6th Workshop on

Knowledge Engineering and Software Engineering (KESE6)

at the 33rd German Conference on Artificial Intelligence

September 21, 2010, Karlsruhe, Germany

Grzegorz J. Nalepa and Joachim Baumeister (Editors)

:-k(E),s(E).

Julius-Maximilians-UNIVERSITÄT WÜRZBURG

AGH

Technical Report No. 475, Würzburg University, Würzburg, Germany, 2010

Preface

Grzegorz J. Nalepa and Joachim Baumeister

AGH University of Science and Technology
Kraków, Poland
gjn@agh.edu.pl
—
Intelligent Systems (Informatik 6)
University of Würzburg
Würzburg, Germany
joba@uni-wuerzburg.de

Intelligent systems have been successfully developed in various domains based on 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 sixth workshop on Knowledge Engineering and Software Engineering (KESE6) was held at the KI-2010 in Karlsruhe, Germany, and brought together researchers and practitioners from both fields of software engineering and artificial intelligence. The intention was to give ample space for exchanging latest research results as well as knowledge about practical experience. Topics of interest includes but were not limited to:

– Knowledge and software engineering for the Semantic Web
– Ontologies in practical knowledge and software engineering
– Business rules design and management
– Knowledge representation, reasoning and management
– Practical knowledge representation and discovery techniques in software engineering
– Agent-oriented software engineering
– Database and knowledge base management in AI systems
– Evaluation and verification of intelligent systems
– Practical tools for intelligent systems
– Process models in AI applications
– Declarative, logic-based approaches
– Constraint programming approaches

This year, we received contributions focussing on different aspects of knowledge engineering: Kluza et al. present a declarative method of inference specification in modularized knowledge bases. Application of knowledge engineering in psychological processes is elaborated by Newo et al. The contribution of Freiberg et al. discusses practical issues of building novel interfaces for knowledge-based systems. Sagrado et al. review optimization techniques and focus on their application for the requirements selection problem in software releases.

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: Cañadas et al. introduce a practical application for rule-based applications integrated ontologies, using the Jess engine and the MVC approach.

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 KI 2010 conference for hosting the KESE6 workshop.

Grzegorz J. Nalepa
Joachim Baumeister

Program Committee

Workshop Organization

The 6th Workshop on Knowledge Engineering and Software Engineering
(KESE6)
was held as a one-day event at the
33rd German Conference on Artificial Intelligence (KI2010)
on September 21, 2010, Karlsruhe, Germany.

Workshop Chairs and Organizers

Grzegorz J. Nalepa, AGH UST, Kraków, Poland
Joachim Baumeister, University Würzburg, Germany

Programme Committee

Klaus-Dieter Althoff, University Hildesheim, Germany
Joaquin Cañadas, University of Almería, Spain
Uli Geske, FhG FIRST, Berlin, Germany
Adrian Giurca, BTU Cottbus, Germany
Jason Jung, Yeungnam University, Korea
Rainer Knauf, TU Ilmenau, Germany
Dietmar Seipel, University Würzburg, Germany
Ioannis Stamelos, Aristotle University of Thessaloniki, Greece
Gerhard Weiss, University of Maastricht, The Netherlands

Table of Contents

A Tool for MDD of Rule-based Web Applications based on OWL and SWRL

Joaquín Cañadas[1], José Palma[2] and Samuel Túnez[1]

[1] Dept. of Languages and Computation. University of Almeria. Spain
`jjcanada@ual.es, stunez@ual.es`
[2] Dept. of Information and Communications Engineering. University of Murcia. Spain
`jtpalma@um.es`

Abstract. TOOL PRESENTATION*: Rule languages and inference engines incorporate reasoning capabilities to Web information systems. This demonstration paper presents a tool for the development of rule-based applications for the Web based on OWL and SWRL ontologies. The tool applies a model-driven approach to an ontology representing a domain conceptualization and inference model of the problem domain. It automatically generates a rich, functional Web architecture based on the Model-View-Control architectural pattern and the JavaServer Faces technology, embedding a Jess rule engine for reasoning and deriving new information.

Key words: Model Driven Development, OWL, SWRL, Rule-based systems for the Web

1 Introduction

Rule languages and inference engines provide Web information systems with reasoning capabilities. Rule-based applications integrate rule engines to deal with rules and execute inference methods for firing appropriate rules in order to deduce new information and achieve results. Rules combined with ontologies enable the declarative representation of the expert knowledge and business logic in an application domain.

Web Ontology Language (OWL) [1] and Semantic Web Rule Language (SWRL) [2], which play a major role in the Semantic Web [3], are also growing in importance in software development [4]. Ontologies can describe the relevant concepts and data structures of an application domain and rule-based languages can be used to formalize the business logic, increasing the amount of knowledge that can be represented in ontologies.

Ontologies are created using authoring tools like Protégé. Recently, a tool to bridge the gap between OWL ontologies and Model Driven Engineering has

* This work was partially financed by the Spanish MEC through projects TIN2009-14372-C03-01 and PET2007-0033, and by the Andalusian Regional Government project P06-TIC-02411

been presented, the TwoUse Toolkit [3] [5]. It is a free, open source tool bridging the gap between Semantic Web and Model Driven Software Development. It has been developed using Eclipse Modeling Project[4] and implements current OMG and W3C standards for developing ontology-based software models and model-based OWL ontologies. Among its functionality, TwoUse provides a graphical editor for specifying OWL and SWRL models based on the Ontology Definition Metamodel (ODM) [6], and an textual editor for the OWL functional syntax [7]. Since it is deployed as an Eclipse plugin, other Eclipse-based tools for designing and executing transformations can be straightforwardly applied to ontologies created in TwoUse, enabling us to design an MDD process based on OWL and SWRL models.

This demo presents a tool which provides a model-driven approach to develop rule-based Web applications based on OWL and SWRL models created with TwoUse. The tool applies MDD to produce the implementation of a functional, rich Web architecture which embeds the Jess rule engine for inferencing tasks. The functionality for the generated rule-based Web application is predefined to enable end-users to create, retrieve, update and delete instances (CRUD). In contrast to current tools for automatic generation of CRUD systems that perform those functions on relational databases, our contribution is that this functionality is executed on the rule engine working memory, enabling the execution of a forward-chaining inference mechanism to drive the reasoning process.

This paper is organized as follows: Section 2 introduces the model-driven approach applied in the tool. Next, Section 3 describes the architecture for the rule-based Web application generated. Finally, the main conclusions and future work are summarized.

2 Model-driven process implemented in the tool

The proposed tool uses OWL and SWRL ontologies created with TwoUse as source models for the model-driven approach. We focus on *OWL DL* sublanguage of OWL. Classes, properties, and individuals define the structure of data, whereas rules describe logical dependencies between the elements of the ontology refereed in the rule's antecedent and consequent.

Figure 1 shows a simplified schema of the MDD process implemented in the proposed tool. Two different results are found from the single ontology model. On one hand, a Jess [8] rule base is generated, a text file that contains the rules converted to Jess syntax. On the other hand, a set of JavaBeans and JSP web pages, making up a JavaServer Faces (JSF) [9] architecture that embodies the Jess rule engine into the Web application.

Both MDD processes can be executed separately, enabling the decision logic of rule-based applications to be changed regardless of the ontology structure. When the rule model changes, the new rule base can be regenerated and deployed

[3] http://code.google.com/p/twouse/
[4] http://www.eclipse.org/modeling/

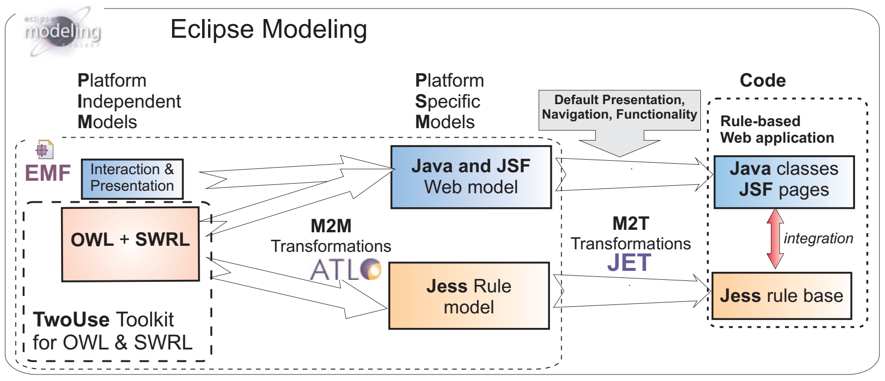


Fig. 1. MDD schema for rule-based Web system generation

into the Web application without affecting the full architecture. This approach makes Web applications easier to maintain and evolve.

The tool was developed using MDD tools provided in Eclipse, and it is supported by the TwoUse toolkit for the creation of ontology and rule models. Metamodels for representing platform-specific models in Jess and the Java/JSF are defined using EMF[5] (Eclipse Modeling Framework).

Model-to-model (M2M) transformations are designed with ATL[6] (Atlas Transformation Language). The first one (bottom flow in Fig. 1) maps an ontology and rule model to a Jess platform-specific model. The second one (top flow in Fig. 1) transforms the ontology model into a Java/JSF Web specific model.

The outputs of both ATL transformations are the respective inputs of two model-to-text (M2T) transformations implemented in JET[7] (Java Emitter Templates). As a result, the code for the rule-based Web application is obtained. On one hand, source files with Jess rules and facts, and on the other hand, the Web application components, the configuration files, the Java classes, and a JessEngine Bean which uses the Jess API (Application Programming Interface) to embed the rule engine into the architecture. Moreover, a set of JSP/JSF web pages is generated for the user interface which are based on the RichFaces library [10] framework that adds AJAX capability to JSF applications. Default configuration is injected to provide presentation templates for pages, predefined navigation between them and default functionality for the generated application.

3 Architecture of rule-based Web applications

Figure 2 shows the target architecture for the generated rule-based Web applications.

[5] http://www.eclipse.org/modeling/emf/
[6] http://www.eclipse.org/m2m/atl/
[7] http://www.eclipse.org/modeling/m2t/?project=jet

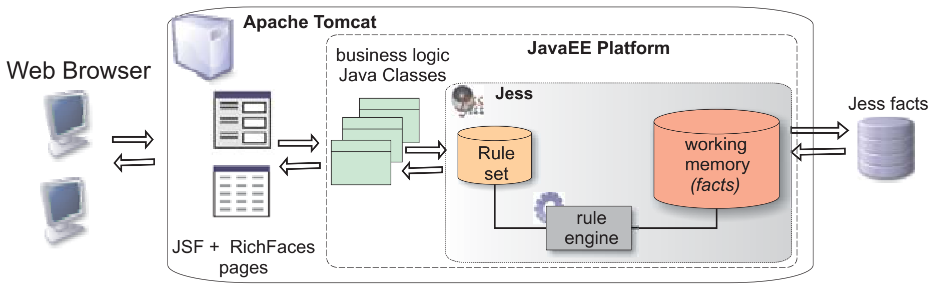


Fig. 2. Architecture of a Web rule-based application

The embedded rule engine manages the Jess rule base and the text file of persistent instances of concepts, called *facts*. Basically, the rule engine consists of three parts: the working memory contains all the information or *facts*, the rule set are all the rules, and the rule engine checks whether the rules match the working memory and executes them then.

The Web application enables the user to perform basic functions for instance management (create, list, update and delete instances). Current tools for automatic generation of that kind of Web applications perform those operations on relational databases. The contribution of our approach is that instance management is executed on the rule engine working memory. The rule engine executes a forward-chaining inference mechanism to drive the reasoning process, firing the rules with conditions evaluated as true, and executing their actions to infer new values or modify existing ones.

The use of both rules and AJAX technology improves instance management since each single value is validated and submitted as it is entered by the user, then the rule engine can fire suitable rules and deduce new information on the fly, driving the instance creation or edition.

4 Conclusion and future work

This tool presentation paper presents a tool that applies MDD to rich Web system development, incorporating a rule engine for deduction and inference. OWL and SWRL formalisms are used as modeling languages by the model-driven approach.

Since expressiveness in rule-based production systems such as Jess is poorer than OWL and SWRL semantics, only a subset of those formalisms is currently supported. Issues related with the combination of rules and ontologies have also an effect on the proposed approach. For example, the semantics of OWL and SWRL adopts an open world assumption, while logic programming languages such as Jess are based on a closed world assumption. As a consequence, only DL-Safe SWRL rules [11] are transformed to Jess. DL-Safe SWRL rules are a restricted subset of SWRL rules that has the desirable property of decidability, by restricting rules to operate only on known individuals in an OWL ontology. Similarly, Jess rules only operate on facts in the working memory.

Another semantic difference between OWL and classic rule engines such as Jess is related to the fact base and state assertions. While in OWL the ABox containing the assertions about the individuals in the domain can not be modified, on the other hand, in a rule-based systems facts can be asserted and modified during the inference.

Although these semantic differences are present, the proposed tool demonstrates how OWL and SWRL can be used as specification formalisms in rule-based Web applications development. The benefits come from the widely use of these formalisms and the many tools available for editing and reasoning over OWL and SWRL specifications.

The tool is planned to be evaluated in several domains such as pest control in agriculture and medical diagnosis. Future work extends the generated Web application with semantic Web functionalities, such as semantic search based on ontology classes hierarchy as well as instance search.

References

1. W3C OWL Working Group: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (2009) Available at http://www.w3.org/TR/owl2-overview/.
2. Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language combining OWL and RuleML. W3C Member Submission (2004) Available at http://www.w3.org/Submission/SWRL/.
3. Eiter, T., Ianni, G., Krennwallner, T., Polleres, A.: Rules and ontologies for the semantic web. In Baroglio, C., Bonatti, P.A., Maluszynski, J., Marchiori, M., Polleres, A., Schaffert, S., eds.: Reasoning Web. Volume 5224 of Lecture Notes in Computer Science., Springer (2008) 1–53
4. W3C: A Semantic Web Primer for Object-Oriented Software Developers. W3C Working Draft (2006) Available at http://www.w3.org/TR/sw-oosd-primer/.
5. Parreiras, F.S., Staab, S., Winter, A.: TwoUse: integrating UML models and OWL ontologies. Technical report, Universität Koblenz-Landau (2007)
6. Object Management Group: Ontology Definition Metamodel. Version 1.0. OMG (2009) Available at http://www.omg.org/spec/ODM/1.0/.
7. W3C OWL Working Group: OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Recommendation (2009) Available at http://www.w3.org/TR/owl-syntax/.
8. Friedman-Hill, E.: Jess in Action: Java Rule-Based Systems. Manning Publications (2003)
9. Geary, D., Horstmann, C.S.: Core JavaServer Faces. 2 edn. Prentice Hall (2007)
10. JBoss: RichFaces (2007) http://www.jboss.org/jbossrichfaces/.
11. Motik, B., Sattler, U., Studer, R.: Query Answering for OWL-DL with rules. Journal of Web Semantics **3**(1) (2005) 41–60

Visual Inference Specification Methods for Modularized Rulebases. Overview and Integration Proposal[⋆]

Krzysztof Kluza, Grzegorz J. Nalepa, Łukasz Łysik

Institute of Automatics,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
kluza@agh.edu.pl,gjn@agh.edu.pl

Abstract The paper concerns selected rule modularization techniques. Three visual methods for inference specification for modularized rulebases are described: Drools Flow, BPMN and XTT2. Drools Flow is a popular technology for workflow or process modeling, BPMN is an OMG standard for modeling business processes, and XTT2 is a hierarchical tabular system specification method. Because of some limitations of these solutions, several proposals of their integration are given.

1 Introduction

Rule-Based Systems (RBS) [1] constitute one of the most powerful knowledge representation formalisms. Rules are intuitive and easy to understand for humans. Therefore, this approach is suitable for many kinds of computer systems. Nowadays, software complexity is increasing. Because describing it using plain text is very difficult, many visual methods have been proposed. For example, in Software Engineering, the dominant graphical notation for software modeling is UML (the Unified Modeling Language). Design of large knowledge bases is non trivial, either. However, in the area of RBS, there is no single visual notation or coherent method. Moreover, the existing solutions have certain limitations. These limitations are especially visible in the design of large systems.

When the number of rules grows, system scalability and maintainability suffers. To avoid this, there is a need to manage rules. Rule grouping is a simple method of rule management. However, it is not obvious how to group rules. One of the most common grouping methods ivolves context awareness and creation of decision tables. Another grouping method takes rule dependencies into account and creates decision trees. This leads to RBS modularization.

This paper describes three possible solutions to modularize rule bases:

– Drools Flow [2], which is a popular technology for workflow modeling,

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

– BPMN (the Business Process Modeling Notation) [3], which is an OMG standard [4] for modeling business processes, and
– XTT2 (EXtended Tabular Trees), which is a result of authors' research project [5] and which organizes a tabular system into a hierarchical structure.

However, these solutions have some limitations. Drools Flow is platform-dependent and not standarized. Moreover, it has some flow design restrictions. BPMN is a notation for business processes, and it is not clearly stated how processes can co-operate with rules. Furthermore, BPMN can be mapped to BPEL (Business Process Execution Language) for execution, but this mapping is non-trivial and execution is not possible for every BPMN model. XTT2, in turn, is not wide-spread, and it is not a universal method.

The general problem considered in this paper is the RBS design and modularization. The article constitutes an overview and a proposal for integration of the three presented methodologies, which can be useful in solving above-mentioned problems. The next two sections present selected rule modularization techniques and an overview of selected visual design methods for rule inference. In Section 4, a proposal of rule translation from XTT2 to Drools is described, and in Section 5, a proposal of XTT2 inference design with BPMN is introduced. Section 6, discusses future work and summarizes the main threads of this article.

2 Rule Modularization Techniques

Most classic expert systems have a flat knowledge base. So, the inference mechanism has to check each rule against each fact. When the knowledge base is large, this process becomes inefficient. This problem can be solved by providing a structure in the knowledge base that allows to only check a subset of rules [6].

CLIPS [7] allows for organising rules into so-called *modules*, that restrict access to their elements from other modules. Modularisation of the knowledge base helps rule management. In CLIPS, each module has its own pattern matching network for its rules and its own agenda. Execution focus can be changed between modules stored on the stack.

JESS [8] also provides a module mechanism. Modules provide structure and control execution. In general, although any JESS rule can be activated at any time, only rules in the focus module will fire. This leads to a structured rule base, but still all rules are checked against the facts. In terms of efficiency, the module mechanism does not influence on the performance of *conflict set* creation.

Drools Flow provides a graphical interface for modelling of processes and rules. Drools 5 has a built-in functionality to define the structure of the rule base, which can determine the order of rule evaluation and execution. Rules can be grouped in ruleflow-groups which define the subsets of rules that are executed. The ruleflow-groups have a graphical representation as nodes on the *ruleflow* diagram. They are connected with links, which determines the order of evaluation. Rule grouping in Drools 5 contributes to the efficiency of the ReteOO algorithm, because only a subset of rules is evaluated. However, there are no policies which determine when a rule can be added to the ruleflow-group.

3 Selected Visual Design Methods for Rule Inference

Efficient inference would not be possible without a proper structure and design of rule-based system. The important issues in dealing with this problem is grouping and hierarchization of rules, as well as addressing the contextual nature of the rulebase. The following subsections describe selected methods and tools, in which visual design of rule inference is possible.

3.1 Drools

Drools is a rule engine which offers knowledge integration mechanisms. The project is run by the JBoss Community, which belongs to the Red Hat Foundation. It is divided into four subprojects: Guvnor, Expert, Flow and Fusion. Each of them supports different part of integration process.

Expert is the essence of Drools. It is the actual rule engine. It collects facts from the environment, loads the knowledge base, prepares the agenda and executes rules. A modified version of the Rete [9] algorithm is used for the inference.

The *knowledge base* in Drools consists of three main elements: rules, decision tables and Drools Flow. The fundamental form of knowledge representation in Drools is a rule. This form is easy to use and very flexible. Rules are stored in text files which are loaded into the program memory by special Java classes. Rules in Drools can be suplemented with attributes which contain additional information. They have a form of name-value pairs and they describe such paramters as rule priority and provide meta information for inference engine.

Rules which have the same schema can be combined into *decision tables*. A decision table is devided into two parts: the *left-hand side*, which represents the conditions of rules and the *right-hand side*, which represent the actions to be executed. One row in a table corresponds to one rule. However, decision tables, are useful only during the design phase. The structure does not improve the performance of the inference. Decision tables are, in fact, transformed into rules. So the inference engine does not recognize which rules come from decision tables and which are just a group of unrelated rules.

Rules form a flat structure. When the inference engine matches rules against facts, it takes all rules into consideration. The user, however, can define the flow of the inference process. Drools Flow offers a workflow design functionality in the form of blocks (See Fig. 1). The user can specify exactly which rules should be executed in which order and under which conditions.

Each model in Drools Flow has to contain two blocks: *start* and *end*. Rules and rule flow are linked together inside the *ruleset* block. Each *ruleset* block has a *ruleflow-group* attribute. Similarly, each rule has the attribute with the same name. Rules belong to the ruleset block with the same values of the ruleflow-group attribute. Additionally, the process can be split and joined. Two blocks, *split* and *join*, are used for that purpose. The block *split* has different types. The AND type defines that the process follows all the outgoing connections. The *join* block also has different types. The AND type waits for all the incomming subprocesses to finish. The OR type waits for the first process to finish, while the n-of-m type waits until specified number of processes finish.

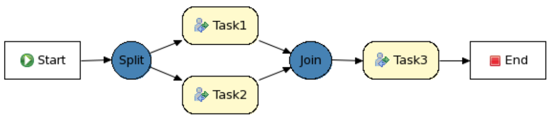
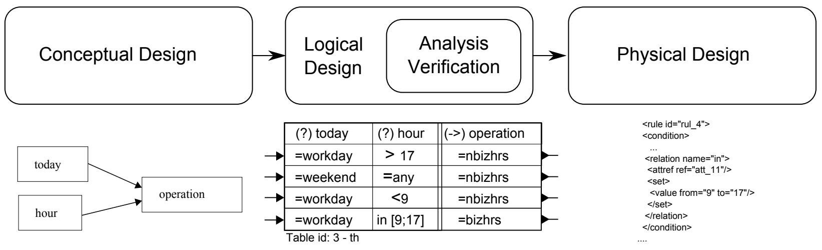


Figure 1. Sample Drools Flow diagram (drools.org)

Drools has some *limitations*. First of all, it is not a standarized solution. The form of knowledge representation still evolves. It could have been seen when version 5 was released - new block were introduced and the format of Drools Flow file has changed. Moreover, Drools does not provide any tools which can be used in the knowledge design phase. It can be problematic in large systems. What is more, the rulebase has a flat structure. Although, Drools Flow complements the strucutre by desribing execution process, the rules still do not have a hierarchy. The last thing is that Drools is language dependent, closely related to Java. Parts of the rules and some Rule Flow blocks contain Java expresions.

3.2 XTT2

XTT2 (EXtended Tabular Trees) [5] is a hybrid knowledge representation and design method aimed at combining decision trees and decision tables. It has been developed in the HeKatE research project (`hekate.ia.agh.edu.pl`), and its goal is to provide a new software development methodology, which tries to incorporate some well-established Knowledge Engineering tools and paradigms into the domain of Software Engineering, such as declarative knowledge representation, knowledge transformation based on existing inference strategies as well as verification, validation and refinement.



(?) today	(?) hour	(->) operation
=workday	> 17	=nbizhrs
=weekend	=any	=nbizhrs
=workday	<9	=nbizhrs
=workday	in [9;17]	=bizhrs

Table id: 3 - th

```
<rule id="rul_4">
<condition>
  ...
 <relation name="in">
  <attref ref="att_11"/>
  <set>
   <value from="9" to="17"/>
  </set>
 </relation>
</condition>
....
```

Figure 2. The HeKatE process

The HeKatE process consists of three design phases (shown in Fig. 2) [5]:

1. **The conceptual design phase**, which is the most abstract phase. During this phase, both system attributes and their functional relationships

are identified. This phase uses ARD+ (Attribute-Relationship) diagrams as a modeling tool. It allows design of the logical XTT2 structure.

2. **The logical design phase**, in which system structure is represented as a XTT2 hierarchy. The preliminary model of XTT2 can be obtained as a result of the previous phase. This phase uses the XTT2 representation as a design tool. During this phase, on-line analysis, verification as well as revision and optimization (if necessary) of the designed system properties is provided.
3. **The physical design phase**, in which the system implementation is generated from the XTT2 model. The code can be executed and debugged.

Some *limitations* of XTT2 can be pointed out. XTT2 provides a support for the entire process. It is used to model, represent, and store the business logic of designed systems. Rules in XTT2 are formalized with the use of the ALSV(FD) [5] logic and are supported by a Prolog-based interpretation. Although XTT2 rules are prototyped with the ARD+ method, the method is quite poor, and does not provide more advanced workflow constructs. Moreover, it is not a widely known methodology and only dedicated tools support it.

3.3 Business Rules and BPMN

BPMN [4] is a visual notation for business processes. A BPMN model defines the ways in which operations are carried out to accomplish the intended objectives of an organization. Visualization makes the model easier to understand. The goal of the notation is to provide such a notation which is easily understandable by business users. The notation provides only one kind of diagram – BPD (Business Process Diagram). There are four basic categories of BPD elements: Flow Objects (Events, Activities, and Gateways), Connecting Objects (Sequence Flow, Message Flow, Association), Swimlanes, and Artifacts. An example describing evaluation process of a student project is presented in Fig. 3.

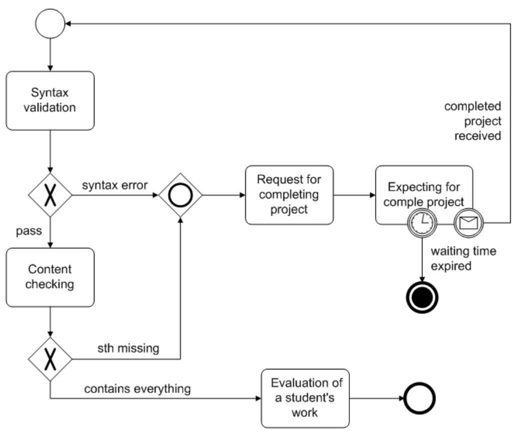
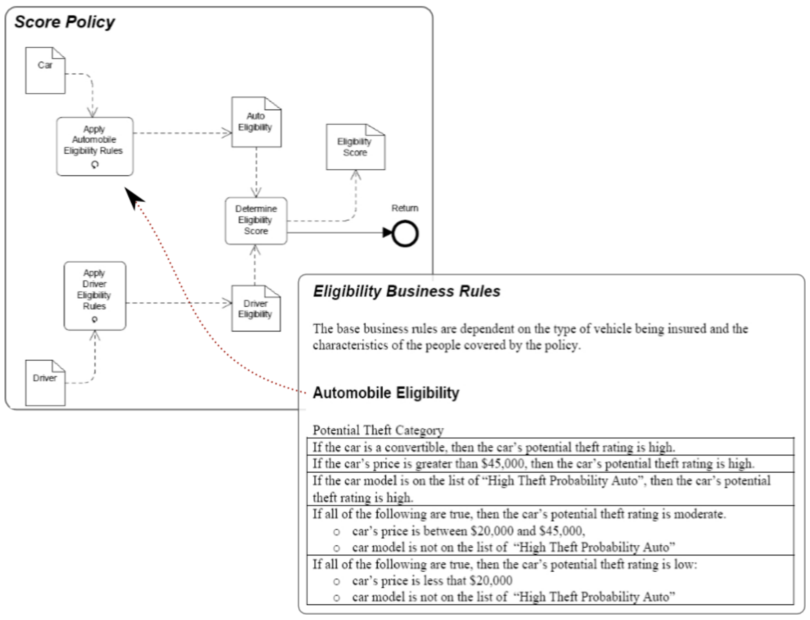

Syntax validation
syntax error
pass
Request for completing project
Expecting for comple project
completed project received
waiting time expired
Content checking
sth missing
contains everything
Evaluation of a student's work

Figure 3. An example of Business Process Diagram

BPMN [4] has been developed by the Business Process Management Initiative (BPMI) and currently is maintained by the Object Management Group. Although the notation is relatively young, BPMN is becoming increasingly popular. According to OMG, there is more than 60 BPMN implementations of BPMN tools. Moreover, BPMN models can be serialized to XML and further processed e.g. into languages for execution of business processes, such as BPEL4WS (Business Process Execution Language for Web Services) [10].

Very often a Business Process (BP) is associated with particular Business Rules (BR), which define or constrain some business aspect, and are intended to assert business structure or to control or influence the business behavior [11]. According to the specification [4], BPMN is not suitable for modeling concepts, such as organizational structures and resources, data models, and business rules. There is a huge difference in abstraction level between BPMN and BR. However, BR may be complementary to the business process. In Fig. 4, an example from the classic UServ Financial Services case study has been shown. This example presents how business processes and rules can be linked.


Score Policy
Car
Apply Automobile Eligibility Rules
Auto Eligibility
Eligibility Score
Determine Eligibility Score
Return
Apply Driver Eligibility Rules
Driver Eligibility
Driver

Eligibility Business Rules

The base business rules are dependent on the type of vehicle being insured and the characteristics of the people covered by the policy.

Automobile Eligibility

Potential Theft Category
If the car is a convertible, then the car's potential theft rating is high.
If the car's price is greater than $45,000, then the car's potential theft rating is high.
If the car model is on the list of "High Theft Probability Auto", then the car's potential theft rating is high.
If all of the following are true, then the car's potential theft rating is moderate. ○ car's price is between $20,000 and $45,000, ○ car model is not on the list of "High Theft Probability Auto"
If all of the following are true, then the car's potential theft rating is low: ○ car's price is less that $20,000 ○ car model is not on the list of "High Theft Probability Auto"

Figure 4. An example of using BR to define a process in Business Process Diagram

BPMN has some *weaknesses*. Although a specification defines a mapping between BPMN and BPEL (standard for execution languages), there is a fundamental difference between these two standards. One of the consequences of this difference is that, for instance, not every BPMN process can be mapped to BPEL and executed. Moreover, execution of the processes requires additional specification, which is not necessarily integrated with the entire design process.

Despite the fact that the BPMN model has well-defined semantics and a particular model should be clearly understood, there can be various models having the same meaning and there can be ambiguity in sharing BPMN models. Last but not least, it is difficult to asses the quality of the model.

3.4 Critical comparison

As one can see from the Table 1, each of these solutions has some pros and cons. Integration of these technologies based on their merits can bring better results than using them separately.

	Drools 4	**BPMN**	**XTT2**
Visual design of the rulebase	no	no	**yes**
Verification	no	some	**yes**
Workflow modeling (OR, AND etc.)	yes	**yes**	no
Runtime environment	yes	no	**yes**
Tool support	yes	**yes**	yes
Standardization	no	**yes**	no

Table 1. Comparison of the three approaches

The disadvantages of the Drools Flow are platform dependency and lack of standarization. Drools Flow supports decision tables and grouping of unrelated rules. XTT2 allows multiple connections between tables. Although Drools only allows for a single connection, it provides *Join* and *Split* blocks.

The XTT2 connections are of the AND type, by default. However, the conection semantics is different than that in Drools or BPMN. In Drools and BPMN, the default inference process is forward chaining, while XTT2 provides various inference modes, e.g. forward chaining (where the connections ore of the AND type) or backward chaining (where the semantics of connections varies).

BPMN is only a notation which has many elements for precise control of flow. However, this solution originally was not based on Rule-Based Systems. Therefore, it does not define the relationship between processes and rules. Although BPMN can be mapped to BPEL and executed, mapping and execution is possible only for selected groups of a BPMN model.

In case of XTT2, the entire design process is supported. What is more, formal on-line analysis can be performed during the design process, and then a prototype of the system can be generated. However, XTT2 is not a wide-spread solution, and does not pretend to be a universal method.

On the one hand, the comparison shows that XTT2 is the only one solution which supports visual modeling of the rulebase (modeling using decision tables). Moreover, only XTT2 provides formal verification. On the other hand, Drools offers workflow modeling. The integration of Drools and XTT gave the opportunity to combine these advantages. The next section describes the proposal of rule translation from XTT2 to Drools in detail, as part of the HeKatE project.

BPMN is already a well-known and standardized notation. In Drools 5, it can be used to model workflow. To facilitate workflow modeling for XTT2 and to provide an executable platform for BPMN, the integration of XTT2 with BPMN is considered. The possible scenarios are identified and described in Section 5. This research is a part of the *BIMLOQ* project (2010–2012).

4 Proposal of Rule Translation from XTT2 to Drools

Knowledge structure represented by Drools is very similar to the one represented by XTT2. In fact, that was one of the main reasons for choosing Drools as an integration platform. Both frameworks have the same goal: to provide rule-based and structurized knowledge representation. On the one hand, XTT2 is a unified structure which contains both rules and inference flow. On the other hand, Drools has both of these features, but rules can exist without Drools Flow. Both solutions can be used to model business processes. Drools Flow even provides special blocks which contain Java source code to be executed. XTT2, however, is more flexible and language independent. It contains rules which do not have any dialect specific parts.

4.1 Generating Drools files

Knowledge represented in XTT2 is stored in XML form. One file contains a tree structure and rules. Drools with the Flow model, on the other hand, stores knowledge in at least two files: a file with rules and a file with a flow. The XTT2-to-Drools integration mechanism separates XTT2 rules from the structure, transforms them and puts into two separate Drools files.

Nevertheless, Drools operates on objects while XTT2 uses primitive types. In Drools, facts are instances of Java classes inserted into the working memory. When the rules are fired, values used during comparison are taken from objects using getters. The workaround would be to create one Java class which contains all XTT2 attributes. The class is called a *Workspace*. To sum up, three files are generated from one XTT2 model file: *Rule Flow* (model structure), *Decision tables* (aggregated rules), and *Workspace* (a Java class with all attributes).

The results of XTT2 into Drools translation are three files. The first one is an XML based file and represents the flow structure. It does not contain the actual rules, but only the nodes (tables' names). The second one, a CSV (Comma separated values) file, contains Decision Tables storing the rules. The last one is a single Java class which holds all the XTT2 attributes.

4.2 Structural difference

While generating Drools files from an XTT2 file, structural differences are revealed. First one was already mentioned above. It is the form of attribute types. There is, however, an easy solution. The type of every XTT2 attribute is exchanged with an appropriate Java type. All XTT2 attributes are wrapped into one *Workspace* class which does not contain any logic but getters and setters.

Another structural difference is the placement of the logical operator. An XTT2 table is translated to a Drools decision table. An XTT2 table contains logical operators in the table cell – together with the value used in comparison. This implies that in one column, many different operators can appear. In Drools, however, the logical operator is placed in a table header. This means that all cells underneath use the same operator. This problem can be solved by decomposing the XTT2 columns into one or more columns in Drools model. Table 2 is the representation of XTT rules, while Table 3 is its Drools equivalent.

today	**hour**	**operation**
= workday	> 17	= nbizhrs
= weekend	= ANY	= nbizhrs
= workday	< 9	= nbizhrs
= workday	in [9,17]	= bizhrs

Table 2. XTT table from the thermostat example

condition	condition	condition	condition	condition	action
Workspace	Workspace	Workspace	Workspace	Workspace	Workspace
Today = "$param"	hour >	hour <	hour >=	hour <=	setOperation ("$param")
workday	17				nbizhrs
weekend					nbizhrs
workday		9			nbizhrs
workday			9	17	bizhrs

Table 3. Decision Table for the thermostat example

There are some structural differences in the flow structure as well. First of all, XTT2 tables allow multiple incoming connections. Furthermore, the connection can be directed to a specific row in a table. It is not possible in Drools Flow. *Ruleset* blocks can have only one incoming and one outgoing connection. This issue can be resolved by placing *split* and *join* blocks before and after the *ruleset* block. Nevertheless, the problem with row-to-row connection is still present and it is to be resolved in a future version of the integration proposal.

Another difference with the representation form of Drools Flow appeared when version 5 of Drools was released. In version 4.0.7, the Drools Flow structure could only be created in a dedicated Eclipse plugin. This is because the file which contained the structure was a Java class serialized using the XStream library (`http://xstream.codehaus.org`). The programmer was dependent on the class, which was included into the plugin. In version 5 however, the file storing the Flow structure was slimmed and now contains only the most important information: blocks defined in the flow and connections between them.

5 Proposal of XTT2 Inference Design with BPMN

The integration of XTT2 with BPMN faces two main challenges.

Different goals: BPMN provides a notation for modeling business processes. Such processes define the order of tasks to accomplish the intended objectives of an organization. Although in BPMN one can define very detailed description of the particular task, it is rather not the proper use of the notation. The XTT2 methodology, in turn, is not only a notation. It provides well-founded systematic and complete design process [5]. This preserves the quality aspects of the rule model and allows gradual system design and automated implementation of RBS.

Different semantics Apart from goals, the semantics of both notations is also different. BPMN describes processes while XTT2 provides the description of rules. Although the semantics of each BPMN element is defined, the implementation of some particular task is not defined in pure BPMN. XTT2 provides a formal language definition and therefore enables automatic verification and execution. Therefore, BPMN and XTT2 operate on different abstraction levels.

Several **integration scenarios** for XTT2 and BPMN are considered:

- **BPMN integration with XTT2**
  This scenario assumes that BPMN and XTT2 have some intersecting parts, in which the integration of the two solutions can be performed. The general idea is as follows: BPMN is responsible for inference specification and hierarchization of the rulebase, and rule tables for some part of the system are designed in XTT2. Another example is a BPMN model of a cashpoint, shown in Fig. 7 and 8.
- **BPMN as a replacement of ARD+**
  Because the abstraction level of ARD+ and BPMN seems to be similar, in this scenario BPMN is proposed to be used instead of present solution – ARD+. This assumes that mapping between BPMN tasks and XTT2 tables is one-to-one. A prototype example of this approach is shown in Fig. 5.
- **BPMN representation of XTT2 table**
  This is not a primary goal of integration. However, this could enable BPMN design of the whole XTT2 methodology, including single tables and rules. An example of this approach can be seen in Fig. 6.

Because the assumed mapping in the first scenario may be not one-to-one, this scenario is highly complex. It requires well-prepared analysis and specification of both solutions as well as a detailed specification of the integration proposal. However, this is the best scenario for real-world cases. In the second one, in turn, the mapping is very simple, because each task is mapped to exactly one table. However, this solution does not provide the table schema, as it was in the case of ARD+. The third scenario is a rather academic one, because tables are already an efficient method of presenting rules, and their visual representation in another form may not be so useful [12].

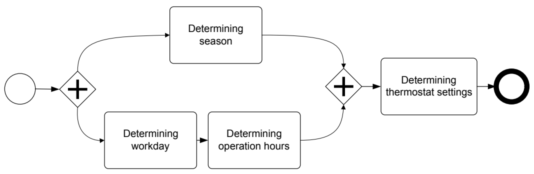


Figure 5. An example of using BPMN instead of ARD+

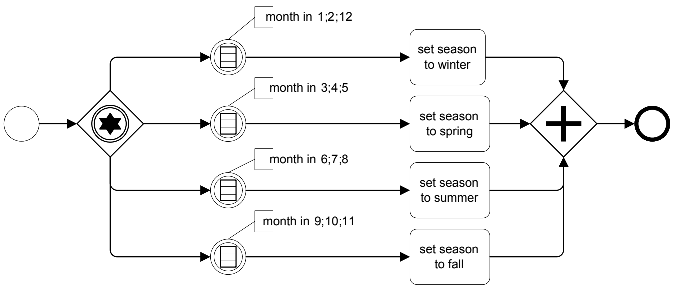


Figure 6. BPMN representation of XTT2 table

6 Conclusions and Future Work

The general problem considered in this paper is the RBS design and modularization. The paper considers possible solutions to modularize rule bases with Drools, BPMN and XTT2. However, these solutions have some limitations. The paper constitutes an overview and a proposal for integration of the three presented methodologies, which can be useful in solving the identified problems.

The work described here is partially in progress. The rule translation from XTT2 to Drools is being developed and implemented. The design is the result of the comparison of both semanticts (XTT2 and Drools) while the translation is achieved by the module to HQEd, writen in C++. Drools 5 has some differences from its predecessor. The most important thing is that Drools Flow focuses more on a process management, rather than on the rule hierarchisation. In the previous version the main part was the block which refers to the rules in the knowledge base. In the new version there are much more blocks which provide strict integration with Java programming langauge.

Moreover, several issues concerning BPMN as an end-user notation are considered. Future work will be focused on integration of the three described solutions. The plan involves analysis of the BPMN notation for the purpose of Rule-Based Systems, which can be useful for implementation and application of the integrated methodology. In a more distant future, the plan involves running selected BPMN models in the rule engine, and comparison of the analysis of BPMN models via rule engine to executable BPEL4WS.

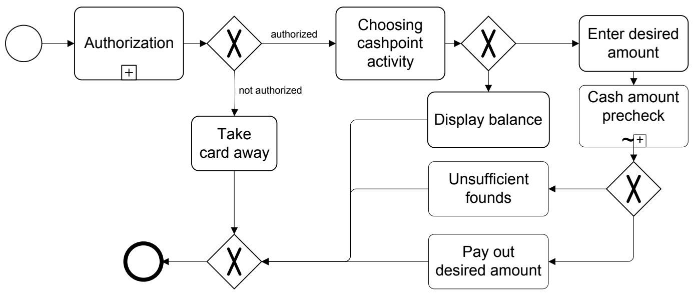


Figure 7. Example of BPMN representation of cashpoint

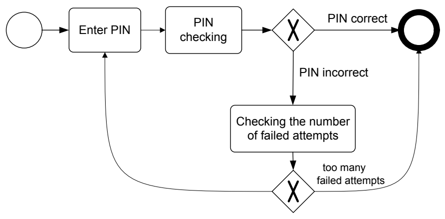


Figure 8. Example of BPMN representation of cashpoint *Authorization* subactivity

References

1. Ligęza, A.: Logical Foundations for Rule-Based Systems. Springer-Verlag, Berlin, Heidelberg (2006)
2. Browne, P.: JBoss Drools Business Rules. Packt Publishing (2009)
3. Owen, M., Raj, J.: BPMN and business process management. Introduction to the new business process modeling standard. Technical report, OMG (2006)
4. OMG: Business process modeling notation (bpmn) specification. Technical Report dtc/06-02-01, Object Management Group (February 2006)
5. Nalepa, G.J., Ligęza, A.: HeKatE methodology, hybrid engineering of intelligent systems. International Journal of Applied Mathematics and Computer Science **20**(1) (March 2010) 35–53
6. Bobek, S., Kaczor, K., Nalepa, G.J.: Overview of rule inference algorithms for structured rule bases. (2010) to be published.
7. Giarratano, J., Riley, G.: Expert Systems. Principles and Programming. 4th edn. Thomson Course Technology, Boston, MA, United States (2005) ISBN 0-534-38447-1.
8. Friedman-Hill, E.: Jess in Action, Rule Based Systems in Java. Manning (2003)
9. Doorenbos, R.B.: Production Matching for Large Learning Systems. Carnegie Mellon University, Pittsburgh, PA, United States of America (2005)
10. Sarang, P., Juric, M., Mathew, B.: Business Process Execution Language for Web Services BPEL and BPEL4WS. Packt Publishing (2006)
11. Hay, D., Kolber, A., Healy, K.A.: Defining business rules - what they really are. final report. Technical report, Business Rules Group (July 2000)
12. Kluza, K., Nalepa, G.J.: Analysis of UML representation for XTT and ARD rule design methods. Technical Report CSLTR 5/2009, AGH University of Science and Technology (2009)

Knowledge Acquisition for Simulating Complex Psychological Processes

Régis Newo, Kerstin Bach, and Klaus-Dieter Althoff

University of Hildesheim,
Institute of Computer Sciences,
Laboratory of Intelligent Information Systems
Email: lastname@iis.uni-hildesheim.de

Abstract. In this paper we introduce an approach that elicits information from psychologists and transfers this knowledge in knowledge representations for a knowledge-based system. Based on SIMOCOSTS, a psychological model for simulating coping strategies in critical situations, we present a case study on a literary character that exemplifies our approach. We describe the simulation sequences in an abstract way and elucidate how we utilise the acquired knowledge. We use Eichendorff's literary character of "From the Life of a Good-for-Nothing" ("Aus dem Leben eines Taugenichts") as an example for a psychological analysis, which is transferred into a simulation scenario according to the SIMOCOSTS model.

1 Introduction

Artificial Intelligence (AI) has been successfully applied in various research disciplines trying to explain complex contexts. Especially when dealing with complex problems, a simulation of the problem helps to understand interdependencies between various components. When a person has to cope with a critical situation in her or his life, many reactions are possible responding this situation, so it is very interesting how the decision they made have been developed. The work presented in this paper picks up on this topic and presents an approach for simulating a person's behavior coping with critical situations. The approach has been developed together with development psychologists who have developed a psychological theory how decision making in stressful situations can be explained [1]. In this approach the decision making process is explained as a market place with various buyers and sellers. Each of them has an individual goal and they bargain with each other trying to receive its goal. The buyers and sellers in the theory are goals that have been developed over years and the result of the negotiation is a strategy how the person deals with the situation.

We picked up the idea of a market place idea where different goals meet and transfered it to an multi-agent-system that simulates the determination of a strategy. The underlying architecture of our approach is the SEASALT (Sharing Experience using an Agent-based System Architecture LayouT) architecture [2] which is based on the CoMES (Collaborating Multi-Expert-Systems) approach [3], the research vision of our group.

SEASALT consists of five components and is built around a multi-agent system containing intelligent agents providing information (Knowledge Provision). Each agent, so-called Topic Agent, is equipped with a knowledge-based system that derives it information from some kind of community (Knowledge Sources) that is pre-processed and formalized in order to be usable by a knowledge based system (Knowledge Formalization). Furthermore a Knowledge Representation component provides shared knowledge like rule sets and domain ontologies. The interaction with the user is realized via a user interface that interacts with the multi-agent system. The simulation of developing coping strategies focuses on the multi-agent- system and the according knowledge formalization which is mainly carried out by psychologists supported by intelligent graphical interfaces. By talking about knowledge based systems, we are currently focusing on case-based reasoning (CBR) systems, because CBR is an established methodology that uses previous experiences to solve new problems. Moreover, CBR already works with representative examples and thus performs well in interdisciplinary projects as ours. Experts have to provide initial examples and by using the application the figure our "wrong behaviors" and provide more examples. So, the application develops successively by integrating the new examples. In comparison to other SEASALT realizations like docQuery [4] that are focusing on co-operating topic agents where the agents deal with complex problems by dividing them into topics that are solved individually and afterwards the solutions are put together. However, in this approach we have competitive agents, because each intelligent agent has its individual goal and they are competing against each other. The knowledge acquisition is almost the same, because a knowledge engineer is supported by intelligent agents or processed developing knowledge models and cases that can further be used by the CBR systems.

The remaining part of this paper presents the SIMOCOSTS model and the briefly introduced marked place idea in section 2.2, followed by the introduction of our case study based on Eichendorff's literary character of "From the Life of a Good-for-Nothing" ("Aus dem Leben eines Taugenichts") [5] in section 3. Section 4 discusses how knowledge can be acquired and gives examples taken from the case study. The final Section 5 sums up the paper and points out the next challenges for us.

2 Simulating human processes

As stated in the previous section, we deal in our project with human behaviour. In this section, we will first present some psychological background that we use for the implementation of the simulation. It should be noted that we want to develop a tool that should be used only by domain experts.
Any time we have to deal with simulations, we have to know exactly which information we need and what we have to take into account. In our case, it means that we have to know which information is processed by a person while facing

a critical situation. Yet, as we all know, it is (almost) impossible to capture the complete knowledge of a human being. This due to the fact that that whole knowledge is saved in the so called *long term memory* which can, according to psychologists, store an infinite amount of information [6].
Luckily, humans never access their whole long term memory at the same moment. We instead just use our so called *short term memory*, which can be seen as a very small activated part of our long term memory. The short term memory contains information related to the situation we are actually experiencing. More on memory activation can be read in [7, 6]. That perception of memory legitimates us to just consider situation related information during our simulations. That means that we do not have to capture all kind of information that a human can have. We can restrict on the situation related knowledge in order to simulate a plausible behavior of human beings in critical situations.

Nevertheless, our simulation will be based on the theory of Brandtstädter and Greve [8]. It is based on the fact that intentions are a key part of psychological theories of action. Except for knee-jerk or automated behaviours, human actions are motivated by intentions. When somebody faces a critical situation, his actual state strongly differs from his goal state (i.e., his intentions). In order to solve the problem, the person essentially can use one of the following three forms of coping processes:

- *Assimilative processes*: the strategy here is to solve the problem by working directly on the actual state. That is, it is an active art to work through a problem, in which the person uses the available resources in a problem oriented way. The available resources can be the person's own resources or external ones.
- *Accommodative processes*: this strategy is used when the person believes he can not change the actual state (i.e. solve the problem) by himself. He then tries to adapt his goal state such that the discrepancy to the actual state can be diminished.
- *Defensive processes*: in this case, the person just ignores the discrepancy between the actual state and his goals. He can for example perform actions that diminishes the meaning of the discrepancy.

It should be noted that a person does not (normally) intentionally apply a given type of process. The person rather just try to find out, which strategy would be the best for him at the moment (depending on his capabilities, environment, etc.). The chosen strategy can then be evaluated to belong to one of the given processes by experts.

We will consider that the goals of a person play a crucial role for his behaviour. We also suppose that these goals are competitive. In fact, we will consider that human's mind is comparable to some kind of market place which contains those competitive goals.

Another question which have to be taken into account is to know which part of the knowledge of a human play a significant or even an essential role when he faces critical situations. This is actually a question that will be answered with the aid of the simulation tool, because even psychologists can not give

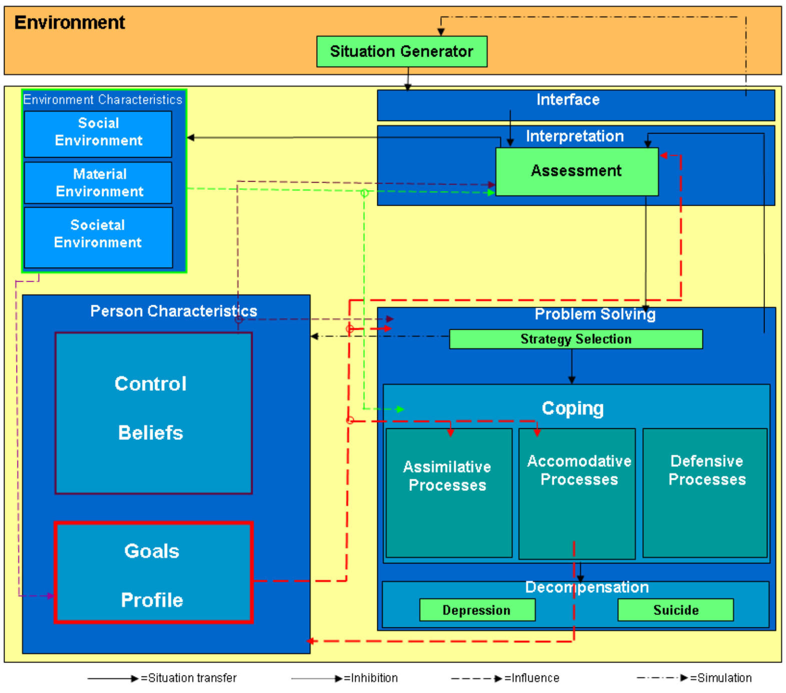


Fig. 1. Simplified Version of SIMOCOSTS (see [9])

a precise answer. Nevertheless, there are some parts (i.e. components) which are believed to a play a major role. To that extend, we developed a model, called SIMOCOSTS [9] (SImulation MOdel for COping STrategy Selection), in which we represent those components and also the interactions between them. A simplified version of the model can be seen on Figure 1

Yet that model can not be directly used for an implementation. We still for example have to figure out how the information is going to gathered and stored. The gathering of information will be discussed in Section 4. While dealing with human processes, many knowledge representation techniques have to be considered, because of the diversity of human knowledge. Yet we believe that humans mostly rely on past experiences (first or second hand), particularly in difficult situations. That is why we will mainly rely on Case-Based Reasoning (CBR) [10] as our main knowledge representation technique.

Considering the organisation the knowledge, we distinguish between two types of knowledge, which differs in their accessibility for the simulated person. The first type is called *shared knowledge* and is always accessible for the person. It contains general facts about the person as well as personal characteristics. The second type of knowledge is called *unshared knowledge* and is only available for the goals. It mainly contains information about the concerned goals and strategies which can be used if there is a critical situation.

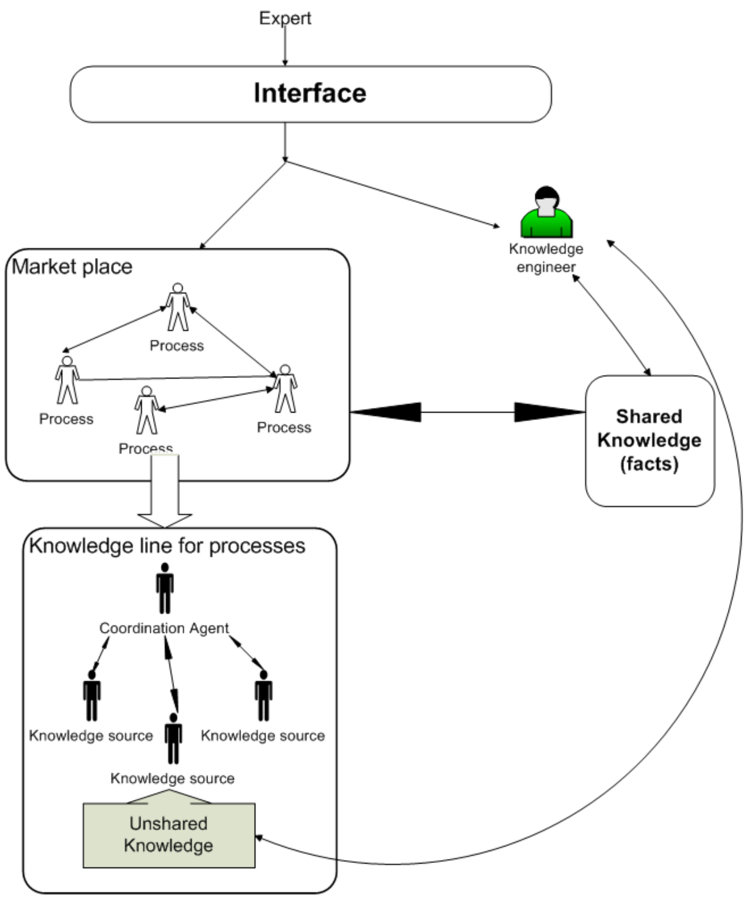


Fig. 2. Architecture for the Knowledge Management in SIMOCOSTS

A generic illustration of the simulation with respect to the knowledge management can be seen in Figure 2. The market place contains several processes, in our case the goals, which interact with each other and are competitive. Each process can access its own unshared knowledge which can in turn consist of multiple sources. As mentioned earlier, the knowledge sources will rely on CBR. Yet we do not rule out the use of other techniques (e.g. ontologies or rules).

This architecture actually displays the knowledge line of the SEASALT architecture [2]. SEASALT thus provides a very good platform for the organisation of the (unshared) knowledge, which can be realised with a case factory [11].

In order to give a better insight on the realisation of the simulation, we will elucidate in the next paragraph the underlying concepts and workflows used for the implementation.

2.1 Concepts of the Simulation

Person The main concept is the person itself. As can be seen on Figure 2, the person can access a so-called general knowledge, which contains for instance general facts about the person. These general fact include psychic as well as physical facts. It means that we assume that we know how the person feels physically and which abilities he has as well as the personality of the simulated person. We use several psychological theories in order to represent all those facts. In the style of the personality theory developed by Asendorpf [12], we describe the psychological facts among other things with the self-concept of a human by using adjectives defined by Müskens in [13]. These adjectives are grouped in several classes which include for example happiness and creativity. The physical abilities are similarly represented by defining the hard skills of the simulated person.
We thus represent all those attributes of the person with attribute value pair in which the attributes display the characteristics and the values whether and to which extend they are existent.

Situations In our simulation, a situation is a description of a state. It might for instance be the description of the actual state of the person, thus indicating how the person feels. We use situations to model two important parts of our architecture.

Events An event is a situation which may have an impact on the characteristics of the person. Events represents the situation that the person faces which might affect him. The simulation therefore starts with the (external) generation of an event with is then passed to the person.

Goals Goals, as stated earlier, are things that the person wants to achieve. They are thus concrete specification of situations that the person wants to achieve. Goals might have priorities which show their importance for the simulated person. An accommodative strategy to achieve is then capable to change the goal or adjust the its priority.

Strategies Strategies are the plans that can be used by a person in order to achieve a certain goal. These plans can consists of several steps which affect either the actual situation (assimilative) or the concerned goal (accommodative).

2.2 Workflows of the Simulation

A simulation run consists of five main steps:

1. The generated event is compared to the goals of the person in order to find out whether we have a critical situation.

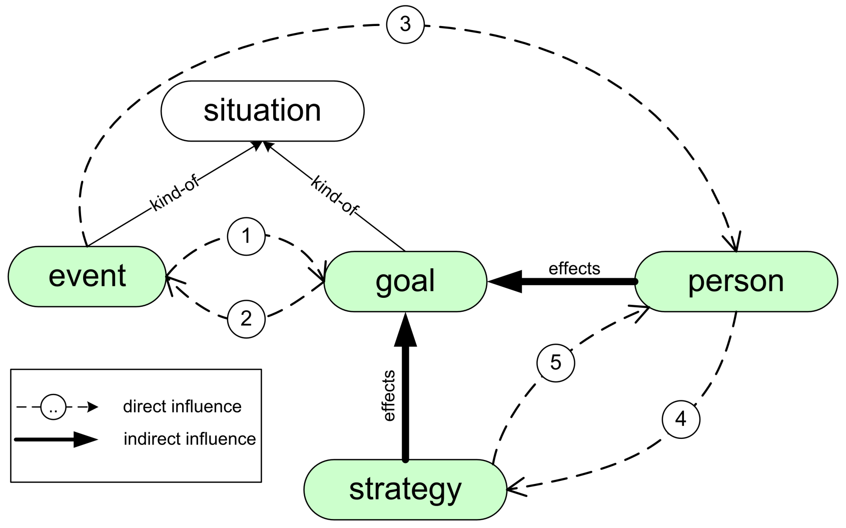


Fig. 3. Correlation between the concepts and illustration of the workflows

2. The goals gives a feedback of that evaluation to the event. That feedback contains a so-called influence factor vector which indicates how the event actually affects the person.
3. The influence factors are applied on the personal characteristics.
4. The best strategies for the affected goals are computed.
5. The influence factor generated by the strategies is applied on the characteristics of the person.

Afterwards, the psychologists analyse the state of the person after the run and also have the possibility to start a new run with different parameters. These workflows are illustrated in Figure 3

3 Case Study

In this section, we will present an example to illustrate the architecture. We decided to stick with an literary example because the (important) information is given and accessible for everyone. Another reason is the fact that we also have the output that our simulation tool should give. It is hence a good way to tune our implementation.

Our example is a literary character of "From the Life of a Good-For-Nothing", a novella of Joseph von Eichendorff [5]. It is the story of a young man, called Good-For-Nothing, who is very lazy (hence the name). He likes to sing and play the violin. Because he does not like to work and does not help at home, his father send him to the wide world. His adventure starts there. Although he is quite happy at the beginning, because he likes journeys, he faces several difficulties during his adventure. Nevertheless, he always finds a way to come through, even if he is sometimes lucky.

Our aim is to simulate the main character, Good-For-Nothing, in different situations. For this example, we will concentrate on the psychological facts about the character, because the physical facts do not play an important role in his decision making. The only hard skill that can be mentioned is his ability to play violin. As for his self-concept, Good-For-Nothing is a positive and preponderant lucky person. He is also authentic, candid, unorderly, educated and creative. Furthermore, he can be seen as a sentimental person and in need of affection.

He has several goals as the story goes on. First, as he leaves his father's home, he soon realizes that he has to earn his keep. This situation is a difficult one because he doe not like to work. Later he gets to know a girl, yet she does not seem to like him at fist sight. Moreover, he has the goal to go to Italy because he heard many good things about the country. Because of his laziness, Good-For-Nothing often applies accommodative strategies to overcome his critical situations. That means, he just changes his goals or at least their priorities. Most of his assimilative strategies are based on luck.

4 Knowledge Acquisition in SIMOCOSTS

With the previous example, we can see that there are many information that should be given before an initial simulation run can be made. Because the tool should be used by domain experts, we have to deal with the knowledge acquisition from experts. Two main problems arise. First, the expert has to give many different information before the start. He does not only have to give a complete and detailed description of the person (i.e. characteristics), he also have to provide the goals, strategies to achieve these goals and also events, which should be evaluated by the person (as critical situations or not).
The second main problem concerns the amount of information needed for each part of the simulation. In order to avoid a trivial simulation, we do not only need decent algorithms, but also enough information. Because of the fact that we intend to use CBR as main knowledge representation technique, we need for example many strategies the person can rely on before we can start simulation runs. These two points lead to the fact we need a good knowledge acquisition methodology. That methodology should of course also take into account that the experts, in our case the psychologists, often have a different perception of the underlying model and the simulation than computer scientists. Furthermore, they want to able to have non trivial simulation runs, which is necessary in order to be able to develop (psychological) theories.

The basic idea of our approach is based upon the fact that the domain expert should not be overwhelmed by the amount of information that has to be entered.

In our concrete example, we aim at having as few input masks as possible. For that purpose, we try to gather similar information in one step instead of

getting each kind of information in different steps. This leads to the need of an intelligent combination of similar information inputs. Determining the appropriate combination is of course a highly domain dependent process. Another point that has to be considered is the use of information extraction techniques. The domain expert should be able to enter the needed information in his preferred form, because this might encourage him to give more input. It is therefore a necessity to use information extraction techniques, in order to be able to structure the given information in our case-based simulation tool.
For our simulation we will have two input masks. The first one allows the expert to give information about the person. That means that the experts do not only give general facts about the person, but also his goals. In the second mask, the expert should be able to enter information about the events as well as some strategies which can be applied in order overcome the critical situations. This is advantageous for the experts because the can give events as well as solutions which should adapted and applied on the event. We should remark that the specification of strategies does not trivialise the simulation, because it is important to find out which factors play a role for the adjustment of the strategies.

With both input masks, we will be able to gather information about the four knowledge areas in our simulation (namely personal characteristics, goals, events and strategies).

4.1 Discussion

We are still implementing the approach. Therefore we can not provide an evaluation yet. Nevertheless we strongly think that it would facilitate the knowledge acquisition from the experts.

If we consider the case study from the previous section, the experts would have to provide in the first mask the characteristics mentioned (lazy, candid, etc.) as well the goals of Good-For-Nothing (e.g. earn his keep or have a relationship with the girl he loves). The second mask offers the possibility to enter events which will start the simulation and also strategies sketches for the event. The experts thus can provide information for several knowledge bases in the same iteration.

5 Summary and Outlook

We presented in this paper an architecture for the implementation of the simulation of complex processes. Our generic approach for the simulation is applied in a psychological realm, namely the simulation of human behaviour in so called critical situations. The developed is intended to be used by domain experts (psychologists) with the aim to develop and test theories. The simulation tool is currently been implemented while using the SEASALT architecture.
One major problem for our simulation is the knowledge acquisition. We gave an approach and elucidate how we intend to cope with that problem. Further steps towards realising our simulation tool include the implementation of both approaches and an evaluation of the tool.

References

1. Greve, W., Wentura, D.: Personal and Subpersonal Regulation of Human Development: Beyond Complementary Categories. Human Development **50** (2007) 201–207
2. Reichle, M., Bach, K., Althoff, K.D.: The SEASALT Architecture and its Realization within the docQuery Project. In Mertsching, B., ed.: Proceedings of the 32nd Annual Conference on Artificial Intelligence, KI-2009, Paderborn, Springer (September 2009) 556–563
3. Althoff, K.D., Bach, K., Deutsch, J.O., Hanft, A., Mänz, J., Müller, T., Newo, R., Reichle, M., Schaaf, M., Weis, K.H.: Collaborative Multi-Expert-Systems – Realizing Knowlegde-Product-Lines with Case Factories and Distributed Learning Systems. In Baumeister, J., Seipel, D., eds.: Proc. of the 3rd Workshop on Knowledge Engineering and Software Engineering (KESE 2007), Osnabrück, Germany, Berlin, Heidelberg, Paris, Springer Verlag (2007)
4. Bach, K., Reichle, M., Althoff, K.D.: Case-based reasoning in a travel medicine application. In Bichindaritz, I., Jain, L., eds.: Computational Intelligence in Medicine. Advanced Information and Knowledge Processing. Springer (2010) to appear
5. von Eichendorff, J.: Aus dem Leben eines Taugenichts. Vereinsbuchhandlung, Berlin (1826)
6. Anderson, J.: Kognitive Psychologie. Spektrum Akademischer Verlag, Heidelberg (2001)
7. Anderson, J.R.: A Spreading Activation Theory of Memory. In Collins, A., Smith, E.E., eds.: Readings in Cognitive Science - A Perspective from Psychology an Artificial Intelligence. Morgan Kaufmann Publishers (1988)
8. Brandtstädter, J., Greve, W.: The aging self: Stabilizing and protective processes. Developmental Review **14** (1994) 52–80
9. Newo, R., Müller, T., Althoff, K.D., Greve, W.: Learning to Cope with Critical Situations - A Simulation Model of Cognitive Processes using Multiagent Systems. In Hinneburg, A., ed.: Proceedings of the LWA 2007: Lernen - Wissen - Adaptivität. (September 2007) 159–164
10. Aamodt, A., Plaza, E.: Case-based reasoning : Foundational issues, methodological variations, and system approaches. AI Communications **1**(7) (March 1994)
11. Althoff, K.D., Hanft, A., Schaaf, M.: Case factory – maintaining experience to learn. In Göker, M.H., Roth-Berghofer, T., Güvenir, H.A., eds.: Proc. 8th European Conference on Case-Based Reasoning (ECCBR'06), Ölüdeniz/Fethiye, Turkey. Volume 4106 of Lecture Notes in Computer Science., Berlin, Heidelberg, Paris, Springer Verlag (2006) 429–442
12. Asendorpf, J.B.: Psychologie der Persönlichkeit. 4 edn. Springer Verlag (September 2007)
13. Müskens, W.: Sedimente der Selbstbeschreibung: Der lexikalische Ansatz der Persönlichkeitsforschung, Berlin (2001)

Interaction Pattern Categories
Pragmatic Engineering of Knowledge-Based Systems

Martina Freiberg, Joachim Baumeister, and Frank Puppe

University of Würzburg, Institute of Computer Science
Dept. of Artificial Intelligence and Applied Informatics
D-97074 Würzburg, Germany
`freiberg/baumeister/puppe@informatik.uni-wuerzburg.de`

Abstract. The application of knowledge-based consultation- and documentation systems is, apart from large industrial projects, often also beneficial for small to mid-sized enterprises. Yet, their design and implementation still is a tedious and costly task. We motivate, that customized UI and interaction patterns constitute a pragmatic technique for supporting especially requirements engineering, and thus are capable of considerably promoting real-world projects. In this paper, we introduce abstract categories—*Guided-*, *Adaptive-*, and *Autonomous Entry*—for classifying tailored patterns for knowledge-based systems. Further, we discuss their role in an overall approach extending the *Agile Process Model* and resulting benefits.

Keywords: Dialog System, User Interface Design, Agile Development

1 Introduction

Knowledge-based systems gained increasing impact also outside academia over the last decades. Apart from large clinical and industrial projects, the application of knowledge-based consultation and documentation systems is also often beneficial for small to mid-sized corporations. Yet, the trade-off between their potential benefits and their mostly still tedious and costly development, is still often perceived as unfavorable, and respective projects are declined.

In general software engineering, *user interface (UI) prototyping* already is an established methodology regarding iterative, rather inexpensive system specification before the final product is implemented [3]. Also, UI prototyping permits the early evaluation of (several) design options. Inspired by that approach, we suggest *Interaction Patterns for Knowledge-Based Systems* as a cornerstone of a tailored, agile knowledge system development methodology. The overall approach integrates pattern- and prototyping-based development into an existing, agile process model, and thus combines the advantages of reusing approved solutions (patterns) and of affordable, iterative system specification (UI prototyping within an agile process model). We argue, that this constitutes a rather pragmatic way to enhance understanding, discussing, and specifying system requirements at project start. This in turn helps to promote respective projects in the first place, and thus renders its application especially interesting when addressing small to mid-sized enterprises as customers.

As a first step into this direction, this paper introduces *Interaction Pattern Categories*, that provide an abstract classification framework for knowledge systems and corresponding interaction/UI patterns. We further discuss the role of such patterns within the proposed, extended agile approach; the details regarding the prototyping and a respective tool are subject of separate work, see [7].

The rest of the paper is organized as follows: Related research is presented in Section 2. In Section 3, we introduce our classification framework of *Interaction Pattern Categories* and the relevant terminology. We further present three categories, identified on the basis of past experiences with conducted projects. In Section 4, we outline the extended, agile process model, and the patterns' specific role as well as resulting benefits. We conclude with a summary of the presented work and a discussion of future research directions in Section 5.

2 Related Work

The process model for knowledge system development, that we suggest in this paper, integrates pattern- and prototyping-based development, thus uniting the advantages of both approaches and especially fostering an enhanced requirements engineering.

Patterns specify proven solutions for recurring (design) problems and are established in many domains: Examples are *software engineering*, *ontology engineering*, or *knowledge formalization*, [8, 9, 14]. They offer the advantage to reuse approved solutions for similar problems, and thus to reduce development efforts and to profit from the lessons learned. Regarding *UI–/interaction design*, tailored, domain-specific pattern collections exist, e.g., [16–18]. Yet, patterns originating from such research cannot be straightly transferred to our context, as knowledge-based systems put specific demands on interaction and UI design.

Regarding knowledge system development, various methodologies have been proposed in the past—see [12] for an overview. More recent works emphasize the relevance of *agility*, e.g., see [2, 10]. We follow that direction by integrating pattern- and prototyping-based techniques into an agile process model. Previous approaches, however, often strongly emphasize the development of the knowledge itself. In contrast, we specifically support knowledge system UI and interaction design by the means of tailored patterns and prototyping as to enhance requirements engineering on the one hand, and to foster a pragmatic, affordable promotion and execution of respective projects on the other hand.

UI prototyping so far has become an established approach in general software engineering [3] as well as in HCI and usability engineering [4]. Main advantages are a strong support of requirements specification, and the opportunity to evaluate (several) UI design(s) at an early stage. In [11], a prototyping tool, that incorporates design patterns for layout support, has been proposed. Though generally related to our approach, that work focusses on cross-device design of general web-style interaction. Contrastingly, we explicitly consider UI and interaction design of knowledge-based consultation and documentation systems.

3 Interaction Pattern Categories

By *Interaction Patterns for Knowledge-Based Systems*, we understand the description of the systems' interaction- and UI design for a specified context. They comprise a compact specification of their applicability, and exemplify the corresponding solution approach. Yet, there exist attributes, the value of which may be common to more than one distinct pattern. Thus, we first introduce an abstract framework—*Interaction Pattern Categories*—for classifying patterns according to such common properties before specifying concrete patterns. In Section 3.1, we first introduce relevant terminology, and in Section 3.2, we present the classifying categories—*Guided-*, *Adaptive-*, and *Autonomous Entry.*

3.1 Relevant Terminology for Specifying Pattern Categories

In the following, we specify the addressed system types as well as the classifying attributes, that characterize the pattern categories, in more detail.

Knowledge-Based Systems: We specifically address *knowledge-based systems* with our approach—by that, we understand knowledge systems, that serve either a *consultation* or a *documentation* task. In both cases, the main user-system interaction is structured data entry—mirrored by *"Entry"* in the pattern category names. Regarding consultation, the system gradually derives solutions for a given problem with the respective, implemented reasoning mechanisms based on the provided user input (answers). Documentation systems emphasize supporting uniform and reliable data input as effectively as possible.

Classifying Attributes: The attributes *User Competence*, *Context Presentation*, and *Data Volume* are common to all patterns of one category. Major classifier thereby is *User Competence*—in the context of knowledge-based systems, lengthy, strictly prescribed interviews can be annoying and inflexible for competent users, that might want to influence the interrogation flow according to their expertise. This makes it essential to tailor the system and interface design to the target users' competence.

A. User Competence: A *naive* data provider follows the prescribed interrogation sequence, with no desire for deviation or adaption; possible reasons can vary from rather low domain competence/lacking experience to a highly stressful usage context (but nevertheless domain expertise). *Experienced* users possess a certain domain expertise, and thus may be interested in system-suggested workflow guidance, yet, additionally require the option to influence the interrogation and to deviate from suggested paths. An *autonomous* problem solver finally possesses sufficient expertise to solve the problem independent from system guidance, based on the (potentially various and complex) information presented.

B. Data Volume: The amount of data that is processed during a typical interrogation session; thus, it corresponds to the number and the complexity of questions required for deriving a solution or entering a complete data set. We roughly distinguish between *small*, *medium*, and *large*. *Data Space*, in contrast, specifies the universal range of possible input data, and thus corresponds to the domain complexity. The respective data volume/data space combination does not influence the pattern categorization, yet the knowledge required for a specific implementation—e.g., large data space and low data volume implies sophisticated interrogation structures to present appropriate questions efficiently.

C. Context Presentation: *No context* means, that during an interrogation, only the required questions are presented, but no further information. Otherwise, we distinguish *support knowledge*—auxiliary information (not interrogation specific), or informal knowledge representations—and *interrogation context*—i.e., additional information regarding, e.g., the progression of the workflow, or indicating the consequences of choosing certain answer alternatives in advance. Type and extent of context presentation highly depend on the respective level of user competence—concerning naive data providers, interrogation context often is not required, yet for complex questions, support knowledge presentation might be advisable; with rising competence, the presentation of interrogation context gains importance for supporting an independent, efficient system usage.

3.2 Interaction Pattern Categories for Knowledge-Based Systems

Based on experiences from past projects, we define three basic categories for UI/interaction patterns: *Guided-*, *Adaptive-*, and *Autonomous Entry*. For each category, we describe the *Problem Statement*, the *Solution*, and the *Applicability*, specifying common properties that apply to all contained patterns. Further, we provide *Examples*—i.e., existing implementations—and *Variants*, that describe in what regard patterns of a category may vary.

The basic interaction specifying each pattern, regards question selection during interrogation. Even if patterns later vary, e.g., regarding the processed data volume, that basic interaction remains the same. For its specification we use the UML sequence diagram notation and the elements: *User*, the system *Interface*, *Questions* (presented to and answered by the user), the *Data* pool (storing data resulting from provided answers or reasoning), and the *Knowledge* component.

A. Guided Entry

Problem Users act as naive data providers, thus for a reliable, effective decision- or documentation support, a high level of system autonomy/guidance regarding the interrogation flow is required. Data volume might vary from small to medium.

Solution An interview metaphor is transferred to the interface, where the user and the system interact alternately. The system flexibly reacts to the provided answers by adapting the interrogation sequence, thus presenting only the question(s) that fit the respective context best. The interview proceeds system-guided, and deviation is mostly not (or only in limited terms) intended. Thus, presenting interrogation context is not mandatory, even though regarding lengthy sequences status feedback may be beneficial. Contrastingly, support knowledge is required in the case of complex/difficult questions for clarification.

Figure 1 depicts the interaction sequence for *Guided Entry.* The interface initi-

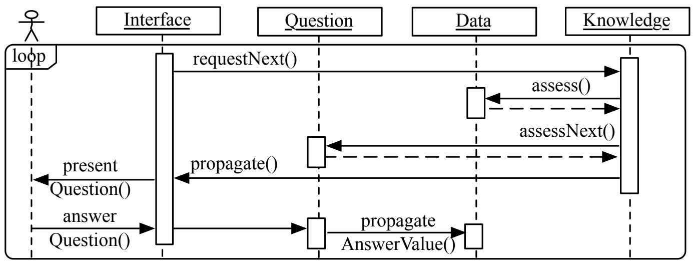


Fig. 1. Guided Entry—Basic interaction sequence for question selection.

ates the question request, whereupon the knowledge component assesses the next question—where available, based on the previously provided user input stored in the data pool—and propagates the result back to the interface. The then provided answer of the user is propagated to the data component and thus made available for the knowledge component hereafter. Those steps are performed iteratively until a defined interrogation sequence is finished.

Applicability Systems based on *Guided Entry* equally fit consultation and documentation tasks. Especially documentation of high quality or frequently recurring data is supported, as specified data entry can be assured by the strict, system-guided interrogation flow. However, if a higher level of user autonomy is desired—e.g., influence on the interrogation, or adaptable question representation—*Adaptive-* or *Autonomous Entry* provide more flexibility.

Examples Figure 2 presents two implementation variants of *Guided Entry. CareMate* (A) is a quick response second-opinion system for emergency situations. Its one-question interaction style creates the literal impression of an interview and supports the intuitive usage in the context of stressful emergency conditions. Continuous status feedback on the current solution states is provided, and the processed data volume is rather small. For a more detailed introduction, see [1]. *SonoConsult* [15] is a consultation and documentation system for the field of abdominal ultrasound. The multiple-question interaction style resembles a paper-based interview (questionary) and helps to cater with the rather large data volume. Both support knowledge (question clarification) and interrogation context (presenting currently derived solutions) are provided.

Variants Pattern variants arise with regards to the type and extent of context

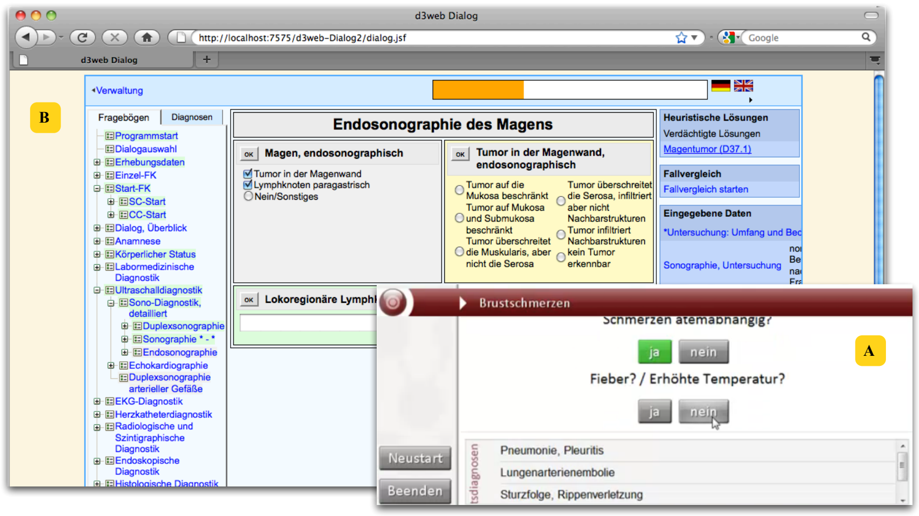


Fig. 2. Implementation variants of *Guided Entry*, both in german: A) Digitalys Care-Mate and B) SonoConsult.

presentation (see above examples), as well as regarding the characteristics of the *naive* user (e.g., expert in stressful context vs. non-expert's ad-hoc usage).

B. Adaptive Entry

Problem Experienced target users have a certain—yet, from user to user potentially varying—domain competence; consequently, both system guidance as well as the option for autonomous decisions regarding the workflow are desired. Also, questions should be presented in a user-adaptable manner.

Solution The system basically suggests the most appropriate workflow to the user; yet, also the option to deviate from that path and choose an adapted interrogation sequence is provided. Where applicable, a hierarchical tree metaphor is applied to cater with varying user competence levels: Questions are defined both on an abstract (aggregate) level, but also subdivided into (several) refined questions, where reasonable. Thus, according to their expertise, users may either answer the aggregate questions—taking less time, but requiring more expertise—or request the presentation of the questions' refinement. To support the user's decision-making, providing interrogation context is strongly recommended. Also, depending on the refinement level and complexity of the questions, support knowledge should be additionally presented.

Figure 3 sketches question selection in *Adaptive Entry*. Basically, the user decides whether to follow the system-guided interrogation or whether to choose an own path. Regarding the first alternative, question selection proceeds as in *Guided Entry* (Figure 1). In the second case, either the user's competence allows for an-

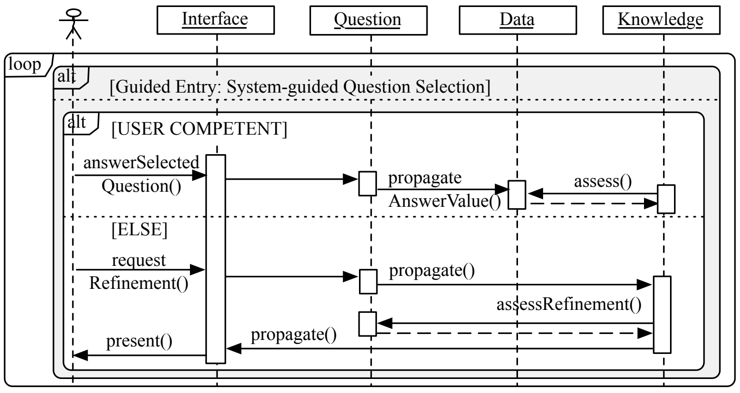


Fig. 3. Adaptive Entry—Basic interaction sequence for question selection.

swering the currently displayed question; then the answer is propagated to the data pool and thus is available for the knowledge component as the interrogation continues. Otherwise, the user can request question refinement whereupon the knowledge component assesses the possible refinement, and propagates the result back to the interface for displaying it to the user.

Applicability Apart from consultation, respective systems can, with limitations, also serve documentation purposes. In that case, special care has to be taken that all required input data is obtained from the user. Regarding effective interrogation of naive data providers *Adaptive Entry* is only marginally suitable.

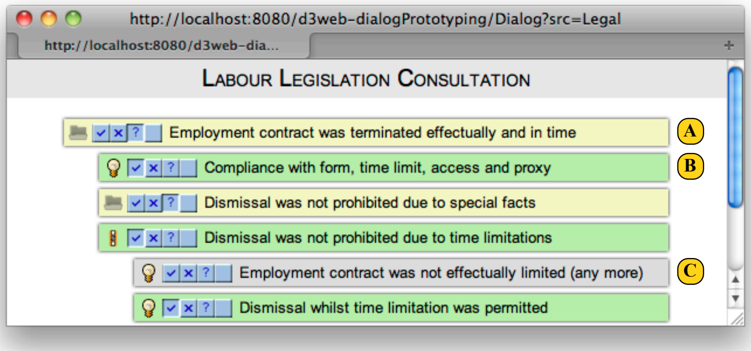
Examples Figure 4 shows the Labour Legislation Consultation, that clarifies, whether a dismissal in a given context is legitimate. Figure 4, A, represents the problem statement. Its current derivation state and the questions' state (e.g., answered) are visually indicated by background coloring and updated with each provided answer. Questions can be processed either in the sequence suggested (i.e., from top to bottom), or in any other order. Further, adaptable question presentation is implemented—e.g., Figure 4, B, was confirmed on the abstract level; question *Dismissal was...* is expanded into refined questions (Figure 4, C).

Variants Possible variants originate from different forms of context presentation as well as from different data volumes that may be processed.

C. Autonomous Entry

Problem Target users are highly competent, autonomous problem solvers, thus no explicit guidance regarding the interrogation sequence is required.

Solution The user explores the (various and potentially complex) information sources presented by the system. Integrated knowledge-based components—e.g., consultation features or automated data entry support—can be used optionally, but are not mandatory to benefit from system usage. The user provides any in-


http://localhost:8080/d3web-dialogPrototyping/Dialog?src=Legal
http://localhost:8080/d3web-dia...
A
B
C

LABOUR LEGISLATION CONSULTATION

Employment contract was terminated effectually and in time
Compliance with form, time limit, access and proxy
Dismissal was not prohibited due to special facts
Dismissal was not prohibited due to time limitations
Employment contract was not effectually limited (any more)
Dismissal whilst time limitation was permitted

Fig. 4. Labour Legislation Consultation—Indication of the solution and its current rating (A), clarifying questions (B), and further refinement of one question (C).

put data voluntarily; based on those data fragments, the system performs rather modularized reasoning, following sort of a *bits and pieces* metaphor. Thus, the system merely provides a second-opinion to the user in presenting reasoning results (e.g., rated solutions, next-input suggestions). Such extensive user control requires a high level of context presentation, regarding both types of context.

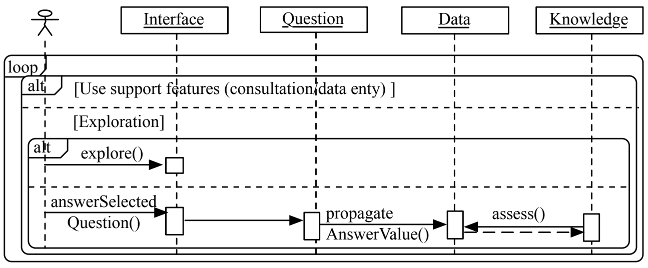


Fig. 5. Autonomous Entry—Basic interaction sequence for question selection.

As Figure 5 shows, the user always can choose between using more formal knowledge components and free exploration. In the first case, potentially any kind of (complex) knowledge component can be integrated into such a system, e.g., according to the *Guided Entry* or *Adaptive Entry* categories. Otherwise, the user can either simply explore the provided information, or answer questions autonomously in a modularized manner. Answers then are propagated to the data component, and from there assessed by the knowledge component; the latter rates solutions, presents context, and recommends next steps piecemeal.

Applicability *Autonomous Entry* can be applied for loose consultation, as well as regarding a more informal, potentially collaborative documentation task. In contrast, it is inappropriate, if rather naive data entry is desired, as no strict workflow guidance for solving the addressed problem is provided. Further it is not suitable for high quality documentation tasks, as any interaction takes place voluntarily, and thus the supply of any data cannot be guaranteed.

Examples Implementation examples are the user-centered consultation approach described in [5], the PEN-Ivory system [13], but also the *Inline Answering* concept provided by the Semantic Wiki KnowWE [1].

Variants Implementation variants arise with respect to the data volume, and the type and extent of context presentation. Despite mainly addressing expert users, systems falling into this category, might to some extent also be suitable for unexperienced ad-hoc usage. Finally, resulting systems can vary regarding the extent of integrating knowledge-based features.

The proposed pattern categories classify basic knowledge system types and corresponding UI/interaction design patterns according to the level of user competence (corresponding to the level of system guidance). Ongoing research addresses the definition of concrete patterns and their categorization accordingly. We proceed by discussing how such patterns can be integrated in an extended, agile process model for developing knowledge-based consultation and documentation systems.

4 Pragmatic Knowledge System Engineering

Regarding knowledge system development and knowledge engineering, there exist diverse approaches today, such as *CommonKADS*, *MIKE*, or adaptions of the classical *stage-based* and *incremental* software development models. Yet, for the success of knowledge system projects in the context of small to mid-sized companies, a pragmatic approach—affordable and efficient regarding time and effort—is essential, c.f. [2]. Especially for promoting such projects in the first place, it is important to quickly come up with first solutions, e.g., in the form of prototypes or example implementations. In this respect, we made positive experiences with applying the *Agile Process Model*, described in [2]. However, that model emphasizes knowledge base development, not yet taking much into account the design of the target system's interface, or usability traits. In extending this model, not only UI/interaction design gains importance in the overall development process, but also the integration of usability activities.

In the following, we introduce the *Extended Agile Process Model*, and afterwards we discuss resulting benefits specifically regarding the integration of tailored UI/interaction patterns. Although prototyping and usability-related activities are included in the model for reasons of completeness, their detailed discussion is part of further work, see [7].

4.1 The Extended, Agile Knowledge System Engineering Model

Figure 6 outlines the *Extended Agile Process Model*—the gray parts represent the original model, consisting of the four phases *System Metaphor*, *Planning Game*, *Implementation*, and *Integration*. For a detailed discussion, see [2].

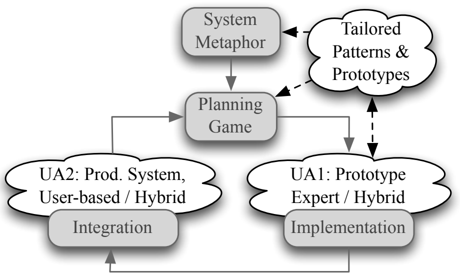

System Metaphor
Tailored Patterns & Prototypes
Planning Game
UA2: Prod. System, User-based / Hybrid
UA1: Prototype Expert / Hybrid
Integration
Implementation

Fig. 6. Extended Agile Process Model.

Basically, tailored patterns and prototyping can support both *System Metaphor* and *Planning Game*. In *System Metaphor*, the system objectives are defined by developers and customers. Based on appropriate patterns and corresponding implementation examples, a basic idea can be developed more easily. Thereby, patterns can be assessed either manually, or by using a tailored recommender system, that suggests patterns matching the target context. Prototypes, that also provide the relevant user-system interactions, further support that process by presenting a realistic simulation of a potentially resulting system as opposed to the static, visual depiction of knowledge system examples provided by the patterns. The *Planning Game* defines the scope and prioritization of development tasks. Here, patterns ease the analysis and valuation of system requirements—taking place during the *Exploration* sub-phase of the planning game—by providing clear specifications of required features and interactions. Additionally, prototyping supports that task by allowing for actually trying out (and thus better evaluating) relevant functionalities.

With regards to *Usability Activities*, the original model can be extended both regarding *Implementation* and *Integration* (Figure 6, UA1, UA2). The basic model defines *Implementation* as a test-first activity—i.e., before actually implementing new or additional features, the corresponding tests for assuring their correctness are developed. This can be expanded by an evaluation-first activity, in the sense that based on the formerly created prototypes, usability issues are assessed and valued first, before continuing with test-first implementation as defined by the model. Without going into detail here, at that stage, expert- or hybrid approaches (according to a categorization suggested in [6]) seem to be most appropriate. During *Integration*, the implemented functionality is added to the productive system, using integration tests for assuring its overall correctness and integrity. Such testing can be extended by usability checks that evaluate, whether the system still meets the specified usability goals. As this results in a running version of the productive system, not only hybrid, but also purely user-based usability evaluation can be beneficial.

4.2 Benefits

The integration of tailored patterns into an extended agile model offers several benefits: First, the patterns uniformly specify common framework conditions of different knowledge-based system types; thus, they provide a descriptive and visual language, that enables customers and developers to discuss at the same competence level. This fosters a clear communication and thus reduces potential misconceptions right away, that otherwise can lead to additional, unnecessary redesign cycles. Next, the patterns present actual implementation examples, that can be assessed, and serve as a inspirational source regarding the concrete project at hand. Even in case none of the provided patterns or examples completely satisfy the project- and customer requirements, those nonetheless are helpful by providing an overview of the possibilities and a basis for further discussions. Despite diverse general pattern collections and UI prototyping tools, to date to the best of our knowledge no tailored patterns/tools exist addressing specifically knowledge-based systems. As the latter exhibit quite specific characteristics, our approach can provide strong support for their development.

5 Conclusions and Future Work

We motivated that tailored patterns can constitute the cornerstone of an extended, agile model for knowledge system development. Especially when targeting smaller to mid-sized companies as customers, the suggested approach is a rather inexpensive, pragmatic technique for promoting and launching respective projects. As a first step, in this paper we introduced three abstract categories for classifying corresponding patterns. Due to our focus on knowledge-based consultation and documentation systems, those categories specifically address the data entry task; yet, the elementary classification—*guided, adaptive,* and *autonomous interaction* might be applied accordingly for other forms of interaction. The categorization arose from practical experiences with implementing knowledge-based systems in the past, such as *SonoConsult* [15], *Digitalys CareMate*, or more recently the *Semantic Wiki KnowWE* [1]. Further research includes the question, whether additional pattern categories are required. Based on those, as well as on an assessment of further existing systems, concrete patterns will be specified. Currently, also a tailored prototyping tool is developed [7], that will be further extended based on an analysis of required knowledge system base components.

References

1. Baumeister, J., Reutelshoefer, J., Puppe, F.: KnowWE: A Semantic Wiki for Knowledge Engineering. Applied Intelligence (2010)
2. Baumeister, J., Seipel, D., Puppe, F.: Agile development of rule systems. In: Giurca, Gasevic, Taveter (eds.) Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches. IGI Publishing (2009)

3. Bäumer, Dirk and Bischofberger, Walter R. and Lichter, Horst and Züllighoven, Heinz: User Interface Prototyping—Concepts, Tools, and Experience. In: ICSE '96 Proceedings of the 18th International Conference on Software Engineering. pp. 532–541 (1996)
4. Beaudouin-Lafon, M., Mackay, W.: Prototyping tools and techniques. In: The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications. pp. 1006–1031. L. Erlbaum Associates Inc., Hillsdale, NJ, USA (2003)
5. Buscher, G., Baumeister, J., Puppe, F., Seipel, D.: User-Centered Consultation by a Society of Agents. In: Proc. of the 3rd International Conference on Knowledge Capture (K-CAP 2005), Banff, Alberta, Canada (2005)
6. Freiberg, M., Baumeister, J.: A survey on usability evaluation techniques and an analysis of their actual application. Tech. Rep. 450, Institute of Computer Science, University of Würzburg, Germany (2008)
7. Freiberg, M., Mitlmeier, J., Baumeister, J., Puppe, F.: Knowledge system prototyping for usability engineering. In: Proceedings of the LWA-2010 (Special Track on Knowledge Management) (2010)
8. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley Longman (1995)
9. Gangemi, A., Presutti, V.: Ontology Design Patterns. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies (2009)
10. Knublauch, H.: Extreme programming of knowledge-based systems. In: Proceedings of eXtreme Programming and Agile Processes in Software Engineering (XP2002) (2002)
11. Lin, J., Landay, J.A.: Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. In: CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems. pp. 1313–1322 (2008)
12. Plant, R., Gamble, R.: Methodologies for the development of knowledge-based systems, 1982–2002. Knowledge Engineering Review 18(1), 47–81 (2003)
13. Poon, A.D., Fagan, L.M., Shortliffe, E.H.: The PEN-Ivory Project: Exploring User-Interface Design for the Selection of Items from Large Controlled Vocabularies of Medicine. Journal of the American Medical Informatics Association pp. 168–183 (1996)
14. Puppe, F.: Knowledge Formalization Patterns. In: Proceedings of PKAW 2000, Sydney Australia (2000)
15. Puppe, F., Atzmueller, M., Buscher, G., Huettig, M., Luehrs, H., Buscher, H.P.: Application and evaluation of a medical knowledge system in sonography (sonoconsult). In: Proceeding of the 2008 conference on ECAI 2008. pp. 683–687. IOS Press, Amsterdam, The Netherlands, The Netherlands (2008)
16. Ratzka, A.: Identifying user interface patterns from pertinent multimodal interaction use cases. In: Mensch & Computer 2008: Viel Mehr Interaktion. pp. 347–356 (2008)
17. Schmettow, M.: User interaction design patterns for information retrieval systems. In: EuroPLoP' 2006, Eleventh European Conference on Pattern Languages of Programs. pp. 489–512 (2006)
18. Tidwell, J.: Designing Interfaces — Patterns for Effective Interaction Design. O'Reilly Media Inc. (2006)

Requirements Selection: Knowledge based optimization techniques for solving the Next Release Problem

José del Sagrado, Isabel M. del Águila, Francisco J. Orellana, and S. Túnez

Dpt. Languages and Computation,
Ctra Sacramento s/n, 04120 University of Almería, Spain
{jsagrado,imaguila,fjorella,stunez}@ual.es

Abstract. The requirements selection for the next software release is a problem always present in Software Engineering. The complex nature of this problem and the difficulty to address it using exact techniques has motivated the application of optimization techniques to obtain near optimal solutions. This work provides a review of the different optimization techniques proposed to accomplish the requirements selection problem. Moreover, it proposes the application of these techniques in a requirement tool in order to be used in real software developments.

Keywords: next release problem, optimization techniques, search based software engineering

1 Introduction

Software development organizations fail many times to deliverer its products within schedule and budget. Statistical studies, and all the Chaos Reports [16] published since 1994, reveal that, frequently, tasks related to requirements lead software project to the disaster. When requirement-related tasks are poorly defined or executed, the software product is typically unsatisfactory [23]. Software requirements express the needs and constraints fixed for a software product that contribute to the solution of some real world problem [18]. Usually stakeholders propose some desired functionalities that software managers must filter in order to define the set of requirements to include in the final software product. All new suggested functionalities cannot be selected to be implemented since resource constraints are always present in development companies; hence each new feature competes against each other to be included in the next release software product. The requirements selection is considered a complex task in every software development since many factors are involved in this decision. A bad or inappropriate choice of enhancements can turn into a source of problems during software development: scheduling problems, dissatisfied customers, and economic losses. This problem is known as next release problem (NRP) [2] and it is considered an optimization problem [2, 17, 12] within the Search Based Software Engineering (SBSE) discipline [13, 5, 14]. The SBSE area is a growing research field which proposes the application of search based optimization algorithms to tackle problems in Software Engineering (SE). The term SBSE was first used in 2001 by Harman and Jones [13] and has been successfully applied to different problems in SE such as requirements, design tools and techniques, software verification and testing and debugging among others [15]. Different approaches can be found in the literature to tackle with requirements selection problem, for example, [2] and [3] apply greedy and simulated annealing (SA) techniques, [12] use genetic algorithms (GAs) in software release planning and [21] propose the use of ant colony optimization (ACO). In this work we provide a comprehensive review of the AI techniques applied to solve the NRP. We also propose the inclusion of these techniques on a CARE (Case Aided REquirement) tool to guide the decision maker to select the best set of requirements for the next release. The rest of this paper is structured as follows. Section 2 introduces and provides the formal description of the NRP problem describing different approaches used when addressing the NRP as an optimization problem. Section 3 analyzes the existing techniques applied in the literature to address the NRP whereas Section 4 describes a proposal to integrate these optimization techniques in a CARE tool. Finally, Section 5 draws conclusions and future works.

2 The NRP Problem

The problem of selecting the subset of requirements among a whole set of candidate requirements proposed by a group of customers, that will be included in the next release of a software product is a well-known problem in Software Engineering. However, it is not a straightforward problem since many factors are involved in this selection problem. Customers, seeking their own interest, demand the set of enhancements they consider important, but not all customer needs can be satisfied; on the one hand, each requirement means a cost in effort terms that the company must assume but company resources are limited; on the other hand, neither all the customers are equally important for the company nor are the requirements equally important for the customers. Market factors can also drive this selection process; the company may be interested on satisfying the newest customers' needs, or they may consider desirable to guarantee every customer have fulfilled at least one of their proposed requirements. Two main goals are usually considered in this kind of problems: find a subset of requirements which maximize the customers' satisfaction and minimize the required effort to implement the subset of chosen requirements. The complexity of the problem increases as the number of customers and requirements grows. Therefore, optimization techniques can be used to find optimal or near optimal solutions in a reasonable amount of time. As Harman defined in [13], it is possible to apply metaheuristic search to numerous problems in SE, but that aim requires a reformulation of the problem which implies to define:

– a representation of the problem which is amenable to symbolic manipulation,
– a fitness function based on this representation and
– a set of manipulation operators.

These are the steps that we are going to follow in order to review how metaheuristic search techniques had been applied to the NRP problem.

2.1 The NRP formulation

Let $R = \{r_1, r_2, \ldots, r_n\}$ be the set of requirements that are proposed by the customers. These requirements represent enhancements to the current software system, suggested by a set of m customers and candidates to be included in the next release. Customers are not equally important for the company. So, each customer i will have an associated weight w_i, which measures its relative importance. Let $W = \{w_1, w_2, \ldots, w_m\}$ be the set of customers' weights. Each requirement $r_j \in R$ has an associated development cost e_j, which represents the effort needed in its development. Let $E = \{e_1, e_2, \ldots, e_n\}$ be the set of requirements' efforts. On many occasions, the same requirement is suggested by several customers. However, its importance or priority may be different for each customer. Thus, the importance that a requirement r_j has for customer i is given by a value v_{ij}. The higher the v_{ij} value, the higher is the priority of the requirement r_j for customer i. A zero value for v_{ij} represents that customer i has not suggested requirement r_j. All these importance values v_{ij} can be arranged under the form of an $m \times n$ matrix. The global satisfaction, s_j, or the added value given by the inclusion of a requirement r_j in the next release, is measured as a weighted sum of the its importance values for all the customers and can be formalized as: $s_j = \sum_i^m w_i v_{ij}$. In every SE project it is common to find dependencies among the features suggested by the customers. Requirements dependencies mean that a set of constraints has to be considered during the requirement selection task, since they force us to check whether conflicts are present whenever we intend to select a new requirement to be included in the next software release. Several kinds of dependencies related to this problem are proposed first in [1] and later in [4]:

- *Implication or precedence.* A requirement r_i cannot be selected if a requirement r_j has not been implemented yet.
- *Combination or coupling.* A requirement r_i cannot be included separately from a requirement r_j.
- *Exclusion.* A requirement r_i can not be included together with a requirement r_j.
- *Revenue-based.* The development of a requirement r_i implies that some others requirements will increase their value.
- *Cost-based.* The development of a requirement r_i implies that some others requirements will increase their implementation cost.

These kind of dependences, that are reviewed in [20], are taken into account in some works about NRP such as [2] and [12]. Thus, the NRP main goal is to search for a subset of requirements $\hat{R}$ within the set of all subsets of n requirements $P(R)$, so the dimension of the search space is 2^n. A subset of requirements $\hat{R}$ can be represented in this space as a vector $x_1, x_2, \ldots, x_n$, where $x_i \in 0, 1$. If requirement $r_i \in \hat{R}$, then $x_i = 1$ and otherwise $x_i = 0$. In this way, the NRP can be considered as an instance of the 0-1 knapsack problem, and in consequence is a NP-hard problem [2] (it is unfeasible to tackle it using exact techniques to find the best solution in a polynomial time).

2.2 Single-objective NRP or Multi-objective NRP

The main goal of optimization problems is to search for the best solution with respect to several objectives. The quality of a candidate solution with respect to each objective is measured throughout the use a previously fixed evaluation function. According to the number of objectives, the problem can be classified as single-objective or multi-objective. Generally, in order to define the next software release, the main goal that we pursuit is to select a subset of requirements $\hat{R}$ from the candidate requirement list R, which maximize satisfaction and minimize development effort. The satisfaction and development effort of this subset $\hat{R}$ can be obtained, respectively, as

$$\text{sat}(\hat{R}) = \sum_{j \in \hat{R}} (s_j), \quad \text{eff}(\hat{R}) = \sum_{j \in \hat{R}} (e_j) \tag{1}$$

where j is an abbreviation for requirement r_j. As the resources available are limited, then development effort cannot exceed a certain bound B. First works [2, 12] in NRP tended to consider this problem as a single-objective problem: maximize customers' satisfaction within a certain development constraints. Their main goal is to find a subset of requirements that satisfies customer requests within a given resource constraints (i.e. availability of resources). That is to say, the selected subset of requirements $\hat{R}$ has to maximize customers' satisfaction within a given development effort bound B. Formally,

$$\begin{array}{l} \text{maximize sat}(\hat{R}) \\ \text{subject to eff}(\hat{R}) \leq B \end{array} \tag{2}$$

Most recent works [25, 9] consider NRP as a multi-objective problem (MONRP), since they consider at least two conflicting objectives; maximize customers' satisfaction and minimize the total effort involved in the development of the selected requirements. Formally, the NRP can be defined as the search for requirement subsets $\hat{R} \subseteq R$ such as

$$\begin{array}{l} \text{maximize sat}(\hat{R}) \\ \text{minimize eff}(\hat{R}) \end{array} \tag{3}$$

Other approaches (see [10]) formulate multi-objective in a different way, applying other criterion to measure customers' satisfaction or defining more than two objectives. In contrast to single objective optimization, which returns a unique solution, multi-objective optimization returns a set of solutions satisfying the proposed objectives. This means an advantage for the software developers as they can choose from a range of different alternatives. The set of non-dominated solutions that fulfill multiple objectives is denominated Pareto optimal front (see Fig. 1) . Whether any of the objectives of these solutions is improved, the others objectives will get worse.

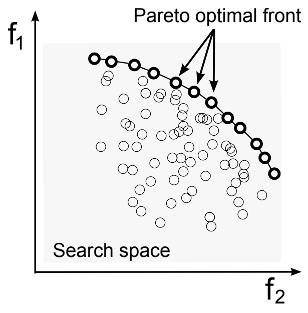


Fig. 1. Pareto optimal front considering two different objectives f_1 and f_2.

3 Analysis of Techniques

Once the problem has been formulated as a search problem and the fitness functions have been defined, metaheuristic techniques are be applied in order to find possible solutions. In the specific case of NRP, the metaheuristic techniques that can be found in the literature are: greedy algorithms, simulated annealing, genetic algorithms or ant colony systems. However, although all these approaches pursuit the same aim, not all of them deal with the NRP in the same way.

3.1 Simulated Annealing

Simulated annealing (SA) is an optimization algorithm which emulates the energy changes that occur in a system of particles when its temperature is reduced till the system reaches a state of equilibrium. At higher temperatures drastic changes in the system are allowed, whereas at lower temperatures only minor changes are allowed. This cooling scheduling has as goal to reduce the energy state of the system, taking the system from an arbitrary initial energy state to a final state with the minimum possible energy. Starting from an initial solution and an initial temperature T_0, the algorithm iterates following a cooling schedule function which decreases the temperature until it reaches a minimum T_{end}. Using some cooling functions, the algorithm stays at the same temperature for a certain number of iterations; then, it is decreased. In each algorithm iteration, a new solution from the neighbourhood is extracted and it can be accepted or not as the current solution. This technique allows to explore the search space at higher temperatures accepting poor solutions, whereas at lower temperatures only moves that improve the current solution are accepted. This algorithm uses an acceptance probability which determines whether a new solution found is accepted as the current one or not. Formally, let S be the current solution and S' be a new solution in the neighborhood of S, $S' \in nei(S)$ (it is said that S' is a neighbour of S, if they differ exactly on one requirement). Let T be the current temperature and $\varDelta E = f(S') - f(S)$ the energy difference between S and S', obtained after applying a fitness function. The probability of making the transition from the current solution S to the candidate solution S', i.e. the acceptance probability, is denoted by

$$p(S, S', T) = \begin{cases} 1, & \text{if } \Delta E > 0 \\ e^{\frac{\Delta E}{T}}, & \text{otherwise.} \end{cases} \tag{4}$$

SA has been applied to NRP by Bagnall et al. [2] and Baker et al. [3]. In contrast to most of the NRP approaches in the literature, which are focused on finding the optimal subset of requirements, the main aim searched in [2] is to find a subset of customers whose needs will be fully covered. Only implication dependencies are considered (i.e. a requirement that needs some others requirements to be implemented), defining precedence relationships for the candidate requirements. Baker et al. [3] focuses on a component selection problem of a software system from a telecommunications company. The aim of this work is to compare the component selection obtained from a group of human experts with the results obtained applying a search technique such as simulated annealing. In this case dependencies among requirements are not considered.

Table 1. Simulated annealing techniques applied to the NRP

	Fitness Function	Cooling Schedule	Initial Temperature	Parameters
[2]	$f(S) = \sum_{i \in S} w_i -$	Geometric $T_{i+1} = \alpha T_0$	$T_0 = 100$	$\alpha = 0.9995$ Iters. $= \{250, 500\}$
	$-\lambda \min\{0, B - \mathrm{eff}(S)\}$	Lundy and Mees $T_{i+1} = T_i/(1+\beta)T_i$	$T_0 = 100$	$\beta = \{5 \times 10^{-7}, 10^{-7}, 10^{-8}\}$
[3]	$f(S) = \mathrm{sat}(S)$	Geometric $T_{i+1} = (1-\alpha)T_0$	$-T_0 = \frac{\Delta E^{max}}{ln(1-p_1)}$	$\alpha = 0.2$ Iters. $= 15000$ $p_1 = 0.8$

Table 1 provides details about the fitness and cooling schedule functions applied in both works, and the parameters used for the experiments. Bagnall's approach [2] combines into a single fitness function two objectives based on the customers' weights and the total effort of the solution, since the requirement priorities are not gathered in this work. By contrast, Baker et al. [3] only takes into account the total satisfaction given by a solution to measure its quality. Applying the specified parameters, the results obtained by Bagnall et al. [2] show that a search technique such as SA is the best choice among the studied alternatives (greedy algorithms or hill climbers). On the other side experiments performed in [2] using the Lundy and Mees cooling schedule slightly outperforms the geometric function approach. Bakeret al. [3] compares SA to a greedy algorithm, and a selection performed by human experts. Best results are also reached by SA. This technique yields the best score in every experiment, followed by the greedy algorithm. Finally, the component selection specified by the human experts demonstrates to be much worse than the returned using SA.

3.2 Genetic Algorithms

A Genetic Algorithm (GA) [11] is a bio-inspired search algorithm based on the evolution of collections of individuals (i.e. populations) as result of natural selection and natural genetics. Starting from an initial population, their individuals evolve into a new generation by means of selection, crossover and mutation operators. This technique emulates the evolution process where best fitted individuals survive through generations. This evolution (i.e. iteration of the algorithm) is performed selecting some individuals according to their quality (measured by a fitness function) from the population. Then some parents are chosen and combined using crossover to produce new individuals (children). Finally, all the individuals in the new population have a certain but very small probability of mutation, i.e. their hereditary structure may be altered. The crossover and mutation operators are in charge of producing new individuals and they are applied with different probabilities i.e. crossover probability and mutation probability, denoted by P_c and P_m, respectively.

NRP addressed using GAs can be found in Greer and Ruhe [12], as a single-objective problem, whereas Zhang et al. [25] , Durillo et al. [9] and Finkelstein et al. [10] tackle the problem using a multi-objective approach, existing important differences among them.

Greer and Ruhe [12] addresses the requirement selection problem from a perspective based on agile methods, considering the iterations in the incremental software development. This work proposes an overall method for optimally allocating requirements to increments, which deals with a single-objective NRP as a combination of two different objectives: maximize the satisfaction and minimize the total cost of the solution. Precedence (implication) and coupling (combination) dependencies are considered and added to the problem as new constraints. The system provides the decision maker a small set of the most promising solutions that can be selected for the next software increment.

Zhang et al. [25] applies GAs to solve the NRP, using first synthetic data in [25] and real data in [24]. As Greer and Ruhe [12], two main goals related to benefit and effort are considered, although in this case the problem is addressed from a multi-objective perspective applying NSGA-II and ParetoGA algorithms. The first is a well-known multi-objective algorithm using an elitist strategy to preserve the solutions from the best front whereas the latter is an extension of the simple GA. Results reported by this work point to the NSGA-II method as the best choice; the solutions belonging to the Pareto front are better than the rest of methods evaluated and it offers a better diversity of solution distribution.

Durillo et al. [9] filled the gap left by this last work, arguing that the algorithms evaluation was performed in a visual way and no statistical analysis of the obtained results was provided. Using the same instances used by [25], they solve NRP by using a Random Search, and two multi-objective metaheuristics, NSGA-II and MOCell. In order to perform the analysis of the results, some quality indicators were used to measure the extent of spread of the set of solutions (i.e. spread) or the volume covered by the set of non-dominated solutions (i.e. hypervolume). According to the obtained results, Random Search results are generally poor, whereas NSGA-II and MOCell obtains good results presenting a similar performance in most of the cases. However NSGA-II outperforms MOCell when the experiment reaches the highest number of requirements.

Finkelstein et al. [10] focuses on satisfying the fairness term related to the requirements selection problem, whose main motivation is to "try to balance the requirement fulfillments between the customers". However, the task of finding this fairness does not result easy to achieve; hence three different multi-objective approaches are proposed. These proposals intend to maximize the satisfaction taking into account the number of fulfilled requirements per customer, the total satisfaction, or the percentage of satisfied requirement per customer. Two different algorithms, NSGA-II and the two-archive algorithm, are studied and applied on a set of real data from a telecommunication company.

Table 2 summarizes the techniques applied in each work and the parameters settings used by authors in the experimental evaluation.

Table 2. Genetic algorithms applied to the NRP

	Techniques	Selection	Crossover	Mutation
[12]	Single-objective GA	Probability curve based on fitness value	Random selection $P_c = \{0.1, 0.2, 0.3 \cdots, 1\}$	Random $P_m = \{0.05, 0.1, 0.15, \cdots, 1\}$
[25]	NSGA-II, Pareto GA, Single-objective GA	Tournament	Single Point $P_c = 0.8$	Bitwise $P_m = 1/n$
[10]	NSGA-II, The Two Archive	Tournament	Single Point $P_c = 0.8$	Bitwise $P_m = 1/n$
[9]	NSGA-II, MOcell	Tournament	Single Point $P_c = 0.9$	Random $P_m = 1/n$

3.3 Ant Colony Optimization

Ant colony optimization (ACO) is a meta-heuristic for combinatorial optimization problems proposed by Dorigo et al. [7], [6]. This technique emulates the cooperative behaviour of real ants in their task to find the shortest path from their colony to a source of food. This process is led by a substance called pheromone that ants leave on the floor as they move along their path. If other ants find and follow the same path, this pheromone trail will be stronger, attracting other ants to follow it. On the other side, the pheromone is periodically evaporated; therefore, the worse paths gradually lose their pheromone trail. Thus, what at first seems to be a random behaviour for ants, when no pheromone trail is present on the ground, turns into a movement influenced by the substance left by other ants in the colony.

The Ant System (AS) was the first ACO algorithm, proposed by Dorigo et al. [8]. Later, a new approach, called Ant Colony System (ACS) [7], was defined.

This approach introduced some changes related to the mechanism used by the ants to select the next vertex, and to update the pheromone.

In ACS, the NRP is represented as a fully connected directed graph. Vertexes represent the candidate requirements $r_i, r_2, \ldots, r_n$ and a pheromone τ is associated to the edges joining pairs of requirements. Ants traverse the graph vertex by vertex constructing a new solution, but their movement is driven by an equation based on the heuristic information and the pheromone values.

The pheromone update [6] is performed both locally and globally. The local update $\tau_{ij} = (1-\varphi)\tau_{ij} + \varphi\tau_0$ (where $\varphi \in (0,1]$ is a pheromone decay coefficient and τ_0 is the initial pheromone value) is applied by each individual ant only to the last edge traversed, when searching for its solution. Its main goal is to expand the search of subsequent ants during one iteration of the colony. The global update $\tau_{ij} = (1-\rho)\tau_{ij} + \Delta\tau_{ij}$ (where ρ is the pheromone evaporation rate and $\Delta\rho_{ij}$ is the amount of pheromone left in each arc), is performed by the ant that has found the best solution during an iteration of the colony. It is a kind of global memory of the colony that stores the best paths (solutions) found.

At the time of building a solution, the ants apply the pseudorandom proportional rule [6]: an ant moves from requirement i to j, depending on a random variable q (that is uniformly distributed on the 0 to 1 range) and a parameter q_0, such that if $q \leq q_0$, then $j = argmax_{l \in nei(i)} \tau_{ij}\eta_{ij}^{\beta}$, otherwise j is selected with a probability [6]

$$p_{ij}^{k} = \begin{cases} \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{h \in N_i^k}[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}, \text{ if } j \in N_i^k, \\ 0, \qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{5}$$

where the set of visible nodes, $nei(i)$, from the current vertex i is denoted by N_i^k. The heuristic information is defined by η_{ij}, whereas the pheromone accumulated in the edge i,j is represented by τ_{ij}. On the other side, the parameters α and β, reflect the relative influence of the pheromone with respect to the heuristic information.

Del Sagrado and del Águila [21] propose applying ACO to the requirement selection problem in the incremental development proposed by agile methodologies. The NRP using a single-objective approach is afforded in [22], and later in [21] applying a multi-objective perspective, defining this problem as NI-RSP (Next Increment Requirement Selection Problem). Both approaches are based on ACS. NI-RSP formulates a multi-objective problem seeking to maximize the total score, and minimize the total effort needed to develop. This technique is compared to other multi-objective optimization techniques, such as GRASP (greedy randomize adaptive search procedure) and NSGA-II, using some indicators to measure the quality of the Pareto front. The obtained results indicate that ACS can be applied efficiently to solve the requirement selection problem; its performance is very similar to NSGA-II and considerably better than GRASP. However, according to the quality indicators, it presents less oscillation in the number of non-dominated solutions.

4 Practical Application

This work has shown how optimization techniques applied to NRP let us find high quality solutions, in order to help developers during the requirement selection tasks. Once the applicability of these techniques has been demonstrated, they still have to be put in practice in real world software development. We strongly believe that having these search techniques available in a CARE tool would be considerably helpful for any development team at the time of dealing with the requirements selection.

InSCo Requisite [19] is a web-based tool early developed by our research group to manage the requirements of software development projects. Therefore, we propose an architecture (see Fig. 2) that integrates these techniques in the InSCo Requisite tool. The tool allows that a group of customers and developers works simultaneously in the same project, specifying the requirements of the system. Each requirement has an associated form which gathers its features, priorities and even a scenario or storyboard.

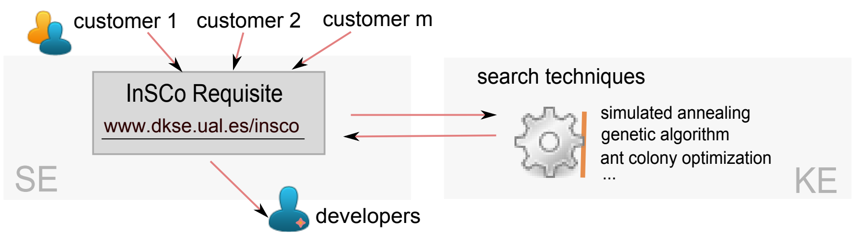


Fig. 2. Integration of search techniques in the architecture of the InSCo Requisite tool.

In order to facilitate the applicability of meta-heuristics algorithms, InSCo-Requisite must generate an interface file that contains all data needed during the execution of the simulated annealing, genetic or ant colony optimization algorithms.

The tool actually allows to export the whole set of specified requirements to an XML file. In a near future we plan to include development effort as a new property of each requirement. In this way, the resulting XML file could be easily used as input to any of the metaheuristic techniques applied in NRP. The result obtained by the metaheuristic techniques will be presented in the interface of InSCo Requisite and will serve as a feedback to developers when facing to the problem of planning the next software release.

5 Conclusions

The paper presented has provided a review of the metaheuristic techniques applied to the requirement selection problem, known as Next Release Problem

(NRP). This problem, within the Search Based Software Engineering (SBSE) discipline, was formulated in 2001 as a search problem and since then it has been addressed by many authors applying different search techniques: SA, GA and ACO. Table 3 summarizes the different works in the literature addressing the NRP, classified according to several factors as dependences are considered or not, and whether a single-objective or multi-objective perspective is applied.

Table 3. Classification of NRP related works

NRP single-objective		NRP multi-objective
With requirements dependences	Without requirements dependences	Without requirements dependences
Greedy, SA: Bagnall et al., 2001 [2]	*SA*: Baker et al., 2006 [3]	*GA*: Zhang et al., 2007 [25], Filkenstein et al., 2009 [10], Durillo et al., 2009 [9]
GA: Greer and Ruhe, 2004 [12]	*ACO*: del Sagrado et al., 2010 [22]	*ACO*: del Sagrado et al., 2009 [21]

Although each technique has been reviewed in an isolated way, since the comparison is not feasible when different datasets are applied, the results obtained by all of them demonstrate their applicability to the NRP. Finally, an integration of these techniques in an existent requirement tool has been proposed in order to take advantage of these techniques in Software Engineering.

Acknowledgments. This work was supported by the Spanish Ministry of Education and Science under project TIN2007-67418-C03-02 and by the Junta of Andalucía under project P06-TIC-02411-02.

References

1. van den Akker, M., Brinkkemper, S., Diepen, G., Versendaal, J.: Software product release planning through optimization and what-if analysis. Information and Software Technology 50(1-2), 101–111 (2008)
2. Bagnall, A.J., Rayward-Smith, V.J., Whittley, I.M.: The next release problem. Information and Software Technology 43(14), 883–890 (2001)
3. Baker, P., Harman, M., Steinhofel, K., Skaliotis, A.: Search based approaches to component selection and prioritization for the next release problem. In: Procs. 22nd IEEE Int. Conf. on Soft. Maintenance, 176–185. IEEE Computer Society (2006).
4. Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., och Dag, J.N.: An industrial survey of requirements interdependencies in software product release planning. In: Procs. 5th IEEE Int. Symp. on Requirements Engineering. p. 84–91 (2001)
5. Clarke, J., Dolado, J.J., Harman, M., Hierons, R., Jones, B., Lumkin, M., Mitchell, B., Mancoridis, S., Rees, K., Roper, M., et al.: Reformulating software engineering as a search problem. IEE Proceedings-Software 150, 161–175 (2003)

6. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. IEEE Computational Intelligence Magazine 1(4), 2839 (2006)
7. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Trans. On Evolutionary Computation 1(1), 53–66 (1997)
8. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Trans. on Sys., Man, and Cybernetics, Part B 26(1), 29–41 (1996)
9. Durillo, J.J., Zhang, Y.Y., Alba, E., Nebro, A.J.: A study of the multi-objective next release problem. In: Procs.1st Int. Symp. on Search Based Soft. Engineering. p. 49–58 (2009)
10. Finkelstein, A., Harman, M., Mansouri, S., Ren, J., Zhang, Y.: A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. Requirements Engineering 14(4), 231–245 (2009)
11. Goldberg, D.E.: Genetic Algorithms in Search and Optimization. Addison-wesley (1989)
12. Greer, D., Ruhe, G.: Software release planning: an evolutionary and iterative approach. Information and Software Technology 46(4), 243–253 (2004)
13. Harman, M., Jones, B.F.: Search-based software engineering. Information and software technology 43(14), 833 (2001)
14. Harman, M.: The current state and future of search based software engineering. In: 2007 Future of Software Engineering, 342–357. IEEE Computer Society (2007)
15. Harman, M., Mansouri, S.A., Zhang, Y.: Search based software engineering: A comprehensive analysis and review of trends techniques and applications. Tech. Rep. TR-09-03 (2009)
16. Johnson, J.: CHAOS chronicles v3.0. Tech. rep. (2003), `http://standishgroup.com/chaos/toc.php`
17. Karlsson, J., Ryan, K.: A Cost-Value approach for prioritizing requirements. IEEE Softw. 14(5), 67–74 (1997),
18. Kotonya, G., Sommerville, I.: Requirements Engineering: Processes and Techniques. Wiley (Aug 1998)
19. Orellana, F.J., Canadas, J., del Águila, I.M., Túnez, S.: INSCO requisite - a Web-Based RM-Tool to support hybrid software development. In: ICEIS (3-1), 326–329 (2008)
20. Ruhe, G.: Software release planning. In: Handbook of software engineering and knowledge engineering, vol. 3, 365–394. S K Chang (2005)
21. del Sagrado, J., del Águila, I.M.: Ant colony optimization for requirement selection in incremental software development. Technical Report, University of Almería (2009)
22. del Sagrado, J., del Águila, I.M., Orellana, F.J.: Ant colony optimization for the next release problem. a comparative study. In: Procs. 2nd Int. Symp. on Search Based Software Engineering (2010)
23. Sommerville, I.: Software engineering (6th ed.). Addison-Wesley Longman Publishing Co., Inc. (2001)
24. Zhang, Y., Finkelstein, A., Harman, M.: Search based requirements optimisation: Existing work and challenges. In: Procs. 14th Int. Conf. on Requirements Engineering: Foundation for Soft. Quality, 88–94. Springer-Verlag, Montpellier, France (2008)
25. Zhang, Y., Harman, M., Mansouri, S.A.: The multi-objective next release problem. In: Procs. 9th Ann. Conf. on Genetic and Evol. Computation, 1129–1137. ACM, London, England (2007)

Author Index