The support concept of software development iteration management

Viktors Krasovskis, Ivans Samusenko, Artis Teilans, Oegs Uzga-Rebrovs

Faculty of Engineering, Rezekne University College of Applied Sciences, ITC, Atbrīvošanas aleja 115, Rēzekne, LV-4601, Latvia
viktors.krasovskis@gmail.com

Abstract: The management process is an important part of almost every software development project. It is a common practice that the same manager is simultaneously involved in multiple projects. It is difficult to focus attention in time to the stages of particular projects, which have increasing problems and risks. Thereby communication between development and management processes is an essential aspect of a successful project. There is a necessity for automatic communication mechanism between development and management processes. Modern development processes are usually divided into multiple iterations, which consist of tasks in order to achieve more transparency and control. The application of ontologies to define the task structure can make it interpretable for data mining. Data can be analyzed using artificial intelligence or modern declarative programming. Thereby, the attributes of current and completed tasks can be analyzed by an applied information system, which will support and/or warn managers to focus on particular projects, basing on the gathered facts, if necessary. Through focusing on the projects, which require management, decision making speeds up. The artefacts of collected development tasks are not wasted and are used to design new standards of tasks and iterations, which are based on successful previous attributes. It can lead to less management. Considering that some projects are almost self-managed and management tasks are applied to the projects, which require management, fewer human resources are necessary. It also positively affects the quality and strictness of management process.

Keywords: data mining, decision making, iteration, management, ontology, software development, support.

Introduction

Few of significant software development problems are process management, qualitative risk evaluation and well-timed problem solving reaction. Since most processes are managed by humans, there is a high probability for human risk factor. Many software methodologies give common recommendations, but do not describe solutions for particular problems. Respectively almost any methodology is reduced to the fact that development management is grounded on experience from previous projects. This approach is good to solve typical problems, which could be met in the past, but the human factor does not always allow identifying the problem in time. Since artefact quantity is growing alongside with formalization in almost every middle-sized project, project management will suffer from limited human capabilities extract non-obvious facts from huge amount of data. Thereby projects, which are led using iterative development methodologies, are also affected (Pugh, 2010). Successful run of all iterations is significant to whole project - overall risk factor does not rise. So it is necessary to introduce a semi-automated software development management process, which would allow recognizing problems in time. This paper will describe a support concept of software development iteration management, which is based on Semantic Web and decision making technologies (Allemang and Hendler, 2008).

Materials and methods

One of the ways to improve software developments iterations is to study the development process and to find entry point for improvements. Considering that a support concept of development process iteration management is the main idea of this paper, an abstract iterative development process model needs to be studied. The IDEF0 standard is capable to model this process as shown in Fig. 1 and Fig. 2.
The process model allows better understanding where it can be improved or changed. One of goals is not only to determine the optimal process modifications for interaction with an expert system for management improvements, but also to determine mechanism and control significance in particular subprocesses. As mentioned previously an expert system would help to manage the development process. The process itself develops a lot of artifacts, which need to be collected. Later the collected data could be used in the expert system to generate possible solutions for particular problem solving. Data storage in such solution is very significant. It is quite problematical to store artifact data in a relational database because data needs to be prepared before it could be recorded into table. On the other hand also a data extraction problem persists - expensive data mining technologies are needed to use reasoning capabilities. Substantially relational database data mining is based on statistical algorithms, which are good for numerical data processing, but not for other information like triples. One of the main problems in iteration management is to choose and evaluate right iteration metrics. The data must provide the state of iteration in various contexts e.g. completeness, overall delay, bug quantity dynamics, resource consumption efficiency and overall iteration planning success.

Results and discussion

An abstract iterative development process was chosen to model the interaction between process itself and the expert system (Fig. 2 and Fig. 3). The artifacts gathered from the development plan would be used in information gathering for further iteration control. This subprocess is controlled by the Semantic Web standards and a certain methodology. In this case the main resource is local semantic data storage. The subprocess results the necessary information, which would be used in an expert system to generate a list of possible decisions.
The iteration execution is also controlled by a certain methodology and by the project’s development plan. The third control element is a separate management decision, which is a result of current iteration state rating (Fig. 4).

The iteration rating is a semi-automated subprocess where both experts and an expert system are involved. Live data from current iteration execution is used for input, while control comes from semantic data storage instance. The expert system can contain some predefined rules, which will be used for learning and possible decision generation. If the expert system recognizes a suspicious situation it would start to try generate possible decision, which will be passed to final decision making subprocess where a human manager (also an expert) with help of senior developer will choose the optimal decision if one is present. In opposite case the decision can be made based on personal knowledge and experience (Fig. 5).

There are following advantages to use a semantic storage to store data for process automation:

- stores data entities
- reasoning support
- SPARQL (DuCharme, 2011)
- no need to develop complex database schemas
- simple and extended rule definition using RDFS/OWL (Klyne and Carroll, 2002; Brickley et al., 2004)
- many platform-independent storage solutions.

Some current triplestore software products were analyzed for their capabilities (Table 1). They all support SPARQL query language, but only few have RDFS/OWL (Dean and Schreiber, 2004) reverse reasoning support. No all solutions have a built-in Web interface for querying and/or storage management. Only the Virtuoso
universal server has graph storage for triples, which can work in combination with data in relational database. Virtuoso is the only product that supports SQL for additional data management. An essential feature for a triplestore is a built-in SPARQL endpoint, which provides an interface for data querying through other applications or systems.

### Comparison of triplestore solutions

<table>
<thead>
<tr>
<th>Product</th>
<th>SPARQL support</th>
<th>RDFS/OWL reasoning support</th>
<th>SQL support</th>
<th>Web interface</th>
<th>Built-in SPARQL endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>4store</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes, limited</td>
<td>Yes</td>
</tr>
<tr>
<td>ARC2</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Bigdata</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Mulgara</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes, limited</td>
<td>Yes</td>
</tr>
<tr>
<td>Virtuoso</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RDF::Trine</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Redstore</td>
<td>Yes</td>
<td>No</td>
<td>Yes, limited</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Redland</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Some object class prototype models were developed, which are involved in the development process. These class objects are described using the XML and RDF standards. Almost in every project the atomic piece of every development process is a task (Fig. 6). Any class instance can contain properties, which are essential to subprocess, where they are used.

```xml
<task:Task rdf:about="http://www.example.org/t3451">
  <task:duration>00:55</task:duration>
  <task:estimationTime>02:00</task:estimationTime>
  <task:complete>70%</task:complete>
  <task:type>1</task:type>
  <task:bugsCaused>1</task:bugsCaused>
  <task:customerMark>6</task:customerMark>
  <task:customerAttention>00:15</task:customerAttention>
  <task:complexity>3</task:complexity>
  <task:priority>1</task:priority>
  <task:iterationSuccessInfluence>0</task:iterationSuccessInfluence>
  <task:hasCustomer rdf:resource="http://www.example.org/p645"/>
  <task:hasOwner rdf:resource="http://www.example.org/p376"/>
</task:Task>
```

Fig. 6. **Task object example in RDF/XML format.**

Objects can contain other objects as properties e.g. a task can contain multiple owners and customers, but an iteration would contain multiple tasks. The `rdf:about` property describes the object’s unique URI, which at the same time is an unique identifier. In turn the `rdf:resource` property indicates the URI of an object in a parent object property. So iterations may contain tasks e.g. using `hasTask` properties (Fig. 7). Thereby objects form relationship with each other or in other words they form a graph.

```xml
<iteration:Iteration rdf:about="http://www.example.org/i343">
  <iteration:hasTask rdf:resource="http://www.example.org/t345"/>
  <iteration:hasTask rdf:resource="http://www.example.org/t3451"/>
  <iteration:deadline>2012-11-25 12:00</iteration:deadline>
  <iteration:start>2012-11-21 12:00</iteration:start>
  <iteration:actualStart>2012-11-21 12:00</iteration:actualStart>
  <iteration:actualEnd>2012-11-25 12:00</iteration:actualEnd>
  <iteration:successful>1</iteration:successful>
  <iteration:description>DS model development</iteration:description>
  <iteration:hadManagementIntervention>1</iteration:hadManagementIntervention>
</iteration:Iteration>
```

Fig. 7. **Iteration object example in RDF/XML format.**
Using the RDFS and/or OWL standards it is possible to create schemas, which may describe the rules between object interactions. These rules can be stored separately in a triplestore and be applied to different graphs. The schemas essentially help the reasoning process for gathering non-obvious facts (Fig. 8).

**Fig. 8. RDFS schema example (graph visualization) for the Task and Iteration classes.**

These facts are used in the expert system for decision generation. Since the conditions where decisions have to be made are often variable the expert system has to be dynamic so it will be constantly retrained on new solutions (Shachter and Bhattacharjya, 2010). It is not excluded that some static methods will be used (e.g. correlational evaluation, chi-square) to search for related factors. Classification decision trees, which are intuitive and are used for rules gathering, could be used for information presentation to the end user.

While making decisions in various situations it is possible to face such proposed or being taken solution rating criteria like authenticity and evaluation of the accruing predicted source situation. It is very important to determine the kind of a being taken solution while rating the risk factor of previously mentioned criteria e.g. is it a undeniable key solution or it is a recommendation like solution, which needs further and more deep analysis.

Notice that automated decision process unlikely will be used to make decision for solving trivial tasks. On turnover it would be more reasonable to use it to solve tasks from which material, financial values and resources depend. In regret the trust level to suchlike systems is noticeably low due banal skepticism. It is explainable with a fact that only the process owner can take the risk, which can not only impact favorable outcome of current, but also of future processes (Prakash and Shenoy, 1994; Shachter and Bhattacharjya, 2007).

**Conclusion**

Currently there are many triplestore solutions, which can be used to store and query the stored data. Unfortunately, these features do not provide the product’s ability to be used in process automation where reasoning support and advanced management options are essential. At this point Virtuoso server is one of products, which meets most requirements. Future product tests are needed to choose an optimal solution for the practical realization of discussed concept.

Probably a model of decision making, which has to solve a question like task with an answer, is not one of the best suited method to make decisions at all. Respectively the best made decision is a result of systematic and detailed analysis of given problem, during which the problem solution is not so mandatory like the problem understanding point itself. In this case for the individual who makes decisions the final solution is not so important like the possibilities for understanding the fact why the particular solution was selected, but not another one.

The ability to use not always obvious key factors for decision making could significantly improve the trust level of the expert system. Since final decisions are almost always made by a user, he will be in an expert role, who will train the system. The ability to store knowledge in series is preferable in many aspects than application of static algorithms. The dynamically built knowledge base could significantly lower the risk level of taking wrong or not enough suitable decisions as the system will store both right decisions and ones, which could lead to negative consequences.

The future perspective of current research includes the following tasks:

- To develop common ontologies for project methodology, project management tools and expert system support.
- To shape process object model classes closer to real-world process examples.
- To develop an expert system, based on dynamic algorithms for decision generation. The semantic storage’s SPARQL endpoint would ensure communication with the expert system not only for data querying, but also to update data e.g. save the decision experience for the future use.

**References**


