KNOWLEDGE SUPPORT FOR SOFTWARE PROCESSES

Michal Košinár, Svatopluk Štolfa and Jan Kožusznik
VŠB - Technical University of Ostrava, Faculty of Electrical Engineering and Computer Science
17. listopadu 15, 708 33, Ostrava, Poruba, Czech Republic

Keywords: Software Process Improvement, Knowledge, Rules, Facts, Knowledge Base, Data Mining, Software Process.

Abstract: Documented software processes and their assessments are the basics of modern software development. Nowadays, the semantic web, knowledge bases and knowledge management support many applications, but still their application within software processes (and business processes generally) are surprisingly being ignored. In this paper we focus on applying a knowledge layer into software processes and on the design of such a knowledge base. After a brief description of some classical fundamental approaches to software processes and knowledge bases, we propose an improvement based on the application of a machine readable knowledge base. We focus, in particular, on optimizing and enhancing software processes and their assessments with the knowledge layer.

1 INTRODUCTION

The main goal of every software company is to develop high-quality software with a minimal cost of development. One way to assure this is to follow good practices that are described in software processes. Since software processes show how to build software, they are therefore an integrated part of every software company. Every software company uses some type of software process. Even if this process is undocumented and/or unknown, it is still there (Thayer, 1997).

Describing and maturing the software process is a key element of a company's strategy, because a more mature software process means higher quality and more inexpensive software. Maturity of the software process is recognized through the assessment and its evaluation. According to the reference standards, a company's software process maturity is rated at a level from 0 to 5 (SEI, 2002). If the company wants to have a more mature process, the process must follow appropriate good practices for a higher level (Makinen, 2008).

Building a knowledge base that describes the practices in a company is an essential practice that assures that everybody in company knows what to do. Nowadays, almost every company has some type of human readable knowledge base (HRKB) that describes a variety of practices in the company. There even exist human readable knowledge bases that describe reference software processes and/or good practices (Alexandre, 2008). What is still missing in this area is the systematic usage of the machine readable knowledge bases (MRKB) and appropriate automated knowledge management.

In this paper, we are going to describe an application of a machine readable knowledge base for the support of a basic assessment of software processes. The assessment result is the evaluation of a comparison between the real software process and the reference software process. This type of assessment and its evaluation is only one part of our proposed comprehensive approach for an assessment and enhancement of the software processes. The aim of this paper is to present this particular part and discuss the integration into our comprehensive knowledge supported approach for the assessment and enhancement of software processes.

This paper is organized as follows: section 2 describes the basics of software processes; section 3 introduces the concept of creating, sharing and comparing knowledge bases. In section 4 we briefly present our process of semi-automated assessments and evaluation of the similarity of the software process to the reference process. Section 5 concludes and discusses future work.

Košinár M., Štolfa S. and Kožusznik J. (2010).
KNOWLEDGE SUPPORT FOR SOFTWARE PROCESSES.
In *Proceedings of the Fifth International Conference on Evaluation of Novel Approaches to Software Engineering*, pages 105-111
DOI: 10.5220/0002999801050111
Copyright © SciTePress

2 SOFTWARE PROCESS

Business processes represent the core of company behavior. They define activities which companies (their employees) perform to satisfy their customers. For a definition of the term *business process*, we use the definition from (WfMC, 1999): "*Business process* is a set of one or more linked procedures or *activities* which collectively realize a business objective or policy goal, normally within the structure defining functional roles and relationships." A process of IS development is called *the software process*. *The software process* is also a kind of business process but it has its specific aspects.

Software engineering is a discipline that is involved in software process development and maintenance (Humphrey, 1995). The main purpose is risk and cost reduction during software development. There exist many methodologies in the domain of software engineering but they could be divided specifically into two areas:

- Software development methodologies – they are the system of methods for software product development,
- Project management methodologies – they are the system of methods for project management.

Software development methodologies often divide software process into separate disciplines:

- requirement specification,
- analysis and design,
- implementation and testing,
- deployment,

There are two base kinds of software process models:

- waterfall model,
- iterative model

During the *waterfall model*, every discipline takes part after the previous discipline ends. The idea of the *waterfall model* is adopted from civil engineering and was popular in the beginning of software engineering.

The *waterfall model* has a big disadvantage, because implemented software is visible only after the end of the whole process. However, it is a practical experience requirement and the needs of stakeholders, users or investors could be changed. This issue solves the *iterative model*. The software development process consists of several iterations. Every iteration contains all disciplines – from requirement specification to deployment. Only selected requirements – based on priority – are implemented and after finishing the iteration the evaluation takes its part. Results of the evaluation come as input in the next iteration and it can modify or add requirements. The paradigm of the *iterative model* uses such robust process frameworks as *Rational Unified Process* or other lightweight process frameworks or methodologies – XP programming, ICONIX and etc…

3 KNOWLEDGE BASE

A knowledge base serves as artificial memory (Brachman, 2004). The content of the knowledge base consists of rules (obtained from reasoning or domain ontologies) and facts (the state of the system and its environment). We will use knowledge bases and ontologies as a basic building block for documenting and evaluating software processes. In this section, fundamentals are described to understand our approach to knowledge (Ciprich, 2008) and (Frydrych, 2008).

Details on how we can transform a human readable knowledge base and how the rules and facts are stored, reconstructed and managed in the knowledge base are described in the next section.

3.1 Human Readable Knowledge Base

This section contains fundamentals of human readable knowledge bases that are used in organizations to document, share and evaluate knowledge.

All of the pros and cons are shown the name. As it is human readable, it is also created by humans. It is simple to understand and share the contents. A big disadvantage also comes from this property – contents of such knowledge bases are typically vague, and suffer from the lack of details.

3.1.1 Documentation

The most commonly used tools to document knowledge in organizations are Wikis, project portals and typical FAQs. Typical examples are applications like web documentation of RUP, portals like MSDN and Wikipedia.

3.1.2 Sharing

Sharing human readable knowledge means typical communication that can be personal (meetings, brainstormings, conferences etc.), combination of personal and electronic (video calls, conference calls etc.), or pure electronic (e-mail client, Exchange, Lotus etc.).

3.1.3 Evaluation

With all the advantages that human readable knowledge bases offer, the assessment is a really big disadvantage. Evaluating a process described in such a knowledge base means manually comparing documented knowledge with a real process.

3.2 Machine Readable Knowledge Base

The basic difference between a human readable and machine readable knowledge base is the use of an inference system in a machine readable knowledge base. Even comparing two or more machine readable knowledge bases is easier because we can use smart algorithms instead of a manual comparison.

So far so good, these possibilities are however dependent on the domain ontology of such knowledge bases. Let's presume that we have a base ontology describing a general software process (terms like request, use case, change, sequence diagram, class etc. and basic relation among them) and one specialized ontology which is an extension of the base ontology. The specialized ontology will serve as a domain ontology for a concrete software process formal semantic description and will allow us to build a knowledge base for assessment purposes. More information about enhancing software processes with knowledge bases and semantic tools is in the section 4.

3.2.1 Creation and Storage

The first difference is that instead of "documenting" the knowledge in human readable knowledge bases we are "creating and storing" it to fill the base with rules and fact. All knowledge must be based on domain ontology and has to be written in some formal language (i.e. Prolog, OWL DL, TIL-Script).

Still, knowledge must be inserted into the base by an expert, as in the case of human readable knowledge bases.

More about formal languages used in knowledge bases in (Ciprich, 2007), (Ciprich, 2008) and (Frydrych, 2008).

3.2.2 Sharing

Sharing as we know it from human readable knowledge bases is out of the question because we are not building machine readable knowledge bases to share the knowledge among experts like in their human readable siblings' case.

Moreover, there are ways to share knowledge stored in a machine readable base. One possibility is to use such a base as a base memory of an intelligent software agent in a multi-agent system. The second way is to translate the knowledge into natural language (using TIL-Script and natural language semantic web like WordNet). The third way is to read the content of the base in its plain form; however, this is possible only for those who know the language used to formalize the content.

3.2.3 Evaluation

The difficult way to build machine readable knowledge base for software processes has its positives. We can evaluate knowledge content using a machine comparison of two (or more) knowledge bases that can be done more easily than a manual comparison of two human readable knowledge bases. In the next section of this paper, this idea is described in more detail.

3.3 Knowledge Base Comparison

As mentioned above, we can compare contents of two knowledge bases.

To do this correctly, we must define one basic condition that will allow us to do that

- **every member of knowledge base must be comparable with another**

This basic requirement is fulfilled as we are using a homogenous data model to store knowledge (every member of the set is stored as a string of text) and usage of more content languages in two bases which will be compared is prohibited.

Now, we can handle knowledge bases as with typical unordered sets (of rules and facts) so it allows us to perform operations like union and intersection.

So far, a simple relation to compare two knowledge bases can be defined.

Definiton 3.4.

Similarity (A, B) = | intersection (A, B) | / | union (A, B) |

This function can compare only KBs where at least one has no zero cardinality..The function returns a number from interval <0 ;1>. This fuzzy value is the value of two knowledge bases similarity.

4 KNOWLEDGE SUPPORT FOR SOFTWARE PROCESSES

The idea of knowledge support for the assessment and evaluation of software processes is based on the fact, that according to us, the assessment, enhancement and monitoring can be supported by the creation and usage of machine readable knowledge bases. A lot of manual procedures can be automated. The goal is to create a more effective environment for the assessment, enhancement and monitoring of software processes.

One of the many tasks of this domain area is the comparison between the reference software process and real software process that is used in the company. The issue is to find the similarity between the real software process and the reference software process and provide an evaluation of the current state. Typically, the real process is assessed and human readable knowledge bases are searched for similarities. Everything is performed manually. Our proposed approach shows the possibility of using a machine readable knowledge base for the automatic evaluation of the similarity between the real and reference processes.

Our approach can be basically described as follows:

1. The first step is the creation of the particular reference knowledge base - Knowledge base transformation.

Documented software processes that are described in the human readable knowledge bases are analyzed and the particular ontology for each software process is created. Next, the ontology and knowledge obtained from the HRKB is transformed into the machine readable knowledge base. A new MRKB is created for every type of the reference software process.

2. The second step is the study and creation of the real knowledge base and the comparison of the reference and real knowledge base - enhancing software processes with knowledge management.

A real software process is modeled and the ontology for this process is created. The created ontology must then be mapped into the reference software process ontology. The mapped ontology set and the knowledge obtained from the model is then transformed into the machine readable knowledge base.

Both knowledge bases are then automatically compared and the result is a number that shows the similarity of real and reference software processes.

It is obvious that the first step can be performed only once for every reference software process. The step is then applied for every real software process that we want to evaluate.

4.1 Knowledge Base Transformation

We have already sketched a basic idea of how to transform a human readable knowledge base into its machine readable clone. Now, it is time to refine it into details.

At first, we should explain the basic reason for transforming the human readable KB into the machine readable KB. The answer is really simple – if we want to evaluate a process that is based on some reference software process we would have to manually compare it with the reference model process described in its human readable knowledge base. With knowledge enhancements and basic approaches described above, we can sort this problem automatically – by transforming the reference knowledge base into the machine readable knowledge base (Fig.1).

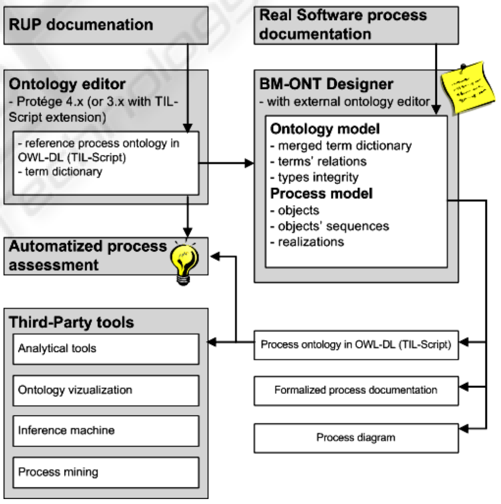

Figure 1: Scheme of the machine readable knowledge base creation – RUP example.

This can be done by building an ontology for a concrete domain (i.e. a software process) using the human readable reference documentation. Then, the knowledge base is defined by such ontology's content. We can see particular transformation in greater detail in figure 2.

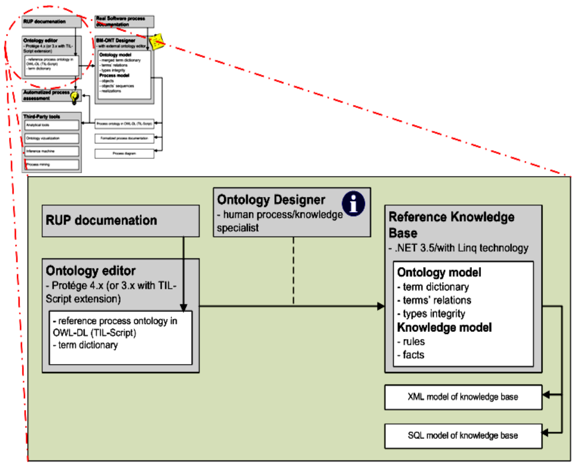
RUP documenation
Ontology Designer
- human process/knowledge specialist
Reference Knowledge Base
- .NET 3.5/with Linq technology
Ontology editor
- Protége 4.x (or 3.x with TIL-Script extension)
- reference process ontology in OWL-DL (TIL-Script)
- term dictionary
Ontology model
- term dictionary
- terms' relations
- types integrity
Knowledge model
- rules
- facts
XML model of knowledge base
SQL model of knowledge base

Figure 2: Knowledge base transformation.

An ontology builder has to describe the domain to the ontology data file (using tools like Protégé etc.) and then fill the reference knowledge base with rules and facts (using some content language like OWL-DL, TIL-Script or PROLOG) describing the reference domain (using terms and relations from domain ontology).

4.2 Building Process Definition with Ontology Background

Example: Let's say that we want to transform a part of RUP's "Change request" process. The simple example of created ontology below describes types of Change request and Change manager where the Change request is a class of all possible instances of change requests and Change manager class of all change managers - individuals (as defined in RUP's documentation). As in real world, both types have some properties – we must define them as well by using Object Property definition. These properties allow us to define a relationship among all described types. In our example we define that Change manager is associated with Change request via "Is manager of" relation and vice versa Change request is connected with manager by "Has manager" relation. We can also use primitive types as Integer, Boolean etc. in Object properties as in "Is approved" property of Change request that says whether the request was or was not approved. However more appropriate usage of primitives are in Data properties of types like in "Change request number" property of Change request. Then we have to associate the Data property to Class with defining an Individual – the concrete instance of the property associated with Class (the Change label that identifies the change request in whole system). In last example we define an individual as instance of Change manager associated to Change request via Change label identification.

```
Class(pp:changeRequest)

Class(pp:changeManager partial
     restriction(pp:uses
     someValuesFrom(pp:ChangeManagementArt
     efacts)))

ObjectProperty(pp:isManagerOf
     domain(pp:changeManager)
     range(pp:changeRequest))

   ObjectProperty(pp:isApproved
domain(pp:changeRequest) range(xsd:bool))
```

```
ObjectProperty(pp:hasManager
     domain(pp:changeRequest)
     range(pp:changeManager))

   DataProperty(pp:changeRequestNumber
range(xsd:integer))

Individual(pp:ChangeLabel
     type(pp:changeRequest)
     value(pp:changeRequestNumber
     "007"^^xsd:integer))

Individual(pp:Michal type(changeManager)
     value(pp:isManagerOf pp:ChangeLabel))
```

Now, with defined ontology background of some RUP process part, we can build a process definition with storyboard method with strong formal system of semantic annotation in every item's background.

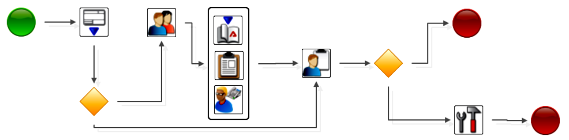

Figure 3: Storyboard diagram example from Storyboard designer application.

4.3 Enhancing Software Processes with Knowledge Management

The ability of transforming human readable knowledge bases of documented software processes into machine readable knowledge bases means that we have a tool for the automatic assessment of software processes.

This can be performed with the creation of two knowledge bases.

i. One knowledge base that holds the optimal software process transformed from human readable base holding the process' documentation (RUP documentation on the Web). Let us name this knowledge set as a "Template".

ii. The second knowledge base that holds an actual software process that is used in an organization. This knowledge base must be filled by an ontology expert in processing and consulting services using the same ontology dictionary as the "Template". The name of this set will be the "Actual".

Now, when we have two knowledge bases whose contents are documented and real software processes we can use the function from ***Definition 3.4*** to evaluate the similarity of them.

An individual content comparison can be performed as an add-on to the assessment. We can search for differences directly between individual members of both sets. However, this means basically a brute-force comparison of two sets, which is not optimal. Results of such a comparison can tell us where gaps are in the "Actual" knowledge base of a process compared to "Template". The way to search for such differences is the main task of our future research.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we presented our new approach for the semi-automated assessment and evaluation of software process similarity. The comparison of the real and reference software processes is done by the usage of machine readable knowledge bases. The template (reference) knowledge base describes the reference software process and the actual knowledge base (assessed process) describes the software process in the company. The template knowledge base is created in advance, using specific ontology for the particular software process and other techniques that are necessary to build a machine readable knowledge base. The actual knowledge base is created during the assessment process. Both knowledge bases are then compared and the result is the number that represents percentage similarity.

This presented process is one part of our comprehensive approach to the assessment, evaluation and improvement of software processes. The first step is the extension of the presented process to the automated comparison of specific parts of software processes. The next steps are then e.g. automated comparisons with more than one specific process at once, involvement of process modeling to the approach that will be used for the automated search, evaluation and improvement suggestion using the template software process models etc… A lot of future work is needed to solve all the problems that arise during the development of this new approach. Our work is also supported by the experience that is gained through the practical experiments of this approach in real software companies.

Although, according to our preliminary use cases studies, this approach seems to be very promising, the further use case studies are needed to continuously develop and enhance the approach and support its inclusion into the software process assessment models and improvement techniques.

ACKNOWLEDGEMENTS

This research has been supported by internal grant agency of FEECS VSB-TU Ostrava - IGA 22/2009 Modeling, simulation and verification of software processes. Author Michal Košinár is also grand aided student of Ostrava City Authority, Czech Republic.

REFERENCES

Alexandre, S. Makinen, T., and Varkoi, T. *Implementation of a Software Process Standard as an Electronic Process Guide*. In proceedings of SPICE 2008 Conference (Software Process Improvement and Capability dEtermination), 26-28 May 2008, Nuremberg, Germany.

Ronald Brachman and Hector Levesque. *Knowledge Representation and Reasoning (The Morgan Kaufmann Series in Artificial Intelligence)*. Morgan Kaufmann, May 2004.

Ciprich, N., Duží, M., Košinár, M.: *TIL-Script: Functional Programming Based on Transparent Intensional Logic*. In *RASLAN 2007*, Sojka, P., Horák, A., (eds.), MasarykUniversity Brno, 2007, pp. 37-42.

Ciprich, N., Duží, M. and Košinár, M.: *The TIL-Script language*. In the Proceedings of the 18th European Japanese Conference on Information Modelling and Knowledge Bases (EJC 2008), Tsukuba, Japan 2008.

Frydrych, T., Kohut, O., Košinár, M.: *Transparent Intensional Logic in Knowledge Based Multiagent Systems*. In *RASLAN 2008*, Sojka, P., Horák, A., (eds.), MasarykUniversity Brno, 2008.

Dragan Gasevic, Dragan Djuric, and Vladan Devedzic. *Model Driven Architecture and Ontology Development*. Springer, July 2006.

Humphrey, W. S. 1995 *A Discipline for Software Engineering*. 1st. Addison-Wesley Longman Publishing Co., Inc.

Makinen, T., and Varkoi, T. *Assessment Driven Process Modeling for Software Process Improvement.* In proceeding of PICMET'08 Conference, 27-31 July 2008, Cape Town, South Africa 2008

Standish Group International. *The Chaos Report* [online]. Boston: The Standish Group International, 1994 – [cited on 17. Sept. 2006]. Available on http://www.standishgroup.com/sample_research/chaos_1994_1.php

Workflow Management Coalition, *Terminology & Glossary*, The Workflow Management Coalition Specification, February, 1999

Software Engineering Institute: CMMI staged-version 1.1 (2002), http://www.sei.cmu.edu/cmmi/

Thayer, R. H., *Software System Engineering: An Engineering Process*, Software Requirements Engineering, R. H. Thayer and M. Dorfmann, Eds., IEEE Press, Los Alamitos, CA 1997.