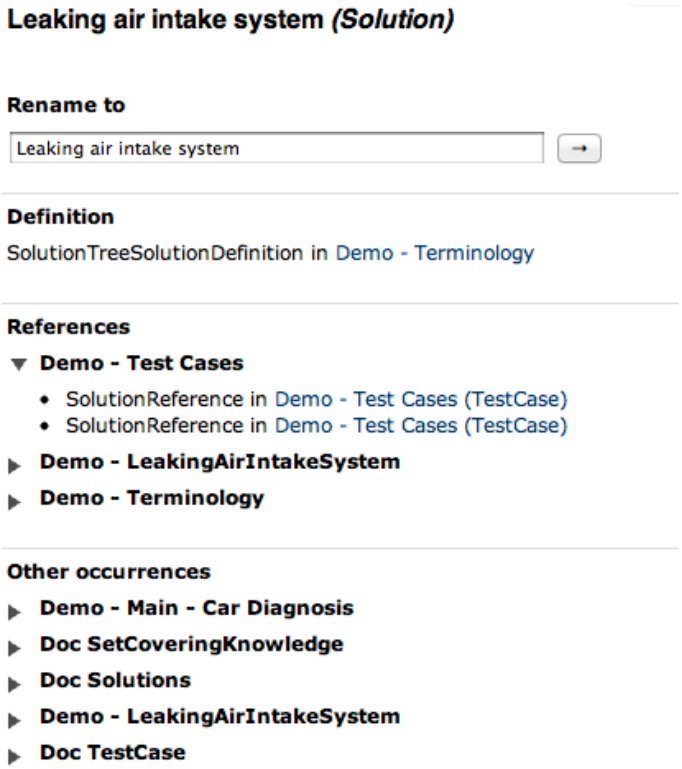


Figure 4. The generated object-info page for every concept allows for the renaming of the concept and it shows the use of the concept across the wiki articles.

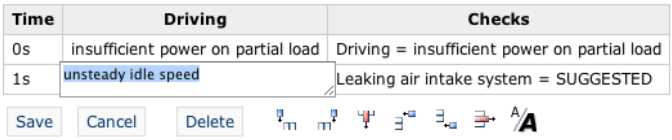


Figure 5. Inline editing of tables by the WYSIWYG interface of the wiki.

give verbose feedback to the knowledge engineers by status messages on the dashboard as shown in Figure 6.

At any time, the dashboard displays the current state of the wiki knowledge base with respect to quality at one glance. Also the history of builds is listed on the left panel of the dashboard. Older builds can be inspected by clicking on the build number, for instance, because the developer wants to check the reason for the build problem. For the selected build the applied tests are shown in the center of the dashboard. In case of errors, the tests give detailed reports on the error s as well as links are provided for further investigation and debugging of the issue. In Figure 6, the top two tests have been passed successfully, while the lower two tests have failed showing more details explaining the actual problem. The tests can be activated by three trigger-modes *onChange*, *onSchedule*, and *onDemand*. In the mode onChange, the tests are executed after each modification of a wiki article which changed the knowledge base. This mode provides the most immediate feedback possible. However, for very time consuming tests this mode can yield inconvenient delays. The mode onSchedule executes the tests on a regular basis according to a specified schedule, for instance at night. This mode is preferable also for tests with considerable high execution time. Further, in the mode onDemand all responsibility for test execution is left to the user, since the user has to explicitly start a continuous integration run. The user has to decide, when the execution is reasonable, which often is an option for tests with high runtime (considering sufficiently experienced users). It is important to note, that the user can define different

dashboards, for instance, one for quick tests running onChange and another one for executing larger/time-consuming tests onSchedule.

Additionally to the dashboard, located on a specific wiki page, KnowWE provides a *CI-Daemon* (daemon for continuous integration) which can be connected to a dashboard. The CI-Daemon is always visible in the KnowWE user interface basically only showing a colored bubble (green, red, or grey) representing the current state of the connected dashboard. In Figure 2 the CI-Daemon is visible as a green bubble on the left of the page below the navigation menu. In this way, the users are always aware of the current quality state not requiring to frequently visit the dashboard article. A very important category of tests for knowledge bases are the competency tests which can be implemented by (sequential) test cases [1]. Figure 7 shows a markup for the definition of sequential test cases in KnowWE. During execution, the test case is performed line-by-line. Equal signs express assignments of input data, added to the current testing session. Expressions containing brackets are expected derivations. The test fails, if the expected derivations do not match the actual ones. That way, input-output behavior of a knowledge base can be covered by automated competency tests which can be attached to a continuous integration dashboard easily.

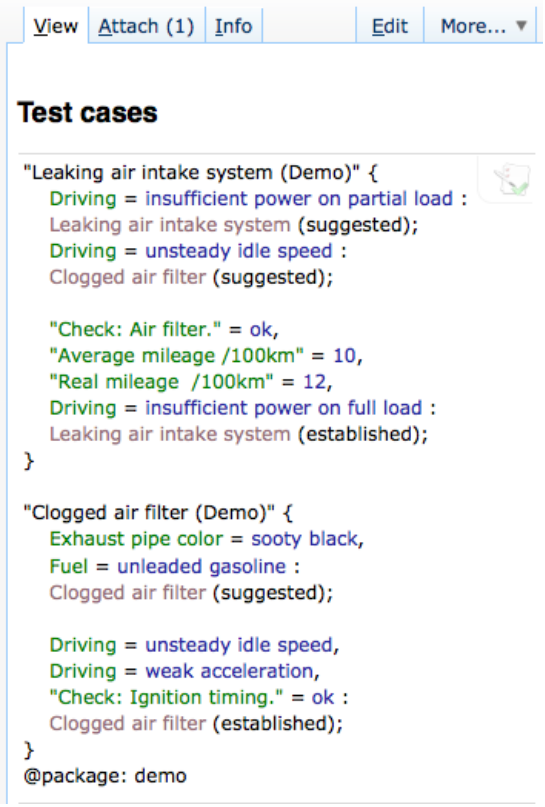


Figure 7. Markup for the definition of sequential test cases.

3.4 Knowledge Use

For instant manual testing of the created knowledge base KnowWE provides an embedded interview component which can be embedded into any wiki article. Figure 8 shows the interview interface which is dynamically generated from the connected knowledge base. It allows the user to answer the input questions and instantly gives feedback of the derived solution concepts. In the shown example, the combination of inputs derived the established solution concept *Bad ignition timing*. The solutions *Clogged air filter*, *Flat battery*, and *Leaking air intake system* are also suggested as potential solutions while *Damaged idle speed system* is marked as an excluded solution.

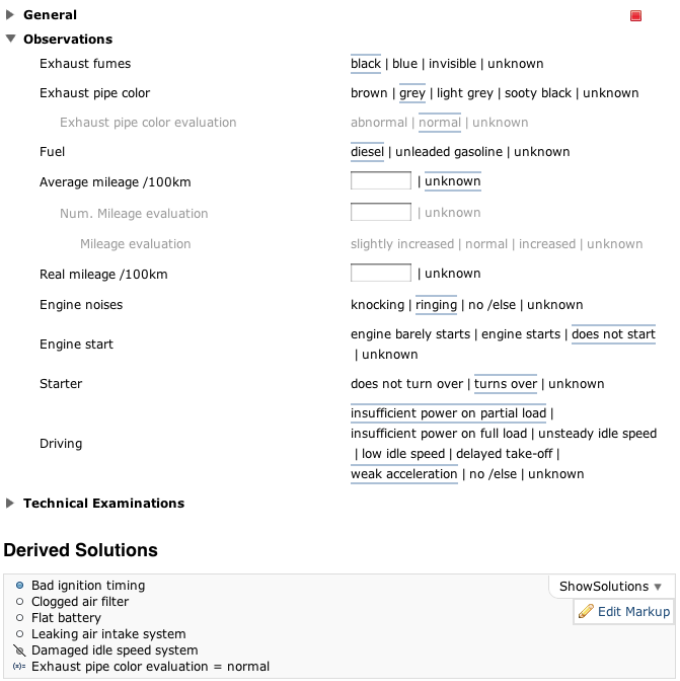


Figure 8. The interview component for manual knowledge base testing.

For developed ontologies KnowWE provides an inline-query mechanism to summarize the knowledge of the ontology as a dynamic content element. Using a markup based on the SPARQL language, queries can be defined within the wiki pages. They are evaluated on page load on the current version of the developed ontology. The result of the query is displayed in the view of the wiki article.

4 Known Uses of KnowWE

KnowWE is currently used in several knowledge engineering projects of different subject domains, both in academic and industrial contexts. In this section, we report on a selection of these projects and we give a brief overview of the use of the system KnowWE.

4.1 Managing Chemical Safety with KnowSEC

KnowSEC (Managing Knowledge of Substances of Ecological Concern) is a group-wide wiki to manage substance-related work(flows) within a group of the German Federal Environmental Agency (Umweltbundesamt). Here, every substance is represented by a distinct wiki article storing important information such as chemical endpoints, relevant literature, or comments of group members. The information is entered in (user-friendly) editors in the wiki and translated into special markups in the background; thus, the information is

also stored in an RDF ontology. That way, the information currently available in the wiki but also the latest knowledge changes can be aggregated and visualized by integrated SPARQL queries.

Besides the storage of weakly formalized knowledge, KnowSEC also offers knowledge-based modules that support the classification of substances for a number of critical chemical characteristics. At the moment, modules are available for supporting the assessment of the relevance, the persistence, the bioaccumulation, and the toxicity of a given substance. These aspects (e.g., relevance, persistence, etc.) are developed in the wiki using different namespaces, so they can be maintained and tested independently from the other aspects. For the users of KnowSEC, a joint knowledge base with all aspects is virtually defined including all above namespaces.

Currently, the knowledge base is still under development. The joint version of the knowledge base consists of 214 questions (user inputs to characterize the investigated substance) grouped by 46 questionnaires, 146 solutions (assessments of the investigated substance), and more than 1.000 rules to derive the assessments. The rules are automatically generated from entered decision tables that allow for an intuitive and maintainable knowledge development process.

Two knowledge engineers are supporting a team of domain specialists, that partly define the knowledge base themselves, partly giving domain knowledge to the knowledge engineers.

4.2 Modeling Clinical Guidelines in KnowWE

Within the project CliWE[5] (Clinical Wiki Environments), KnowWE is extended by plugins to allow for the collaborative development of Computer-Interpretable Guidelines (CIGs). Clinical guidelines are based on evidence-based medicine and improve patient outcome by providing standardized treatments. Their computerization allows for decision-support systems at the point of care, or even the automated application by closed-loop systems in the setting of Intensive Care Units. The goal of CliWE is to create a platform that supports the engineering of CIGs by spatially distributed domain specialists. Therefore, the graphical CIG language DiaFlux was created. Its focus lies on the direct applicability and understandability by domain specialists [9]. By offering only a small set of intuitive language elements, the guidelines can in the best case be built and maintained by the domain specialists themselves. Currently, the extensions developed within CliWE are used in the project WiM-Vent[6]. Its goal is to integrate medical expertise concerning mechanical ventilation and physiological models into an automated mechanical ventilator [11]. In the course of this project, one knowledge engineer guides and supports one domain specialists (backed up by a committee of further experts) during the knowledge engineering process. The latest version of the guideline contains 17 DiaFlux modules, that in total contain 295 nodes and 345 edges. During its development, the testing capabilities of KnowWE are extensively used. So far, about 1.100 continuous integration builds were automatically executed. Especially the empirical testing feature is applied to define and process local test cases, as well as ones that are created using external tools, e.g., a Human Patient Simulator. Those simulated patient sessions can then be replayed in KnowWE for introspecting and debugging the guideline execution. A high-lighting of the taken paths within the DiaFlux models serves as an accessible means of explanation for the domain specialists.

[5] funded by Draegerwerk AG & Co. KGaA, Lübeck, Germny, 2009-2012

[6] "WiM-Vent" - Knowledge- and model-based Ventilation, funded by BMBF (Federal ministry of education and research)

4.3 ESAT: Selecting Assisting Technologies for Handicaped People

ESAT (Expertensystem für Assistierende Technologien [german]) is an expert system designed to determine an appropriate set of human-computer interaction devices for handicapped people. In the application scenario a detailed profile of the physical capabilities (e.g., visual or motorical abilities) for a person is entered into the system. The knowledge base derives a set of input and output devices, that together provide optimal computer interaction for that specific person. In advance, the underlying domain knowledge has been elaborated by a comprehensive study in 2008. The actual implementation of a corresponding executable knowledge base using KnowWE has started in spring 2011. Currently, the ESAT knowledge base has been completed and the system will be launched for a testing phase at the project's initiator (FAB[7]). The knowledge base has been implemented by mainly one knowledge engineer using KnowWE. For knowledge representation production rules are used. In total the ESAT knowledge base currently contains 654 rules distributed on 74 wiki articles. Also in this single-user context the possibility of free structuring allows for reasonable and clear distribution of the knowledge. The terminology is defined on different wiki articles dealing with vision, hearing, motoric and haptic abilities and general skills (e.g., braille) respectively. The about 50 different types of input and output devices (e.g., various kinds of keyboards, sensors, displays) are each described in distinct wiki articles also containing the rules relevant for the derivation of the particular device. Five heuristics have been established within a theoretical study, describing solutions for major categories of handicaps. These are implemented on distinct wiki articles forming the core of the derivation knowledge. The testing framework for continuous integration discussed in Section 3.3 is extensively used to guarantee the save development process by uncovering undesired side-effects of modifications including at least one sequential test case for each device and heuristic. More details about the project are given by Kreutzer [13].

4.4 Continuous Medical Cataract Knowledge with WISSKONT

The WISSKONT project considers the creation of an intelligent information system in the medical domain of cataract surgery. The system is currently under development and it will support the ophtalmologist during the treatment process before, in-between, and after the cataract surgery. That way, the system needs to present relevant knowledge of the domain, which is integrated at varying degrees of formality. For instance, textbook content with images describe particular aspects of a treatment process, whereas temporal relations of the treatment phases are represented by ontological annotations. Here, informal content is correlated by ontological relations. In consequence, a semantic search mechanism provides the presentation of the relevant information at any stage of the treatment process. Additionally, for a number of decision tasks occurring during the treatment, distinct decision-support modules are created, e.g., the selection of an appropriate lens for the surgery based on the patient's parameters. The integration of formalized and informal knowledge allows the ophtalmologist to verify the recommendations of the knowledge base by analyzing the comprehensive support information provided with the recommendation.

The WISSKONT project is part of the WISSASS project, a cooperation of the Karlsruhe Institute of Technology, Germany (KIT) and

[7] http://www.vo-fab.at/

the denkbares GmbH. It is funded as a ZIM-KOOP[8] project by the German Federal Ministry of Economics and Technology (BMWI).

5 Conclusion

In this paper, we presented the current state-of-the-art of the semantic wiki KnowWE. The tool is used in knowledge engineering projects that have a distributed and collaborative nature. Also, KnowWE is capable to jointly represent and use knowledge at different levels of formalization and therefore allows for the flexible organization and elicitation of knowledge. We showed notable features of the tool, such as dedicated markups and editors for knowledge acquisition and use, but also features for (continuously) testing the developed knowledge base. Publicly known projects and applications were reported, that use KnowWE as their primary knowledge engineering environment.

REFERENCES

[1] Joachim Baumeister, 'Advanced empirical testing', *Knowledge-Based Systems*, **24**(1), 83–94, (2011).

[2] Joachim Baumeister, Jochen Reutelshoefer, and Frank Puppe, 'Markups for knowledge wikis', in *SAAKM'07: Proceedings of the Semantic Authoring, Annotation and Knowledge Markup Workshop*, pp. 7–14, Whistler, Canada, (2007).

[3] Joachim Baumeister, Jochen Reutelshoefer, and Frank Puppe, 'Engineering intelligent systems on the knowledge formalization continuum', *International Journal of Applied Mathematics and Computer Science (AMCS)*, **21**(1), (2011).

[4] Joachim Baumeister, Jochen Reutelshoefer, and Frank Puppe, 'KnowWE: A semantic wiki for knowledge engineering', *Applied Intelligence*, **35**(3), 323–344, (2011).

[5] Joachim Baumeister, Dietmar Seipel, and Frank Puppe, 'Agile development of rule systems', in *Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches*, eds., Giurca, Gasevic, and Taveter, IGI Publishing, (2009).

[6] B.G. Buchanan and E.H. Shortliffe, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, 1984.

[7] John H. Gennari, Mark A. Musen, Ray W. Fergerson, William E. Grosso, Monica Crubézy, Henrik Eriksson, Natalya F. Noy, and Samson W. Tu, 'The evolution of protégé: An environment for knowledge-based systems development', *Int. J. Hum.-Comput. Stud.*, **58**(1), 89–123, (January 2003).

[8] Chiara Ghidini, Barbara Kump, Stefanie N. Lindstaedt, Nahid Mahbub, Viktoria Pammer, Marco Rospocher, and Luciano Serafini, 'MoKi: The enterprise modelling wiki', in *ESWC'09: The Semantic Web: Research and Applications*, volume 5554 of *LNCS*, pp. 831–835. Springer, (2009).

[9] Reinhard Hatko, Joachim Baumeister, Volker Belli, and Frank Puppe, 'DiaFlux: A graphical language for computer-interpretable guidelines', in *KR4HC'11: Proceedings of the 3th International Workshop on Knowledge Representation for Health Care*, (2011).

[10] Reinhard Hatko, Jochen Reutelshoefer, Joachim Baumeister, and Frank Puppe, 'Modelling of diagnostic guideline knowledge in semantic wikis', in *Proceedings of the Workshop on Open Knowledge Models (OKM-2010) at the 17th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, (2010).

[11] Reinhard Hatko, Dirk Schädler, Stefan Mersmann, Joachim Baumeister, Norbert Weiler, and Frank Puppe, 'Implementing an automated ventilation guideline using the semantic wiki knowwe', in *EKAW 2012: 18th International Conference on Knowledge Engineering and Knowledge Management*, eds., Heiner Stuckenschmidt, Annette ten Teije, and Johanna Voelker, (2012).

[12] Matthew Horridge, Nick Drummond, John Goodwin, Alan Rector, Robert Stevens, and Hai H Wang, 'The manchester owl syntax', in *Proceedings of OWL: Experiences and Directions (OWLED'06)*, eds., Bernardo Cuenca Grau, Pascal Hitzler, Connor Shankey, and Evan Wallace, Athens, Georgia, USA,, (2006).

[8] http://www.zim-bmwi.de/

[13] S. Kreutzer, *Ein Expertensystem zur Unterstützung körperbehinderter Menschen*, Diplomica, 2012.

[14] Markus Krötzsch, Denny Vrandecić, and Max Völkel, 'Semantic MediaWiki', in *ISWC'06: Proceedings of the 5th International Semantic Web Conference, LNAI 4273*, pp. 935–942, Berlin, (2006). Springer.

[15] Dennis Merritt, *Building Expert Systems in Prolog*, Springer, Berlin, 1989.

[16] Grzegorz J. Nalepa, 'PlWiki - a generic semantic wiki architecture', in *ICCCI'09: Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, number 5796 in LNCS, pp. 345–356. Springer, (2009).

[17] Frank Puppe, 'Requirements for a Classification Expert System Shell and their Realization in MED2', *Applied Artificial Intelligence*, **1**, 163–171, (1987).

[18] Frank Puppe, 'Knowledge Reuse among Diagnostic Problem-Solving Methods in the Shell-Kit D3', *International Journal of Human-Computer Studies*, **49**, 627–649, (1998).

[19] Frank Puppe and Bernhard Puppe, 'Overview on MED1: An heuristic diagnostics system with an efficient control structure', in *Proceedings of the German Workshop on Artificial Intelligence (GWAI-83), Informatik-Fachberichte 76*, pp. 11–20. Springer, (1983).

[20] Jochen Reutelshoefer, Florian Lemmerich, Joachim Baumeister, Jorit Wintjes, and Lorenz Haas, 'Taking OWL to Athens – Semantic Web technology takes ancient greek history to students', in *ESWC'10: Proceedings of the 7th Extended Semantic Web Conference*, pp. 333–347. Springer, (2010).

[21] Sebastian Schaffert, François Bry, Joachim Baumeister, and Malte Kiesel, 'Semantic wikis', *IEEE Software*, **25**(4), 8–11, (2008).

[22] Guy L. Steele and Richard P. Gabriel, 'The evolution of lisp', in *The second ACM SIGPLAN conference on History of programming languages*, pp. 231–270, (1993).

[23] W3C. RDF - resource description framework recommendation: http://www.w3.org/rdf/, February 2004.

[24] W3C. SPARQL recommendation: http://www.w3.org/tr/rdf-sparql-query, January 2008.

Overview of BPMN Model Equivalences. Towards normalization of BPMN diagrams[1]

Krzysztof Kluza and **Krzysztof Kaczor**[2]

Abstract. In various application domains, there is a desire to standardize modeling techniques. Business Process Model and Notation (BPMN) is currently the most widespread language used for modeling Business Processes (BP). Although there are some guidelines how to use this notation, the issue of modeling technique is not standardized. The same semantics can be represented in BPMN using various but behaviorally equivalent model structures. In this paper, we present an overview of the BPMN models equivalences topic. We point out various possibilities of equivalence patterns. This can help to structure diagrams and decrease their semantic complexity. Such research can be further useful for such tasks as analyzing similarities or measuring compliance of processes.

1 Introduction

Business Process (BP) models constitute a graphical representation of processes in an organization. Business Process Model and Notation (BPMN)[3] [1, 23] is a notation for modeling Business Processes, which contributed significantly in Software Engineering when it comes to collaboration between developers, software architects and business analysts. Although there are many new tools and methodologies which support the BPMN notation, they neither support some recommended modeling techniques nor make BPMN models easily comprehensible.

Two models with different structure, but behaviorally equivalent, can be both correct and unambiguous. This stems from the BPMN specification allowing for expressing the same semantics using various syntactic structures. However, this can cause difficulties in modeling or understanding of the model – the *modeling challenge*.

Although behaviorally equivalent structures can be replaceable, some of them may be not translatable to other languages in order to be analyzed or verified [29, 33]. This makes practical problems with model analysis – the *analysis challenge*. Thus, to avoid such problems, a set of best practices for modelers is needed, and it would be useful to *normalize* the preffered model structures.

The first step towards such a structure normalization process is to identify behaviorally (or semantically) equivalent structures. One model can be transformed to the equivalent model to make it consistent in a way which it might not have been before [14]. While this may be done manually, and usually is in the case of ad hoc modeling, it is possible to support a normalization task with tools. The goals of such a normalization can be to maintain compatibility, interoperability, safety, repeatability, or quality of models.

[1] The paper is supported by the *BIMLOQ* Project funded from 2010–2012 resources for science as a research project.

[2] AGH University of Science and Technology, Poland,
Email: {kluza,kk}@agh.edu.pl

[3] See: http://www.bpmn.org/.

Although there are several research papers concerning equivalences of Business Process models, most authors do not consider using of the BPMN notation, but analyze equivalences of models for Petri nets [5, 32] or web services [12, 25]. The thorough research in the area of BPMN models equivalences was carried by Vitus Lam and can be found in his papers [14, 15, 16]. Although Lam's equivalences of models are formalized, he analyzes only several equivalence patterns. Thus, it is advisable to address the issue of BPMN models equivalences in a wider range.

In this paper, we present an overview of the BPMN models equivalences and show various possibilities of equivalent structures. This research can be useful in different areas of BPMN application, such as: process matching [36], identifying the differences between process models [13], analyzing similarities [3, 6, 19] or measuring compliance of processes [2, 8].

The rest of this paper is organized as follows. In Section 2, BPMN models and elements are introduced. Section 3 provides a review of various equivalence patterns in BPMN models. The conclusion with suggested course of action is presented in Section 4.

2 BPMN models and elements

A Business Process [34] can be defined as a collection of related tasks that produce a specific service or product (serve a particular goal) for a particular customer. BPMN constitute the most widespread language for modeling BPs. It uses a set of predefined graphical elements to depict a process and how it is performed. The current BPMN 2.0 specification defines three models to cover various aspects of processes:

1. *Process Model* — describes the ways in which operations are carried out to accomplish the intended objectives of an organization. The process can be modeled on different abstraction levels: *public* (collaborative Business 2 Business Processes) or *private* (internal Business Processes).
2. *Choreography Model* — defines expected behavior between two or more interacting business participants in the process.
3. *Collaboration Model* — can include Processes and/or Choreographies, and provides a Conversation view (which specifies the logical relation of message exchanges).

In most cases, using only the Process Model is sufficient. In our research, the internal Business Process Model is considered. Four basic categories of elements used to model such processes, presented in Fig. 1, are: flow objects (*activities*, *gateways*, and *events*), connecting objects (*sequence flows*, *message flows*, and *associations*), swimlanes, and artifacts [23].

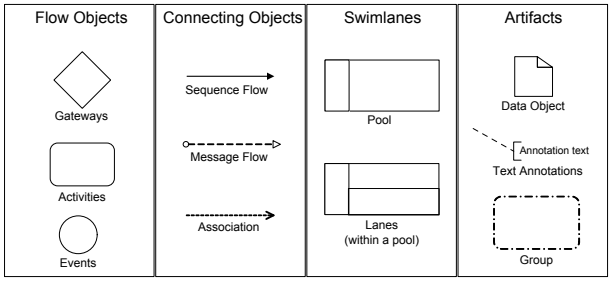


Figure 1. BPMN core objects

Activities constitute the main BPMN elements. They denote tasks that have to be performed and are represented by rectangles with rounded corners. The sequence flow between activities, the flow of control, is depicted by arcs. The directions of arcs depict the order in which the activities have to be performed.

Events, represented by circles, denote something that happens during the lifetime of the process. The icon within the circle denotes the event type, e.g. envelope for *message event*, clock for *time event*.

Gateways, represented by diamond shapes, determine forking and merging of the sequence flow between tasks in a process, depending on some conditions.

3 Equivalences of BPMN Models

In various application domains there is a need to compare process models [32]. One of the possible results of such a comparison can be that two structurally different graphical representations of a business process are behaviorally (and semantically) equivalent. Thus, BPMN processes can be regarded as equivalent if both of them can be transformed into a common graphical representation [14].

There is ongoing research in the area of process models equivalences [5, 32, 35]. However most of the researchers do not consider BPMN notation, but e.g. Petri nets [5, 32]. There are tools which can prove selected equivalences of BPMN processes [14]. However, this topic still remains an open research problem [16].

3.1 Basic equivalent structures

Some basic equivalences that follow directly from the semantics of model elements described in the BPMN specification [23] are presented in Table 1.

Other basic equivalences have been presented by Wohed et al. [35] when defining the five simple control-flow patterns for process control based on the concepts defined by Workflow Management Coalition [4], such as:

1. *sequence* — the ability to depict a sequence of activities,
2. *parallel split* — the ability to capture a split in a single thread of control into multiple threads which can execute in parallel,
3. *synchronization* — the ability to capture a synchronization of multiple parallel subprocesses/activities into a single thread,
4. *exclusive choice* — the ability to represent a decision point in a workflow process where one of several branches is chosen,
5. *simple merge* — the ability to depict a point in the workflow process where two or more alternative branches come together without synchronization.

Apart from the sequence, the other patterns can be modeled in several ways. The models in each column of the Table 2 are equivalent.

One can also observe that in many cases multiple gateway structure can be replaced by a single gateway, as shown in Table 3. Moreover, Gruhn and Laue described patterns in BPMN models that deal with OR-gateways which can be replaced by AND- or XOR-gateways [9], as presented in Table 4 (each row contains an equivalent pair of structures). They claimed that the equivalent model is easier to understand, as it is cognitively less complex. Such transformation is also consistent with a study on the comprehensibility of BPM carried out by Sarshar and Loos [28], which shows that OR-gateways are significantly less comprehended than AND or XOR gateways. Thus, Mendling et al. recommended to avoid OR-gateways [20].

Several researchers noticed that in several situations it is possible to reduce number of repeated activities [14, 17]. The first example in Table 5 shows a situation where the same activity is located at the last position of all incoming sequence flow paths before a join gateway. It is possible to reduce the number of nodes by moving this activity behind the join gateway. The second one is similar but concerns a situation in which the repeated activity is located at the first position after a split gateway.

In [10], Jung et al. proposed a transformation from the BPMN-formed business process to its semantically equivalent XPDL process. Although both BPMN and XPDL are conceived of as a directed graph structure and the mapping should be straightforward, there are some differences between BPMN and XPDL. Thus, in the paper [10] several BPMN transformations are considered.

One of them concerns a loop mechanism. A loop in a process can be depicted as in Fig. 2a. The BPMN 2.0 specification defines the "testBefore" standard loop attribute, which constitutes a flag that controls whether the loop condition is evaluated at the beginning (testBefore = true) or at the end (testBefore = false) of the loop iteration. Instead of using this attribute, a loop can be depicted explicitly as in Fig. 2b (test time: before) and Fig. 2c (test time: after).

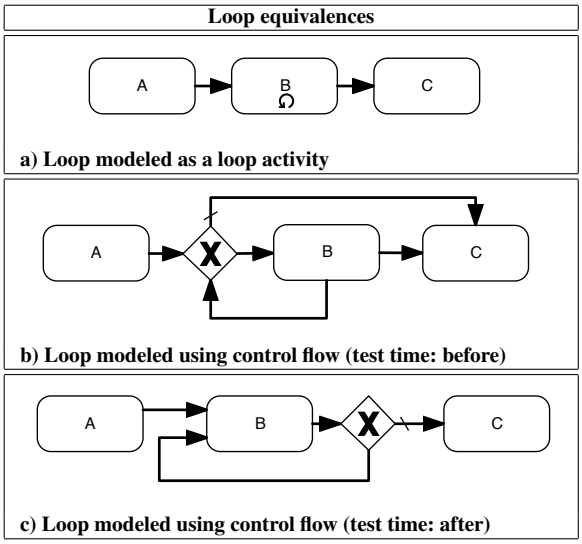


Figure 2. Variants of a loop structure [10]

Another transformation of loops in graphs was proposed by Zhongjun Du and Zhengjun Dang in [7]. Based on the graph reduction technique [27], they proposed an algorithm which transforms the loop in the workflow to an acyclic sub-graph. Although their solution does not use BPMN, it is rather general and should be applicable to BPMN models as well.

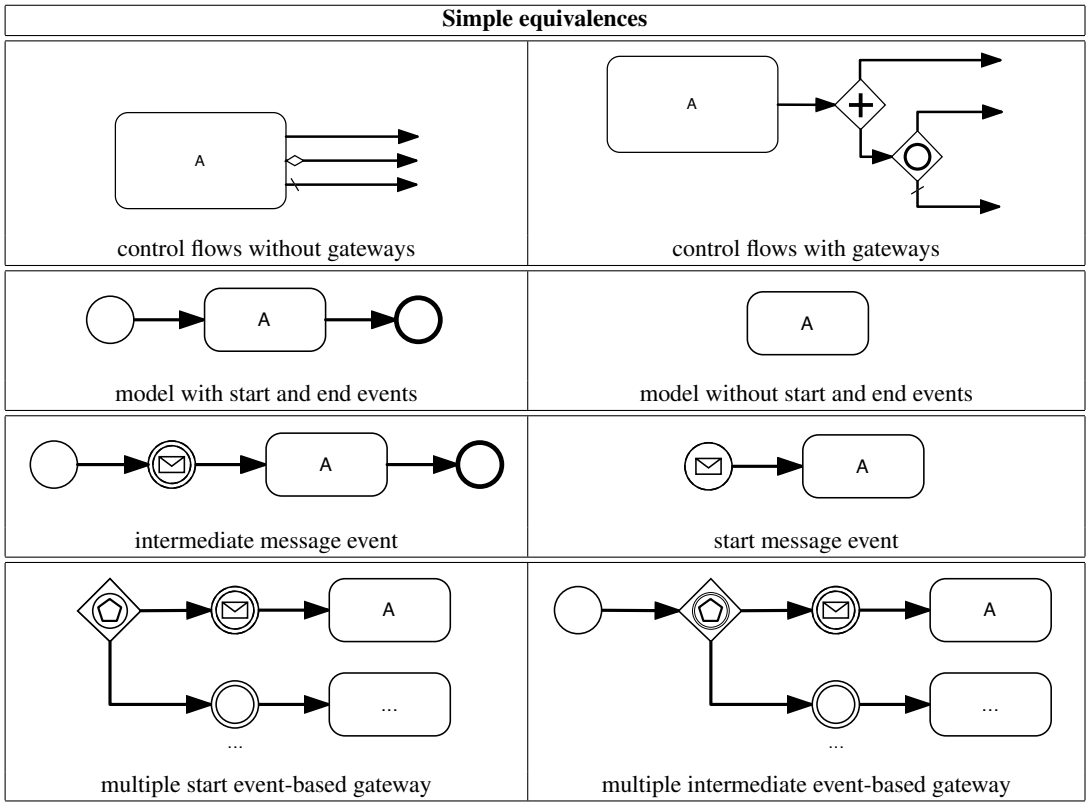


Table 1. Equivalences of BPMN structures based on the semantics of elements (based on the BPMN specification [23])

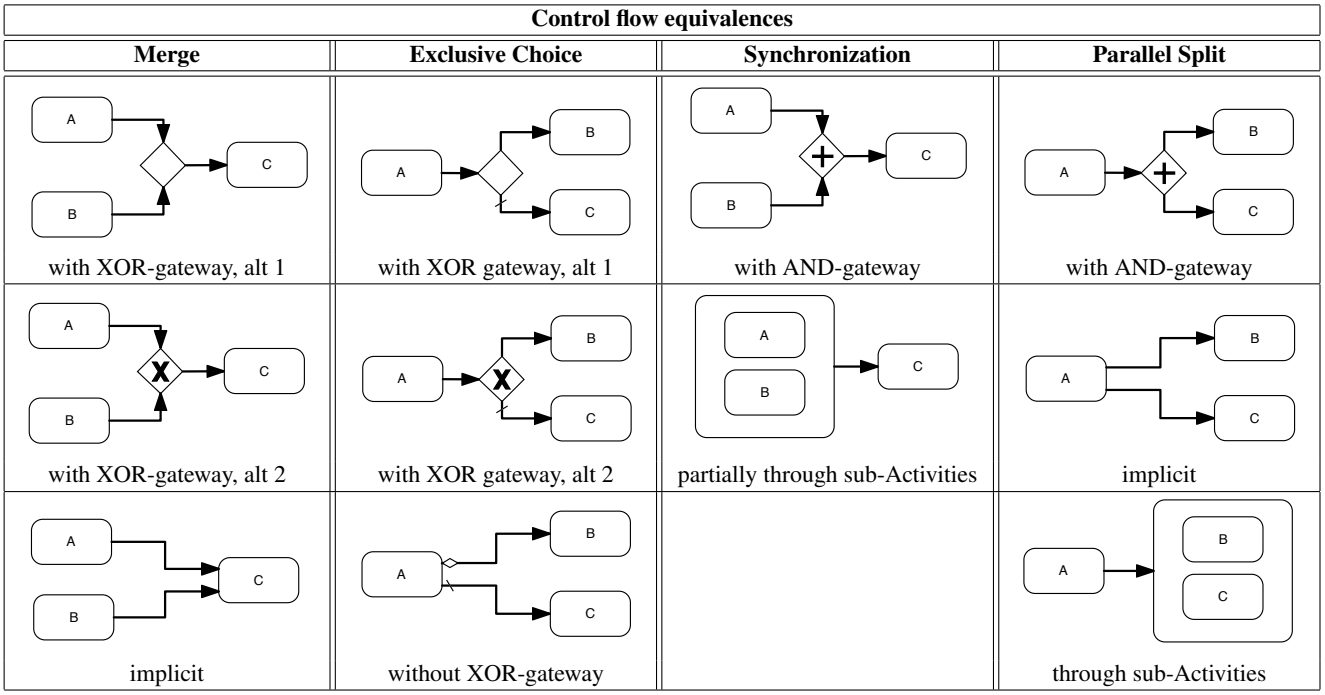


Table 2. Basic control-flow patterns in BPMN [35]

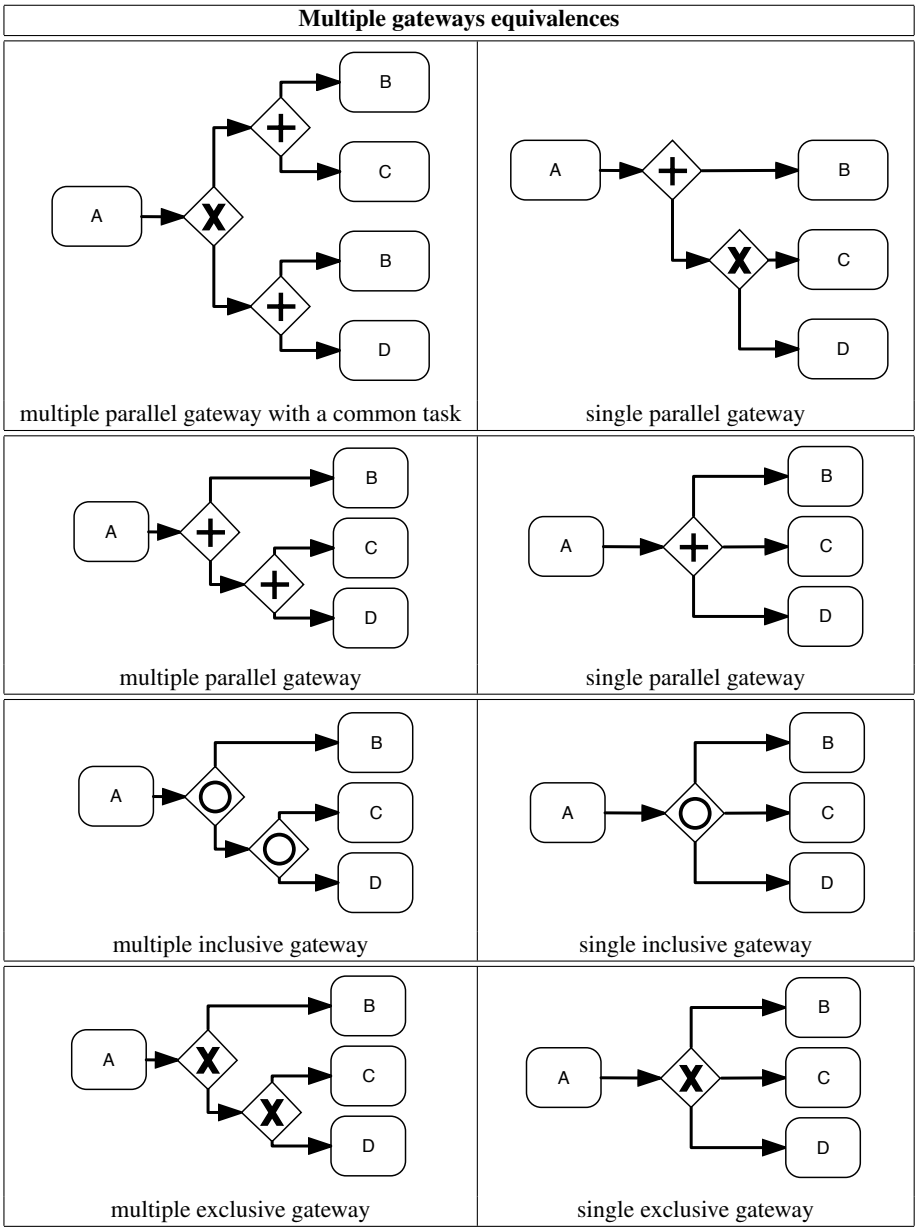


Table 3. Equivalences of BPMN structures based on multiple gateway elements

3.2 Complex Equivalences of BPMN structures

Other transformations considered in [10] concern discrimination and serialization mechanisms. In Table 6 several examples of the application of the discriminator transformation to selected BPMN elements are presented. The serialization examples, which transform something serialized implicitly to another thing serialized explicitly, are shown in Table 7.

Qing-xiu et al. [24], in order to verify a workflow model based on Petri net, proposed several reduction actions, such as reduction of sequential, iterative, or adjacent structure. However, the proposed reductions are not directly applicable to BPMN models.

Tantitharanukul and Jumpamule [31] defined Generalized Business Process Modeling Notation (GBPMN) as a notation for diagrams which nodes are labeled with the process expression. They presented an algorithm which converts any BPMN into GBPMN form. It is important to mention that the GBPMN is not a standardized solution, thus it is not very useful in practice. However, one of the steps of their algorithm is taken if the existing diagram has more than one start event or end event. In such a case, they stipulate adding a new single start event and/or a new single end event, and connecting these events to the existing diagrams by using inclusive gateway which is capable of capturing whether they simultaneously start or not. Using single start and end events should be taken into account when modeling, and such a procedure should be considered as a part of a normalization algorithm for business processes as well.

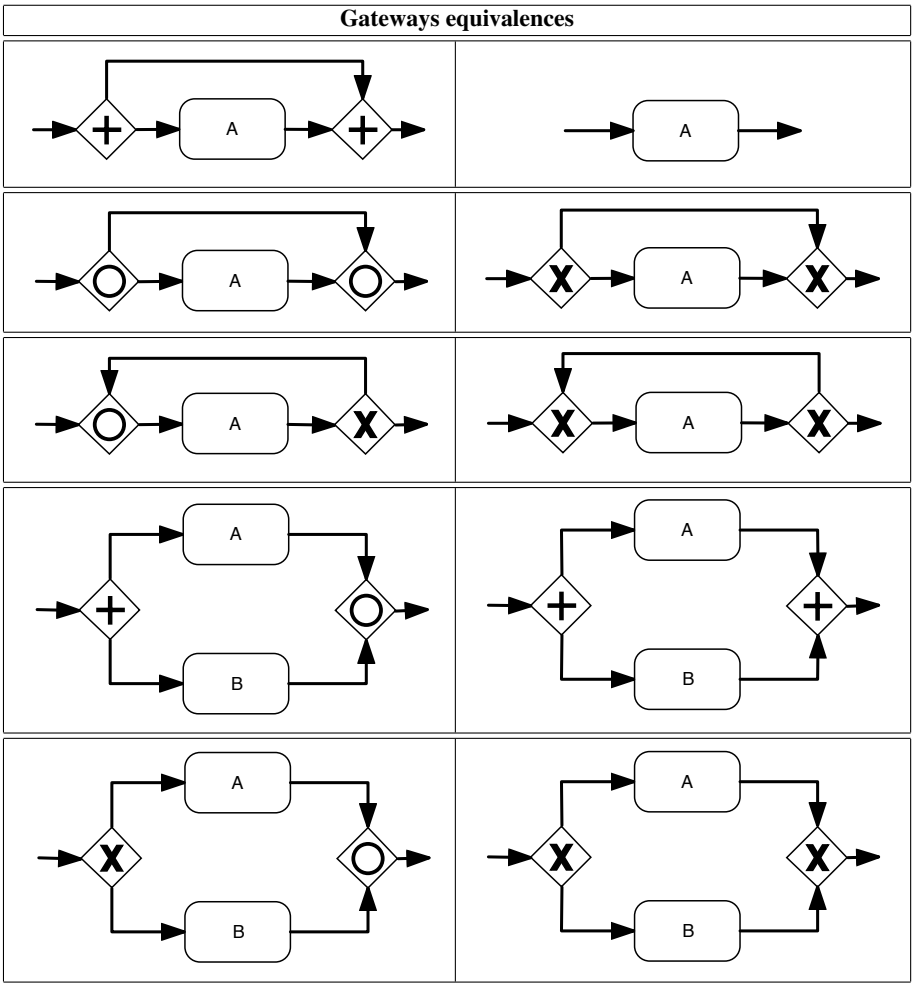


Table 4. Gateways equivalences of BPMN structures (based on [9])

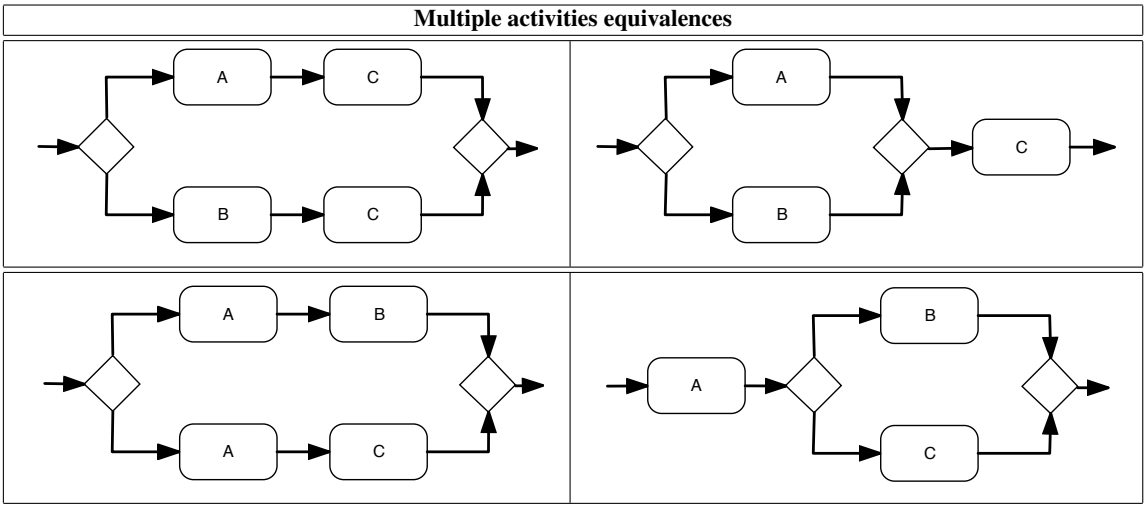


Table 5. Multiple activities equivalences of BPMN structures (based on [14, 17])

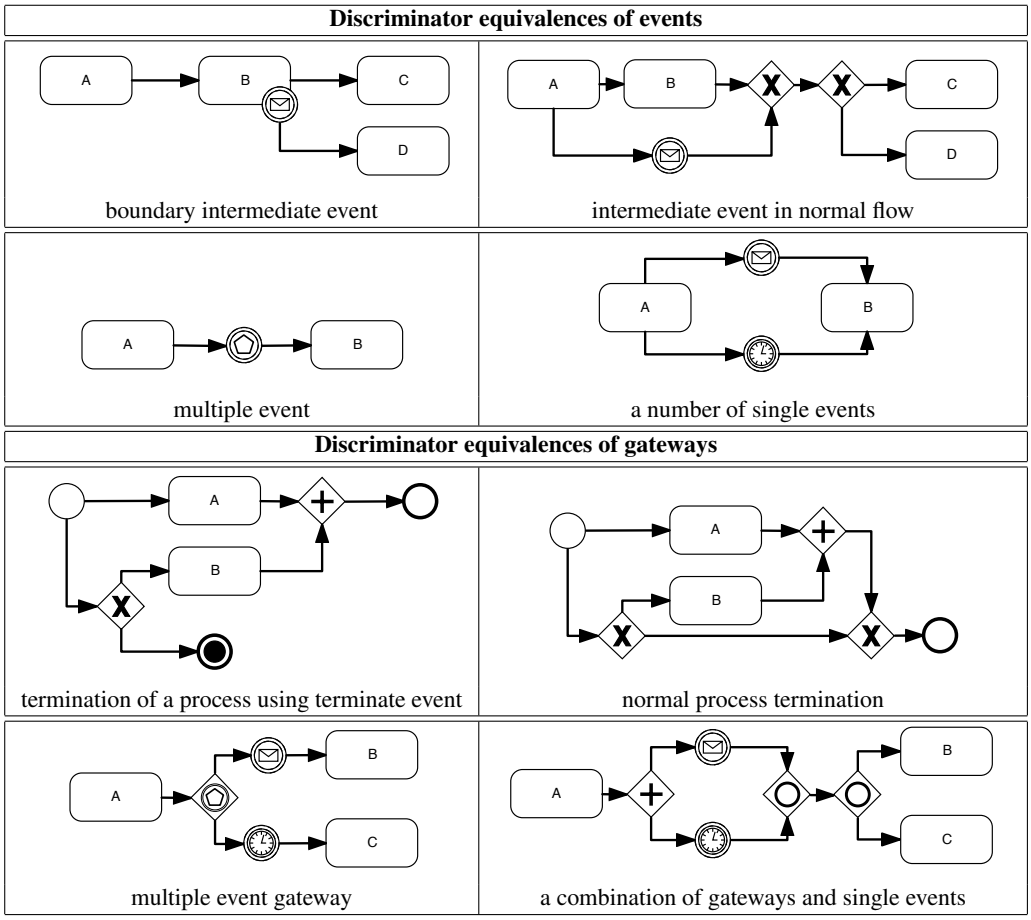


Table 6. Discriminator equivalences of BPMN structures (based on [10])

3.3 Guidelines for modelers

The normalization process should also take into account the existing guidelines for business modelers. Most of the existing tools do not require to comply with any guidelines or modeling requirements, so a user has to adhere to them itself.

One of the papers with most impact in the business process modeling field by Mendling et al. [20] concerns guidelines for business process modelers, which should be taken into account when modeling business processes. They formulated seven guidelines and prioritized them with the help of industry experts [20]:

1. Model as structured as possible.
2. Decompose a model with more than 50 elements.
3. Use as few elements in the model as possible.
4. Use verb-object activity labels.
5. Minimize the routing paths per element.
6. Use one start and one end event.
7. Avoid OR routing elements.

La Rosa et al. [26] performed a systematic analysis and proposed a number of concrete syntax modifications for business process models to manage their complexity. They presented a collection of patterns that generalize and conceptualize various existing mechanisms to change the visual representation of a process model. Their goal was to simplify the representation of processes. Thus, they identified eight patterns which reduce the perceived model complexity without changing the abstract syntax of the model and classified them according to the following hierarchy [26]:

1. Layout Guidance — describes features to modify the process model layout.
2. Outline visual mechanisms to emphasize certain aspects:
   (a) Enclosure Highlight — for visually enclosing close a set of logically related model elements,
   (b) Graphical Highlight — to change the visual appearance of model elements, such as shape, line thickness and type, etc.
   (c) Pictorial and Textual Annotation — to assign pictorial elements, such as icons or images, to modeling elements, or to visually represent free-form text in the canvas, which can be attached to modeling elements without changing semantics.
3. Two representation patterns:
   (a) Explicit Representation — to capture process modeling concepts via a dedicated graphical notation,
   (b) Alternative Representation — to capture process modeling concepts without the use of their primary graphical notation.
4. Naming Guidance — naming conventions or advice for model elements' labels, which can be syntactic (e.g. using a verb-object style) or semantic (e.g. using a domain-specific vocabulary).

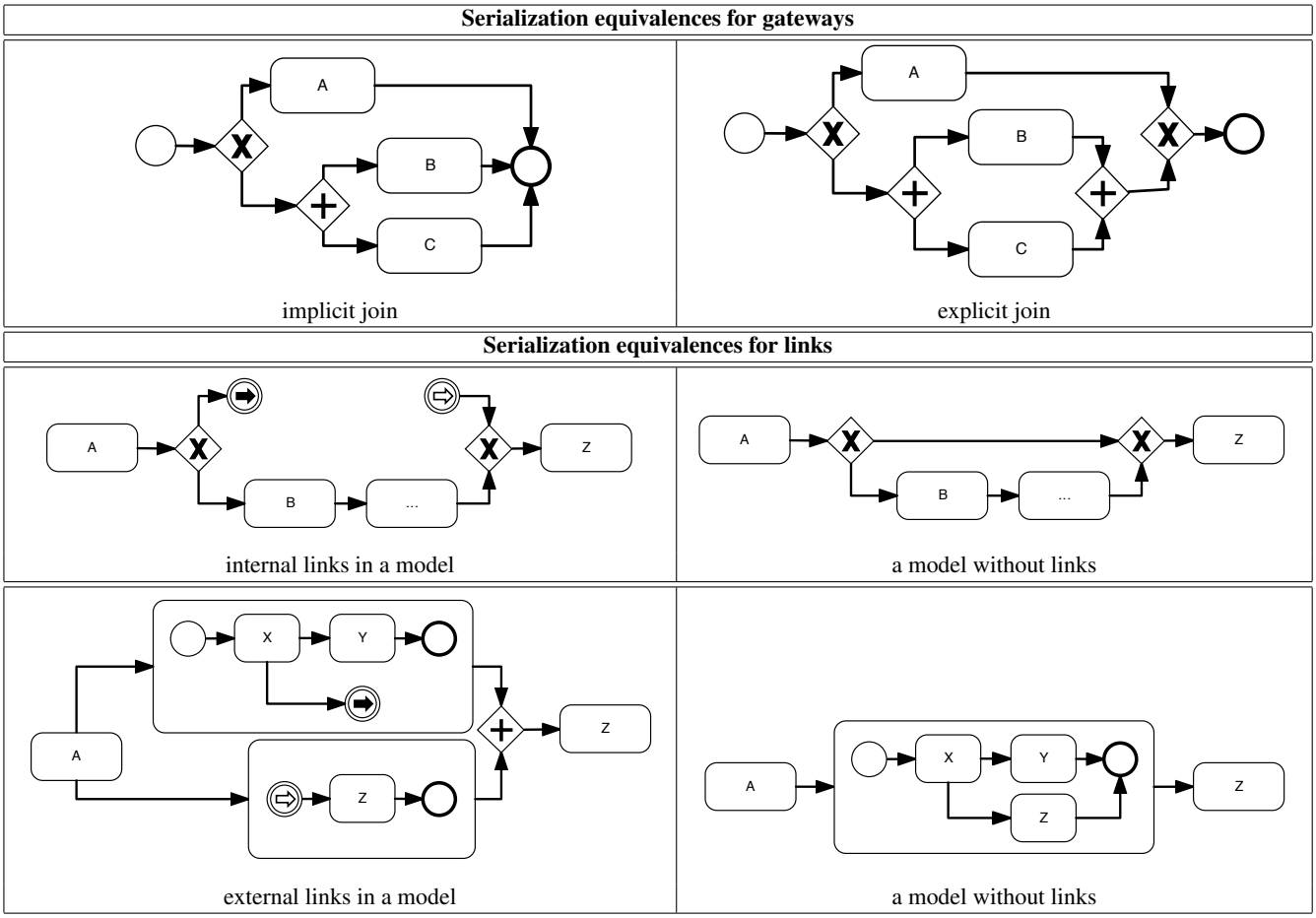


Table 7. Serialization equivalences of BPMN structures (based on [10])

4 Conclusion

Although BPMN is the most widespread notation used by software architects and business analysts for modeling Business Processes, it is not clear which structures should be preferred and which avoided. The BPMN specification does not clarify how the notation should be used for modeling various processes. Thus, the standardization of such modeling technique in BPMN is desired.

As BPMN allows for expressing the same semantics using various syntactic structures, this can cause the modeling and analysis challenges. Cognitive understanding of model semantics can vary in case of complex syntactic differences. Furthermore, a behaviorally equivalent but syntactically different structures can be analyzed in different ways or even can be untranslatable to other languages in order to be verified. To address these issues, a set of best practices for modelers as well as normalization of BPMN models are needed.

In this paper, we prepared the first step towards such a normalization process – based on a literature review, we presented an overview of the topic of BPMN models equivalences, identified various behaviorally (or semantically) equivalent structures, and pointed out possibilities of equivalent patterns.

Moreover, we presented several guidelines for modelers, which should be taken into account when modeling, and considered as a part of a normalization algorithm for business processes.

While normalization can be performed manually, and usually is in the case of ad hoc modeling, it is possible to support such a process with tools. However, most of the existing tools do not require to comply with any guidelines or modeling requirements, so a user has to adhere to them itself.

Furthermore, normalization can help in the future research on structuring diagrams in order to decrease their semantic complexity. Our research can be further useful for many purposes, such as process matching, identifying the differences between process models, analyzing similarities or measuring compliance of processes.

In our future research, we will formalize the presented equivalences. This will allow for implementing a tool for proving that two models are equivalent or using some of the existing tools for analyzing BPMN patterns for this purpose [15, 17, 18, 30]. Our goal is to define the preferable structures of the model, which will constitute a normalization process and a part of a modeling methodology for modeling business processes integrated with rules [22, 21]. Such process can be further supported by a proper tool framework [11].

References

[1] Thomas Allweyer, *BPMN 2.0. Introduction to the Standard for Business Process Modeling*, BoD, Norderstedt, 2010.

[2] Ahmed Awad, Matthias Weidlich, and Mathias Weske, 'Visually specifying compliance rules and explaining their violations for business processes', *Journal of Visual Languages & Computing*, **22**(1), 30–55, (2011). Special Issue on Visual Languages and Logic.

[3] Michael Becker and Ralf Laue, 'A comparative survey of business process similarity measures', *Computers in Industry*, **63**(2), 148–167, (Feb 2012).

[4] Workflow Management Coalition, 'Workflow management coalition terminology & glossary', Technical Report WFMC-TC-1011, WfMC, United Kingdom, (Feb 1999).

[5] Ana Karla Alves de Medeiros, Wil M. P. van der Aalst, and A. J. M. M. Weijters, 'Quantifying process equivalence based on observed behavior', *Data Knowl. Eng.*, **64**(1), 55–74, (2008).

[6] Remco Dijkman, Marlon Dumas, Boudewijn van Dongen, Reina Käärik, and Jan Mendling, 'Similarity of business process models: Metrics and evaluation', *Information Systems*, **36**(2), 498–516, (Apr 2011).

[7] Zhongjun Du and Zhengjun Dang, 'A new algorithm based graph-search for workflow verification', in *Proceedings of the 2nd International Conference on Information Engineering and Computer Science (ICIECS), 25-26 Dec. 2010*, pp. 1–3. IEEE, (2010).

[8] Kerstin Gerke, Jorge Cardoso, and Alexander Claus, 'Measuring the compliance of processes with reference models', in *On the Move to Meaningful Internet Systems: OTM 2009*, eds., Robert Meersman, Tharam Dillon, and Pilar Herrero, volume 5870 of *Lecture Notes in Computer Science*, 76–93, Springer Berlin / Heidelberg, (2009).

[9] V. Gruhn and R. Laue, 'Reducing the cognitive complexity of business process models', in *Proceedings from the 8th IEEE International Conference on Cognitive Informatics, 15-17 June 2009. ICCI'09.*, pp. 339–345, (2009).

[10] Moonyoung Jung, Hak Soo Kim, Myung Hyun Jo, Kyung Hyun Tak, Hyun Suk Cha, and Jin Hyun Son, 'Mapping from BPMN-formed business processes to XPDL business processes', in *Proceedings of the Fourth International Conference on Electronic Business – Shaping Business Strategy in a Networked World ICEB*, pp. 422–427. Academic Publishers/World Publishing Corporation, (2004).

[11] Krzysztof Kluza, Krzysztof Kaczor, and Grzegorz J. Nalepa, 'Enriching business processes with rules using the Oryx BPMN editor', in *Artificial Intelligence and Soft Computing: 11th International Conference, ICAISC 2012: Zakopane, Poland, April 29–May 3, 2012*, eds., Leszek Rutkowski and [et al.], volume 7268 of *Lecture Notes in Artificial Intelligence*, pp. 573–581. Springer, (2012).

[12] Li Kuang, 'A formal analysis of behavioral equivalence for web services', in *Proceedings from the IEEE Congress on Services - Part I, 2008*, pp. 265–268, (2008).

[13] Min-Hsun Kuo and Yun-Shiow Chen, 'A method to identify the difference between two process models', *Journal of Computers*, **7**(4), 998–1005, (2012).

[14] Vitus S. W. Lam, 'Equivalences of BPMN processes', *Service Oriented Computing and Applications*, **3**(3), 189–204, (2009).

[15] Vitus S. W. Lam, 'Formal analysis of BPMN models: a NuSMV-based approach', *International Journal of Software Engineering and Knowledge Engineering*, **20**(7), 987–1023, (2010).

[16] Vitus S. W. Lam, 'Foundation for equivalences of BPMN models', *Theoretical and Applied Informatics*, **24**(1), 33–66, (2012).

[17] Ralf Laue and Ahmed Awad, 'Visual suggestions for improvements in business process diagrams', *Journal of Visual Languages & Computing*, **22**(5), 385–399, (2011).

[18] Antoni Ligęza, 'BPMN – a logical model and property analysis', *Decision Making in Manufacturing and Services*, **5**(1-2), 57–67, (2011).

[19] N.M.b. Mahmod and S.b.A. Radzi, 'An approach to analyse similarity of business process variants', in *Proceedings from the IEEE International Conference on Progress in Informatics and Computing (PIC), 2010*, pp. 640–644, (2010).

[20] J. Mendling, H. A. Reijers, and W. M. P. van der Aalst, 'Seven process modeling guidelines (7pmg)', *Information & Software Technology*, **52**(2), 127–136, (Feb 2010).

[21] Grzegorz J. Nalepa, 'Proposal of business process and rules modeling with the XTT method', in *Symbolic and numeric algorithms for scientific computing, 2007. SYNASC Ninth international symposium. September 26–29*, eds., Viorel Negru and et al., pp. 500–506, Los Alamitos, California ; Washington ; Tokyo, (september 2007). IEEE Computer Society, IEEE, CPS Conference Publishing Service.

[22] Grzegorz J. Nalepa, Krzysztof Kluza, and Sebastian Ernst, 'Modeling and analysis of business processes with business rules', in *Business Process Modeling: Software Engineering, Analysis and Applications*, ed., J.A. Beckmann, Business Issues, Competition and Entrepreneurship, 135–156, Nova Science Publishers, (2011).

[23] OMG, 'Business Process Model and Notation (BPMN): Version 2.0 specification', Technical Report formal/2011-01-03, Object Management Group, (January 2011).

[24] Liu Qing-xiu, Cao Bao-xiang, and Zhao Yi-wei, 'An improved verification method for workflow model based on petri net reduction', in *Proceedings of the 2nd IEEE International Conference on Information Management and Engineering (ICIME), 2010*, pp. 252–256. IEEE, (2010).

[25] S. Rinderle-Ma, M. Reichert, and M. Jurisch, 'Equivalence of web services in process-aware service compositions', in *Proceedings from the IEEE International Conference on Web Services, 2009. ICWS 2009*, pp. 501–508, (2009).

[26] Marcello La Rosa, Arthur H. M. ter Hofstede, Petia Wohed, Hajo A. Reijers, Jan Mendling, and Wil M. P. van der Aalst, 'Managing process model complexity via concrete syntax modifications', *IEEE Transactions on Industrial Informatics*, **7**(2), 255–265, (2011).

[27] Wasim Sadiq and Maria E. Orlowska, 'Analyzing process models using graph reduction techniques', *Information Systems*, **25**(2), 117–134, (2000).

[28] Kamyar Sarshar and Peter Loos, 'Comparing the control-flow of epc and petri net from the end-user perspective', in *Business Process Management*, eds., Wil van der Aalst, Boualem Benatallah, Fabio Casati, and Francisco Curbera, volume 3649 of *Lecture Notes in Computer Science*, 434–439, Springer Berlin / Heidelberg, (2005).

[29] Marcin Szpyrka, Grzegorz J. Nalepa, Antoni Ligęza, and Krzysztof Kluza, 'Proposal of formal verification of selected BPMN models with Alvis modeling language', in *Intelligent Distributed Computing V. Proceedings of the 5th International Symposium on Intelligent Distributed Computing – IDC 2011, Delft, the Netherlands – October 2011*, ed., Frances M.T. Brazier et al., volume 382 of *Studies in Computational Intelligence*, 249–255, Springer-Verlag, (2011).

[30] N. Tantitharanukul, P. Sugunnasil, and W. Jumpamule, 'Detecting deadlock and multiple termination in bpmn model using process automata', in *Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON), 2010 International Conference on*, pp. 478–482, (May 2010).

[31] Nasi Tantitharanukul and Watcharee Jumpamule, 'Detection of livelock in BPMN using process expression', in *Advances in Information Technology*, eds., Borworn Papasratorn, Kittichai Lavangnananda, Wichian Chutimaskul, and Vajirasak Vanijja, volume 114 of *Communications in Computer and Information Science*, 164–174, Springer Berlin Heidelberg, (2010).

[32] Wil M. P. van der Aalst, Ana Karla A. de Medeiros, and A. J. M. M. Weijters, 'Process equivalence: Comparing two process models based on observed behavior', in *Business Process Management, 4th International Conference, BPM 2006, Vienna, Austria, September 5-7, 2006, Proceedings*, volume 4102 of *Lecture Notes in Computer Science*, pp. 129–144, (2006).

[33] Matthias Weidlich, Gero Decker, Alexander Grosskopf, and Mathias Weske, 'Bpel to bpmn: The myth of a straight-forward mapping', in *Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part I on On the Move to Meaningful Internet Systems*, OTM '08, pp. 265–282, Berlin, Heidelberg, (2008). Springer-Verlag.

[34] Stephen A. White and Derek Miers, *BPMN Modeling and Reference Guide: Understanding and Using BPMN*, Future Strategies Inc., Lighthouse Point, Florida, USA, 2008.

[35] Petia Wohed, Wil M. P. van der Aalst, Marlon Dumas, Arthur H. M. ter Hofstede, and Nick Russell, 'On the suitability of bpmn for business process modelling', in *Business Process Management, 4th International Conference, BPM 2006, Vienna, Austria, September 5-7, 2006, Proceedings*, volume 4102 of *Lecture Notes in Computer Science*, pp. 161–176, (2006).

[36] Jian Zhu and Hung Keng Pung, 'Process matching: A structural approach for business process search', in *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD '09. Computation World*, pp. 227–232, (2009).

Critical evaluation of the XTT2 rule representation through comparison with CLIPS[1]

Krzysztof Kaczor and **Grzegorz J. Nalepa**[2]

Abstract. There are two main approaches to the design Business Rules. The first one involves formalized methods that strictly define the syntax and semantics of rules. This approach usually requires technical skills or conceptual knowledge and therefore is not appropriate for everyone. In the second approach, dedicated rule languages are used for facilitating rules specification. Nevertheless, such languages are usually programming solutions without a precisely defined semantics. This may cause ambiguities in knowledge interpretation and thus, the efficient rule interoperability becomes impossible. The goal of our work is to develop a formalized model for a rule representation which will allow for an effective rule interchanging. For this purpose, we want to combine the above mentioned approaches by tailoring the formalized rule representation called XTT2 to languages provided by CLIPS or Drools. This paper is the first step in our research providing an identification of the most important differences between the XTT2 and CLIPS rule languages.

1 Introduction

Rule-Based Systems constitute a mature technology in the field of Artificial Intelligence. Over the years, they were applied in many domains like medicine, engineering [7] or decision support [10]. Despite their maturity, many ideas, algorithms and solutions that are applied in new technologies, such as Business Rules (BR) [22], Semantic Web [2] or Complex Event Processing [12], are derived from the classic Rule-Based Systems [9].

Business Rules are one of the latest application of classic rules. They are intended to be created by business people in order to define logical aspects of business. Despite the fact that business people may not have any technical skills or scientific knowledge, BR must be appropriate to be used by such users. Currently, many techniques are used for the specification of BR, from description in natural language to design by using formalized methods. There is no single method that is considered to be the best. Usually, a designer chooses one according to his or her own preferences.

Rules specification in natural language is very intuitive and does not require any specialized skills. Moreover, such a method allows for an easy specification of very complex rules. However, such informal description may be very vague, especially in case of complex rules which may be hard to understand or in the worst case may be misunderstood. This type of problems can be prevented by using formalized methods having the following advantages:

- they provide a clear framework enabling uniform knowledge modeling with well-defined expressive power,
- speed up the design process – formalized rule language opens possibility to partially formalize the design process which can, in turn, lead to better detection of design errors, possibly at early development stages,
- allow for a superior knowledge base quality control – formal methods can be used to identify logical errors in rule formulation,
- simplify knowledge interoperability – partially formalized translation to other knowledge representation formats are possible, and
- allow for custom inference modes – structured rule bases require inference strategies alternative to the classic inference algorithms.

This paper is organized as follows: Section 2 gives a short motivation for our work. A short introduction to the XTT2 method is provided by Section 3. Section 4 is the main part of this paper discussing the most important differences between XTT2 and CLIPS. The paper is concluded with Section 5 providing short summary and information concerning future works.

2 Motivation

Together with the development of BR design methods, a number of development tools also increases. Among them very important are Drools [4] and OpenRules[3]. Sometimes, it is desirable to have a mechanism for exchanging knowledge between different tools. This makes maintenance of the rule bases easier and allows for more efficient usage of these tools. Nevertheless, the existing tools usually allow for BR modeling in an informal way and do not provide any common rule representation model. This provides ambiguity in the rule semantics and in turn, does not allow for efficient interchanging.

A problem of knowledge interoperability is known since classic rule-based expert systems and still remains an open issue. During the years, several approaches to this problem were proposed. The most important of them are: Knowledge Interchange Format[4], Rule Markup Language [3], Rule Interchange Format [8] and REWERSE Rule Markup Language [23]. Nevertheless, the above mentioned methods provide a very general model of rule-based knowledge representation, what makes their practical application hard or even impossible. Hence, practical tools supporting any of these methods do not exist or provide only partial support.

The main objective of our current research is to develop a formalized method for an efficient rule interoperability. We assume that this can be done by providing a common and logic-based rule representation model. Thanks to such a model, the semantics of rules, specified in other representations, can be clarified or defined. What

[1] The paper is supported by the AGH UST Grant.

[2] AGH University of Science and Technology, Poland, email: kk,gjn@agh.edu.pl

[3] See: `http://openrules.com`

[4] See: `http://www.upv.es/sma/teoria/sma/kqml_kif/kif.pdf`

is more, this model will allow to specify which representation can be losslessly translated to another and how this translation should be performed. We assume that the model will be based on the formalized rule representation method called XTT2 [19] which is provided by the Semantic Knowledge Engineering (SKE) methodology [16]. The XTT2 method (see Section 3) is a visual method for modeling structured rule bases. This method is intended to be a rigorously formalized rule language. Nevertheless, a rigorous formalization has restricted the expressiveness of the language. Thus, in comparison with other methods, XTT2 has several limitations and significant differences. This is why our current work is focused on the extension of XTT2 towards such languages as CLIPS[5] [6] or Drools. These two languages have been selected as the reference because they proved to be successful implementations of rule-based systems.

CLIPS is a classic rule-based expert system shell developed by NASA in 1984. The original intent for CLIPS was to gain useful insight and knowledge about the construction of expert system tools and to lay the groundwork for the construction of a replacement tool for the commercial tools being used in that time. Because of its portability, extensibility, capabilities, and low cost, it has received widespread acceptance throughout the government, industry, and academia. Development of this tool has improved the accessibility to expert system technology throughout the public and private sectors for a wide range of applications and diverse computing environments. CLIPS became one of the most commonly known rule language that was used for e.g. image processing or recognition.

As a classic rule-based tool, CLIPS became a reference tool also for other tools like Jess which is a rule engine and scripting environment providing rule language. It is written in Java. Jess was originally a clone of the essential core of CLIPS, but has begun to acquire a Java-influenced flavor. Therefore, it is a convenient tool for giving Java applets and applications the ability to reason.

Drools is a much younger project which was started in 2001. Currently Drools is widely used by Business environment as Business Logic Integration Platform providing a unified and integrated platform for Rules, Workflow and Event Processing. Drools-based rules are specified using dedicated rule language and processed by dedicated rule engine called Drools Expert. Similarly to Jess, this engine is also written in Java and allows for easy integration with other applications written in this language.

The above mentioned tools are not intended to provide a formalized rule representation that is necessary for efficient rule interchange preserving their semantics. The provide only programming solutions for rapid development of the rule bases. In our work, we try to combine the advantages of formalized methods and programming solutions. This paper describes the first step of this work. The main contribution is the comparison of the XTT2 method with the CLIPS language, by identifying the differences and limitations of XTT2 in terms of CLIPS. It describes the most important aspects of extending XTT2 towards CLIPS and challenges that must be overcome for an efficient rule interoperability between these two representations.

3 Overview of XTT2

This section gives a short introduction to XTT2 (*eXtended Tabular Trees*) [17, 18]. XTT2 can be considered a multidimensional concept. It involves many aspects of the rule-based systems design:

- Rule Base — this aspect involves issues related to structure and maintenance of a rule base.
- Rule Syntax — defines how the knowledge can be expressed and what are the limitations of a provided rule language i.e. this issue concerns rule language syntax as well as its expressiveness.
- Rule Semantics — defines the semantics of the rules and how they should be interpreted.
- Rule Processing — is related to inference mechanism as well as the way how the knowledge processing is performed.

This section is divided into four subsections describing XTT2 in terms of above mentioned aspects.

3.1 Rule Base

An XTT2-based rule base contains attributes that store values. Each attribute-value pair can be considered as a single fact. The set of all pairs is called system state and is defined as follows:

$$s\colon (A_1 = S_1) \wedge (A_2 = S_2) \wedge \ldots \wedge (A_n = S_n) \qquad (1)$$

where A_i are the attributes and S_i are their current values.
It is important to notice, that the number of attributes (facts) is constant during the inference process. The knowledge base modification can be made only by changing attribute value.

XTT2 provides modularized rule base, where rules working together are placed in one context. Contrary to the majority of other systems, where a basic knowledge item is a single rule, in the XTT2 formalism the basic component displayed, edited and managed at a time is a single *context*. A single context corresponds to a single decision table. Thus, only those rules which have the same conditional and decisions attributes can be placed in one context i.e. each rule in a decision table determines values of the same set of attributes.

XTT2 is a hybrid knowledge representation combining a decision network and decision tables. Tables are linked together forming a network-like structure of the XTT2 decision tables. Links define order in which tables should be processed. Considering a single table as a blackbox for determining attribute value, the links corresponds to functional dependencies between attributes.

3.2 Rule Syntax

The XTT2 rule language provide a dedicated syntax called *HeKatE Meta Representation* (HMR). This is a textual representation that can be easily read by human and automatically processed by an inference engine. Moreover, the HMR language is suitable for visual representation (see Figure 1). Thanks to this, such a knowledge representation provides not only high density of knowledge visualization, but assures transparency and readability. Additionally, the visual representation is fully supported by the HQEd [20] graphical editor. Using this editor, a HMR-based representation can be automatically generated for a given visual model. Then, the HMR representation is processed by the HeaRT [15] tool which is the dedicated inference engine for reasoning with the XTT2 rule bases [1].

For study purposes, an example below presents the same rule in three representations: natural language, HMR representation and XTT2-based visual representation. The rule comes from Cashpoint case study [5] and is as follows:

```
if
  driver is younger than 25 years
  and
  it has driving licence at least three years
then
  increase the driver current discount by 50%
```

[5] See: `http://clipsrules.sourceforge.net`

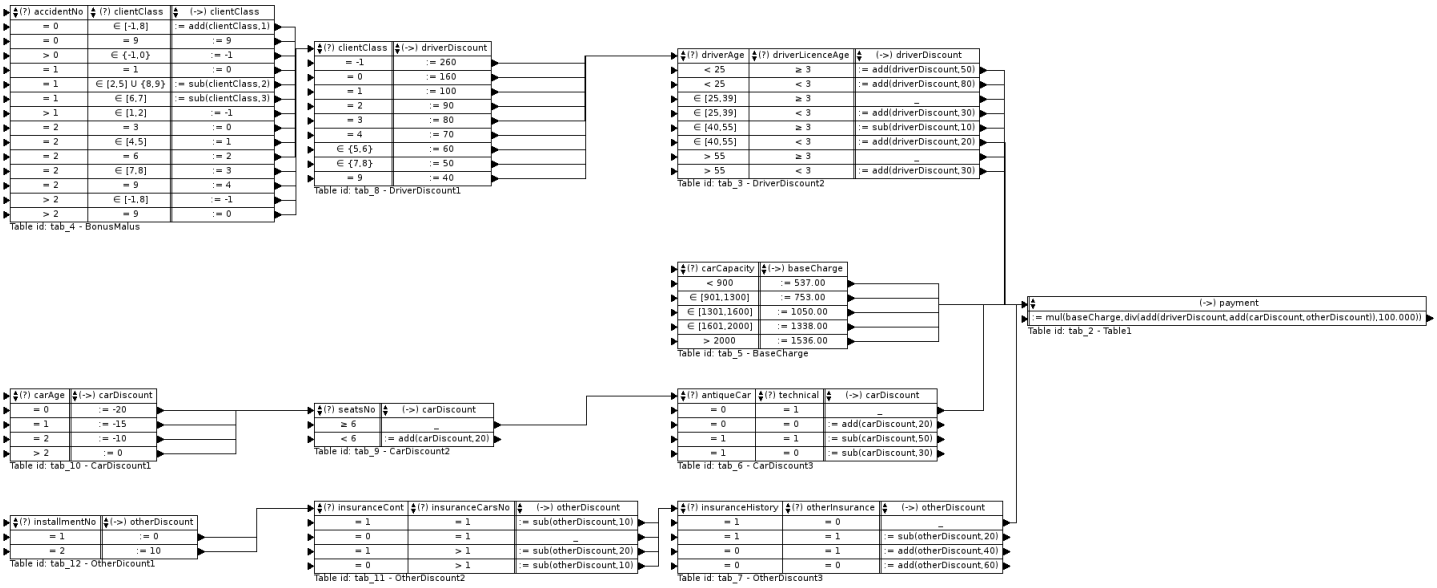


Figure 1. An example of visual representation of XTT2 rule base

(?) driverAge	(?) drLicAge	(->) driverDiscount
< 25	>= 3	= driverDiscount+50

Figure 2. An example of visual representation of XTT2 rule

This rule can be easily expressed with HMR syntax:

```
xrule 'Table1/1':
  [driverAge lt 25,
   drLicAge gte 3]
==>
  [driverDiscount set (driverDiscount+50)]
```

The figure 2 depicts the equivalent visual representation of the provided HMR syntax.

3.3 Rule Semantics

XTT2 is based on the *Attributive Logic with Set of Values over Finite Domains* (ALSV(FD)) logic [11, 18]. ALSV(FD) is a formal framework for attributive logic that provides syntax, semantics and some notes on inference rules for a logical calculus in which attributes can take *set values* (generalized attributes). In comparison with other attributive logics, its expressive power is increased through the introduction of new relational symbols enabling definitions of atomic formulae. This logic provides very strict and rigorous definition of rule semantics allowing for knowledge definition which can be unambiguously interpreted.

A single rule in ALSV(FD) is defined as a set of ALSV(FD) triples. The exemplary rule from Section 3.2 can be expressed in ALSV(FD) in the following way:

$$r_1 : (driverAge, <, 25) \wedge (drLicAge, >=, 3) \rightarrow \\ \rightarrow (driverDiscount, :=, driverDiscount + 50)$$

The complete formalization of XTT2 can be found in [19].

3.4 Rule Processing

XTT2 provides a dedicated rule processing mechanism. This is an advanced inference algorithm that can work in one of four modes: Fixed Order, Data, Token and Goal Driven (for more details see [13]). The inference mechanism is responsible for evaluating and executing (firing) rules. The rules are processed in predetermined order which is specified by taking the following issues into account: inference mode, links between modules, order of the rules in the XTT2 table.

4 Challenges in the Rule Interoperability between XTT2 and CLIPS

The goal of our work is to develop a common unified rule representation model for efficient rule interoperability between different rules representations. Our starting point is formalization of the XTT2 method which provides the underlying ALSV(FD) logic. Nevertheless, the current form of the method does not use the ALSV(FD) logic effectively (lack of support for complex types) and has several limitations. In comparison to CLIPS, this language constitutes a subset of CLIPS. In this context, the knowledge interchange between XTT2 and CLIPS requires many improvements and changes that must be done in XTT2 in order to provide a better coverage of CLIPS. Some of the limitations stem from the made assumptions and others stem from the visual representation. This section provides a detailed description of the most important challenges in the context of extending XTT2 and efficient rule interoperability with CLIPS. The section is divided into subsections according to aspects introduced in Section 3.

4.1 Rule Base

Rule base modification CLIPS provides a mechanism allowing for modifying a Knowledge Base (KB) by asserting, retracting and modifying facts. This can be done using the following commands: `assert`, `retract` and `modify`. Thanks to this, when the system is running, the number of facts in the KB can be changed. In contrast to CLIPS, the XTT2-based knowledge base defines a system with a constant number of facts described by attributes. During the execution, attributes are neither created nor removed from the knowledge

base. Only the current value of the attribute can be changed. Nevertheless, the process of asserting and retracting facts in XTT2 can be achieved using generalized attributes (see paragraph *Multivalued attributes* in Section 4.2).

Modularization of Knowledge Base Both, XTT2 or CLIPS provide mechanism for creating modularized knowledge base. In XTT2, set of rules that work together are grouped into so called contexts. Each context corresponds to a single XTT2 table and contains rules which have the same attributes in their conditional and conclusion parts. In turn, CLIPS provide modules which can be defined using `defmodule` construct. In contrast to XTT2, CLIPS modules do not provide any policy determining which construct can be placed in a module. In particular any rule (or other construct) can be placed in any module. In fact, this is also possible in XTT2 and can be achieved by extending rules LHS and RHS by all attributes that appears in the other rules in this context. However, this can lead to formation of large tables in which majority of cells contain always true comparisons (an example of such table is depicted in Figure 3).

CLIPS modules allow a set of constructs to be grouped together such that explicit control can be maintained over restricting the access of the constructs by other modules. This type of control is similar to global and local scoping used in languages such as C. The default behavior in CLIPS restricts constructs in one module to be accessible in another. However, this can be modified and selected elements can be permitted to be visible from other modules. In turn, the XTT2 rules placed in one context are not accessible from another.

In both CLIPS or XTT2, modules are used by rules to provide execution control. In CLIPS, each module has its own pattern-matching network [13], and thus only rules from the active module can be activated and executed. Similarly, in XTT2 only rules from the active context are evaluated and can be executed.

Variables CLIPS provides two elements which allow for storing information: facts and variables. Nevertheless, the semantics of these two elements is different. Facts are knowledge-based elements which defines what is currently known. Any change made in a set of facts invokes the pattern-matching process. In contrast to facts, variables are used for defining non knowledge-based values e.g. values of some factors, constant values, etc. Variables can be used as a part of pattern-matching process, however changes of their values do not invoke pattern-matching. CLIPS variables can be defined using `defglobal` construct e.g.:

```
(defglobal ?*high-priority-factor* = 100)
```

In turn, XTT2 does not provide any concept having the same semantics as CLIPS variables. The system designed with XTT2 consists of attributes. According to ALSV(FD), the state of the XTT2-based system is defined as a set of current values of all attributes specified within the knowledge base. From logical the point of view, the state of the system is represented as a logical formula (1). According to this definition, all XTT2 attributes are considered to be knowledge-based elements.

It is important to notice that the inference mechanism from XTT2 works in different way than in CLIPS. It evaluates rules in predetermined order and changes in attribute values do not affect it.

4.2 Rule syntax

Complex types The first and most important limitation of the XTT2 is related to complex types. A complex type is a data type that provides its own structure and aggregates a fixed set of labelled fields, possibly of different types, into a single type. An example of such type is a structure that is known from C programming language. The ALSV(FD) logic provides support for complex types and objects throughout attribute function which denotes a property of an object and allows for accessing its value using property name. However, currently XTT2 uses only atomic types for defining all attributes in the knowledge base and assumes that only one object (in this case it is the system being described) with a specific property name exists. In turn, CLIPS provides `deftemplate` element that allows for defining complex facts consisting of number of typed properties (called slots in CLIPS-based vocabulary):

```
(deftemplate person
  (slot name (type SYMBOL))
  (slot surname (type SYMBOL))
  (slot gender (type SYMBOL))
  (slot height (type INTEGER))
  (slot age (type INTEGER))
)
```

This example defines a template of person which allows for creating complex facts consisting of five typed properties: `name`, `surname`, `gender`, `height` and `age`.

Multivalued attributes ALSV(FD) provides a generalized attribute that can take more than one value at any point of time. This is very important and useful feature of ALSV(FD), however it is hard to assess to which element of the CLIPS language it corresponds. There are two obvious possibilities:

- facts list of the same type – a generalized attribute can be used for aggregation of values having the same type. A value of generalized attribute is defined as set. Such sets can be modified using set theory operators. In particular union of sets or difference of sets can correspond to CLIPS operations of asserting or retracting facts to/from knowledge base.
- multivalued slots – the `deftemplate` construct in CLIPS allows for defining multivalued slots which can take more that one value:

```
(deftemplate person
  (slot name (type SYMBOL))
  (slot surname (type SYMBOL))
  (slot gender (type SYMBOL))
  (slot age (type INTEGER))
  (slot height (type INTEGER))
  (multislot friends (type SYMBOL))
)
(assert (person (name Tom) (surname Joe)
  (gender M) (age 18) (height 180)
  (friends John Alex Emma))
)
```

This defines a man (`M`) `Tom Joe` that is `180` cm tall and `18` years old and has three friends: `John`, `Alex` and `Emma`. It is important to notice that the list of friends is not treated as one string containing spaces, only as the list of three separate values.
Usually multislot contains values with the same semantics (informally described by a slot name). Apart from the support for complex types, the generalized attribute in XTT2 can be used in the same context as the multivalued slots in CLIPS.

(?) authorizated	(?) failedAttempts	(?) userRequestedAction	(?) udAmountDifference	(?) cdAmountDifference	(->) cashPointActivity
= false	< 3	= any	= any	= any	:= askForPIN
= false	= 3	= any	= any	= any	:= takeCardAway
= true	= any	= balance	= any	= any	:= displayBalance
= true	= any	= withdraw	>= 0	>= 0	:= payOut
= true	= any	= withdraw	= any	< 0	:= msgNotEnoughFoundsInMachine
= true	= any	= withdraw	< 0	>= 0	:= msgNotEnoughFoundsOnAccount

Table id: tab_2 - Table1

Figure 3. An example of a large XTT2 table

Expressions in LHS The XTT2 method provides mechanisms for logical quality analysis called HalVA [14]. It allows for discovering logical anomalies such as inconsistency, redundancy, contradictions etc. In order to assure higher efficiency of HalVA, the LHS of the rule can contain only a simple attribute-to-value comparisons e.g.:

```
A = 12    B > 23    C in {1,2,3}
```

Such comparisons test a specific attribute against its value. Thus, an attribute is always on the left hand side of a comparison and constant value or set of constant values on the right hand side. Neither attributes nor expressions are allowed on the right hand side e.g.:

```
A = 11+1   B < A-3
```

In turn, the *Right Hand Side* (RHS) of a rule can contain complex expressions and attribute references:

```
A := A+1   B := 4*3
```

In contrast to XTT2, CLIPS allows for any complex expressions in conditional part of the rules. This limitation of XTT2 can be omitted by creating an additional decision table having required expression in its RHS. The figure 4 depicts the equivalent construction in CLIPS and XTT2. The rules comes from the Cashpoint example [21] and are intended to check if a user has entered a correct PIN. This is done by comparing `enteredPIN` and `correctPin` attributes. The equality of this two attributes is a condition that must satisfied in order to authorize a user. In CLIPS this condition can be placed directly in LHS of a rule, while XTT2 required an additional table (`Table3`) and attribute (`pinDifference`).

Constraints ALSV(FD) provides a concept of attribute domain. A domain is a finite set of admissible values that attribute can take. Each domain is based on one of two primitive types *symbolic* or *numeric*. In XTT2, for each attribute a domain must be specified. The domain concept plays important role because it is used by verification mechanism for discovering logical anomalies in knowledge base. The example below shows a definition of types (in HMR language) restricting values of the attributes describing a person. We assume that:

name is not restricted and can contain any list of characters,
gender can take only two values: `M` for male and `F` for female,
height can take a value from the interval $[0, 300]$,
age can take a value from the interval $[0, 120]$.

```
xtype [name: name,
       base: symbolic].
xtype [name: gender,
       base: symbolic,
       domain: [M,F]].
xtype [name: height,
       base: numeric,
       domain: [0 to 300]].
xtype [name: age,
       base: numeric,
       domain: [0 to 120]].
```

In CLIPS, a value of a slot can be restricted using similar concepts: primitive types, list of values, ranges. However, CLIPS provides more primitive types than XTT2: `SYMBOL`, `STRING`, `LEXEME`, `INTEGER`, `FLOAT`, `NUMBER`, `INSTANCE-NAME`, `INSTANCE-ADDRESS`, `INSTANCE`, `EXTERNAL-ADDRESS`, and `FACT-ADDRESS`. Moreover, CLIPS allows for restricting a number of elements in multivalued slots.

The equivalent CLIPS-based definition of slot constraints describing person may look like this:

```
(deftemplate person
  (slot name (type SYMBOL) )
  (slot surname (type SYMBOL))
  (slot gender (type SYMBOL)
    (allowed-symbols M F))
  (slot height (type INTEGER) (range 0 300))
  (slot age (type INTEGER) (range 0 120))
  (multislot friends (type SYMBOL))
)
```

The one advantage of XTT2 in comparison with CLIPS is that the XTT2 allows for defining symbolic ordered domains. Such concept is similar to `enum` construct from C programming language. Thanks to ordering, the symbolic values can be treated as ordinary integer values e.g.:

```
xtype [
  name: weekdaytype,
  base: symbolic,
  domain: [mon/1,tue/2,wed/3,thu/4,
    fri/5,sat/6,sun/7],
  ordered: yes].
```

In this example a type describing weekdays is defined. Each day has assigned an equivalent numeric value. Thanks to that one can write:

```
mon > tue    A = tue+wed
```

The results of this expressions are equal to results of corresponding expressions where symbolic values were replaced with numeric.

Values binding In some cases, it is very hard or even impossible to define LHS of a rule by using only logical and relational operators. Let us consider the following example: The knowledge base contains information about a number of people described by properties defined in paragraph *Complex types*:

```
xrule 'Table3'/1:
    [enteredPin eq any,
      correctPin eq any]
  ==>
    [pinDifference set
      (correctPin-enteredPin)]
    :'Table2'.

xrule 'Table2'/1:
    [pinDifference neq 0]
  ==>
    [authorizated set false,
      failedAttempts set (failedAttempts+1)]
    :'Table1'.
xrule 'Table2'/2:
    [pinDifference eq 0]
  ==>
    [authorizated set true,
      failedAttempts set failedAttempts]
    :'Table1'.
```

```
(defrule rule-1
    ?a <- (atm (enteredPin ?e)
      (correctPin ?c)
      (failedAtempts ?f))
      (test (neq ?e ?c))
 =>
    (modify ?a (authorizated false)
      (failedAtempts (+ ?f 1))))

(defrule rule-2
    ?a <- (atm
      (enteredPin ?e)
      (correctPin ?c))
    (test (eq ?e ?c))
 =>
    (modify ?a (authorizated true)))
```

Figure 4. The equivalent construction in XTT2 (on the left) and CLIPS (on the right)

```
(person (name Tom) (surname Joe)
  (gender M) (age 18) (height 180)
  (friends John Alex Emma))

(person (name Emma) (surname Johnson)
  (gender F) (age 19) (height 180)
  (friends Tom Julia))

(person (name Alex) (surname Johnson)
  (gender M) (age 17) (height 170)
  (friends Tom Emma Julia))
```

Our task is to define a rule selecting all allowed pairs of persons which can dance together. Two person can dance together when satisfy the following conditions: 1) They have different gender and 2) they have the same growth. Such a rule can be easily written using mechanism allowing for value binding. This mechanism allows for retrieving desired value during inference and storing it in a user-defined variable. Then, this variable can be used in further conditions as well as conclusion part. The rule for our task can look like this:

```
(defrule rule-1 "Our solution"
  (person (name ?n1) (surname ?s1)
    (gender M) (height ?h))
  (person (name ?n2) (surname ?s2)
    (gender F) (height ?h))
=>
  (printout t ?n1 " and " ?n2 crlf)
)
```

The LHS of the rule contains two conditions that refers to `person` template. Thanks to this, the inference algorithm would try to match all possible pairs of `person` facts. When a single match is performed, then the variables (which names start with question mark) are bound to the current value of the matched fact. Binding is made only one time during a single match and the variable stores bounded value until this particular match is finished. Thus, usage of bounded variable in further conditions restricts the set of elements that can be matched because matching algorithm must take its value into account. So, the variable binding can be used for defining restrictions across several objects. In our example, the `?h` variable is bound in the first condition and then its value is used in the second condition. This restricts the set of possible facts that can be matched to the second condition, because apart from the value `F` of the `gender` slot, a matched fact must have the same value of the `height` slot as the fact matched in the first condition.

Variable bindings is currently not supported in XTT2. Thus, definitions of equivalent rule is currently not possible.

Functions CLIPS allows for defining functions. It is possible to define a user-defined external functions that can be written in an external language e.g. C and then linked with CLIPS during recompilation. Such functions can be later executed directly in CLIPS in ordinary manner. Moreover, CLIPS provides a second mechanism allowing for defining function directly in CLIPS by using CLIPS-based syntax. This can be done with the help of the `deffunction` construct. The CLIPS-based functions have all features that an ordinary function can have i.e.: unique name, list of parameters, sequence of actions, returned value, recursion. An example function that calculates the factorial of an argument can be written as follows:

```
(deffunction factorial (?a)
  (if (or (not (integerp ?a)) (< ?a 0)) then
    (printout t "Factorial Error!" crlf)
  else
    (if (= ?a 0) then
      1
    else
      (* ?a (factorial (- ?a 1))))))
```

It is important that each function can be invoked from any part of a rule and can modify a knowledge base.

XTT2 provides a similar mechanism to CLIPS user-defined external functions through callbacks. Callback function is an external function written in Prolog or Java language. Then, such function is invoked by Prolog interpreter directly or by using JPL plugin for callbacks written in Java.

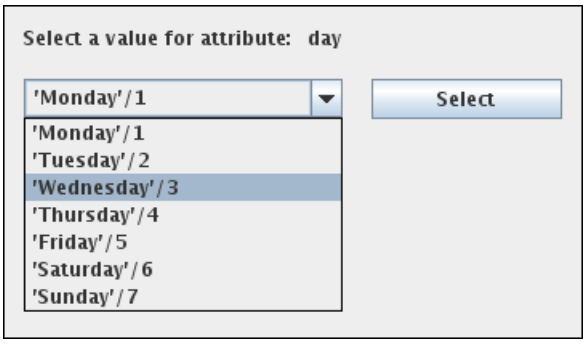


Figure 5. An example of dialog invoked by callback function

Callbacks in XTT2 are strictly related to attributes. Each attribute can have two callback functions assigned: *input* callback and/or *output* callback. The *input* callbacks are used for retrieving attribute value from outside system when value of an attribute is not defined. Thus, this type of callback function can modify a knowledge base. In contrast to *input* callbacks, the *output* callbacks cannot modify state of the system and are used only for presentation purposes. The order and time when a callback is invoked is determined by inference algorithm and cannot be redefined. The example below depicts the definition of *input* callback and attribute to which is assigned:

```
xcall ask_console_symbolic: [AttName] >>> (
  alsv_domain(AttName,Domain,symbolic),
  write('availible answers are '),
  write(Domain), nl,
  write(AttName), write(': '), read(Answer),
  (member(Answer,Domain) ->

xattr [name: weekday,
  abbrev: weekday,
  class: simple,
  type: weekdaytype,
  comm: in,
  callback: [ask_console_symbolic,[day]]
```

This callback function invokes dialog allowing user to provide value of an attribute. The example of such dialog is depicted in Figure 5. The list of possible values is created according to attribute type. The definition of the attribute `weekday` type can be found in Section 4.2 in paragraph *Constraints*.

4.3 Rule semantics

Ordered structures CLIPS facts defined by using `deffact` construct are also called non-ordered facts. This is because the fact structure consists of fields that are referred by named slots. Additionally, CLIPS provides an ordered facts which encode information positionally. To access the information, a user must know not only what data is stored in a fact but which field contains the data. The first field of an ordered fact specifies a *relation* that applied to the remaining fields in the ordered fact e.g.:

```
(father-of jack bill)
```

This fact defines that `bill` is the `father of jack`.

The current XTT2 method does not provide any concept with similar semantics. ALSV(FD) provides support only for complex types, where properties of object are referred by attribute function.

Rules properties The way, a rule is processed by CLIPS can be modified by changing rule properties. CLIPS provides support for two properties `auto-focus` and `salience`.

The `auto-focus` property allows an automatic focus command to be executed whenever a rule becomes activated. If the `auto-focus` property for a rule is `true`, then a focus command on the module in which the rule is defined is automatically executed whenever the rule is activated. This property can be used for defining rules responsible for values validation:

```
(defrule VIOLATIONS::bad-age
  (declare (auto-focus TRUE))
  (person (name ?name) (age ?x&:(< ?x 0)))
=>
  (printout t ?name " has a bad age value."))
```

The above rule is activated whenever the `VIOLATIONS` module receives focus and checks if all the person facts accessible in that module have correct value of the slot `age`.

The `salience` property allows for assigning a priority to a rule. This property is a part of conflict resolution mechanism, which uses a `salience` value for determining order of rules to be fired. Rules with higher value have precedence to be executed.

XTT2 does not provide any rules properties directly. However, the ALSV(FD) logic defines the decision component (table) as follows:

$$t = (r_1, r_2, \ldots, r_n)$$

This means that rules placed in an XTT2 table are ordered. The inference engine uses this order for determining precedence of rules evaluation and execution. This precedence can be changed by moving rules in the table.

This behavior corresponds to CLIPS `salience` rule property. However, XTT2 forces the different values of rules priority in contrast to CLIPS that allows for rules with the same priority. This is why, the XTT2 method do not provide conflict resolution strategies.

4.4 Rule processing

Facts maintenance Any modification of KB in CLIPS that is done by using commands like `assert`, `retract` and `modify`, executes a pattern-matching algorithm which attempts to match rules to the current state of the system (as represented by the fact-list and instance-list). Each rule that has satisfied their conditional part (LHS – *Left Hand Side*) with respect to the modification is activated for execution. CLIPS allows non monotonic inference because each rule firing may again modify the KB. This inference process continues while KB is being modified. During this time, any rule can be activated and executed many times.

As it was mentioned, the XTT2 knowledge base contains a constant number of attributes (facts). The only modification that can be made is changing of the attribute value. However, in contrast to CLIPS, such modifications of the KB do not execute pattern-matching algorithm in order to find the rules that have satisfied their conditional parts against to a new system state.

This behavior is deliberate and follows from the method assumptions. According to this assumptions, the user defines the functional dependencies between attributes (links between tables). Thus, if a rule should be checked for execution when a value of an attribute is changed, then a user must define an appropriate dependency. This allows for optimized rule activating and more advanced inference control in comparison with CLIPS.

5 Summary

The main focus of this paper is to compare XTT2 with the CLIPS language. The scope of the provided comparison covers only the basic CLIPS language elements. In fact, the CLIPS language provides fully object oriented syntax called *CLIPS Object Oriented Language* (COOL). However, in the context of this paper the COOL syntax has not been taken into account. This paper tries to identify differences between these two languages in terms of the following aspects:

- Rule Base — differences related to knowledge maintenance and representation,
- Rule syntax — comparison of the languages expressiveness,
- Rule semantics — differences in knowledge interpretation,
- Rule processing — issues related to different knowledge evaluation and processing.

As it can be concluded from this paper, expressiveness of the XTT2 language (in comparison with CLIPS) is limited in each of the considered aspect. What is more, this paper shows that the ALSV(FD) logic, on which XTT2 is based, has also several limitations. On the other hand, in contrast to CLIPS, the XTT2 language provides strong underlying formalism playing a key role in rule interoperability. Due to the fact that CLIPS language is only a programming solution, a definition of an efficient CLIPS-based knowledge interchanging cannot be done. This is why, the extension of both the ALSV(FD) logic and XTT2 is a must in order to define an unified and formalized knowledge interoperability method. This extended formalism will allow for preserving rule semantics during interchanging. What is more, this method is intended to be supported by tools.

We selected the CLIPS language because it is considered to be successful in the Artificial Intelligence research community and have been used for many AI software projects. What is more, similarly to CLIPS, the XTT2 language is intended to be rule-based systems modeling method in their classic form. On the other side, the current application of rules (Business Rules) differs from the classic systems. One of the most important difference lies in different rule types. The classic systems usually provide only one rule type called *production rule*, while the BR-based languages provide five rule types: *Denotic Rules*, *Derivation Rules*, *Integrity Rules*, *Reaction Rules* and *Transformation Rules*. This rule classification is based on the specific rule properties (e.g. monotonicity of KB modification) and purposes (e.g. reaction on events). We do not discuss the differences between these types in details, because this is out of scope of this paper. These five types of Business Rules slightly extend the semantics of the *production rules*. Nevertheless, each type of Business Rule can be represented in classic rule-based systems using the *production rules*. This is why, despite the classic nature of CLIPS or XTT2, these languages can also be used for BR modeling.

The mentioned in Section 2. methods for rule interoperability (e.g. RIF) try to take rule properties and purpose into account. This is why, such a language is divided into so called dialects. RIF provides two standard dialects for rule representation: BLD (Basic Logic Dialect) and PRD (Production Rules Dialect). In general, the BLD and PRD dialects divide rules into two types: allowing for monotonic and non-monotonic changes in the Knowledge Base. In terms of BR types, usually the *Derivation*, *Denotic* and *Transformation rules* can be expressed in the BLD dialect while remaining in the PRD dialect.

Working on extension of XTT2 and ALSV(FD), the different types of rules will be taken into account and different formalisms will be provided. We assume, the unified rule representation model will be based on the Attributive Logic. However, this issue will be elaborated in details in the future work.

REFERENCES

[1] Weronika T. Adrian, Szymon Bobek, Grzegorz J. Nalepa, Krzysztof Kaczor, and Krzysztof Kluza, 'How to reason by HeaRT in a semantic knowledge-based wiki', in *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2011*, pp. 438–441, Boca Raton, Florida, USA, (November 2011).

[2] Grigoris Antoniou and Frank van Harmelen, *A Semantic Web Primer*, The MIT Press, 2008.

[3] Harold Boley, Said Tabet, and Gerd Wagner, 'Design rationale for ruleml: A markup language for semantic web rules', in *SWWS*, eds., Isabel F. Cruz, Stefan Decker, Jérôme Euzenat, and Deborah L. McGuinness, pp. 381–401, (2001).

[4] Paul Browne, *JBoss Drools Business Rules*, Packt Publishing, 2009.

[5] Tim Denvir, Jose Oliveira, and Nico Plat, 'The Cash-Point (ATM) 'Problem'', *Formal Aspects of Computing*, **12**(4), 211–215, (Dec 2000).

[6] Joseph Giarratano and Gary Riley, *Expert Systems. Principles and Programming*, Thomson Course Technology, Boston, MA, United States, 4th edn., 2005. ISBN 0-534-38447-1.

[7] Adrain A. Hopgood, *Intelligent Systems for Engineers and Scientists*, CRC Press, Boca Raton London New York Washington, D.C., 2001.

[8] Michael Kifer and Harold Boley, 'RIF overview', W3C working draft, W3C, (October 2009). http://www.w3.org/TR/rif-overview.

[9] *The Handbook of Applied Expert Systems*, ed., Jay Liebowitz, CRC Press, Boca Raton, 1998.

[10] Antoni Ligęza, 'Expert systems approach to decision support', *European Journal of Operational Research*, **37**(1), 100–110, (1988).

[11] Antoni Ligęza, *Logical Foundations for Rule-Based Systems*, Springer-Verlag, Berli, Heidelberg, 2006.

[12] David Luckham, 'Complex event processing (CEP)', *Software Engineering Notes*, **25**(1), 99–100, (2000).

[13] Grzegorz Nalepa, Szymon Bobek, Antoni Ligęza, and Krzysztof Kaczor, 'Algorithms for rule inference in modularized rule bases', in *Rule-Based Reasoning, Programming, and Applications*, eds., Nick Bassiliades, Guido Governatori, and Adrian Paschke, volume 6826 of *Lecture Notes in Computer Science*, pp. 305–312. Springer, (2011).

[14] Grzegorz Nalepa, Szymon Bobek, Antoni Ligęza, and Krzysztof Kaczor, 'HalVA - rule analysis framework for XTT2 rules', in *Rule-Based Reasoning, Programming, and Applications*, eds., Nick Bassiliades, Guido Governatori, and Adrian Paschke, volume 6826 of *Lecture Notes in Computer Science*, pp. 337–344. Springer, (2011).

[15] Grzegorz J. Nalepa, 'Architecture of the HeaRT hybrid rule engine', in *Artificial Intelligence and Soft Computing: 10th International Conference, ICAISC 2010: Zakopane, Poland, June 13–17, 2010, Pt. II*, eds., Leszek Rutkowski and [et al.], volume 6114 of *Lecture Notes in Artificial Intelligence*, pp. 598–605. Springer, (2010).

[16] Grzegorz J. Nalepa, *Semantic Knowledge Engineering. A Rule-Based Approach*, Wydawnictwa AGH, Kraków, 2011.

[17] Grzegorz J. Nalepa and Antoni Ligęza, 'A graphical tabular model for rule-based logic programming and verification', *Systems Science*, **31**(2), 89–95, (2005).

[18] Grzegorz J. Nalepa and Antoni Ligęza, 'HeKatE methodology, hybrid engineering of intelligent systems', *International Journal of Applied Mathematics and Computer Science*, **20**(1), 35–53, (March 2010).

[19] Grzegorz J. Nalepa, Antoni Ligęza, and Krzysztof Kaczor, 'Formalization and modeling of rules using the XTT2 method', *International Journal on Artificial Intelligence Tools*, **20**(6), 1107–1125, (2011).

[20] Grzegorz J. Nalepa, Antoni Ligęza, Krzysztof Kaczor, and Weronika T. Furmańska, 'HeKatE rule runtime and design framework', in *Proceedings of the 3rd East European Workshop on Rule-Based Applications (RuleApps 2009) Cottbus, Germany, September 21, 2009*, eds., Adrian Giurca, Grzegorz J. Nalepa, and Gerd Wagner, pp. 21–30, Cottbus, Germany, (2009).

[21] Pascal Poizat and Jean-Claude Royer, 'Kadl specification of the cash point case study', Technical report, IBISC, FRE 2873 CNRS - Universite d'Evry Val d'Essonne, France, Genopole Tour Evry 2, 523 place des terrasses de l'Agora 91000 Evry Cedex, (January 2007).

[22] Barbara von Halle, *Business Rules Applied: Building Better Systems Using the Business Rules Approach*, Wiley, 2001.

[23] G. Wagner, A.Giurca, and S. Lukichev, 'R2ml: A general approach for marking up rules', in *Principles and Practices of Semantic Web Reasoning, Dagstuhl Seminar Proceedings 05371*, eds., F. Bry, F. Fages, M. Marchiori, and H. Ohlbach, (2005).

Template-based Extensible Prototyping for Creativity- and Usability-Oriented Knowledge Systems Development

Martina Freiberg and **Frank Puppe** [1]

Abstract. In knowledge-based systems (KBS) development, there still is a lack of research regarding user interface (UI) design and (usability) evaluation. Thus, especially KBS UIs still often are developed in a rather ad hoc manner, lacking reusability of proven solutions and potentially valuable experimentation with design alternatives and their thorough evaluation. We propose the tailored KBS prototyping and engineering tool *ProKEt* for practically supporting *Template-based Extensible Prototyping*, a technique for more efficient, affordable, and UI design/usability evaluation oriented KBS development. Further, we report current projects where both the approach and the tool provided valuable support.

Keywords: Knowledge-based System, Knowledge System Engineering, Extensible Prototyping, UI Design, Usability Evaluation

1 Introduction

Knowledge-based systems (KBS) engineering still constitutes an effortful, expensive task in terms of development time and costs; also, the focus often is on knowledge base development whereas UI design, creativity/experimentation, or even formal usability evaluation are considered rather lower priority task—if considered at all. Probably due to the numerous benefits of web-based systems, an increasing number of knowledge-based/expert systems seems to be developed for the web. However, such systems apparently often are being developed for quite specialized contexts in a rather ad hoc manner and not (re)using (neither providing) any patterns or best practices especially regarding the UI/interaction design. Amongst the reasons for this may be the lack of research—c.f. Duan et al. [9]—and tool support for encompassing KBS development, i.e., particularly integrating UI design and usability evaluation. An important premise for creative KBS (UI) development, for reusability of existing solutions and their usability-related evaluation is the availability of an affordable development methodology and tool. With regards to general KBS development there exist various software tools—such as JavaDON [15], or KnowWE [6]—as well as development methodologies—e.g., CommonKADS [14], or the Agile Process Model [5]. Yet, such approaches mostly focus on knowledge base design and evaluation. In contrast, we propose *ProKEt* as tailored development tool for web-based KBS that seamlessly couples agile KBS development—with particular focus on UI/interaction design—with semi-automated usability evaluation activities; therefore, the tool particularly supports *Template-based Extensible Prototyping*—a tailored form of evolutionary prototyping—and fosters reuse of existing KBS solutions. Concerning usability evaluation—specifically collecting click log data—there exist a vast range of both research-based and commercial tools; however, those mostly need to be separately installed and configured. In contrast, ProKEt seamlessly integrates appropriately tailored evaluation mechanisms.

In Section 2, we propose *Template-based Extensible Prototyping* in more detail. We then introduce the KBS engineering tool *ProKEt* in Section 3 for practical support of the described, tailored prototyping approach for KBS. In Section 4, we report experiences with the approach and the tool during current projects. We conclude with a summary of the presented research and an outlook on prospective future work in Section 5.

[1] University of Würzburg, Germany, email: freiberg/puppe@informatik.uni-wuerzburg.de

2 Template-based Extensible Prototyping

Evolutionary prototyping—see e.g. [7]—in particular evolves mature prototypes continuously into productive systems; yet the process, until a productive stage is reached may be quite lengthy.

Template-based Extensible Prototyping (TEP) We propose Template-based Extensible Prototyping (TEP) as a tailored form of online evolutionary [7] prototyping that additionally (re-)uses certain template or pattern sets for accelerating development. In contrast to basic evolutionary prototyping, TEP particularly focusses on the anytime production of functional systems. TEP basically consists of the two stages *pure prototyping*, and *productive prototyping*; consequently, it results in two types of prototype artifacts: An interactive, potentially slightly stripped-down user interface prototype (pure prototype), that can be transferred into an entirely productive, non-prototypical system with no effort. In the context of KBS, we think of pure prototypes as a specific excerpt of the system that mirrors only the core KBS specific UI and interactions, but not yet contains general required functionality such as session persistence or login mechanisms. In the productive prototyping stage, the pure prototype is transferred into a productive system by associating it with the respective knowledge base and aforementioned add-on functionality. For a detailed introduction of basic Extensible Prototyping and how it can be integrated with agile KBS development, see [12]. The additional usage of proven KBS solutions in the form of templates further enriches Extensible Prototyping by fostering efficiency and affordability as copying & and adaption/extension can be exploited. Templates thereby are applied directly at the pure prototyping stage when developing the UI of the prototype and future system, respectively. The range of templates should encompass more generic, system-level templates—e.g. for the entire framing UI design—to fine granular templates—e.g. for single UI elements such as buttons or the representation of questions and their answer alternatives. We propose a set of (system-level) templates derived

from practical project experiences in the next section. Besides from UI templates, knowledge patterns—for creating the knowledge base, such as proposed in [13]—are an opportunity for further leveraging the overall KBS development process.

Due to the application of reusable UI/KBS templates where reasonable and due to the deliberate exclusion of certain system aspects, pure prototyping becomes an affordable and straightforward task—even the more when TEP-tailored tools such as *ProKEt*, see Section 3, are available. Thus, it particularly supports the development of multiple alternative KBS prototypes in parallel and/or to develop in a highly iterative manner. Also, a more creative, experimental KBS design process is fostered, as e.g. novel KBS UI forms can be experimentally tried out while there is no need to deal with selecting—or newly developing—the appropriate required knowledge representation immediately. It can be argued, that template-based development and using a specific and thus potentially restricted tool could rather hinder than unfold creativity; there, we argue that it is no strict prerequisite to always make use of all or even any existing templates, but they are more to be seen as additional option to accelerate development in cases where system requirements and framing conditions are similar. Moreover, we claim it a major important feature of such template sets to be assembled of modular entities that built on each other and can be most easily extended; this allows for reusing just the templates that match the given requirements (and save time and efforts) and to get creative with other parts. Regarding template selection, this is currently intended as a manual process, depending on the project requirements and on the experiences of the knowledge engineer; however, we also plan to further enrich the approach with a template selection KBS which could—based on some entered framing properties—propose and setup the most appropriate template for a given context. Further, the affordability of frequent iterations supports usability-oriented development both implicitly and explicitly. Implicitly, as iterative development most often naturally detects shortcomings and flaws of the system which are more likely to be refined the more development iterations are performed. Explicit usability support is provided, as it becomes possible to create several alternative pure prototypes—which, as described above, exhibit a mature UI and the core interaction—and to formally evaluate them in a straightforward manner under quite realistic framework conditions. Due to the possibility to create alternative prototypes by simply adding adapted/other knowledge bases to the pure prototype, both UI and knowledge base can be assessed and refined in a highly iterative and visual manner.

Exemplary KBS UI Templates Due to practical experiences in past and current KBS projects, several system-level templates for web-based KBS could be identified. The **Questionary** style displays questions in resemblance to paper-based questionnaires. Two exemplary realizations of the Questionary template are shown in Figure 1, A (1-column style) & B (3-column style). For a more compact UI, the **Daily** template was developed; an exemplary 3-column Daily prototype is depicted in Figure 1, C. There, questionnaires and their included questions form a column-wide, visual entity similar as in common newspapers. Both Questionary and Daily style can be applied for documentation KBS—where the focus is on collecting data uniformly and correctly—as well as for consultation KBS—that derive one or several solutions based on the user input provided for the questions. Questionary and Daily style are introduced somewhat more elaborately in [12]. As an example for an efficient, skill-building KBS UI, we propose the **iTree** template, particularly apt for clarification consultation KBS—i.e., systems, where only a single issue is rated. An exemplary implementation is shown in Figure 2, A. The core issue as well as the questions–a tailored form of yes/no questions with additional value neutral/uncertain—that determine the core issue rating are presented in a hierarchical, tree-like manner. The core issue rating is derived from its top-level questions—placed directly underneath the core issue and are interactively and recursively navigable. We refer to [10] for a more detailed introduction of the iTree. Also applicable for clarification KBS, yet also for multiplex consultation KBS—where one issue/solution out of a potentially extensive set of solutions is to be derived due to the provided input—is the **One-Question** template. An example is shown in Figure 2, B. It basically aims at closely imitating a conversation between the system and a user by always presenting only the one appropriate next question at a time. The intention of such a strict conversational style is to ease the interaction as that way the user can always fully concentrate on the current question at hand, letting the KBS guide the problem solving workflow. In [10] also more details on the One-Question style are given. Of the proposed templates, so far only iTree and One-Question contain explanation modules, i.e., parts of the UI where the results of the KBS session are displayed and explained—in iTree above the tree part and in One-Question above the main, conversational question display panel. This is mostly due to the fact, that Questionary and Daily style were so far only applied in the context of documentation KBS where no solutions/diagnoses/explanations are required; nevertheless, there exist rough, alternative Questionary prototypes that also include prototypical explanation modules realized, e.g., as additional side panels.

3 ProKEt: Practical KBS Development Support

ProKEt is a tailored **Pro**totyping and **K**nowledge systems **E**ngineering **t**ool for web-based documentation and consultation KBS; it additionally provides support for various usability evaluation activities and fosters Template-based Extensible Prototyping (TEP). Pure prototypes are constructed in ProKEt by simply specifying a certain template name—e.g. *oqd* for the One-Question template—when defining the prototype-knowledge in a tailored XML format; then ProKEt automatically selects the required system-level and sub-templates and assembles them into a KBS prototype (pure prototyping). Templates thereby are defined by using the StringTemplate [4] technique, whereas the specific design/styling of UI elements is mostly done by separate CSS; relevant core interactivity—e.g., value abstraction—which needs to be imitated in pure prototypes is realized by JavaScript and is included automatically. When switching to productive prototyping, the basic KBS framework remains the same, making productive prototyping as easy as linking a productive knowledge base and potentially slightly adapting the base specification regarding, e.g., the CSS to be used. ProKEt currently supports exclusively *d3web* [1] knowledge bases which allow for defining a vast range of knowledge representations, such as (heuristic) rules, decision trees, or set-covering knowledge. This straightforward pure-to-productive-prototyping switch is supported for a bunch of basic KBS templates—as summarized in the previous section—out of the box. Thus ProKEt allows for a straightforward and affordable prototyping and engineering process in cases where framing conditions and system requirements are similar. Yet, also creativity is fostered, as existing templates and/or style files can easily be adapted or even completely rewritten, whereas the ProKEt framework—that finally assembles prototypes and productive KBS and enriches them by the required interactivity—needs not to be altered normally. For a more

extensive introduction of particularly the agile prototyping and engineering process with ProKEt and a detailed description of the tool, see [12]. It has to be noted, that when used as a prototyping environment alone, ProKEt (is not intended to and) does not provide any way to create (*d3web*) knowledge bases. However, when additionally using the semantic wiki KnowWE [6] for knowledge base development, both UI front-end and KB back-end can be developed in a tightly interconnected manner: Changes made to the knowledge base in the wiki can directly be deployed to the ProKEt artifact by a simple button click, making the changes immediately visible in the UI, which in turn eases the direct investigation of the recent changes and the potentially resulting side-effects regarding the UI.

Regarding usability, ProKEt directly offers integrated functionality to perform usability evaluations. This fosters the seamless integration of more or less extensive or formal evaluations into the KBS development process. Therefore, ProKEt basically offers *quantitative and qualitative* data collection mechanisms, which can be added for both prototypes and productive KBS by a simple property in the knowledge specification. As a result, e.g. questionnaires are included within the prototype UI and/or click logging is activated. Thus, developers can setup and conduct various evaluation scenarios and assess the current development state in a favorable way any time. Regarding *quantitative data*, ProKEt provides a tailored, mouse click and keyboard event logging mechanism that records all relevant actions during KBS usage sessions. Based on that data, ProKEt furthermore automatically calculates a bunch of known usability metrics—such as *Success Rate*, or *Average Task Time*. For *qualitative data* collection, ProKEt supports both the integration of form-based questionnaires/surveys—standard measures as e.g. the SUS [8] are provided out of the box, yet own questionnaires can be integrated equally easily—and of anytime feedback—a mechanism for collecting free user feedback at any time during a KBS session. All recorded data—quantitative as well qualitative—can be exported to a standard CSV format for further processing e.g. in statistical software. For more details on ProKEt's usability extension, see [11].

4 Case Studies

Several current projects so far showed the general applicability as well as the value of the Template-based Extensible Prototyping approach and the ProKEt tool.

Mediastinitis The Mediastinitis Registry [3] is a german national project for improving patient care in a cardiac medical context. Therefore, certain medical data are collected and statistically evaluated as to develop appropriate future treatment strategies—for more details, see [12]. For best supporting data entry by the medical staff, a *knowledge-based documentation system* was implemented. Based on a first specification of the underlying knowledge, the first prototype—Figure 1, A—was created; based on that, ProKEt allowed for creating also the two alternative designs in a straightforward manner by just adapting the respective UI templates and style files, and linking them with the existing knowledge. Thus the entire KBS framework, that was working for the first prototype, was reused, which greatly shortened the development efforts required for the UI alternatives (shown in Figure 1, B & C). After selecting the prototype fitting the requirements and expectations of the medical doctors best—Figure 1, B—a productive knowledge base was created and included with the chosen prototype UI (productive prototyping stage). In the further course, one of the doctors from the project reviewed the respectively current prototype by entering exemplary cases; the required adaptions—both regarding the knowledge base and its representation in the UI—were made in a timely manner and the expert continued reviewing the adapted prototype; thereby, the possibility to adapt UI and knowledge base separately from each other, but immediately re-merge them into new productive KBS for further reviewing was particularly valuable. This highly iterative process allowed for detecting and removing several non-obvious flaws regarding both knowledge base and UI, and thus for improving the system's overall usability.

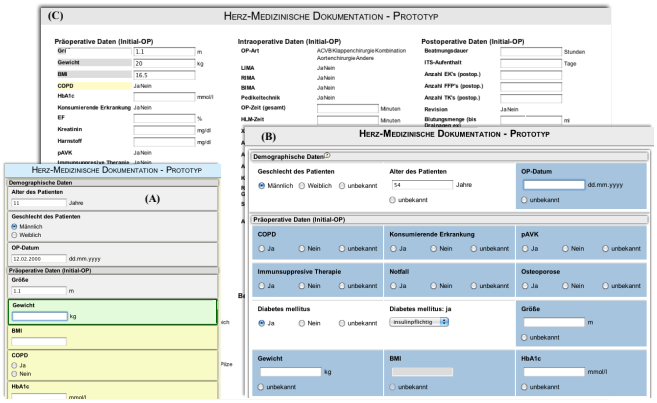


Figure 1. The three initial Mediastinitis prototypes (in german). 1-column questionary style (A); 3-column questionary style (B); daily style (C).

EuraHS EuraHS [2] is a project of the European Hernia Society (EHS) with the goal to improve patient care and increase knowledge regarding abdominal wall hernia surgery. Similar as in Mediastinitis, relevant data is to be collected and statistically evaluated; due to the similar basic framing conditions and application context, the first EuraHS prototype could be quickly built by (re)using the basic Mediastinitis prototype framework and just adapting the initial, exemplary knowledge specification. Based on that prototype, a first phase of iterative development began, where the expert participation remained passive, as he reviewed the respective prototypes and just reported what to refine. However, once the knowledge was transferred into a productive d3web knowledge base—starting the productive prototyping process—the expert was enabled to actively participate in the further development. This was possible due to the mechanism to immediately deploy adapted knowledge to the dialog system via the direct linkage between the knowledge base development tool KnowWE [6] and the dialog UI. This extensive expert participation was perceived highly beneficial as it led to a high satisfaction on the side of the expert due to his active involvement and resulting identification with the system; it further saved time and efforts, as on the one hand the expert knowledge was formalized in an unsophisticated manner and thus contained less flaws, and on the other hand the parallel development of KBS/UI (university team) and KB (expert) led to quicker overall results. The highly iterative process again enabled many KB and UI refinement cycles, thus enhancing the overall quality of the system. The final EuraHS implementation is quite similar to the final Mediastinitis system—c.f. Figure 1, B—however enhanced by several additional features including image questions (where answers can be selected visually) and a more comprehensive mechanism for flexibly fading in and out parts of the UI depending on already provided answers. For a more detailed description of EuraHS, see [12].

JuriSearch *JuriSearch* was started in 2012 as a cooperation between the university of Würzburg and the RenoStar corporation and aims at building a freely accessible, web-based knowledge-based system for the legal domain for various topics, such as right of cancellation or the law of tenantry. The target system is intended to integrate both a standard consultation (entrance) module—helping the user to preselect the specific problem definition—and various clarification modules for each potential problem—which then validate the concrete rating of that issue. Target users range from legal laymen—searching for a basic understanding/estimation of their case to (fresh) lawyers seeking for guidance regarding legal (sub)domain(s) that are not exactly their special field of work. So far, the focus lay on

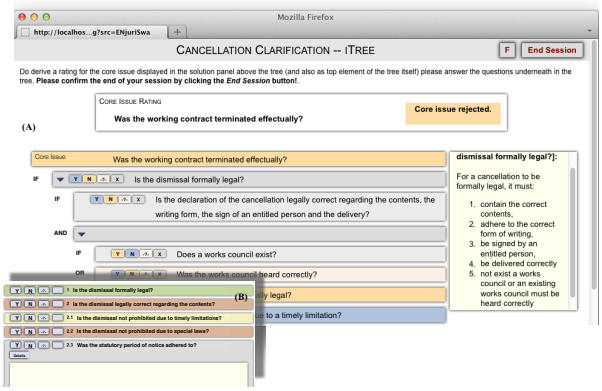


Figure 2. The two JuriSearch prototypes: interactively navigable iTree clarification style (A), and One-Question clarification style (B).

the clarification modules each of which rates exactly one distinct core issue, e.g. "Was the cancellation legally correct" (labour legislation domain). Initially, we experimented with two alternative yet distinct UI forms: An iTree implementation, depicted in Figure 2, A, and a One-Question UI, depicted in Figure 2, B. Therefore, first an *iTree* prototype was implemented based on a rough specification of the underlying knowledge. The possibility, to create various prototypes by simply exchanging the knowledge specification again proved valuable, as that way the prototypes could be reviewed highly iterative by a RenoStar staff member; this strongly supported the refinement and correction of both the underlying knowledge but also its most appropriate UI representation. ProKEt further allowed for creating the alternative One-Question UI in an affordable and timely manner in parallel to the iTree development. Based on those two alternative prototypes, so far several comparative assessments were performed. As first goal of the studies, it was assessed whether the iTree or the One-Question UI style were more suitable—if any—for the target context in general; there, the results of the studies indicate, that for the specific context of legal clarification consultation—a domain of highest expertise which needs to be mirrored adequately yet understandably by the KBS—the iTree is perceived more suitable and intuitively usable than the One-Question UI. Elaborate details on that study can be found in [11]. Furthermore, studies were conducted as to asses two distinct alternative knowledge base structures for the iTree style—one adhering to a legal specialist deduction scheme, the other specifically intended to provide more guidance and overview for legal laymen users; there, so far no significant distinction could be identified whether one scheme works better than the other. However, both the knowledge base as well as the UI could be drastically improved by refining them according to the respective insights and user comments gained in the user studies.

5 Conclusion

For leveraging the issue of a lacking integration of UI-related creativity and usability activities in KBS current development, we proposed *Template-based Extensible Prototyping* as KBS development technique that despite originally being developed specifically for the KBS domain may as well be applicable in general software engineering. For practical support of the approach, we introduced the KBS engineering tool ProKEt and we reported case studies that showed the applicability and value of the approach and tool. Regarding future work, current and upcoming projects raised the need for extending the collection of KBS classes and UI templates supported by ProKEt. Also, integrating mouse tracking mechanisms as addition to the existing click logging seems promising as to gain even more detailed insights regarding the UI usage evaluation. Equally, an automated, visual evaluation aid—that compares the solutions derived by the users with the correct solutions—could strongly support usability related evaluations. Further, a more formal classification of existing KBS types and respective suitable UI styles/interaction forms—e.g. in the form of a KBS pattern catalogue or also an interactive pattern selection KBS—could enrich the overall approach; thereby, the combination of UI templates/patterns and KB patterns [13] seems promising for encompassing, reusability-enabling KBS development.

REFERENCES

[1] http://d3web.sourceforge.net/ , last checked Jun. 1st, 2012.
[2] http://eurahs.drwontwikkeling.nl/, last checked Jun. 1st, 2012.
[3] http://www.dgthg.de/register, last checked Jun. 1st, 2012.
[4] http://www.stringtemplate.org/, last checked Jun. 1st, 2012.
[5] Joachim Baumeister, *Agile Development of Diagnostic Knowledge Systems*, IOS Press, AKA, DISKI 284, 2004.
[6] Joachim Baumeister, Jochen Reutelshoefer, and Frank Puppe, 'KnowWE: A Semantic Wiki for Knowledge Engineering', *Applied Intelligence*, **35**(3), 323–344, (2011).
[7] Michel Beaudouin-Lafon and Wendy Mackay, 'Prototyping Tools and Techniques', in *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, pp. 1006–1031, Hillsdale, NJ, USA, (2003). L. Erlbaum Associates Inc.
[8] J. Brooke, 'SUS: A quick and dirty usability scale', in *Usability evaluation in industry*, eds., P. W. Jordan, B. Weerdmeester, A. Thomas, and I. L. Mclelland, Taylor and Francis, London, (1996).
[9] Y. Duan, J. S. Edwards, and M. X. Xu, 'Web-based expert systems: benefits and challenges', *Information & Management*, **42**, 799–811, (September 2005).
[10] Martina Freiberg and Frank Puppe, 'itree: Skill-building user-centered clarification consultation interfaces (to appear)', in *KEOD 2012 - Proceedings of the International Conference on Knowledge Engineering and Ontology Development*, (2012).
[11] Martina Freiberg and Frank Puppe, 'Prototyping-based Usability-oriented Knowledge Systems Engineering', in *To appear in Proceedings of Mensch und Computer 2012*, (2012).
[12] Martina Freiberg, Albrecht Striffler, and Frank Puppe, 'Extensible prototyping for pragmatic engineering of knowledge-based systems', *Expert Systems with Applications*, **39**(11), 10177 – 10190, (2012).
[13] Frank Puppe, 'Knowledge Formalization Patterns', in *Proceedings of PKAW 2000, Sydney Australia*, (2000).
[14] Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Robert de Hoog, Nigel Shadbolt, Walter Van de Velde, and Bob Wielinga, *Knowledge Engineering and Management - The CommonKADS Methodology*, MIT Press, 2 edn., 2001.
[15] Bojan Tomic, Jelena Jovanovic, and Vladan Devedzic, 'JavaDON: an open-source expert system shell', *Expert Systems with Applications*, **31**(3), 595 – 606, (2006).

Towards Collaborative Knowledge Engineering for Improving Local Safety in Urban Environment

Antoni Ligęza and **Weronika T. Adrian** and **Przemysław Ciężkowski**[1]

Abstract. Web systems supporting collaborative knowledge engineering have attracted much attention recently. By using social software techniques and attractive yet simple user interface, the motivation of users increases and the process can be significantly improved. The willingness of community to invest their time as well as mutual encouragement can be achieved when users are convinced that their contribution is important and useful. We propose a social platform called Social Threat Monitor (STM) aimed at improving safety of local communities in urban environment. The main assumption of the system is the collaboration of users to build and maintain a knowledge base about threats in their neighborhood. Knowledge gathered in the system can be used by the citizens as well as local authorities and police. The system supports collaborative knowledge engineering and management using semantic methods and a GIS component.

1 Introduction

Web-based information systems are used for gathering, storing and processing diversified information for various purposes. In Web 2.0. era, users can actively participate in building such systems. Projects such as Wikipedia have shown that collaborative knowledge acquisition (KA) and management (KM) can be successful if people see the importance of the project and the KA process is relatively easy. Mechanisms such as voting, commenting and discussions increase the possibility of building reliable and useful knowledge bases. Semantic technologies enable adding metadata to regular content and facilitate automatic knowledge extraction and processing [5].

In this paper, we present a Web-based system for collaborative knowledge acquisition and management. It is a thematic portal which aims to gather knowledge about threats of various kinds in local environment. This information may be used by citizens as warnings and by local police as notifications. The system is being developed within the INDECT project: "Intelligent information system supporting observation, searching and detection for security of citizens in urban environment" [2]. The original contribution of this paper consists in presenting the collaborative knowledge engineering (KE) possibilities including knowledge exchange with external sources with use of a dedicated Application Programming Interface (API).

The paper is organized as follows: In Sect. 2 the motivation for this research is given. Sect. 3 provides an overview of the system functionality, user interface and implementation. Mechanisms applied to facilitate KE with the system are discussed in Sect. 4. The API of the system is presented in Sect. 5. Related work is outlined in Sect. 6, followed by a summary in Sect. 7 and future work in Sect. 8.

[1] AGH University of Science and Technology, Poland, email: {ligeza,wta}@agh.edu.pl

[2] See `http://indect-project.eu`.

2 Motivation

The aim of the Task 4.6. of the INDECT project is to develop a system for distributed knowledge acquisition and management with GIS [9] integration. The research is motivated by a hypothesis that local communities can effectively collaborate to build a useful knowledge base about threats in the neighborhood that can be used by both the citizens and local services or police. A system promoting collaborative knowledge acquisition and management should improve the communication between the citizens and the services, encourage cooperation within the community and thereby improve the local safety.

A social software platform facilitate collaborative knowledge engineering in an unintrusive way. Pieces of information shared by different persons are insignificant alone, but connected make a rich diversified picture. In popular social software platforms, people build personal knowledge bases half-consciously by acknowledging things and events shared by friends or "followed" people. We want to leverage this dynamics and develop a system that would provide useful information while seamlessly integrating with daily life.

Within the INDECT project, several prototypes have been developed [3, 8], each of which constitute a information silo Web-based application. We claim that it is necessary to extend the existing prototypes to be more flexible and better adapted to knowledge interoperability. Knowledge gathered in the system should be easily exchanged with other applications that can use its data to process it in an arbitrary way (custom notifications, aggregation, statistics).

3 System Overview

The main goal of the system is to serve as a distributed knowledge acquisition system for data, information and knowledge provided by citizens, as well as to enable knowledge management and exchange. Principles and a conceptual model of the system have been described in [4]. The general idea of the system can be observed in Figure 1. The input data, in general, may be composed of: a text description of a threat, its spatial location, and multimedia documentation. The data, stored in a relational database equipped with spatial features, should be presented to the audience in a combined visual and textual form. The system provides means for searching, filtering, aggregation and grouping information for users, according to their preferred form and level of detail. The threats can be presented in a convenient and transparent way as icons on the map, in reports or notifications.

To enhance the automated knowledge processing of the system, semantic technologies for GIS were analyzed and discussed in [6]. The semantic research thread led to the development of a prototype described in [8] which investigates the integration issues of databases and ontologies. In the ontology, the general categories of threats were stored, whereas in the database the actual data about selected areas in

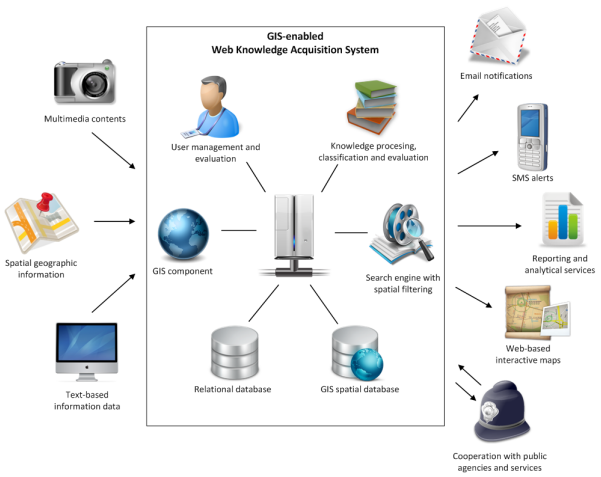


Figure 1. Conceptual model of the system [3].

particular time were located. This prototype provided interesting insights and ideas for future investigations. However, for the INDECT purposes, more lightweight semantics and reasoning has been chosen. Three systems, referenced in [3] use lightweight reasoning and metadata annotations of threat such as simple tags. In the newest prototype [2], codenamed Social Threat Monitor (STM), only basic semantics is added with use of tags and categories (see Section 4). Summary of improvements with respect to the previous implementations can be found in [1].

The following groups of users are defined in the system:

Guest is a user with the anonymous web account. He is able to use basic features of the application. In order to gain more privileges, a guest need to register and log in into the system.

Member is a user with registered account in the system. With this account user can add threats, manage his own threats, comment and vote threats of other users and edit his profile.

Services User is a special account with features helping threats monitoring.

Moderator is a user with full access to threat records, able to ban users.

Administrator is a user with full access to the application.

Main part of application is the map (see Fig. 2) that covers all space user have and resize immediately when needed. The interactive map implemented with use of AJAX technology – Asynchronous JavaScript and XML – provides easy and quick reloading as few times as possible. Because the URL does not change when only partial reloading is done, the identification of the state of the map (visible location or threat) is done with *hashing* (every time map position is changed or other action is performed, the hash changes).

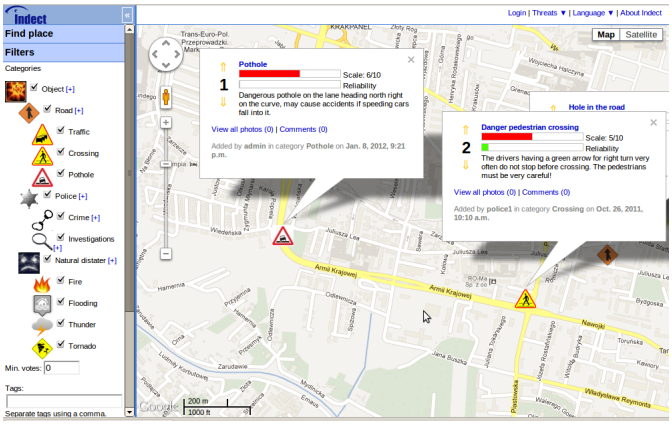


Figure 2. User Interface of the Social Threat Monitor.

The map have two editing modes: for browsing and adding threats. Mouse events are treated adequately to the mode. The map locates the user's position, but it can be moved (with mouse or keyboard) and zoomed (with mouse scroll or scroll on the left on the map).

Users browsing the system can immediately view short threat details (votes, one picture if available, number of comments and quick edit link). Voting is available only for registered users, but anonymous are able to see how many votes the threat got. Full gallery, list of tags and all comments are available on new page for each threat.

Top menu toolbar allows the user to toggle options of the login panel. If a user is already logged in, the login link is replaced with his account name and five options: 1) Profile (where the user can change account settings), 2) Logout, 3) Threats (where the user can list all threats, last added threats, top reliability threats and most dangerous threats), 4) Language (which allows user to change site language) and 5) About Indect – which is a link to the INDECT project page[3].

Most of the functionality is available form the left menu: (1) adding threats, including selecting area for a new threat, (2) browsing map by location, and (3) searching for threats using various filters. Each section of the left menu can be shown or hide.

The system has been implemented using widely-accepted, cost-free Web technologies: HTML, CSS, JavaScript, jQuery, Google Maps API version 3 and the Django framework (for details see [2]). It has been deployed on a dedicated server and is available at: `http://vigil.ia.agh.edu.pl`. In the wiki system there is a short description of the system, as well as user and admin manual.

4 Towards Collaborative Knowledge Engineering

Semantic annotations for the threats The basic piece of knowledge in the system represents a single *threat*. Each threat may be described using a set of attributes, such as: *geometry* – the information about the shape and the location of the danger, *name*, *category* – each danger is assigned to one category, *comments* – all logged in users may post their comment on the danger info, *severity* – a number telling how severe the danger is, *reliability* – a number telling to what degree the information is reliable, *photo gallery* – a relation to an object being a set of pictures illustrating the threat, *date added* – when the information has been added into the system, *modification date* – when the information has been last modified, and *tags* – in order to search for interesting information.

Tags Tags are non-hierarchical keyword or terms that describe an object, specifically a threat. One threat can be assigned several tags. They help searching and categorizing threats. Threats are tagged while adding by user. Well tagged threats are more reliable for other users and usually get more votes.

Categories Categories constitute a hierarchical way of describing threats. One threat has one category, but categories can contain many threats. Categories are organized into a tree structure. The root of the categories is not visible on site. There is no children limit for

[3] `http://www.indect-project.eu/`

categories. As relational databases are not designed for storing hierarchical data, retrieving category and its all parents with use of procedural SQL is complicated. Also, fetching the whole tree requires additional operations after database query execution. To solve this problems in STM, Nested Set Model was implemented through `django-mptt` module. It provides an efficient way to retrieve categories from database. Modifying the tree is more complicated, thus slower, but it is only possible for admin user and rarely executed.

User Groups Except for categories for threats, the users can also be grouped. Groups allows to publish threats for specified users. When adding a threat, a user (who is at least in Services group) can decide if the threat will be public or visible only for selected groups. This solution allows special groups have their own threats that will never be published for all system users.

An exemplary use-case has been presented in [1]. In this use-case, three police departments cooperate on an investigation. However, each group has their own sub-investigation and landmarks important to these investigations can be marked on map and visible only to selected users.

5 Towards Interoperability: the System API

Parts or the system data can be imported and exported to JSON, XML and YAML formats. In order to enable export of knowledge for further custom processing and import from another knowledge base, a simple Application Programming Interface (API) has been developed. Standarized knowledge representation using *attribute-value* pairs describing threats allows for using the system knowledge in various semantic applications (where triples of the form: *object-attribute-value* or *subject-predicate-object* are used). External systems can communicate with STM and use it as a web service. The API is available over HTTP protocol: `http://application.url/api/method_name/`.

API access and exchange format Part of the functionality of the Social Threat Monitor is available for external applications without authorization. For instance, by preparing an appropriate request conforming to the system API one can get all the threats defined for a given location or filter. In order to use the whole functionality, including adding threats to the system, the application must be authorized. Its user must be defined in the STM and assigned to the `API` group.

The API uses POST requests and HTTP cookies. All responses are in JSON format. Each successful request returns HTTP 200 header and the 400 header is returned, if a method does not exist. Moreover, if an anonymous user wants to access a method that requires authorization, a HTTP 200 header with appropriate content is returned.

Methods The system API provides three basic methods: 1) logging in, 2) adding a new threat, and 3) retrieving existing threats. Each method accepts arguments that must be sent using POST or COOKIES. Below, the methods with required (marked with an asterisk "*") and optional parameters are presented:

1. Method **login**:

   Description: A method allowing to log into the API.

   Parameters:

   - POST:
     - *username:`string` – user's name.
     - *password:`string` – user's password.

2. Method **add**:

   Description: A method allowing to add a threat.

   Parameters:

   - POST
     - *title:`string`
     - *description:`string`
     - *latitude:`float`
     - *longitude:`float`
     - *category:`int`
     - *scale:`int` from range $[1, 10]$ (severity of the threat)
     - *date_end:`int` from range $[1, 3]$ (the number indicates how many months a threat is active)
     - *tags:`string` – tags separated with comma
     - groups:`array(int)` – group ids for whom danger will be shown (empty for "all groups")
   - COOKIES
     - *sessionid:`string` – session id returned in login method

3. Method **threats**:

   Description: A method allowing to get filtered list of threats.

   Parameters:

   - POST
     - *polygon: `string` – string of coordinates that define the area of interest, for example: `lng1 lat1, lng1 lat2, lng2 lat2, lng2 lat1, lng1 lat1`
     - votes: `int` – minimal votes number.
     - images: `int` from set $\{0, 1\}$ where 1 selectes threats only with photos.
     - category: `int` – category id.
     - tags: `string` – tags separated by commas.
     - scale: `int` from set $\{1, 4, 7, 10\}$ indicating threats scale.
     - date: `string` from set $\{12h, 24h, week\}$.
     - date_start: `datetime` – threats added after date.
     - date_stop: `datetime` – threats ending before date.
     - groups: `array(int)` – threats for groups.

   An example response:

   Listing 1. Response to **threats** method on success.

```
{"threats": [
  {"category": 2,
    "user__username": "admin",
    "votes": 6,
    "scale": 5,
    "description": "Threat␣description",
    "point": {"latitude": 50.087389,
              "longitude": 19.891606},
    "title": "Crime",
    "category__img": "http://url/remont.png",
    "comments": 5,
    "date_add": "3␣kwietnia␣2011␣22:57:34",
    "points": 2,
    "category__title": "Crimes",
    "images": 7,
    "image__img": "http://url/name.jpg",
    "id": 1},
  {another_threat},
  {another_threat}]}
```

Demo implementation An example system using API has been developed and is available for testing at: `http://home.agh.edu.pl/kk/stm`. It has been implemented in PHP and uses cURL library. The library allows for connecting to and communicating with various types of servers and different protocols. An example screenshot showing STM response is presented in Figure 3.

Figure 3. API demo: A threat list returned upon request.

6 Related Work

Crime Mapping systems were originally a class of systems that map, visualize and analyze crime incident patterns using Geographic Information Systems (GIS). However, the name has been later extended to incorporate all applications that aid in improving the public safety. This include natural disasters monitoring systems, often designed for specific regions, which scope of functionalities is limited to the specific types of disasters that are most common and dangerous in those regions, systems monitoring threats on the roads and crime monitoring systems. A detailed survey of existing crime mapping systems is given in [10]. To the best of our knowledge, none of existing system works as a social platform supporting collaboration (voting, comments and collaborative evaluation of information).

7 Summary

One of the tasks within the INDECT Project is the development of a Web-based system for knowledge acquisition and management. Once the main assumptions and requirements were defined, the development has been done iteratively. Semantic description, categories and tags constitute the basis for further development of intelligent information processing and knowledge management. System API allows for easy integration with other applications and facilitate knowledge exchange and integration from various sources.

8 Future Work

The approach to semantics representation now used in the STM can be extended. Currently, only basic semantics is represented with use of tags and categories. They are closer to the model of folksonomies, where users provide custom tags that can be a foundation for a simple hierarchy of categories. A possible direction of future work is to refactor the hierarchy currently existing in STM with the use of a selected OWL 2.0 profile. All of the important relations should be identified and formalized. This will allow for having a complete formal model of the threat ontology. It is also planned to work on the rule-based engine [7] to manage and customize output channels.

Although the system works in a regular Web browser and thus can be accessed from any mobile device that has a browser, further adaptation for smartphones is planned. In particular, the system should use the GPS embedded in mobile devices to facilitate adding threats.

ACKNOWLEDGEMENTS

The research presented in this paper is carried out within the EU FP7 INDECT Project: "Intelligent information system supporting observation, searching and detection for security of citizens in urban environment" (`http://indect-project.eu`).

References

[1] Weronika T. Adrian, Ciężkowski, Krzysztof Kaczor, Antoni Ligęza, and Grzegorz J. Nalepa, 'Web-based knowledge acquisition and management system supporting collaboration for improving safety in urban environment', in *Multimedia Communications, Services and Security*, eds., Andrzej Dziech and Andrzej Czyżewski, volume 287 of *Communications in Computer and Information Science*, 1–12, Springer Berlin Heidelberg, (2012).

[2] Przemysław Ciężkowski, *Functionality Analysis and Design and Implementation of User Interface for Threats Enregistration in Internet System*, Master's thesis, AGH University of Science and Technology, 2011.

[3] Antoni Ligęza, Weronika T. Adrian, Sebastian Ernst, Grzegorz J. Nalepa, Marcin Szpyrka, Michał Czapko, Paweł Grzesiak, and Marcin Krzych, 'Prototypes of a web system for citizen provided information, automatic knowledge extraction, knowledge management and gis integration', in *Multimedia Communications, Services and Security*, eds., Andrzej Dziech and Andrzej Czyżewski, volume 149 of *Communications in Computer and Information Science*, 268–276, Springer Berlin Heidelberg, (2011).

[4] Antoni Ligęza, Sebastian Ernst, Grzegorz J. Nalepa, and Marcin Szpyrka, 'A conceptual model for web knowledge acquisition system with GIS component', *Automatyka: półrocznik Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie*, **13**(2), 421–428, (2009).

[5] Grzegorz J. Nalepa, 'Collective knowledge engineering with semantic wikis', *Journal of Universal Computer Science*, **16**(7), 1006–1023, (2010).

[6] Grzegorz J. Nalepa and Weronika T. Furmańska, 'Review of semantic web technologies for GIS', *Automatyka: półrocznik Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie*, **13**(2), 485–492, (2009).

[7] Grzegorz J. Nalepa and Antoni Ligęza, 'HeKatE methodology, hybrid engineering of intelligent systems', *International Journal of Applied Mathematics and Computer Science*, **20**(1), 35–53, (2010).

[8] Jarosław Waliszko, Weronika T. Adrian, and Antoni Ligęza, 'Traffic danger ontology for citizen safety web system', in *Multimedia Communications, Services and Security*, eds., Andrzej Dziech and Andrzej Czyżewski, volume 149 of *Communications in Computer and Information Science*, 165–173, Springer Berlin Heidelberg, (2011).

[9] *The Handbook of Geographic Information Science*, eds., John P. Wilson and A. Stewart Fotheringham, Blackwell Publishin Ltd, 2008.

[10] Maciej Żywioł, *Analysis and Evaluation of Crime Mapping Systems*, Bachelor's thesis, AGH University of Science and Technology, 2012.