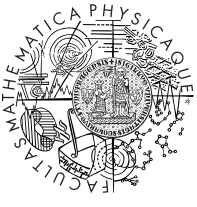
Charles University in Prague, MFF, Department of Software Engineering
Czech Technical University in Prague, FEE, Dept. of Computer Science & Eng.
VŠB–TU Ostrava, FEECS, Department of Computer Science
Czech Society for Cybernetics and Informatics

Proceedings of the Dateso 2012 Workshop

DATESO 2012

Databases, Texts, Specifications, and Objects

http://www.cs.vsb.cz/dateso/2012/
http://www.ceur-ws.org/Vol-837/



Supported by

http://www.mirlabs.org/

http://arg.vsb.cz/ieee-smc/

April 18 – 20, 2012
Zernov, Rovensko pod Troskami

DATESO 2012

Technical editor:
Pavel Moravec, pavel.moravec@vsb.cz
VŠB – Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Computer Science

Page count:	164
Publication:	365^{th}
Impression:	100
Edition:	1^{st}
First published:	2012

This proceedings was typeset by PDFLaTeX.
Cover design by Pavel Moravec (pavel.moravec@vsb.cz) and Tomáš Skopal.
Printed and bound in Ostrava, Czech Republic by TiskServis.

Published by MATFYZPRESS publishing house of Faculty of Mathematics and Physics
Charles University in Prague, Sokolovská 83, 186 75 Praha 8, Czech Republic
as its 365^{th} publication.

Workshop Partner

DATESO 2012 workshop proceedings was supported by SoSIReCR project (CZ1.07/2.4.00/12.0039). The project is co-financed by ESF and Czech state budget.

SoSIReČR

SOCIAL NETWORK OF COMPUTER SCIENTISTS IN THE REGIONS OF THE CZECH REPUBLIC

SoSIReČR

Social Network of Computer Scientists in the Regions of the Czech Republic

The project aims to promote communication and establish cooperation between the Czech IT communities in academic and corporate sectors. The main output will be a Web portal that will build a social network of computer scientists., which will differ from the usual social and professional networks as Facebook or LinkedIn by providing unique tools to search for IT specialists, to create teams (for joint projects, tenders), and in general to allow cooperation. The aim of the project activities is to increase the competitiveness of the Czech Republic in the field of computer science, as well as improvement of the status of IT in the Czech Republic and its contribution to society.

Personal interconnection of educational and research activities and the cooperation of academic and applied research are required for high-quality tertiary education and its fruitful cooperation with industry and Government. From the social network will benefit students and graduates, scientific personnel, schools, and businesses. It will be possible to search here for a job or an interesting project, skilled workers or whole teams, even the schools can compare their learning plans with the situation on the labor market. The social network portal SítIT will be available for general public in June, 2012 at https://www.sitit.cz/.

Another component of the project is to support personal meeting of the IT experts in the regions (both among themselves and with companies), the sharing of experience and the identification of needs. These are the conferences, regional seminars and workshops, aimed at improving communication between the academic, corporate and public sector. One of the following activities are for example Conversations with computer scientists (Hovory s informatiky), organised in the framework of this project by Institute of Computer Science, Academy of Sciences of the CR.

On the SoSIReČR project cooperate:

Faculty of Mathematics and Physics, Charles University in Prague
Institute of Computer Science, Academy of Sciences of the Czech Republic
Czech Technical University in Prague
The Faculty of Informatics and Statistics, University of Economics, Prague
Higher Vocational School and Technical High School, Šumperk

The social network portal SítIT will be at disposal at the address https://www.sitit.cz/.

Project (no CZ.07/2.4.00/12.0039) is running in period 01.11.2009 - 31.10.2012.

For more information please visit the **sosirecr** *web pages.*

INVESTMENTS IN EDUCATION DEVELOPMENT

Steering Committee

Jaroslav Pokorný	Charles University, Prague
Václav Snášel	VŠB-Technical University of Ostrava, Ostrava
Karel Richta	Czech Technical University, Prague

Program Committee

Jaroslav Pokorný (chair)	Charles University, Prague
Václav Snášel	VŠB-Technical University of Ostrava, Ostrava
Karel Richta	Czech Technical University, Prague
Vojtěch Svátek	University of Economics, Prague
Peter Vojtáš	Charles University, Prague
Dušan Húsek	Inst. of Computer Science, Academy of Sciences, Prague
Michal Krátký	VŠB-Technical University of Ostrava, Ostrava
Tomáš Skopal	Charles University, Prague
Pavel Moravec	VŠB-Technical University of Ostrava, Ostrava
Irena Mlynková	Charles University, Prague
Michal Valenta	Czech Technical University, Prague
Pavel Loupal	Czech Technical University, Prague
Martin Nečaský	Charles University, Prague
Jiří Dvorský	VŠB-Technical University of Ostrava, Ostrava
Radim Bača	VŠB-Technical University of Ostrava, Ostrava
Tomáš Knap	Charles University, Prague

Organizing Committee

Jaroslav Pokorný (chair)	Charles University, Prague
Irena Mlynková	Charles University, Prague
Martin Nečaský	Charles University, Prague
Pavel Moravec	VŠB-Technical University of Ostrava, Ostrava

Preface

DATESO 2012, the international workshop on current trends on Databases, Information Retrieval, Algebraic Specification and Object Oriented Programming, was held on April 18 – 20, 2012 in Zernov, Rovensko pod Troskami.

The 12th year was organized by Department of Software Engineering MFF UK Praha, Department of Computer Science and Engineering FEL ČVUT Praha, Department of Computer Science VŠB-Technical University Ostrava, and Working group on Computer Science and Society of Czech Society for Cybernetics and Informatics. The DATESO workshops aim for strengthening connections between these various areas of informatics, particularly this year, Sematic Web, semistructured data, social networks, and formal specifications.

The proceedings of DATESO 2012 are also available at DATESO Web site: `http://www.cs.vsb.cz/dateso/2012/` and CEUR Workshop Proceeding site: `http://www.ceur-ws.org/Vol-837/` (ISSN 1613-0073). The Program Committee selected 14 papers (11 full papers and 3 posters) from 22 submissions, based on three independent reviews.

We wish to express our sincere thanks to all the authors who submitted papers, the members of the Program Committee, who reviewed them on the basis of originality, technical quality, and presentation. We are also thankful to the Organizing Committee and Amphora Research Group (ARG, `http://www.cs.vsb.cz/arg/`) represented by Pavel Moravec, for preparation of workshop proceedings. Special thanks belong to Czech Society for Cybernetics and Informatics and the SoSiReČR project (Social Network of the Computer Scientists in the Regions of the Czech Republic) for their support of publishing this issue.

April, 2012 J. Pokorný, V. Snášel, K. Richta (Eds.)

Table of Contents

Full Papers

Posters

Invited Papers

On General-purpose Textual Modeling Languages

Martin Mazanec and Ondřej Macek

Department of Computer Science, FEL, Czech Technical University,
Karlovo namesti 13, Praha, Czech Republic
{mazanma3, macekond}@fel.cvut.cz

Abstract. Modeling is an important part of the software development process because it allows for a better understanding of the domain as well as an understanding of the software structure and function. Among general-purpose modeling languages dominate the graphical ones such as UML; textual modeling languages are not as popular though they have a big potential. In this paper we define the important features of textual modeling languages and then we compare existing general-purpose textual modeling languages according to these criteria to show if they meet their potential. Based on the comparison results and our experience, we propose our own modeling language called Earl Grey whose basics are presented in this paper together with our experience from creating this language.

Keywords: textual modeling, modeling languages, model driven development

1 Introduction

Modeling is an integral part of the software development process, where it helps to explain the static part of the system (data the software works with and software inner structures and states) and the dynamic part of the system (how the software works). A lot of modeling languages exists; the best known and most widespread is the Unified Modeling Language (UML) [12], which represents the so called general purpose modeling languages. Besides the general purpose modeling languages, domain specific languages (DSL) [3] exist, whose aim is to describe a concrete domain only, therefore their usage is limited. Nevertheless the DSLs are very popular nowadays, in contrast with UML a lot of DSLs are textual. This may be caused by the fact that general purpose language is primarily focused on the possibility to describe many various problems, where the graphical representation could be helpful even if it could cause a problem later with the ambiguity of graphical structures and their meanings [2]; moreover many graphical models are extended by some textual information which completes or refines the model interpretation (e.g. together with UML models the Object Constraint Language (OCL) [11] is used). On the other hand DSLs are focused on a single domain and are often integrated into the software code so no ambiguity is allowed. The textual modeling languages (TML) can benefit from the popularity of DSLs and they can be improved by using DSL's best practices.

The unambiguity of model symbols and constructs interpretation is particularly important for Model Driven Development (MDD) because transformations between various models and layers of software require clearly defined inputs in order to maintain consistency between models. That is why there are attempts to define the UML language formally [2] or to create a general-purpose modeling language with exact specification and therefore with no problems with interpretation of models. The attempts for new general-purpose language are often created as textual languages because the textual languages allow easy formal definition, and moreover they are not limited by modeling tool capabilities and maturity [4].

The next reason why the new modeling languages are textual is that it is quite easy to create a new language and integrate it into a development environment such as Eclipse or NetBeans; another possible reason for the textual modeling is the simplicity of creating a textual model (especially for developers which are used to create programs this way) compared to intrusive form filling in graphical modeling tools [4].

The textual modeling languages have big ambitions, but there is no framework that could help to evaluate TML capability and maturity; therefore we define a set of features a TML should have to help us decide which language is the best one. Requested features are based on our experience and experiments with existing TMLs and are discussed in detail later, as well as the evaluation of existing TMLs. Based on the evaluation and experiments with TMLs we decide to create an alternative TML, which will fulfill the defined requirements better than existing TMLS - its name is Earl Grey (EG). The creation of the EG language was not an easy task and we had to solve several serious problems during the language proposal. There is a discussion of these problems together with possible solutions in this paper.

The paper is organized as follows: a set of required features of TML is defined in Section 2, then existing TMLs are evaluated in Section 3 and experimental language is presented in Section 4 together with a discussion of problems connected with the proposal of a general-purpose TML.

2 Required Features of TML

The required features of a TML are defined and discussed in this section. The list provides an overview of the most important ones and all mentioned features should be considered during the creation of a TML so it can be used without problems. Defined features should help with user experience with the language as well as with its usage in MDD or other automatic processing. It is important to realize the TML features are different from features of general programming languages such as Java or C, because the TMLs have a different purpose, and thus rather than focus on type system or object-orientation of the language, it is important to focus on other features, such as readability and unambiguity, which are similar to the features of DSLs [5] [4]. A lot of these features cannot

be evaluated by an exact measurement, however they can help the user to decide whether to use the language or not.

The Ability to Describe Whole Software As we focus on general purpose TMLs the ability to describe whole Software is an important feature of such a language. According to [6] there are five views on the software - use case, logical view, process view, development view and physical view. This means each TML has to be able to describe static and dynamic part of the system from the customer and developer point of view. There is probably no such a language that can describe all views at once; rather there is a set of languages each describing one view of the software. This construction is similar to the UML, where there are different kinds of models for different views. It is important that a meta-model of the TML exists so that the constructs used in one view description can be recognized in another view and so the dependencies between models can be traced and used. TML has to provide enough expression power to describe a modeled subject. This criterion is hard to evaluate because there is no way how to measure the completeness of modeling language (what should be part of the model and what exceeds models limits); sometimes a compromise between completeness and easy or general usage has to be made.

The Ability to Describe Various Levels of Abstraction In the MDD we differ four levels of model abstraction - a level describing a domain on computing independent level (CIM), a level describing a software independently on a concrete implementation platform (PIM), a level describing the software in the context of concrete implementation platform (PSM) and physical deployment; the last level is the software code itself. In the situation when we use TML, the last two levels can be considered the same - from our point of view it is not necessary to distinguish between the source code (and configuration files) and the PSM model, because both defines the same situation with respect to concrete platform.

The description on the CIM level has to be able to cover the customer domain in a way in which the model could be understood by a customer, and at the same time it should provide enough information for a software analyst or developer. The description on the PIM level should extend the information from CIM by adding some implementation details that explain how the software will be implemented, however they will not be specific for a concrete platform or framework.

Readability and Simplicity of Language The readability is very important for each language especially if it should be used for communication with a customer (CIM level). The TML has to be easy to read and easy to write so that models can be created or validated by a customer with almost no technical skills. The simplicity of a language is not only a customer requirement, because developers appreciate a language that is easy to write and read, too.

Unambiguity of TML Expressions The lack of unambiguity is the main problem of most graphical modeling languages therefore it is important for each TML to provide expressions with no ambiguities. This criterion is important for MDD because the ambiguities in models lead to misinterpretations during model-to-model transformations or code generation. Part of the expression unambiguity is the definition of relations between TML constructs - such as the meaning of association between objects or the extension of one object by another.

Supportability and Integrability The requirement on supportability and integrability is defined by [5], where supportability means TMLs feasible to provide DSL support via tools, for typical model and program management, e.g., creating, deleting, editing, debugging, transforming and integrability means the language, and its tools, can be used in unison with other languages and tools with minimal effort. This is essential to integrating the TML with other facilities used in the engineering process. An alternative requirement for TML is extensibility, i.e., that the TML can be extended to support additional constructs and concepts. It means the stable core language that could be extended is over frequent changes of the language.

We believe the most important features of a TML are readability for people, unambiguity of language expressions and capability to describe the whole software, because these features determines the acceptance of the TML by users.

3 Comparison of Existing TMLs

This section provides an overview of existing textual modeling languages and their comparison according the criteria defined in Section 2. The overview provided in this paper could not cover all existing languages; on the other hand it should provide an representative overview (for next languages evaluation see [9]).

To provide a comparison we decide to create a set of UML models that represent different views on a system and that contains several nontrivial constructs of the language. Three of the used models can be seen in Figure 1 there is a class model, a model of activities and a state model. The models do not cover the whole expressive capability of UML, rather they represent only a part from all models we used for evaluation. The purpose of this chapter is to illustrate the method and show the basic concepts of presented TMLs.

3.1 PlantUML

PlantUML [13] is a language that allows the describing of UML models directly within the source code of software. The UML models are then part of the code (as specialized comments) which is useful as there is one source of information. PlantUML can describe all required views on the software on different levels of abstraction; it contains definitions for modeling of use cases, class models,

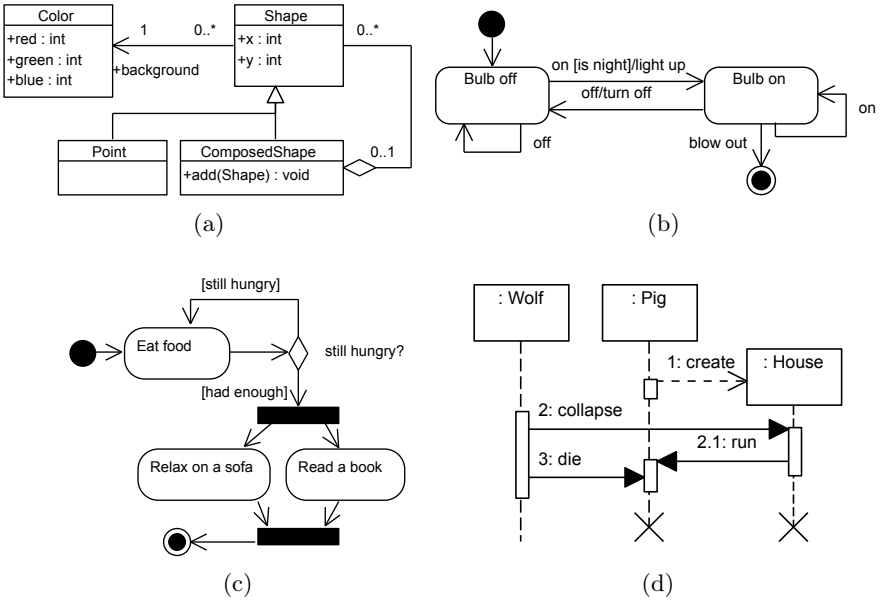


Fig. 1: A selection from the set of UML models used for TMLs evaluation. There is a class model (1a), state model (1b), activity model (1c) and sequence model (1d) in the figure; together these models represent both the static and dynamic view on the software.

state and activity models. The integrability is guaranteed by the integration of PlantUML into Eclipse IDE.

The problem of PlantUML is its readability, as the language copies not only the UML standard, but also the graphical constructs (see Listing 1) that expect the user to be familiar with UML. The next problem is with the usage of state or activity models that become confusing and unreadable.

```
class Color {
  +red : int
  +green : int
  +blue : int
}
class Shape { ... }
class Point {}
class ComposedShape { ... }

Shape <|-- Point
Shape <|-- ComposedShape
ComposedShape "0..1" o-- "0..*" Shape
Shape "0..*" --> "1" Color
```

Listing 1: The class model from Figure 1a in the PlantUML language, the usage of arrows expects the user is familiar with the UML syntax.

```
state "Bulb Off" as bulbOff
state "Bulb On" as bulbOn

[*] --> bulbOff
bulbOff --> bulbOff : off
bulbOff --> bulbOn : on [is night]/light up
bulbOn --> bulbOff : off/turn off
bulbOn --> bulbOn : on
bulbOn --> [*] : blow out
```

Listing 2: The state model from Figure 1b in the PlantUML language, usage of pseudostates could be confusing.

```
(*) --> "Eat food"
if "still hungry?" then
  -->[still hungry] "Eat food"
else
  ->[had enough] ===Fork===
endif
===Fork=== --> "Relax on a sofa"
===Fork=== --> "Read a book"
"Read a book" --> ===Join==
"Relax on a sofa" --> ===Join==
===Join== --> (*)
```

Listing 3: The activity model from Figure 1c in the PlantUML language is hard to read for larger models.

```
package testpackage;
  class Color
    attribute red : Integer;
    attribute green : Integer;
    attribute blue : Integer;
  end;
  class Shape end;
  class Point specializes Shape end;
  class ComposedShape specializes Shape end;

  association
    navigable role background : Color[1];
    role shape : Shape[*];
  end;
  aggregation
    navigable role child : Shape[*];
    navigable role parent : ComposedShape[0, 1];
  end;
end.
```

Listing 4: TextUML version of the class model from Figure 1a is quite verbose therefore it is not simple to create a model.

3.2 TextUML

Text UML [1] is TML that is specialized only at class model, therefore the capability of describing various views on the software is of course limited. This limitation is partially compensated by the readability of its models that are readable even for non-developers. The grammar of the language can be understood intuitively as it refers to UML and common programming language, however we were not able to find any formal definition of the language. In contrast with PlantUML, TextUML does not suppose the user knows UML concepts; on the other

hand it could be considered verbose as it requires a large amount of information about classes and relations, even though some of them are not necessary.

3.3 Umple

The main goal of the Umple language [8] is model-oriented programming that is based on class and state modeling and code generation into Java, Ruby and other languages. The mentioned models are the only one that can be used for modeling so some views on software are missing (use cases, processes). The Umple is the only language that provides the definition of its grammar.

The models are readable and no major complication with language usage were observed. The Umple language is integrated in Eclipse IDE or the online service Umple Online [7] can be used.

```
class Color {
  int red;
  int green;
  int blue;
}
class Shape { ... }
class ComposedShape {
  isA Shape;
  void add(Shape e);
}
```

```
class Point {
  isA Shape;
}

association {
  0..1 ComposedShape -- 0..* Shape;
}
association {
   0..* Shape -> 1 Color;
}
```

Listing 5: Umple version of the class model from Figure 1a, there are no serious problems.

```
class Bulb {
  state {
    Initial {
      init -> BulbOff;
    }
    BulbOff {
      on [isNight] -> / {lightUp();} BulbOn;
      off -> BulbOff;
    }
    BulbOn {
      off -> / {turnOff();} BulbOff;
      on -> BulbOn;
      blowOut -> Final;
    }
    Final { }
  }
}
```

Listing 6: Umple version of the state model from Figure 1b, there are no serious problems.

3.4 yUML

The purpose of the yUML [15] tool is fast and easy creation and publication of UML class model and activity model diagrams. The language is focused on only two languages, and its syntax does not allow for modeling large models in a readable way. Therefore yUML will probably remain a tool for the creation and sharing of small code snippets rather than become a widely accepted standard for general-purpose TML.

```
[Color|red;green;blue]<1-0..*[Shape]
[Shape]^[Point]
[Shape]^[ComposedShape]
[ComposedShape]<>0..1-0..*[Shape]
```

Listing 7: The yUML version of the class model from Figure 1a shows that a definition of a class with many attributes could be confusing.

```
(start)->(Eat food)
(Eat food)-><if>had enough->|fork|
<if>still hungry->(Eat food)
|fork|->(Relax on a sofa)->|join|
|fork|->(Read a book)->|join|->(end)
```

Listing 8: The yUML activity model is quite verbose and becomes confusing for large models.

3.5 Comparison Summary

The experiments show that existing TMLs do not meet the defined criteria and so the potential of TMLs. The main problem is that the TMLs designer tries to describe the UML model, not the domain (classes, states etc.) and that is in contrast with the recommendations of [16] and it reduces the usability and readability of large models.

Feature	PlantUML	TextUML	Umple	yUML	EG	UM
Multiple views on software	yes	no	no	no	yes	yes
Readability and Simplicity	no	yes	yes	no	yes	yes
Provided Language Definition	Grammar	-	Grammar	-	Grammar	MO
Integrability	Eclipse	Eclipse	Eclipse/Online	Online	Eclipse	man

Table 1: The overview of experiments with existing TMLs and UML. Most of the tested TMLs could not describe all views on software; some of them have problems with readability.

4 Experimental Modeling Language

The previous sections show that the existing TMLs do not meet the potential of textual modeling and they are not usually able to cover all requested views on software and their semantic was not specified, rather it was based on a previous knowledge of UML or common programming languages and intuitive understanding of concepts such as generalization, association or aggregation. Moreover, the TMLs are often hard to read and write and they are not usable for large models. Therefore we decide to create our own textual modeling language called Earl Grey (EG), whose concepts and creation is explained in the following text. In

this paper we do not focus on formal specification, as it is beyond the scope of this paper; instead we will focus more on user experience with the language. The main purpose of this section is to show Earl Grey's fundamental differences from other textual modeling languages.

The EG is implemented as an Eclipse plugin using Xtext [14] and the latest version of its grammar is available on-line [9]. There is an implementation of class and state model at the moment. Use case and activity model languages should be finished in the near future, therefore EG should cover all requested views on the software. All EG models are connected with the CIM or PIM group of models and the PSM is represented by the code itself.

The next sections discuss problems we have to solve during the Earl Grey implementation. The majority of the problems are caused by differences in presentation of information in graph and textual environment. We will compare our construct mostly with PlantUML because from the aforementioned TMLs, it is the most complex one.

4.1 Language for Class Modeling

When creating each model we focus on creating a language that will fit the modeling problem, whereas a lot of TMLs try to rewrite the UML models by text. A typical example is the PlantUML language and its description of associations between classes in a class model. There are examples of PlantUML associations in Listing 1, and you can see the symbols are visually close to the UML symbols, which is good if users already understand UML but for users who will meet modeling for the first time these symbols will be confusing. In contrast we decide to use a textual representation of each association type as you can see in Listing 4.

```
class Color
  red : int
  green : int
end
class Shape
  /*...*/
end
class Point isA Shape
end
class ComposedShape isA Shape
  /*...*/
end
```

```
aggregation
  0..1 ComposedShape /*start*/
  0..* Shape /*end*/
end

association
  0..* Shape
  1 Color
end
```

Listing 9: EG version of the class model

We believe the representation proposed in Listing 9 is more readable for users of the language, and they obtain much more information than from PlantUML symbols. The associations are represented as a sentence so it can be read and understood with no need to know the meanings of pseudo-graphical symbols in PlantUML. Next we prefer `isA` for inheritance indication over `extends` or graphical symbol `<|--`, the reason is educational and expression `isA` should help with inheritance usage (the problem of bad inheritance usage is described e.g. in [10]). We believe the `isA` construct will improve the design of future code.

4.2 Language for State Model

The PlantUML state model language (in Listing 2) does not allow logic structuring of a created model; all states and transitions are in one large cluster that decreases the readability of the model.

The proposed language allows the splitting of the model into small sections – one section for each state and its transitions. This structure improves not only the readability of a model but also the change of transitions. In a PlantUML model a user has to check the whole model to find the changed transition, whereas in the EG state model the information is right in the state, where the transition starts.

The next change we made against PlantUML is that we omit the usage of graphical symbols – arrows (as in class model) and asterisk that PlantUML uses for representation of initial and final (pseudo)state of the machine. Instead of asterisk symbol `(*)` we use the keywords `initial` or `final`.

```
initial "Init"
  do
    light up -> "Bulb off"
  end
end

state "Bulb off"
  off do
    -> "Bulb off"
  end
  on do
    if is night then
     light up -> "Bulb on"
  end
end
```

```
state "Bulb on"
  off do
    turn off -> "Bulb off"
  end
  on do
    -> "Bulb on"
  end
  blow Out do
      -> End
  end
end

final "End"
end
```

Listing 10: EG version of the state model

4.3 Language for Behavior Modeling

To model the behavior is an important part of software modeling. The UML language provides two models - activity and sequence model; we use them as templates because we want to preserve user experience with these models and their expression abilities.

The first sequence model describe the communication (message sending) among objects (for example of a model see Figure 1d). In the case that we try to describe the same information textually, we face the problem of low readability of the model because it is hard to provide a textual description of object interactions so that the model is logically structured and allows easy orientation. There is the textual version of the model from Figure 1d created according EG grammar in Listing 11. We believe the list of messages in the EG model becomes confusing for large models and it will not satisfy the condition of readability. The expression capability of UML sequence model lies in the lifelines representation of objects and in the time ordering of messages in left-to-right and top-to-bottom directions; therefore we try to create a similar user experience in TML, but all attempts end up with constructs that were unreadable even for small models.

The visual representation of messages and their sequences provides the next level of user experience that the textual language cannot provide.

```
sequence WolfAndPig
  Pig creates House
    Wolf calls House.collapse
      House calls Pig.run
    Wolf calls Pig.die
end
```

Listing 11: The textual sequence model is not as readable as the graphical; in the case of parallelism or conditional branches it becomes confusing.

Next possibility for the behavior modeling in UML is an activity diagram that focuses on business processes modeling and workflow representation. The representation of workflow is quite hard in TML; the existing languages yUML and PlantUML only rewrite the workflow as a set of activities and transitions between them, and the resulting model has no logical structure and it is unsuitable for large models.

The TML is capable of describing activities and transitions in sufficient detail in a single swim-lane; on the other hand there are difficulties with modeling of decisions (conditional branching), parallelization and cross swim-lanes transitions and relationships. The reason is the loss of graphical information during the rewriting of a model from graphical to textual form. When the model is rewritten we are able to capture the processes, however we lose the information about their flow.

There is not a solution that will help to solve problems with capturing the sequence or flow in TML in a readable and structured way. The compromise can be made between flow capturing and logical structure of a textual model.

5 Conclusion and Future Work

The textual modeling languages are said to have a great potential, and in this paper we discussed features required for the success of a general-purpose TML among users, the most important are readability for people and unambiguity of language expressions.

Already existing TMLs suffer from lack of both important features, which is often caused by a usage of UML-like symbols in textual language. The UML-like symbols and concepts are hard to read and often hard to interpret ambiguously. Therefore we decided to create our own TML that does meet defined criteria. There are presented several basic language grammar constructions that should improve the requested features.

Our experience is that it is easy to create a language for class and state modeling, but on the other hand the creation of a language describing behavior is very complicated, because it is hard to textually represent sequences of messages in a structured, well-arranged way. The future work in the area of TML should focus on finding a way of sufficient behavior modeling, and at the moment a pseudocode looks to be the best way.

References

1. R. Chaves. TextUML Toolkit - textuml. `http://sourceforge.net/apps/mediawiki/textuml/index.php?title=TextUML_Toolkit`, 2011.
2. A. Evans, R. France, K. Lano, and B. Rumpe. The UML as a formal modeling notation. *The Unified Modeling Language.«UML»'98: Beyond the Notation*, pages 514–514, 1999.
3. M. Fowler. *Domain-Specific Languages.* Addison-Wesley Professional, 1st edition, 2010.
4. H. Grönniger, H. Krahn, B. Rumpe, M. Schindler, and S. Völkel. Text-based Modeling. *4th International Workshop on Software Language Engineering*, 2007.
5. D. Kolovos, R. Paige, T. Kelly, and F. Polack. Requirements for Domain-specific Languages. *Proc. of ECOOP Workshop on Domain-Specific Program Development.*
6. P. Kruntchen. The 4+1 View Model of Architecture. *IEEE Software*, 12:42–50, 1995.
7. T. Lethbridge. Umple Online. `http://try.umple.org/`, 2012.
8. T. Lethbridge, A. Forward, and O. Badreddin. Umplification: Refactoring to Incrementally Add Abstraction to a Program. *Reverse Engineering (WCRE), 2010 17th Working Conference on*, pages 220–224, 2010.
9. O. Macek and M. Mazanec. tea-pot/earl-grey - GitHub. `https://github.com/tea-pot/earl-grey`, 2012.
10. B. Meyer. The many faces of inheritance: a taxonomy of taxonomy. *IEEE Computer*, 29(5):105–108, 1996.
11. Object Management Group. Object Constraint Language 2.0, 2006.
12. Object Management Group. Unified Modeling Language Specification 2.3, 2011.
13. A. Roques. PlantUML. `http://plantuml.sourceforge.net/index.html`, 2012.
14. The Eclipse Foundation. Xtext. `http://www.eclipse.org/Xtext/`, 2012.
15. H. Tobbin. Create UML diagrams online in seconds, no special tools needed. `http://yuml.me/`, 2012.
16. D. Wile. Lessons learned from real DSL experiments. *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 1–10, 2003.

P systems: State of the Art with Respect to Representation of Geographical Space

Zbyněk Janoška and Jiří Dvorský

Department of Geoinformatics, Palacký University,
Třída Svobody 26, 771 46, Olomouc, Czech Republic
zbynek.janoska@cdv.cz, jiri.dvorsky@upol.cz

Abstract. Membrane computing is an emergent branch of natural computing, taking inspiration from the structure and functioning of a living cell. P systems, computing devices of this paradigm, are parallel, distributed and non-deterministic computing models which aim to capture processes taking place in a living cell and represent them as a computation. In last decade, a great variety of extensions of model, introduced by Paun in 1998, were presented. In this paper we present a comprehensive review of current progress in the field of membrane computing, focusing on representation of geographical space in P systems. Two approaches are commonly used in Geographic Information Science (GIS) for representation of entities: field-based and object-based. Both approaches are discussed from the point of P systems and possibilities of using inherent hierarchical structure of P systems in spatial modeling are mentioned.

Keywords: membrane computing, P systems, Geographical Information Systems, representation of space

1 Introduction

Membrane computing represents new and rapidly growing branch of natural computing, which starts from observation that the processes taking place in a living cell can be understood as a computation. Membrane computing and its computational device – *P system* – were introduced by Păun [32] and gained a lot of interest in last decade. P systems start from observation, that membrane plays a fundamental role in the functioning of a living cell. Membranes act as three-dimensional compartments which delimit various regions of a living cell. They are essentially involved in a number of reactions taking place inside cell and moreover act as selective channels of communication between different compartments of a cell [5].

P systems take inspiration from cell on two levels – the structure and the functioning. Structure of cell is represented by its membranes and functioning is governed by biochemical reactions. Every P system therefore has three main elements: a *membrane structure*, where *object* evolve according to given *evolution rules* [35]. Some authors add fourth basic element of membrane systems – *communication* [5, 34]. Communication is always encoded in rules (they are called

communication rules instead of *evolution rules*) and will be dealt with later in the text. From the point of view of *Geographical Information Systems* (GIS), communication (e.g. topology) is essential feature of most real world phenomena.

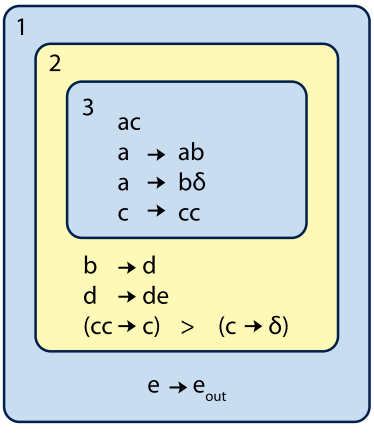


Fig. 1. Graphical representation of P system, [46]

Simple example of P system is depicted in Fig. 1. Membrane structure is hierarchically arranged set of membranes, contained in a distinguished outer membrane, called *skin* membrane. System is surrounded by the *environment*, which may collect objects leaving the system, or in some variants of P systems, the environments can actively support system with objects [4, 11]. Membrane structure can be represented in many ways – as Venn diagram, as rooted tree, or by linear notation. Membrane structure of P system depicted in Fig. 1 in linear notation is written as $[_1\ [_2\ \ [_3\ ]_3\ ]_2\ ]_1$. Membranes delimit *regions*, with which they are in one-to-one relation. Therefore the terms *membrane* and *region* are mostly interchangeable. Each membrane is identified by its *label*, which can be with membranes in one-to-many relation. The position of inner membranes does not matter; we assume, that in membrane *there is no ordering, everything is close to everything else* [35]. Please note the difference with the first law of geography: *everything is related to everything else, but near things are more related than distant things* [39]. This applies not only to membranes, but also to objects in them. From the biological point of view, inner membranes are considered floating free in their parental membranes and therefore definition of topological or metric relations between them makes no sense. This is of course not valid for geographical space and we will discuss it later.

Second basic element of P systems are *objects*. By objects in biological sense are meant chemicals, ions, molecules etc. Those substances are present in a cell in enormous amount, but the ordering again does not matter. What matters is the concentration, the population, the number of copies of each molecule [35]. Abstracting from biological reality, we represent each substance by a symbol from given alphabet and since the multiplicity matters, instead of objects we use *multisets* of objects. Common notation of multisets in P systems is following: if, for example, objects a, b, c are present in 7,2 and 5 copies, they will be represented by multiset $a^7b^2c^5$.

In basic variant of P systems, multisets of objects are considered to be floating in inner regions of membrane systems. They evolve by the means of *evolution rules*, which are localised with the regions of the membrane structure. There are three main types of rules [35]: (1) multiset-rewriting rules, (2) communication

rules and (3) rules for handling membranes. In this section only first type of rules will be described.

Multiset-rewriting rules take form $u \rightarrow v$, where u and v are multisets of objects. For example, rule $ab \rightarrow cd^2$ says, that one copy of a and one copy of b are consumed and one copy of c and two copies of d are produced. A number of possible extensions of rules will be discussed later.

Two crucial features of P systems have to be mentioned at this point. As mentioned earlier, in membranes everything is close to everything else. Therefore, if one instance of an object can be processed by two or more rules, the rule to be applied is chosen *non-deterministically.* All rules have the same probability to be chosen. The rules also have to be used *in maximally parallel manner.*

More specifically, the objects are assigned to rules, non-deterministically choosing the objects and the rules, until no further assignment is possible. An evolution step in a given region of membrane system consists of finding the maximal applicable multiset of rules, removing from region all objects specified in the left hand of the chosen rules and producing the objects on the right hand side of the rules.

After giving short introduction to basic notions of P systems, let us continue to more detailed survey on spatial properties of P systems. In Sect. 2 we will give a formal definition of *transition P system* and in Sect. 3 some possible extensions of P systems are presented. In Sect. 4 we will discuss how geographical space is represented in Geographical Information Systems, in Sects. 5 and 6 we will describe object-based and field-based representation of geographical space in P systems and in Sect. 7 we will discuss how hierarchical structure of P systems can be used to represent geographical phenomena. We will conclude with some final remarks in Sect. 8.

2 Transition P system

P systems based on application of *multiset-rewriting rules* are called *transition P system.* Formally, transition P system is a construct of the form:

$$\Pi = (O, C, \mu, w_1, w_2, \ldots, w_m, R_1, R_2, \ldots, R_m, i_o), \tag{1}$$

where:

- O is the finite and non-empty alphabet of objects,
- $C \subset O$ is the set of catalysts,
- μ is a membrane structure, consisting of m membranes, labeled $1, 2, \ldots, m$; one says, that the membrane structure, and hence the system, is of degree m,
- $w_1, w_2, \ldots, w_m$ are strings over O representing multisets of objects present in regions $1, 2, \ldots, m$ of membrane structure,
- $R_1, R_2, \ldots, R_m$ is finite set of evolution rules associated with regions $1, 2, \ldots, m$ of membrane structure,

– i_o is either one of the labels $1, 2, \ldots, m$ and then the respective region is the *output region* of the system, or it os 0 and then the result of the computation is collected in the environment of the system.

A sequence of transitions of P system constitutes a *computation*. A computation is successful if it halts, it reaches a configuration where no rule can be applied to the existing objects, and output region i_o still exists [35].

The rules are of form $u \rightarrow v$, where $u \in O$ and $v \in (O \times Tar)$, where $Tar = \{here, in, out\}$. Target indications Tar extend transition P system in following way: rule $ab \rightarrow c_{here}d_{in}e_{out}$ consumes one instance of each a and b and produces one copy of c in current membrane, one copy of d in a child of current membrane and one copy of e in the parent of current membrane. If current membrane is skin membrane, object e is send to environment of the system. If current membrane does not have a child, rule can not be applied.

Another extension comes from the existence of *catalysts*. Catalysts are objects, which participate in a chemical reaction, but are not consumed or produced by it. They just enable the application of rule. Rule with catalysts takes following form: $ac \rightarrow bc$, with object c being the catalyst.

One more extension must be mentioned at this place, and that is *dissolution* of membranes. During dissolution, membrane disappears and its content (both objects and inner membranes) are left free in the surrounding membrane. Dissolution rule takes form $u \rightarrow v\delta$, where δ denotes the action of dissolution.

3 Possible Extensions of P systems

In this section we will mention some elementary extensions of P system, which however constitute only a fracture of possibilities. We refer reader to The P systems Webpage [45] for complete list of publications and further information. Already in the text, three types of rules were mentioned. Evolution rules were briefly covered in previous sections.

Communication rules were introduces in [34]. Basic idea of communicating P systems is, that computation is achieved only by transporting object between membranes. Direct inspiration from biology are *symport* and *antiport*. When two chemical pass through membrane only together, in the same direction, the process is called *symport*. When the two chemicals pass only with help of each other, but in opposite directions, the process is called *antiport* [34]. Symport rules take a form (ab, in) or (ab, out), and antiport rules take a form $(a, out; b, in)$, where a, b are object from alphabet of all possible objects. Meaning of rules is following: for symport rule (ab, in) or (ab, out), if objects a, b are present in current membrane, they are sent together into child (or parent, in second case) of the membrane. For antiport rule $(a, out; b, in)$, if a is present in current membrane and b is present in the parent of a membrane, than a exits current membrane and b enters it. Universality of P systems with symport and antiport have been proven [34] and simplified version of communication, *conditional uniport* have been studied [40]. Comprehensive review of communication strategies in P systems can be found in [41].

Third type of rules are *rules for handling membranes*. Dissolution of membranes has already been mentioned, but other ways to obtain dynamical membrane structure, evolving during the course of computation, have been presented. Most simple of those is assigning *electrical polarization* $+, -, 0$ to each membrane. Polarization replaces target indicators $in, out, here$. Polarization of objects is introduced by the rules and polarized objects can enter only membranes with opposite polarization. For example, $ab \rightarrow c^+d^-$ means, that one instance of c enters inner membrane with negative polarization and one instance of d enters inner positive membrane. Rules can also be used to change the polarization of membranes during the computation.

Another possibilities to alter membrane structure have been proposed. Division of membranes can be used to obtain exponential working space in linear time [33] and have been used to solve **NP**-problems [31]. An optimalization algorithms based on membrane computing were proposed [29, 24]. Also biological processes of *exocytosis, endocytosis* and *gemmation* were translated into the language of P systems and examined in detail [25].

Two issues seem essential, when P systems are used to simulate biological phenomena rather than for computation. Non-determinism is first of the issues. Some chemical reactions are more likely to occur than others. First attempt to solve this is by assigning priorities to rules. Firstly, set of rules with the highest priority is chosen and according to the principle of maximal parallelism, all rules which can be applied, are applied. Then, the rules with second highest priority are selected and the procedure repeats, etc.

Second approach is to assign probabilities to all rules. Probability can be introduced to P system on different levels [30], but here we will mention only probability on the level of rule selection. Different approaches have been proposed [4, 13, 36]. Basic idea is to associate each rule with a constant k, so the rule takes a form $u \xrightarrow{k} v$, where u, v are multisets of objects and k can be interpreted either as a probability, or as a "stoichiometric coefficient", using which the true probability is calculated.

Last extension, which we will mention at this point, is representation of time of P systems. In real world, every biochemical reaction takes some time. Representation of time in P systems is similar to representation of probability. A constant t is assigned to each rule, so the rule takes the form $u \xrightarrow{t} v$, where u, v are multisets of objects and t is number of time units, which must pass to complete the application of the rule [12]. In the first time step, multiset u is consumed and removed from the current membrane. After $t-1$ more time steps, multiset v is introduced into the system. Time can also be introduced into P systems as a lifetime of objects or even membranes [1].

For the sake of brevity, we will not discuss more extensions of P systems, although many possibilities were explored within this framework. Every real-world application of P systems requires careful and accurate definition of system to be modeled and once the real-world system is defined, P system as a mathematical model for simulation can be developed. It is very unlikely, that any of presented variants of P systems would accurately describe the complex real-world pheno-

mena, however when considered as a modelling paradigm, P systems offer great variety of extensions, and arbitrary P system for concrete application can be developed.

4 Geographical Representation of Space

Object-based and field-based models of space are accepted as two alternative approaches for conceptualization and geographical modelling [20]. Object-based conceptualization understands geographical entities as sharply bounded and therefore represented by mostly polygonal boundaries. Objects are located in space, i.e. location is attribute. Fields are continuous phenomena and characterize space by properties – functions and values – related to locations [42]. Raster and vector represetations are dual to field-based and object-based conceptualization of space [37].

In field-based model, every location in a spatial framework is associated with a set of attributes. Fields are spatially continuous by definition. Field can be viewed as a mapping between spatial location and an attribute domain [43]. The most common field-types are scalar, vector or tensor; with scalar fields being the most commonly used in GIS modelling. Representation of field must be always approximate and rectangular, triangular or hexagonal tessellations are used [16]. Fields are usually stored as a georeferenced raster.

In an object-based perspective, space is viewed as a container populated by objects. Location is an attribute of each object. Object's spatial projection is mostly represented in GIS environment by points, lines (polylines, networks) or polygons [16].

Possible merge of field-based and object-based approaches have been discussed in literature [16, 42], but GIS applications include only two basic representations of space.

5 Object-based Representation of Geographical Space in P systems

Two basic terms in representation of geographical space are *distance* and *topology*. The mathematical theory of metric spaces is well-known to be inadequate as a formal foundation for distance measures in geographic spaces [28]. Contextual knowledge is a key feature of human apprehension of geographic space [44]. For example distance between cities A and B is different from the point of view of cyclist and pilot of an airplane. There is an important distinction between global view (top-table space) and geographical view (geographical space).

Top-table space can be viewed from one single point, whether geographical space is context-based [44]. Therefore classic definition of metric in mathematical space does not apply (geographical space is asymmetric and triangle inequality does not apply).

In geographical space, neighborhood relations are commonly treated as prior to metrics. For example we know that Austria and Germany share the border,

but we are unable to estimate the length. Regarding the fractal nature of geographical boundaries [26], measuring the length may not even make sense.

Definition of topology between geographical entities is therefore essential for any geographical analysis.

As mentioned earlier in the text, object-based representation of geographical space understands space as a container with entities, which are defined by their locations. The relation of entities is described by their topology. Nearness of two neighboring entities can be quantified as a distance between them. Distance can be context-based (i.e. time necessary to overcome distance between two points can depend on the mean of transportation) and also dependent on direction. Those relations can be formalized using graph theory. Entities are represented as nodes of the graph and links represent topological relation between them. Cost of links represents distance between geographical entities.

Special variant of P systems with membranes arranged in the net have been proposed as *tissue-like P system* [27]. In this variant, membranes are arranged in an arbitrary graph instead of in a hierarchy. The computation is achieved using symport/antiport rules, but generalization using evolution rules can be considered. Links between membranes are represented using *synapses*. Formal definitions are here omitted and can be found in [19, 27, 35]. This formalization is suitable for representing topological relations between entities in geographical space. Entities can be represented as membranes (nodes of the graph) and their topology could be stored in links. Adding cost to synapses will achieve further representation of distances between entities. This approach has been adapted by [6, 7] to simulate interaction between spatially separate regions (so called metapopulations). This research was however only theoretical.

Cardona et al. in [8] started research on modelling population of Beraded Vulture in the Pyrenees using model with several spatially separated regions, which could however interact with each other by sending objects to the environment and retrieving them. This model was later expanded to model population of 12 animals [9, 10], modules for modelling biomass of plants were added [14] and model was also used for management of the area [15]. Also, [10] used similar model to simulate population growth of invasive species of zebra mussel in water reservoir. The water body was represented by 17 regions with different regime of water temperature fluctuations with their topology represented by oriented graph.

Object-based representation of space can be coherently represented using P systems. Different geographical entities can be represented by membranes, topology can be stored in a graph and distances between entities could be represented using costs of links of graph. Moreover, each node – membrane – can have inner hierarchical structure. Take agglomeration of larger city as an example. This agglomeration is connected with other cities by roads and rails, therefore represented as a membrane with synapses to other membranes – cities. In the same time the agglomeration can have inner structure, represented either by smaller set of interconnected membranes (public transportation network with nodes representing stations) or a hierarchy of membranes, representing for example ad-

ministrative zoning with several levels. Moreover, both representations can be stored within this membrane separately.

6 Field-based Representation of Geographical Space in P systems

Field-based representation of space understands geographical space as a continuous field, where every location has a set of attributes (temperature, air pressure or elevation, for example). For analytical purposes, continuous fields must be approximated by grid, mostly rectangular. This representation of space is similar to cellular automata.

Recently, [3] introduced *spatial P systems*, which embody concept of space and position inside membranes in similar manner that cellular automata do. Rules, as usual, specify the objects which are consumed and which are produced, moreover, the position of produced objects can be specified. Although P systems were used before to model processes in geographical space (see previous chapter), [3] was first to inherently include space into P systems. In classical view of P systems, position does not matter. Also [7, 9] and others, who worked with space, did consider position of membranes only as an attribute of membranes, and position of object inside membrane was never considered before.

We will not give formal definition of spatial P system and will focus on possible applications instead, because up to date, there are none. Barbuti et al. [3] defines spatial P systems in two-dimensional space, but living cells are three-dimensional compartments, therefore extension to 3D would be suitable to enhance expressiveness of the model.

Cellular automata are used in GIS for various applications, including modelling of forest fires, urban growth and dispersion of pollution. Among these application, modelling of spread of pollution seems as most promising ground for P systems. Pollutants are mostly chemicals and their consumption, creation and alteration can be naturally described using P systems. Their dispersion can be described using spatial P systems.

Another example would be simulating growth of collonies of bacteria (both in microscale and macroscale), where P systems can accurately describe behavior of such simple organisms. In many other application, like modelling of deforestation, urban growth and qualitative changes in landscape, spatial P systems could achieve similar results like cellular automata.

7 Using Hierarchy to Represent Space

Geographical space is without doubts structured in a hierarchical manner. Administrative division of Czech republic is an example. Hierarchical data structures such as quadtree or octtree [38] are widely used in GIS and spatial databases. Even methodology for adapting such data structures to globe was developed [21].

Also hierarchical spatial reasoning gained interest in the community of geographers in last two decades [23]. This hierarchical approach mimics human reasoning when performing spatial operations: an appropriate scale is selected and results are computed. The quality of results obtained is assessed, and if it is satisfactory, the computation stops. If not, more detailed level of spatial data is consulted.

Our literature search showed, however, that spatial modelling of hierarchically structured phenomena is rare. Eckhardt and Thomas [17] used multilevel regression models for inspection of patterns of road accident occurrences. Other publications dealing with hierarchical modelling also exist, but mostly use hierarchical Bayesian models, where the structure of model is hierarchical, but not in geographical sense [2]. Some research has been dedicated to possibilities of visualization of geographical hierarchies [22].

P systems offer different approach, where modelling can be taking place on multiple levels of the model simultaneously. In Fig. 2, a simple system of forest is depicted (rules are omitted). This model has two levels. On upper level, the population of animals (deers) is modelled. On lower level, the competition between two parts of forests coniferous and hardwood is modelled. Because population of deers is not dependent on the inner structure of the forest, it can be modelled separately, on upper level, and inherent hierarchical structure of P systems can be exploited.

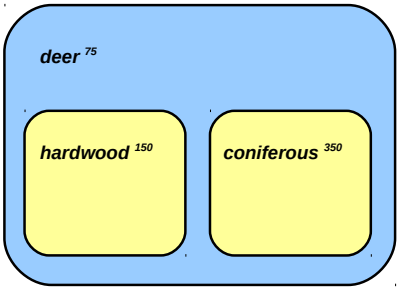


Fig. 2. Hierarchical structure of geographical space in P systems

In natural systems, ecosystems are hierarchically structured and this structure plays prominent role [18]. P systems for ecological application therefore offer expressiveness, which most other computational models do not posses.

To our knowledge, no application, where multiple levels of a single system were simultaneously modelled, were presented and also no theoretical research of this topic was conducted. Hence, further research on application of P systems should focus on this aspect of modelling.

8 Concluding Remarks and Future Work

We have presented brief introduction to membrane computing with emphasis on description of handling of geographical space in P systems. Both representations of geographical space – object-based and field-based were already discussed in the literature and some application are available for object-based representation.

Only one publication so far was dedicated explicitly to computing with space in P systems.

Currently, many extensions of classical transitional P system exist, and expressiveness of this model can be significantly enhanced. For real-world application, however, unique models must be defined. We proposed some possible application of P systems in geography, from which modelling the spread of pollution seems the most promising, given the nature of the prenomenon.

Also perspectives of multi-level modelling were mentioned. This approach exploits inherent hierarchical structure of P systems to simulate the behavior of systems on multiple levels. Possible application can be seen in ecological studies, since ecosystems are deeply hierarchized structures.

Our current research is focused on modelling of transportation using P systems. Individual based modelling, paralellism and evolution of components of a system are key features needed to model complex behaviour of transport systems. However this research is in its initial stage and experimental results are not available at the moment.

References

1. Aman, B., and Ciobanu, G. Adding Lifetime to Objects and Membranes in P Systems. *International Journal of Computers Communications and Control 5 (3).* 2010. Pages 268-279.
2. Banerjee, S. *Hierarchical Modeling and Analysis for Spatial Data.* Chapman and Hall, 2004, London, UK.
3. Barbuti, R., and Maggiolo-Schettini, A., and Milazzo, P., and Pardini, G., and Tesei, L. Spatial P systems. *Spatial P systems 10 (1).* 2011. Pages 3-16.
4. Bernardini, F., and Manca, V. Dynamical aspects of P systems. *BioSystems 70 (2).* 2003. Pages 85-93.
5. Bernardini, F., and Gheorghe, M., and Krasnogor, N., and Muniyandi R.C., and Perez Jimenez, M.J., and Romero-Campero, F.-J. On P Systems as a Modelling Tool for Biological Systems. *Pre-Proc. of the sixth Workshop on Membrane Computing.* Vienna, Austria, 2005. Pages 114-133.
6. Besozzi, D., and Cazzaniga, P., and Pescini, D., and Mauri, G. Seasonal variance in P system models for metapopulations. *Pre-proceedings of International Conference on Bio-Inspired Computing - Theory and Applications, BIC-TA 2006, Membrane Computing Section.* Wuhan, China, 2006. Pages 27-36.
7. Besozzi, D., and Cazzaniga, P., and Pescini, D., and Mauri, G. Modelling metapopulations with stochastic membrane systems. *Biosystems 91 (3).* 2008. Pages 499-514.
8. Cardona, M., and Colomer, M.A., and Margalida, A., and Pérez-Hurtado I., and Pérez-Jiménez, M.J. and Sanuy, D. Modeling Ecosystems Using P Systems: The Bearded Vulture, a Case Study. *Membrane Computing.* Springer-Verlag, 2009, Berlin/Heidelberg. Pages 137-156.
9. Cardona, M., and Colomer, M.A., and Margalida, A., and Pérez-Hurtado I., and Pérez-Jiménez, M.J. and Sanuy, D. A P System Based Model of an Ecosystem of Some Scavenger Birds. *Workshop on Membrane Computing.* 2009. Pages 182-195.
10. Cardona, M., and Colomer, M. and Margalida, A., and Palau, A., and Pérez-Hurtado, I., and Pérez-Jiménez, M.j., and Sanuy, D. A computational modeling for real ecosystems based on P systems. *Natural Computing.* 2010. Pages 39-53.

11. Cavaliere, M. Evolution-Communication P Systems. Proceeding WMC-CdeA '02 Revised Papers from the International Workshop on Membrane Computing. 2003. Pages 134-145.
12. Cavaliere, M., and Sburlan, D. Time-independent P systems. *Membrane Computing. International Workshop WMC5.* Milano, Italy, 2005. Pages 239-258.
13. Cazzaniga, P., and Pescini, D., and Romero-Campero, F.-J., and Besozzi, D., and Mauri, M. Stochastic Approaches in P Systems for Simulating Biological Systems. *Fourth Brainstorming Week on Membrane Computing.* Seville, Spain, 2006. Pages 145-164.
14. Colomer-Cugat, M.A., and Fondevilla, Ch., and Valencia-Cabrera, A New P System to Model the Subalpine and Alpine Plant Communities. *Ninth Brainstorming Week on Membrane Computing.* Seville, Spain, 2011. Pages 91-112.
15. Colomer, M.A., and Lavín, S., and Marco, I., and Margalida, A., and Pérez-Hurtado, I., and Pérez-Jiménez, M.J., and Sanuy, D., and Serrano, E., and Valencia-Cabrera, L. Modeling population growth of Pyrenean Chamois (Rupicapra p. pyrenaica) by using P systems. *Lecture Notes in Computer Science.* 2011. Pages 144-159.
16. Cova, T.J., and Goodchild, M.F. Extending geographical representation to include fields of spatial objects. *Journal of geographical information science 16 (6).* 2002. Pages 509-532.
17. Eckhardt, N., and Thomas, I. Spatial nested scales for road accidents in the periphery of Brussels. *IATSS Research 29 (1).* 2005. Pages 66-78.
18. Forman, R.T.T. *Land Mosaics: The Ecology of Landscapes and Regions.* Cambridge University Press, 1995, Cambridge, Uk.
19. Freund, R., and Păun, Gh., and Pérez-Jiménez M.J. Tissue-like P systems with Channel-States. *Second Brainstorming Week on Membrane Computing.* Seville, Spain, 2004. Pages 101-116.
20. Goodchild, M.F. Geographical data modelling. *Computers and Geosciences 18.* 1992. Pages 401-408.
21. Goodchild, M.F., and Yang, S. A hierarchical spatial data structure for global geographic information systems. *CVGIP: Graphical Models Image Processing 54 (1).* 1992. Pages 31-44.
22. Hadlak, S., and Tominski, Ch., and Schulz, H.-J., and Schumann, H. Visualization of Attributed Hierarchical Structures in a Spatio-Temporal Context. *International Journal of Geographical Information Science.* 2010. Pages 1497-1513.
23. Hirtle, S.C., and Frank, A.U. Spatial Information Theory: A Theoretical Basis for GIS. *Lecture Notes in Computer Science 1329.* 1997. Pages 1-15.
24. Huang, L., Suh, I.H., and Abraham, A. Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants. *Information Sciences 181.* 2011. Pages 2370-2391.
25. Krishna, S.N., and Păun, Gh. P Systems with Mobile Membranes. *Natural Computing 4 (3).* 2005. Pages 255-274.
26. Mandelbrot, B. How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension. *Science 156 (3775).* 1967. Pages 636-638.
27. Martin-Vide, C., and Pazos, J. and Păun, Gh., and Rodríguez-Patón, A. A New Class of Symbolic Abstract Neural Nets: Tissue P Systems. *Proceedings of the 8th Annual International Conference on Computing and Combinatorics.* Springer-Verlag, 2002, London, UK. Pages 290-299.
28. Montello, D.R. The Geometry of Environmental Knowledge. *Spatio-Temporal Reasoning.* 1992. Pages 136-152.

29. Nishida, T.Y. An application of P systems: A new algorithm for NP-complete optimization problems. *Proceedings of the 8th World Multi-Conference on Systems, Cybernetics and Informatics*. 2004. Pages 109-112.
30. Obtulowicz, A., and Păun, Gh. (In search of) Probabilistic P systems. *Biosystems 70 (2)*. 2003. Pages 107-121.
31. Pan, J., Alhazov, A. Solving HPP and SAT by P Systems with Active Membranes and Separation Rules. *Acta Informatica 43 (2)*. 2005. Pages 131-145.
32. Păun, Gh. Computing with Membranes. *Technical report, Turku Center for Computer Science-TUCS*. Turku, Finland, 1998.
33. Păun, Gh. P Systems with Active Membranes: Attacking NP Complete Problems. *Journal of Automata, Languages and Combinatorics 6*. 1999. Pages (75-90)
34. Păun, A., and Păun, Gh. The Power of Communication: P Systems with Symport/Antiport. *New Generation Computation 20 (3)*. 2002. Pages 295-306.
35. Păun, Gh. Introduction to Membrane Computing. *First brainstorming Workshop on Uncertainty in Membrane Computing*. Palma de Mallorca, Spain, 2004. Pages 1-42.
36. Pescini, D., and Besozzi, D., and Mauri, G., and Zandron, C. Dynamical probabilistic P systems. *International Journal of Foundations of Computer Science 17 (1)*. 2006. Pages 440-447.
37. Peuquet, D.J. A Conceptual Framework and Comparison of Spatial Data Models. *Cartographica*. 1984. Pages 66-113.187-260
38. Samet, H. The Quadtree and Related Hierarchical Data Structures. *ACM Computing Surveys 16 (2)*. 1984. Pages 187-260.
39. Tobler, W. A computer movie simulating urban growth in the Detroit region. *Economic Geography 46 (2)*. 1970. Pages 234-240-
40. Verlan, S., and Bernardini, F., and Gheorghe, M., and Margenstern, M. Computational completeness of tissue p systems with conditional uniport. *Proceedings of the 7th international conference on Membrane Computing*. Springer-Verlag, 2006, Berlin/Heidelberg. Pages 521-535.
41. Verlan, S., and Bernardini, F., and Gheorghe, M., and Margenstern, M. Generalized communicating P systems. *Theoretical Computer Science 404 (1-2)*. 2008. Pages 170-184.
42. Winter, S. Bridging vector and raster representation in GIS. *Proceedings of the sixth ACM international symposium on Advances in geographic information systems GIS 98*. ACM Press. 1998. Pages 57-62.
43. Worboys, M.F. *GIS: a computing perspective*. Taylor and Francis, 1995, London, UK.
44. Worboys, M.F. Metrics and topologies for geographic space. *Advances in Geographic Information Systems Research II: Proceedings of the Symposium on Spatial Data Handling*. Delft, The Netherlands, 1995. Pages 170-184.
45. The P Systems Web Page. `http://ppage.psystems.eu/`. Last revision: 2012-03-18.
46. P system - Wikipedia, the free encyklopedia. `http://en.wikipedia.org/wiki/P_system`. Last revision: 2012-03-18.

Methodology for Estimating Working Time Effort of the Software Project

Jakub Štolfa, Svatopluk Štolfa, Ondřej Koběrský, Martin Kopka, Jan Kožuszník, and Václav Snášel

Department of Computer Science
VŠB – Technical University of Ostrava, Faculty of Electrical Engineering and Computer Science
17.listopadu 15, Ostrava-Poruba, Czech Republic
{jakub.stolfa, svatopluk.stolfa, ondrej.kobersky, jan.kozusznik, vaclav.snasel}@vsb.cz, martin.kopka@c4u.cz

Abstract. The precise estimation of the time effort of the project is one of the key limits of its success. One of the ways how to achieve a correct valuation of the project is developing of a detailed analysis, which output is a structured solution that uses use cases. This paper focuses on developing a methodology for estimating working time effort of the project for one particular company. An important part of the methodology is to build up and maintain comparative database of valued use cases and time progress of realized projects. The aim of the methodology is to deliver data for evaluating a new project.

1 Introduction

The proper estimation of the project is a goal which wants to achieve almost every project manager. It is not easy quest and it is not essential. It is hard task which takes a lot of effort to do it right. And the question is how to do it right. There are several ways how to fulfill this task. Which one is the best is depending on the concrete company, concrete types of the projects etc.

However, one thing is clear, if we know supposed project progress (supposed project progress of its activities), we can find out in which phase the project is. Thereby we can figure out if the project plan is in time, late or ahead. So, we can determine the effort and plan resources of the project. For example, since some point of time we know that project will not need analyst activities anymore, so we can move analysts (they do analyst activities) out of the project, to the another project.

1.1 State-of-the-art of estimation project approach

Many formal methods were published in the area of effort estimation for software development projects. Heemstra wrote down the basic ideas why, when and how to estimate projects in paper “Software cost estimation .In Information and Software

Technology" [12] in early 90s. This paper speaks about importance of estimation of the project. One of the mentionable information is that lot of companies do not make an estimation. And even if they make estimation, it is mainly not corresponding with a reality. Since our method is based on the use cases, we will mention only some methods utilizing the use case approach here (referenced also as parametric models).

The COCOMO methodology [1] computes the effort of the software projects as a function of program size and a set of cost drivers on separate project phases. It is the Constructive Cost Model, which has been originally developed by Dr. Barry Boehm and Publisher in 1981 [13]. He found out a formula computing the classification of separate cost drivers.

Similar access as COCOMO basic is used in [2] where author estimates that size of a project is a function of the size of definition that was written into the use cases definition. Function points are popular method to estimate size of proposed application. The ISO/EIR organization [6] created functional size metric standard which supports that method.

Methodology introduced by [3] computes the project estimation using complexity of use cases and its transactions applying the set of adjustment factors. Building the database from really measured projects with several technologies is important approach.

J. Smith in 1999 speaks about use cases and their complexity [4]. It thinks about how big should a use case be and about the complexity of the big or small use cases and how much effort particular use case takes. It brings us an idea that we used for categorizing standardized use cases in our assessed company.

Almost all methods, which use estimation based on use case points, are based on method of Gustav Karner. He first described use case points. [2] Use case point is described like function of the following:

- the number and complexity of the use cases in the system,
- the number and complexity of the actors on the system,
- various non-functional requirements (such as portability, performance, maintainability) that are not written as use cases,
- the environment in which the project will be developed (such as the language, the team's motivation, and so on).

Like Gustav Karner says, the basic formula for converting these parameters into single measure, which is mentioned use case point, there is that it is necessary to weight the complexity of the use cases and actors and then to adjust their combined weight to reflect the influence of the nonfunctional and environmental factors.

The case studies, which are based even on use case points, are described in the paper of Bente Anda [11].

We can say that our methodology is even based on the use case point, but it looks more precisely on the use case realization, that means text of the use case. It evaluates rows, words and paragraphs of the standardized use case realization. How it works, there is described later in our paper.

Our paper is organized as follows: Section 2 introduces the problem of the project estimation in the particular company; Section 3 describes the concept of our methodology: filling of the database by the data from finished projects, categorization of the

use cases and example that describes new project effort estimation. Concluding Section 4 provides a summary and discusses the planned future research.

2 Definition of the problem

Our goal was to set up an estimation working time effort of projects in the particular company. Even that the company has 130 workers and lot of finished projects, people in that company were not used to estimate a new project by some methodology. The common issue was to estimate a new project working time effort by theirs own decisions. That decisions are based on experiences obtained by solving past projects. The problem of that solution of the estimation of the project's working time effort is that it is so much human dependable. For example it means that an estimation can be wrong because the particular worker had not enough experience to do that, or he simply don't know how to make the particular estimation. The solution to this problem is the supporting tool for estimating time effort of new projects based on our methodology. This can help workers to estimate a new project by showing them average project progress of similar projects. Thanks to that methodology, worker simply knows the progress of projects with similar working time effort and can easily estimate a new one.

It is important to mention that out methodology is based on particular company and their software developing standards. The methodology was supposed to use only in that particular company at the beginning. It means we do not declare that this is general approach which can essentially work in other companies. On the other hand the aim of this paper is to provide also the guidance for other companies, which develop software in same way like our one. They can apply our methodology to their processes as well as we did it in that particular company.

2.1 Initial state

Initial state of the estimation in the company was described before. Thorough it is important to mention how the company makes analysis and how is a progress of the project captured.

The analysis in the company is made by use cases. These use cases are standardized. That means, if the use case deals with same repeatable issue, then it is almost similar to other use cases that are dealing with the same issue. It gives us the possibility to use these standardized use cases for the projects comparisons.

The capturing of the project progress is made by CRM system. In the CRM system you can see how much time was spent on each activity. More information about captured activities is written in next sections.

3 Proposal of the methodology

The following chapter is focusing on the logical principles of the methodology. Methodology consists of two main processes. Rectangle is an activity; oval is input or output data to the activity.

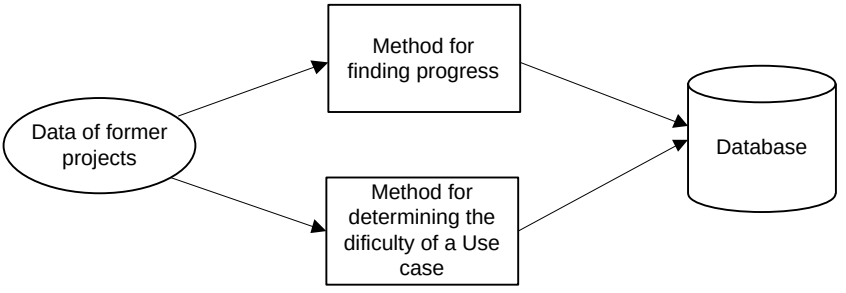


Fig. 1. Filling of the database.

The first process is named "Filling of the database" (Fig. 1). It is a process where data about a recent project are filled into the database. It starts with data of recent projects in particular company. These data are separately executed in two main methods. First one is the "Method for finding progress" and second one is the "Method for determining the difficulty of a Use case".

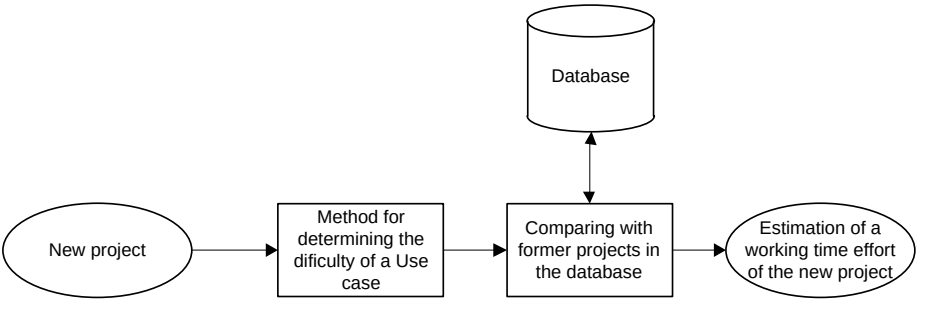


Fig. 2. Estimation of the new project

In case, that database is filed by the first process, the second process can be run. The Name of that process is "Estimation of the new project". The input of that process is a new project. The exact input is the finished analysis of that new project. It means that we have all required use cases. These use cases are elaborated in the activity "Method for determining the difficulty of a Use case". It is the same method like in the process named "Filling of the database". After that, process continues with the method named "Comparing with former projects in the database. The method compares former project in the filled database with data of the new project, finds similar project and estimate working time effort based on the similar projects progress. When the project is finished, the database could be extended by the new finished project. Next sections of this paper introduce particular steps of both processes.

3.1 Method of finding Progress

The supposed project progress, it means how much effort and time particular tracked activities should consume, is discovered by the comparison to the data of finished projects in the company. According to these data, we can find out if the company is following project methodology. If it is true, than all projects have the same or almost same progress of tracked activities.

Our method tracks progress of five main activities of the project.

- Consultation (in the system(CRM) like PORA)
- Analysis (ANAL)
- Programming (PROG)
- Testing (TEST)
- Implementation (IMPL)

Steps of the method finding progress.

1) **Input project data**. We need to know number of worked hours in each day for particular activities, when that activity was practiced (see the table below).
2) **Setting number of segments** to which we will split timeline of the project. Methodology set default number of segments to 10, but we can set even another number (5, 20, 100, etc.). The reason why to split timeline to the same segments is that, we need to normalize timelines of projects. So that we can compare projects whit each other. For example, one project last 10 months a second one last 1 month. If it is slip up only to the weeks, then first project is split to the 40 segments and second one to the 4 segments. Comparing these two projects is impossible in this way. If we split up these two projects to the same number of segments, we can compare them. First project has a 10 segments and one segments contains data of 28 days (project last 40 weeks = 280 days, we split up to 10 segments = 28 to each segment). Second project contains in one segment 2,8 days.

   This example shows that the last segment is not same as the others every time. It depends on the technique in the methodology, if we round up (default) or down. If we round up, one segment contains 3 days (2,8 days > 3 days). A it means that last segment contains only 1 day (28 - 28/3 = 1). It was proved by the experiments that this is not significantly important for the comparison of the projects. It is only good to have in on your mind for later refinement of the method.
3) **Choosing the technique of the evaluation.** Our methodology has two types of evaluation of the project duration, which is later split up to the, before mentioned, segments.

a. **Counting all days.** It means that we count all days – from the first day to the last day, when some of tracked activities were performed. Then the duration of the project (number of days) is counted. Number of days is divided by number of segments (default 10). So that we know how many days the segment contains. After that calculation we add to first date number of days in segment. That date is border line. Then we add number of day in the segment to that border line and establish second border line. We have second segment. That technique we have to do same way to the penult segment. To the last segment we put remaining days of the project. For example first date is 1.7.2010 and one segment has 14 days. First segment contains data of dates from 1.7.2010 to 14.7.2010. Second segment from 15.7.2010 to 28.7.2010, etc.

b. **Counting only days, where there was some work done.** It means that, we count on only days, when some of the tracked activities were performed. We do not count empty days. Days where there was not done any work on tracked activities. We count all effort hour and divide them by number of segments. Then, there is a difference from the first technique, we count only days, where there was some work. So segment of size 11 hours may contain for example 1.7.2011 5 hours of analysis, 13.7.2011 6 hour of programming.

   This technique has one week aspect. When we reach the end of the segment, actual summary of hours in that segment is 9 and next date contains for example 3 hours, so we close segment with actual summary 9 hours. Because we do not want to past out the limit of the segment (limit is 11, 9+3=12, 12>11). It has an effect that segments could contain various numbers of hours. This might be a problem if we count projects with low number of total hours. The affection is minimal to the projects with big number of total hours.

4) **Calculation.** Inputs are fulfilled segments (day o hour method). Then we need to count number of hours spent on particular activity in each segment.

5) **Saving a result to the database.** Saving the result of the methodology is made by vectors. We can group these vectors and so that we can find out similar projects. Vectors are easily transferable to the graph.

   a. **Structure of the vector** (5 segments):

      First is a name of the project. Then three numbers of particular difficultly of use cases. And then are worked hours of the activity in particular segment. First is consultation, then analysis, programming, testing and last implementation. Each activity has 5 numbers divided by semicolon. It shows how much hours was worked in each segment.

Name of the project	Difficulty of Use cases			Hours of consultation per each segment					Hours of analysis per each segment				
Project	H	M	E	13,75	0	18,25	255,75	237,25	0	0	0	122,5	116,5

Hours of programming per each segment					Hours of testing per each segment					Hours of implementation per each segment				
0	0	0	138	2704	0	0	0	0	1071	4	0	11,75	8,5	171,25

Table 1. Structure of the vector. Table shows structure of the vector. It is one log table, but for that paper was divided to two.

3.2 Method for determining the difficulty of a Use case

Following chapter describes how we evaluate particular use case.

Basic complexity according to the number of rows.

Complexity of a project is given by the difficulty of UC realizations. The difficulty of UC is hard to determine, there is no easy benchmark for their comparison. Simplest way is determining the complexity based on number of rows in every UC. We have set 3 levels of difficulty based on number of rows:

- E (easy) – number of rows < 70
- M (medium) – number of rows ε <70;110>
- H (hard) – number of rows > 110

We can obtain number of hard, medium and easy use cases for one project with this type of evaluation.

Extended complexity.

In terms of our method objectivity we decided to extend complexity with number of paragraphs and number of words in each UC. The overall difficulty of UC is then derived from the individual complexities.

Difficulty according the number of words.

- E (easy) – number of words < 200
- M (medium) – number of words ε <200;500>
- H (hard) – number of words > 500

Difficulty according the number of paragraphs.

- E (easy) – number of paragraphs <= number of paragraphs + X -> easy
- M (medium) - number of paragraphs > number of paragraphs + X AND number of rows < number of paragraphs + Y -> medium
- H (hard) – number of paragraphs => number of paragraphs + Y -> hard

Where X=10, Y=25

The overall difficulty.

Overall difficulty of UC can be determined as follows. Difficulty of words has the highest weight, then difficulty of rows and difficulty of paragraphs. First of all we compare difficulty of words with difficulty of rows, If they are in the same level, then the overall difficulty is in this level. If not, we compare the difficulty of words with difficulty of paragraphs, if they are on same level, then the overall difficulty is in this level. If any of them differs then we compare the difficulty of rows with paragraphs in the same way. If the comparison does not help us, overall difficulty is difficulty of words, because we consider it the most important aspect. The difficulty types and levels were checked by the correlation matrix. The result was that the actual dependency setting varies between 60-100 %. Most of the difficulty types and levels more than 2/3) are more than 95% dependable.

3.3 Comparing to former projects in the database

Input of this activity is evaluated by use cases of a new project. This overall difficultness of project use cases was determined in the activity Method for determining the difficulty of a Use case.

After that we find recent project are in the database which have almost similar difficultness of a use cases. The similarity is determined by next proclamation. We proclaim that two projects are similar if their particular difficulties differ by +/- 2. This simple differ was set up by several experiments made on real data in the company.

3.4 Example

This section describes an example of using a described methodology. Inputs are analysis of four projects (Project1, Project2, Project3, Project4). It is only the example so that's why, we show only four representative projects. We cannot publish the names of the project, and that is the reason why we call them like that. Important is that all of these projects are development projects. That means they include all steps of the life cycle of the project (consultation, analysis, programming, testing, implementation).

We fulfill the database by data (vectors) of these projects. After that we take a new project, evaluate its use cases (by difficultness of UC) and find out estimated progress and difficulty.

1) Determination of difficultness of UC of particular projects:

Difficulty according to the number of rows

	T	S	L
Project1	6	8	57
Project2	1	1	12
Project3	4	0	17
Project4	0	2	4

Table 2. Difficulty according to the number of rows.

Difficulty according to the number of paragraphs

	T	S	L
Project1	12	17	42
Project2	1	11	2
Project3	3	5	13
Project4	0	4	2

Table 3. Difficulty according to the number of paragraphs.

Difficulty according to the number of words

	T	S	L
Project1	16	11	44
Project2	3	7	4
Project3	5	3	13
Project4	2	2	2

Table 4. Difficulty according to the number of words.

Total difficultness

	T	S	L
Project1	12	15	44
Project2	1	9	4
Project3	3	4	14
Project4	0	4	2

Table 5. Total difficultness.

At the table 5 we can see the total difficultness of particular projects. This view is most important for further processing. The section 3.2 describes how the particular difficultness was set up.

2) Compilation of vectors and graphical representation of these vectors for the projects progress

This section shows the vector for the project progress of the particular project. Structure of these vectors was described in the table 1 above.

- Project1;12;15;44;284,75;28;48;36,75;29;60,5;27,55;2,5;0;122,5;0,75;8,25;19;27,5;27,25;22,2;2;4,5;0;52,5;444,75;332,75;317,5;341;320,5;256,75;275,25;259;0;0;10,5;93,25;109,5;91,25;7;9,5;160;188,75 ;185,5;0;24,25;3;9,25;1,75;0;0,5;27,25;16;37;0
- Project2;1;9;4;23,75;20;12;30;15;40,25;20;10,5;1;130,50;20;10;18;25,5;20,25;10,25;1;3;2;2;20,25;130,75;256;432;203;150,75;25,25;259;30;0;14,5;23,25;19,75;19,75;10;17;163;129,75;174,25;2;12,25;6;10,25;4,75;0;3,5;20,25;13;20;35

- Project3;3;4;14;5,25;13;18,25;5;12,75;7,5;8,5;11,25;8,25;5;2;10,75;1,5;0;10;6;5,5;0,5;0;5,5;3,75;1,5;2;0;0;1;0;6;9;8,25;5,75;9,75;10,25;0;3,25;0,5;3;0;0;1,75;0;0;1,5;1;0;0;0;0;0;0;0;0;0;0;0,5
- Project4;0;4;2;12;8;6;14;12,25;31,25;5;2;1;10,50;3;7,75;20,50;9,75;15;13,50;5;7,75;3,25;5;10,75;30,50;26;42;20;10,25;2,75;29;18;2;5,5;2,25;9,75;4,75;1;8;16;29,25;4,25;1;6,75;6,50;2,25;3;1;0,0;0;25;13,50;18,25

3) Estimation of a new project – Project5

- Input is the new project – Project5
- Determination of difficultness of UC of the Project5:
  - Hard; Medium; Easy = 4;6;13
- Finding similar projects in database:
  - 4+-2; 6+-2; 13+-2 – we find out project with difficultness of UC +-2 according a new project. From these specified projects we make average of their progress. And that is set as estimation of the new project.
  - In our example it is only Project3, so we do not need to do average.
  - Estimated progress of the Project5 is:

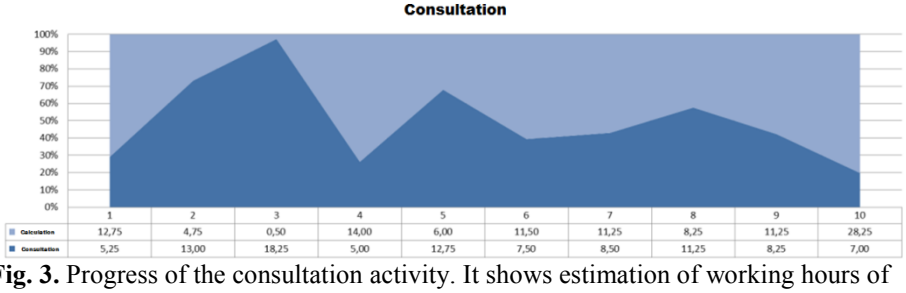


Fig. 3. Progress of the consultation activity. It shows estimation of working hours of current activity in particular segments of the project.

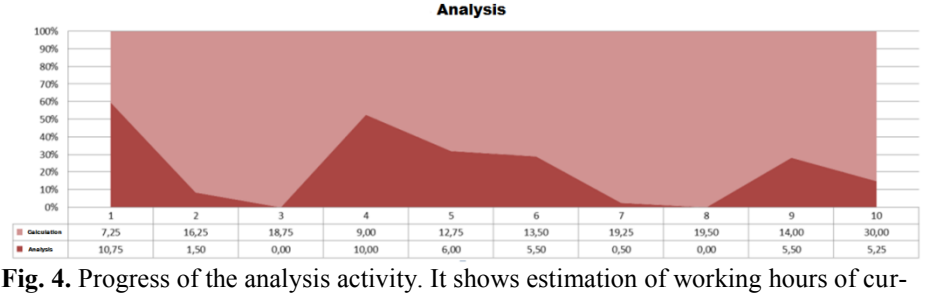


Fig. 4. Progress of the analysis activity. It shows estimation of working hours of current activity in particular segments of the project.

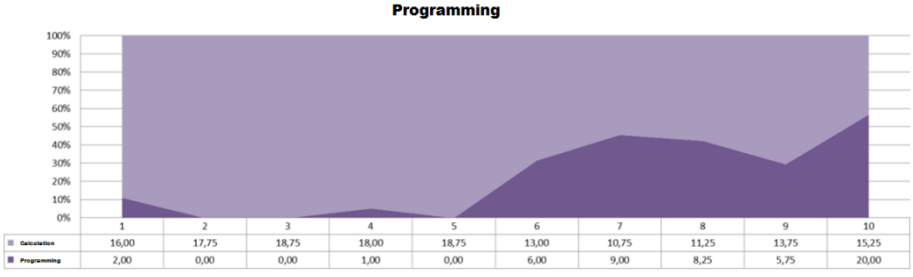


Fig. 5. Progress of the programming activity. It shows estimation of working hours of current activity in particular segments of the project.

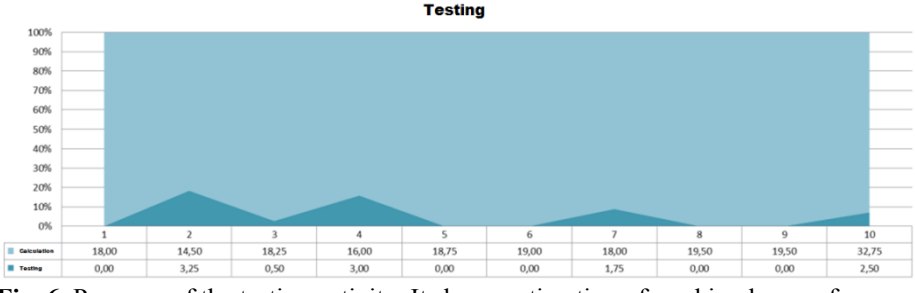


Fig. 6. Progress of the testing activity. It shows estimation of working hours of current activity in particular segments of the project.

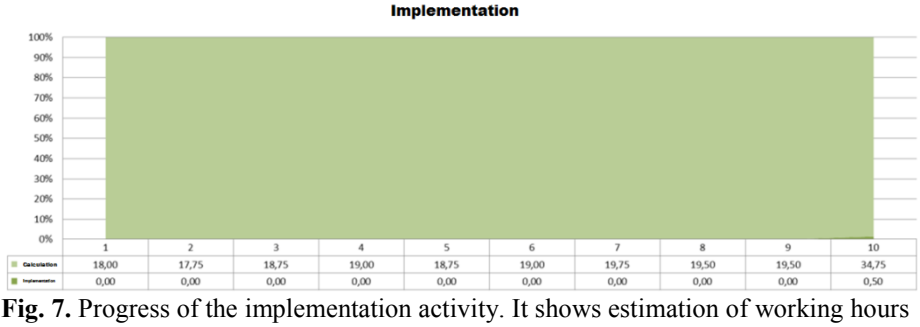


Fig. 7. Progress of the implementation activity. It shows estimation of working hours of current activity in particular segments of the project.

These figures show progress of particular activities of the Project5. We can see estimation of working hours of particular activity in particular segments of the Project5. We can have a look for example on activities consultation and programming. It is interesting to note that estimation of working hours of consultation activity is almost stable for every segment. On the other hand estimation of working hours of programming activity is divided to the two parts. At the beginning of the project are hour spend on programming activity minimal, but at the end are major. It is essential because at the begging is not too much programming work.

4 Conclusion and Future Work

Our methodology was tested on the 20 past projects made by the company. Fifteen projects were taken to the process of filling the database. Five projects were proclaimed as new projects. The result of our methodology – supposed project progress (effort of each activity) was compared to the historical data of these projects. The difference between the predicted progress and the actual data for the tested projects was approximately 30%. The 30% inaccuracy seems not to be a good, but the previous ad-hoc human based estimation had inaccuracy 40%. We found out these data by comparing their original estimations on the beginning of the projects with result of these projects at their end. In 40 percent there was huge deflection of estimation and the real execution. Therefore our methodology brings approximately 10% improvement, which is a good result of the methodology. But we know that 30% is still huge number of inaccuracy, so that there is place to improve our methodology in future.

In the future, we plan to improve our methodology by neuron nets as tools which find out groups of similar projects. Otherwise, we plan to elaborate parameters describing the influence of the customer to every single project. Then we have to elaborate if is better to use more parameters to describe particular use cases for an execution of the methodology. At least but not last we have to have on our mind that we estimate project after first steps of analysis. Especially, after the use cases are finished. That's important to mention, because we have to include that work to the estimation of the project.

I any case the topic of estimation project's effort is huge place for research and improvements. The methodology that works for one company does not have to work for others. Our goal is to overcome that gap by trying to develop methods and that will be used in more companies than one and the results will be more accurate.

5 References

1. Boehm, B., Abts, Ch., Brown, W., Chulani, S., Clark, B., Horowitz, E., Madachy, R., Reifer, D., Steece, B.: Software Cost Estimation with COCOMO II. Englewood Cliffs, NJ:Prentice-Hall, 2000. ISBN 0-13-026692-2
2. Cohn, M.: Estimating With Use Case Points, Methods and Tools, Fall 2005 (Volume 13, number 3), ISSN 1661-402X.
3. Ochodek, M., Nawrocki, J., and Kwarciak, K.: Simplifying effort estimation based on Use Case Points. Information and Software Technology, 53(3):200–213, 2011.
4. Smith, J.: The Estimation of Effort Based on Use Cases, Rational Software white paper, 1999
5. Ruhe, M., Jeffery, R., Wieczorek, I.: Using web objects for estimating software development effort for Web applications. In: Proceedings of the ninth international software metrics symposium, Sydney, Australia 3–5 September 2003, p 30
6. ISO/IEC 14143-1:1998 (1998) Functional size measurement.www.iso.org
7. Ribu, K.: Estimating Object-Oriented Software Projects with Use Cases. 2001. Master of Science Thesis, 2001, University of Oslo, Department of Informatics.
8. Ochodek, M., Nawrocki, J.: Enhancing use-case-based effort estimation with transaction types. Foun- dations of Computing and Decision Sciences, 35(2):91–106, 2010.

9. Kemerer, Ch.: An empirical validation of software cost estimation models, Communications of the ACM, Volume 30, Issue 5, New York 1987.
10. Anda, B.: Comparing Effort Estimates Based on Use Case Points with Expert Estimates, Empirical Assessment in Software Engineering (EASE), Staffordshire 2002.
11. Bente A, Hege D., Dag I. K. Sjoberg, and Magne Jorgensen. 2001. Estimating Software Development Effort Based on Use Cases-Experiences from Industry. In Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools (UML'01). Springer-Verlag, London, 2001.
12. F. J. Heemstra. Software cost estimation. In Information and Software Technology, Vol. 34, No 10, October 1992, Elsevier.
13. Boehm B.: Software Engineering Economics, Prentice Hall, 1981.

Developers' Cooperation based on Terms of Project Description

Štěpán Minks, Jan Martinovič, Pavla Dráždilová, Alisa Babskova, and Kateřina Slaninová

VŠB - Technical University of Ostrava,
Faculty of Electrical Engineering and Computer Science,
17. listopadu 15/2172, 708 33 Ostrava, Czech Republic
{min111,pavla.drazdilova,jan.martinovic,alisa.babskova.st,
katerina.slaninova}@vsb.cz

Abstract. Very interesting specialized web portal for collaboration of developers is CodePlex [1]. Registered users can participate in multiple projects, discussions, adding and sharing source codes or documentations, issue a release, etc. In the article we deal with strength extraction between developers based on their association. The research presented in this article is motivated by our previous work [10]. From this paper we have used the approach for extraction of initial metadata, and we have used modified Jaccard coeficient for description of the strength of associations between developers. Method is usable for creation of derived collaborators network, where as input is used the set of words, which will describe the network (the developers used these words in project description).

1 Introduction

In the library science, keywords are used to describe the theme of the book and its inclusion in the catalog, mostly controlled by selecting words from the register. By using keywords, it is possible to search for books with similar content. In the same way, the keywords are used on the Web Search for websites with specific content.

Recently the concept of social networks and online communities is becoming still more and more popular. As a result, the number of their users significantly increasing. Reasons for communication between people and creation of social networks in our time are various: study, dating, travelling and tourism, work, games and programming is not the exception.

Many programmers on the Internet are looking for interesting ideas, or assistance when implementing their own solutions. Online collaboration is no longer a novelty in our times and it is run by people all over the world. However, searching for suitable and capable people who could implement a particular idea at reasonable deadlines and high quality is an eternal problem.

OSS (Open Source Software) is a example of a dynamic network, as well as a prototype of complex networks emerging on the Internet. By working through the Internet, interactions between developers can be considered as relations in the synthetic network

[1] CodePlex: http://codeplex.com

of collaborators. These relations arise when the developers join the project and begin to communicate with others. OSS network consists of two entities - developers and projects. An examples of such OSS social network established on the basis of interaction between the participants is CodePlex.

In this paper we try to determine the strength of relationship or similarity between CodePlex developers in the context of projects they work on. To determine the context, we used project key words, which in the case of the CodePlex are extracted from project descriptions.

Some related work dealing with the terms extraction in the social network. In the article [11] author illustrate ontology emergence by a novel method for the extraction of community-based ontologies from Web pages. Other approache is in the articel [12], where authors examine the dynamics of social network structures in Open Source Software teams but data were extracted monthly from the bug tracking system in order to achive a longitudinal view of the interaction pattern of each project.

2 CODEPLEX

CodePlex is a specialized web portal operated by Microsoft. It is mainly used by developers for collaboration on projects, sharing source codes, communication and software development. Generally, registered users can participate in multiple projects, discussions, adding the source code and documentation, issue a release, etc. Some of the users have defined a specific role within the project for which they work. Each user has his own page, where he can share information about himself, his projects on which he currently works, and the most recent activities. The CodePlex projects themselves can be considered as a very interesting source of information. In addition to the list of users and roles, CodePlex enables register keywords, add description of the project, the number of visits, status, date of creation, url and other information about the project. All activities are carried out on CodePlex by a particular user within a specific project.

Database which was created as a result of data obtained from CodePlex.com, consists of 6 main tables: User - 96251 records, Project - 21184 records, Discussions - 397329 records, RecentActivity - 72285 records, Membership - 126759 records and SourceCode - 610917 records.

In CodePlex, we can see two types of entities: users and projects. Both are represented by tables that contain specific characteristics (see Table 1).

The undirect connection between the user and the project is implemented through activities within the scope of the project. These activities are in the database CodePlex divided into different types: SourceCode, Discussion, RecentActivity and Membership.

- In the **SourceCode**, there are records about added projects.
- **Discussion** describes discussions about the project and the responses of individual users.
- **RecentActivity** records activities such as check-in, task records, add project to the Wiki information, note about Release version etc.
- In the table **Membership**, we are able to trace the users' participation in the projects and their assigned role in them.

Entity	Attribute	Type
User	login	character(32)
	personalStatement	character(255)
	createdOn	date
	lastVisit	date
	url	character(255)
Project	nameProject	character(255)
	tags	text
	createdOn	date
	status	character(255)
	license	character(255)
	pageViews	integer
	visits	integer
	url	character(255)
	description	text

Table 1. Tables User and Project

We can represent CodePlex as a bipartite graph of users and projects, where the edge between the user and the project is a user's activity in a project.

The Tables User, Project and Activity store in the CodePlex database information about time of occurrence, as well as the last modification or the last visit. Time of creation or last modification is not defined for all activities. Membership activities have no time component and Activities, SourceCode and RecentActivity do not track or change the last visit.

The execution time for casual activity component is often defined by verbal description, such as Today, Last Week, Monday. This data format is not suitable for analysis, and so the time was ignored for component-type Activities, Discussions and SourceCode. Projects and users have the time records in the correct format. Furthermore, all entities were analyzed for a better overview what data are available.

For each table were found a maximum and minimum time values of individual entities (users, projects and activities). We introduce a table of data creation and changes in the CodePlex database (see Table 2).

	createdOn/InsertTime		lastVisit/updateTime	
Entity	max	min	max	min
User	6/3/2011	14/4/2006	20/3/2011	15/1/2009
Project	28/2/2011	28/4/2006	not defined	
Discussions	can't be defined			
RecentActivity	20/3/2011	19/1/2011	not defined	
Membership	not defined			
SourceCode	can't be defined		not defined	

Table 2. Table Project

If we look at the data that we have in the Table User, we are not able to define the user's profile. It consists of the field of interest, what he deals with, the programming language he uses and at what level. PersonalStatement attribute is used to describe the user, but from the total set of our users downloaded, there was not a single one, who would fill it up. On the other hand, the project has enough information defined – which fields are concerned, how long it lasted, whether it is completed, which technology it is used, etc.

The main attribute, carrying the largest set of information, is the project Description – the description of the project itself.

Using activities such as user links to the projects, we are able to determine with some probability an area of specialization and a work of each user. For example, if a user is working on three projects written in .NET and one in Java, we could include him in .NET programmers with high probability, and less likely recommend him as a Java programmer.

In other words, terms or description of the project may not only help us to provide more information about projects, but also to determine the user's area of interests or abilities. As a result, the way we are able to compare user attributes determines the similarity to other network participants.

3 How to Construct Graph of Collaborators

Whenever we think about collaboration between two persons, we not only look at the relationship itself, but also at the context. It is clear that depending on context, the strength of relationship changes. Therefore, we divide collaboration into two main parts *Persons' Relationship* and *Relationship Context*.

Comparing with definition in article [10], where the basis is relation between a person and a term, and another colleague is seen as a context, we now consider the relation between persons as a main part, while the term describes the context. Although the computation process is almost the same, we think this reflects the reality better.

When we describe collaboration as a part of reality, we always start with defining main set of collaborated persons P. Although persons P could in fact represent any object in reality, process was designed specifically for the real persons.

Persons has additional attributes. Usually it could be publications, teams, organizations, projects, etc. We called it attribute domain. Let us define a sample organizations set as $D_O = \{Microsoft, Oracle, IBM, \ldots\}$. To specify organizations of person P_i. We can then specify the set of all persons' organizations as attribute set $O = \{O_{P_0}, O_{P_1}, \ldots, O_{P_n}\}$. If we want to express that person has some attribute, we create a subset from set which defines all the possible values. For example we can create a subset $O_{P_i} \subseteq D_O$, so O_{P_i} could be $\{Microsoft, Oracle\}$.

Generally, let D_X be a set of attribute domain, then X are attributes for all persons P, where object $X_{P_i} \in X$ is one person's attributes described as $X_{P_i} \subseteq D_X$.

3.1 Persons' Relationship

We describe a persons' relationship as commutative operation $\bullet$ on cartesian product of person's attribute $X\,x\,X$, where output is mapped to the set of real numbers $\mathbb{R}$.

$$Attribute_{Score}(X_{P_i}, X_{P_j}) = X_{P_i} \bullet X_{P_j} \in \mathbb{R} \tag{1}$$

An easy implementation of operation $\bullet$ are standard set operations like intersection or union. To get a real number, we just compute cardinality. Jaccard coefficient is the most typical operation we can use:

$$Attribute_{Score}(X_{P_i}, X_{P_j}) = \frac{|X_{P_i} \cap X_{P_j}|}{|X_{P_i} \cup X_{P_j}|} \tag{2}$$

It is clear to see that no matter what order of cartesian product we use; the result is the same. Other implementations could be simple matching coefficient, mutual information, Dice coefficient, overlap coefficient and many others.

Listing 1.1. Exported CodePlex data from XML file.

```
<codeplexProject key="...">
        <developer>hongmin0813</developer>
        <developer>lengleng3898</developer>
        <developer>lengleng38982nd</developer>
        <description>???</description>
        <year>29.4.2010 0:00:00</year>
        <url>http://lengleng3898.codeplex.com/</url>
        <meta>SS = 0</meta>
        <meta>RA = 0</meta>
        <meta>D = 0</meta>
        <meta>M = 0</meta>
</codeplexProject>
```

Applying to the projects in CodePlex, the base set Developers D is chosen as persons at first. We read sequentially the whole file and create sets D and CodePlex project CP as an attribute. For recording, we firstly read the author of the project, and if he is not in the set D, we add him as well the project he is working on, into CP_{D_i}. CP consists of all person's projects $\{CP_{D_0}, CP_{D_1}, \ldots, CP_{D_n}\} = CP$. After the whole analysis of the file, we can define AttributeScore computation for CodePlex:

$$Attribute_{Score}(CP_{D_i}, CP_{D_j}) = \frac{|CP_{D_i} \cap CP_{D_j}|}{|CP_{D_i} \cup CP_{D_j}|} \tag{3}$$

3.2 Relationship Context

As we discussed above, every person has it's attributes. Moreover, each person has a description text. If we use lexical analysis on this text, we can define a term set (or a m-gram set) for every person as T_{P_i}. Term set T consists of all persons term sets $\{T_{P_0}, T_{P_1}, \ldots, T_{P_n}\} = T$, when the domain for terms D_T could be easily obtained as union of all terms extracted for each person $D_T = T_{P_0} \cup T_{P_1} \cup \ldots \cup T_{P_n}$.

The whole process of obtaining term sets is described in [10], so we just remiding $(t_k \; in \; T_{P_i})$ stands for the number of terms t_k in the titles of articles by T_{P_i} and $(t_k \; in \; T)$ for the number of terms in titles in all articles.

We can evaluate association between the selected term $t_k \in D_T$ and a person $P_i \in P$:

$$R(T_{P_i}, t_k) = \frac{(t_k \; in \; T_{P_i})}{(t_k \; in \; T) + |T_{P_i}| - (t_k \; in \; T_{P_i})} \tag{4}$$

$$R_{Norm}(T_{P_i}, t_k) = \frac{R(T_{P_i}, t_k)}{MAX(R(T_{P_i}, t_1), \ldots, R(T_{P_i}, t_{|T_{P_i}|}))} \tag{5}$$

Evaluation of the whole relationship context of two persons P_i and P_j has two steps. First, we compute association between P_i and selecte term t_k, and between the second person P_j and t_k separately. Afterwards, because each part is already evaluated by real number, we combine both results in the same way; we can combine the whole result in equation one. However, the most usual is again muliplication, so we could write:

$$Context_{Score}(T_{P_i}, T_{P_j}, t_k) = R_{Norm}(T_{P_i}, t_k) \, R_{Norm}(T_{P_j}, t_k) \tag{6}$$

In CodePlex we see the description text for the developer as the all description of all projects he is working on, joined together.

$$Context_{Score}(T_{D_i}, T_{D_j}, t_k) = R_{Norm}(T_{D_i}, t_k) \, R_{Norm}(T_{D_j}, t_k) \tag{7}$$

3.3 Collaboration – Whole Score

The last step is to define Score, which consists of $Attribute_{Score}$ and $Context_{Score}$:

$$Score(X_{P_i}, X_{P_j}, T_{P_i}, T_{P_j}, t_k) = Attribute_{Score}(X_{P_i}, X_{P_j}) \, Context_{Score}(T_{P_i}, T_{P_j}, t_k) \tag{8}$$

We obtain for CodePlex:

$$Score(CP_{D_i}, CP_{D_j}, T_{D_i}, T_{D_j}, t_k) = Attribute_{Score}(CP_{D_i}, CP_{D_j}) \, Context_{Score}(T_{D_i}, T_{D_j}, t_k) \tag{9}$$

3.4 Building the Graph

To describe the network of collaboration, we use standard weighted graph $G(V,E)$, where weighted function is defined as $w : E(G) \mapsto \mathbb{R}$, when $w(e) \geq 0$.

The determination of set V is generally simple, because objects of vertices set V match with objects of set P, so $V = P$. However, we can do the same with all the possible pairs from set P to assign a set of edges E; it is better to design the algorithm to each implementation at first, and to reduct the number of useless computations. In addition, we must choose term t_k for function w, which reflects the context. Because only the commutative operations are used, we do not need to take into consideration the order of attribute objects in function parameters. Moreover E is two-object set, where the order of objects does not matter, so the evaluating is done just once.

When we construct graph based on developers' projects relationship, we use $Attribute_{Score}(CP_{D_i}, CP_{D_j})$ as w, where no term is needed, then simply $V = D$, which means that every developer is a vertex in the graph. Then, for each developer $D_i \in D$ we find collaborators D_{i_C} and for each collaborator $D_j \in D_{i_C}$ we create two-object set $\{D_i, D_j\}$, which corresponds with an edge in the graph. Equation 3 is then used to evaluate the edge.

The function $Score(CO_{D_i}, CO_{D_j}, T_{D_i}, T_{D_j}, t_k)$ is used for evaluating the edges in the context of the term. The only difference is, that majority of developers has not chosen term in their description text, so the result will be 0 and no edge would exists. Hence, we first determine subset of developers $D_{t_k} \subseteq D$ for those that have a term in their description text, followed by the same steps described in the last paragraph to compute developers' projects relationship. Then, the term t_k is used for computation of the second part in $Context_{Score}(T_{D_i}, T_{D_j}, t_k)$. Finally, we calculate the whole *Score* by multiplication of both parts.

4 Experiments

For the basic computation of the collaboration, we chose the term "team" and apply it to the formula 5. The results were limited to the collaborators with whose the person has worked together on the project at least once. We show in the Table 3 values of $Attribute_{Score}$ for person with nickname CareBear and in the Table 4 for person with nickname shanselman.

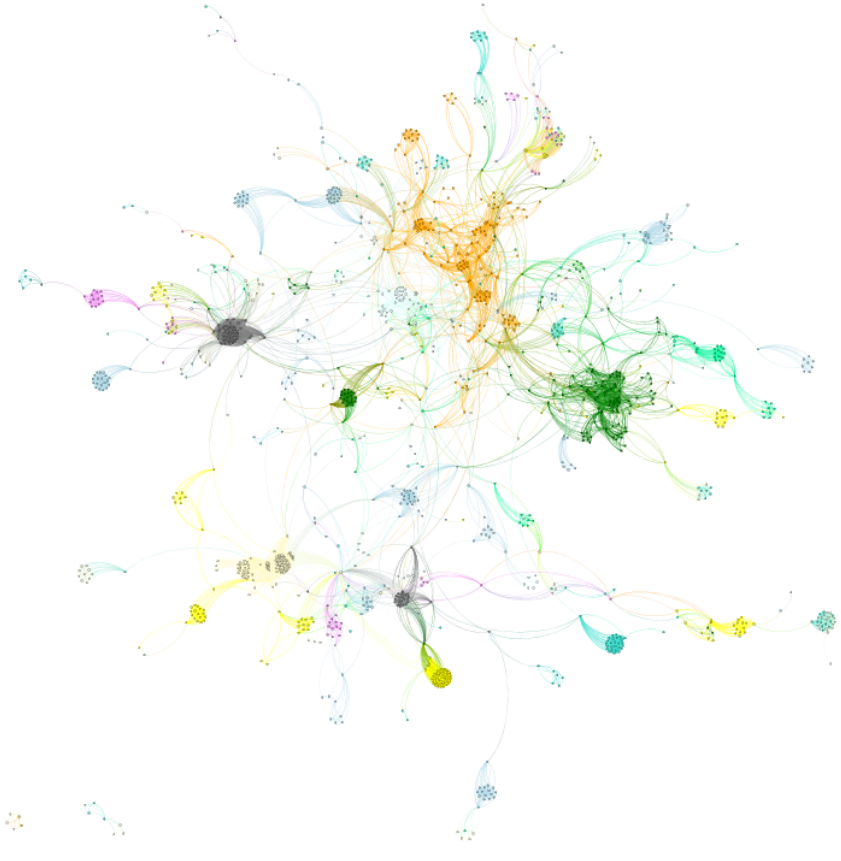


Fig. 1. Synthetic collaborators network for the term *team* - edge weights are computed by *Score*

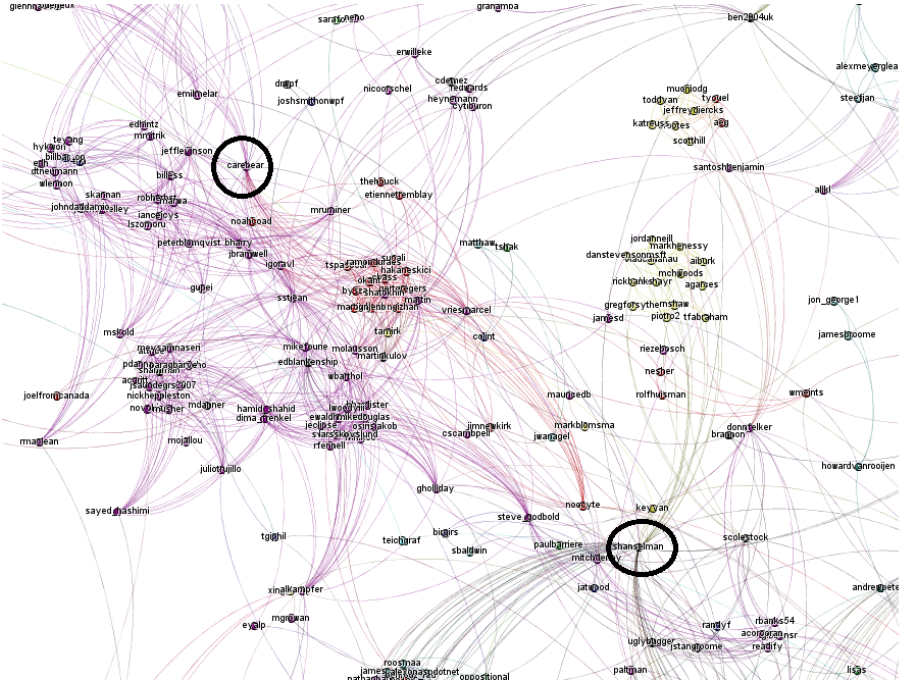


Fig. 2. Selected subnetwork with developers Carebare and shanelsman

Number	Coworkers	Projects	Common projects	$Attribute_{Score}$
1	CareBear	13	13	1
2	EmilMelar	2	2	0,1538462
3	Maggie	2	2	0,1538462
4	Kudzu2	2	2	0,1538462
5	kudzu	12	3	0,1363636
6	hhariri	6	2	0,1176471
7	amccool	1	1	0,07692308
8	arundeep	1	1	0,07692308
9	badmaash	1	1	0,07692308
10	frasse	1	1	0,07692308
...				
145	shanselman	20	1	0,03125
146	Microsoft	537	1	0,001821494

Table 3. Coworkers of CareBear

We can immediately notice that even though shanselman do not participate on many projects with CareBear (they have one common project), the AtributeScore is 0.03125. Conversely then, although shanselman (or CareBear) has with Microsoft 4 common projects, Microsoft cooperate with many other persons. Therefore, the shanselman (or CareBear) has not such a strong AtributScore with Microsoft.

Number	Coworkers	Projects	Common projects	$Attribute_{Score}$
1	shanselman	20	20	1
2	Haacked	14	3	0,09677419
3	jongalloway	14	3	0,09677419
4	SteveSanderson	4	2	0,09090909
5	shahineo	4	2	0,09090909
6	JasonHaley	6	2	0,08333334
7	ben2004uk	7	2	0,08
8	bsimser	8	2	0,07692308
9	AArnott	8	2	0,07692308
10	dcazzulino	20	2	0,05263158
11	agsmith	1	1	0,05
12	BartRead	1	17 0,05	
...				
147	CareBear	13	1	0,03125
...				
152	ReedMe	31	1	0,02
153	Microsoft	537	4	0,007233273

Table 4. Coworkers of shanselman

4.1 Key Terms Computation

At first, we have calculated the keywords for the CareBear and shanselman. We have selected only the first 15 terms for illustration (see Table 5 and Table 6). For comparison we marked some terms (bold text).

number	t_k	t_k *in* $TP_{CareBear}$
1	flickr	1
2	cosmo	0,9894736
3	automaton	0,9415065
4	ovik	0,8872708
5	downloadr	0,8238943
6	weeb	0,7605178
7	scrum	0,571498
8	photo	0,5248883
9	**team**	0,4844927
10	tf	0,404665
11	associ	0,3172817
12	flickr downloadr	0,3168824
13	**store**	0,3070892
14	process templat	0,2949297
15	team foundat	0,2901235
...		
926	set	0,007442362

Table 5. Key Terms for the person CareBear

number	t_k	t_k *in* $T_{P_{shanselman}}$
1	syndic	1
2	administr	0,75
3	peer	0,6934211
4	creatur	0,6413794
5	terrarium	0,5994475
6	argot	0,5570145
7	reflector	0,5359043
8	ecosystem	0,5102881
9	nuget	0,3803681
10	browser	0,3582435
11	assembl	0,3461412
12	consum	0,3358614
13	mobil	0,3216281
14	ad	0,309705
15	nerddinn	0,2954391
...		
82	**team**	0,1306442
...		
174	**store**	0,08637504
1444	system	0,007958922

Table 6. Key Terms for the person shanselman

In the Figure 1 is whole network of collaborators for the term *team*. Here is 31 connected components (communities) with collaborating developers. Figure 2 shows graphs of synthetic collaborators network generated for the term "team" and for selected developers.

5 Conclusion

Research presented in this article is oriented to the strength extraction between persons based on their context in the CodePlex. The method was presented using the data collection from the CodePlex database, which contains information of the activities of developers in the project. The proposed method is usable for the development of collaboration network. The description of this network is based on the set of terms (as the input), which are used in the description of projects by the given developer. Using this method, we have obtained the new weight in the synthetic collaborators network. By means of the set of selected term, belonging to one (or more) persons, we can construct the subnetwork with only the context-related collaborators. This subnetwork can be very helpful in searching of the persons who are interested in the same area, defined by the selected term. It is usable for members of the project management, who need to find suitable developers specialized to certain area. It follows that this method can be used to a certain extent for prediction as well.

Acknowledgment

This work was supported by SGS, VSB – Technical University of Ostrava, Czech Republic, under the grant No. SP2012/151 Large graph analysis and processing.

References

1. E. Deza and M. Deza. Dictionary of distances. 1-391, 2006.
2. Ch. Jacquemin and B. Didier. Term Extraction and Automatic Indexing. Handbook of Computational Linguistics. Oxford University Press, 599-615, 2003.
3. M. Konchady. Text Mining Application Programming (Programming Se-ries). Charles River Media, Rockland, MA, USA, May 2006.
4. A. H. Lashkari, F. Mahdavi, V. Ghomi. A Boolean Model in Information Retrieval for Search Engines, 2009.
5. P. Lopez and R. Laurent. HUMB: Automatic Key Term Extraction from Scientific Articles in GROBID. Computational Linguistics July, 248-251, 2010.
6. J. Mori, Y. Matsuo, M. Ishizuka, B. Faltings. Keyword extraction from the Web for FOAF metadata, In Proceedings of the 1st Workshop on Friend of a Friend, Social Networking and the (Semantic) Web, 2004.
7. M. F. Porter. An algorithm for suffix stripping. Program, 14:130-137, 1980.
8. Y. Ding. Scientific collaboration and endorsement: Network analysis of coauthorship and citation networks. Journal of informetrics 5.1,187-203, 2011.
9. R. R. T. Santamaría. Overlapping Clustered Graphs: Co-authorship Networks Visualization. Lecture Notes in Computer Science 5166, 190-199, 2008.
10. S. Minks, J. Martinovic, P. Drazdilova and K. Slaninova. Author Cooperation based on Terms of Article Titles from DBLP, IHCI 2011, Praha, 2011.
11. P. Mika. Ontologies are us: A unified model of social networks and semantics. Web Semantics Science Services and Agents on the World Wide Web 5, 5-15, 2007.
12. Y. Long, K. Siau. Social Network Structures in Open Source Software Development Teams. Journal of Database Management 18, 25-40, 2007.

Dynamic Time Warping in Analysis of Student Behavioral Patterns

Kateřina Slaninová[1], Tomáš Kocyan[2], Jan Martinovič[2], Pavla Dráždilová[1], and Václav Snášel[2]

[1] VŠB - Technical University of Ostrava,
Faculty of Electrical Engineering and Computer Science,
17. listopadu 15/2172, 708 33 Ostrava, Czech Republic
(katerina.slaninova,pavla.drazdilova)@vsb.cz
[2] VŠB - Technical University of Ostrava,
IT4Innovations,
17. listopadu 15/2172, 708 33 Ostrava, Czech Republic
(tomas.kocyan,jan.martinovic,vaclav.snasel)@vsb.cz

Abstract. E-learning systems store large amount of data based on the history of users' interactions with the system. These pieces of information are usually used for further course optimization, finding e-tutors in collaboration learning, analysis of students' behavior, or for other purposes.
The paper deals with an analysis of students' behavior in learning management system. The main goal of the paper is to find, how selected methods can influence finding of behavioral patterns in learning management system and how we can reduce the amount of extracted sequences. The methods of process mining and sequential pattern mining were used for extraction of behavioral patterns. The authors present the comparison of selected methods for the definition of students' behavior with the focus to influence of dynamic time warping. Obtained patterns and relations between them are presented using complex networks; the visualization and pattern clusters extraction is optimized by spectral graph partitioning.

1 Introduction

E-learning is a method of education which utilizes a wide spectrum of technologies, mainly internet or computer-based, in the learning process. It is naturally related to distance learning, but nowadays is commonly used to support face-to-face learning as well. *Learning management systems* (LMS) provide effective maintenance of particular courses and facilitate communication within the student community and between educators and students [4]. Such systems usually support the distribution of study materials to students, content building of courses, preparation of quizzes and assignments, discussions, or distance management of classes. In addition, these systems provide a number of collaborative learning tools such as forums, chats, news, file storage etc.

LMS based on computer and web-based education environments provide storage of large amount of accessible information. These systems record information about students' actions and interactions onto log files or databases. Within these records, data about students learning habits can be found including favored reading materials, note

taking styles, tests and quizzes, ways of carrying out various tasks, communication with other students in virtual classes using chat, forum, and etc. Other common data, such as personal information about students and educators (user profiles), student results and user interaction data, is also available in the system databases [1].

Such data collections are essential for analyzing students' behavior and can be very useful in providing feedback both to students and educators. For students, this can be achieved through various recommended systems and through course adaptation based on student learning behavior. For teachers, some benefits would include the ability to evaluate the courses and the learning materials, to detect the typical learning behavior or to find possible students suitable for collaborative learning [18].

Regardless of LMS benefits, huge amount of recorded data in large collections makes often too difficult to manage them and to extract useful information from them. To overcome this problem, some LMS offer basic reporting tools. However, in such large amount of information the outputs become quite obscure and unclear. In addition, they do not provide specific information of student activities while evaluating the structure and content of the courses and its effectiveness for the learning process [23]. The most effective solution to this problem is to use data mining techniques [1].

The main goal of the paper is to compare selected data mining methods suitable for the extraction of students' behavioral patterns performed in LMS Moodle. The behavioral patterns are obtained using methods of process mining and sequential mining, the patterns are presented using methods from graph theory. The organization of the paper is as follows: Section 2 consists of the background related to the methods used for the analysis of students' behavior. Process mining issues and selected methods for comparison of sequences are presented here. In Section 3 is presented an extraction of sequences used for students' behavior description from log file of e-learning system. Then, we are presented results of experiments provided on e-learning system Moodle. The experiments are focused to the extraction of students' behavioral patterns and to the comparison of selected methods. For easier analysis of the students' behavior is important the reduction of amount of sequences. We have used spectral clustering algorithm, which determine number of important clusters with behavioral patterns. The last Section 4 contains the conclusion.

2 Analysis of Students' Behavior

Several authors published contributions with relation to mining data from e-learning systems to extract knowledge that describe students' behavior. Among others we can mention for example [11], where authors investigated learning process of students by the analysis of web log files. A 'learnograms' were used to visualize students' behavior in this publication. Chen et al. [3] used fuzzy clustering to analyze e-learning behavior of students. El-Hales [6] used association rule mining, classification using decision trees, E-M clustering and outlier detection to describe students' behavior. Yang et al. [22] presented a framework for visualization of learning historical data, learning patterns and learning status of students using association rules mining. The agent technology and statistical analysis methods were applied on student e-learning behavior to evaluate findings within the context of behavior theory and behavioral science in [2].

However, contributions oriented to analysis of students' behavior in e-learning systems describe the behavior using statistical information, for visualization and representation of obtained information are mostly used only common statistical tools like figures or graphs. They usually do not provide information about behavioral patterns with effective visualization, nor information about relations between students based on their behavior.

2.1 Process Mining

Our subject of interest in this paper is student behavior in LMS, which is recorded in form of events and stored in the logs. Thus, we can define the student behavior with the terms of process mining which are used commonly in business sphere. Aalst et al. [20, 19] defines event log as follows:

Definition 1. *Let A be a set of activities (also referred as tasks) and U as set of performers (resources, persons). $E = A \times U$ is the set of (possible) events (combinations of an activity and performer). For a given set A, A^* is the set of all finite sequences over A. A finite sequence over A of length n is mapping $\sigma =< a_1, a_2, \ldots, a_n >$, where $a_i = \sigma(i)$ for $1 \leq i \leq n$. $C = E^*$ is the set of possible event sequences. A simple event log is a multiset of traces over A.*

Then, student behavior in LMS can be described by set of event sequences. More detailed description is presented in Section 3.

The paper is oriented to finding behavioral patterns. Behavioral patterns are discovered using similarity of extracted sequences. A sequence is an ordered list of elements, denoted $< e_1, e_2, \ldots, e_l >$. Given two sequences $\alpha =< a_1, a_2, \ldots, a_n >$ and $\beta =< b_1, b_2, \ldots, b_m >$. α is called a subsequence of β, denoted as $\alpha \subseteq \beta$, if there exist integers $1 \leq j_1 < j_2 < \ldots < j_n \leq m$ such that $a_1 = b_{j1}, a_2 = b_{j2}, \ldots, a_n = b_{jn}$. β is than a super sequence of α.

In the problem of finding similar behavior, we do not use traditional methods of sequential pattern mining where usually frequently repeated patterns are extracted. For finding the behavioral patterns, we need to use the methods for the sequence comparison, described in Section 2.2.

2.2 Comparison of Sequences

There are generally known two basic groups of algorithms for the comparison of two or more categorical sequences. The first group divides the algorithms by the fact, whether the sequences consist of ordered or unordered elements. The second group of algorithms focuses on the comparison of the sequences with the different lengths and with the possible error or distortion.

The basic approach to the comparison of two sequences, where the order of elements is important, is *The longest common substring* (LCS) method [10] (see example in Table 1). As obvious from the name of the method, the main principle of the method is to find the length of the common longest substring. Given the two sequences x and y, we can find such subsequence $z =< z_1, z_2, \ldots, z_p >$, where $z_k = x_{i+k-1} = y_{j+k-1}\ \forall k = 1, \ldots p$ and $p \leq m, n$.

The LCS method respects the order of elements in the sequence. However, the main disadvantage is, that it can find only identical subsequences, where no extra element is presented in the sequence. For some domains, typically where is large amount of different sequences, gives this fact too strict limitation.

As a solution of this problem we can consider *The longest common subsequence* (LCSS) described for example in [12] (see example in Table 1). Contrary to The longest common substring, this method allows (or ignores) the inserted extra elements in the sequence, and therefore, it is immune to slight distortions.

	a	b	c
Sequence X	E**ABC**F	E**A**E**BC**E	**AB**B**C**C
Sequence Y	Z**ABC**T	F**ABC**F	E**ABC**E
Longest Common substring	ABC	BC	AB
Common subsequence (LCSS)	ABC	ABC	ABC
Common subsequence (TWLCS)	ABC	ABC	ABBCC

Table 1. Results of algorithms for subsequence detection

Whether we define the similarity of compared sequences as a function using a length of common subsequence, we can find one characteristic of this method. The length of the common subsequence is not immune to recurrence of identical elements, which can occur only in one of the compared sequences. We can find such situations, for example due to inappropriate sampling or due to any kind of distortion.

In some applications, it is suitable (or sometimes even required) to eliminate such type of distortions and to work with them like with equivalent elements. The solution is in another method, *The time-warped longest common subsequence* (T-WLCS) [9] (see example in Table 1). The method combines the advantages of LCSS method with dynamic time warping[13]. Dynamic time warping is used for finding the optimal visualization of elements in two sequences to match them as much as possible. This method is immune to minor distortions and to time non-linearity. It is able to compare sequences, which are for standard metrics evidently not comparable.

The method emphasizes recurrence of elements in one of the compared sequences. Due to this fact the length of the common subsequence can be longer than the shorter length of the compared sequences.

In the experiments described in the paper, the authors compare the impact of LCSS and T-WLCS methods to the construction of derived network based on similar behavior of students in e-learning system.

3 Sequence Extraction in LMS Moodle

In this section is presented the extraction of students' behavioral patterns performed in the e-learning educational process. The analyzed data collections were stored in the Learning Management System (LMS) Moodle logs used to support e-learning education at Silesian University, Czech Republic.

The logs consist of records of all events performed by Moodle users, such as communication in forums and chats, reading study materials or blogs, taking tests or quizzes etc. The users of this system are students, tutors, and administrators; the experiment was limited to the events performed only by students.

Let us define a set of students (users) U, set of courses C and term *Activity* $a_k \in A$, where $A = P \times B$ is a combination of activity prefix $p_m \in P$ (e.g. course view, resource view, blog view, quiz attempt) and an action $b_n \in B$, which describes detailed information of an activity prefix (concrete downloaded or viewed material, concrete test etc.). *Event* $e_j \in E$ then represents the activity performed by certain student $u_i \in U$ in LMS. On the basis of this definition, we have created a set S_i of sequences s_{ij} for the user u_i, which represents the students' (users') paths (sessions) on the LMS website. *Sequence* s_{ij} is defined as a sequence of activities, for example $s_{ij} =< a_{1j}, a_{2j}, \ldots, a_{qj} >$, which is j-th sequence of the user u_i.

The sequences were extracted likewise the user sessions on the web; the end of the sequences was identified by at least 30 minutes of inactivity, which is based on our previous experiments [5]. Similar conclusion was presented by Zorrilla et al. in [23].

Using this method, we have obtained a set of all sequences $S = \cup_{\forall i} S_i$, which consisted of large amount of different sequences s_l performed in LMS Moodle. We have selected the course Microeconomy A as an example for the demonstration of proposed method. In Table 2 is presented detailed information about the selected course.

Records	Students	Prefixes	Actions	Sequences
65 012	807	67	951	8 854

Table 2. Description of Log File for Course Microeconomy A

Sequence appearance in the selected course follows the power law distribution.

As mentioned in Section 3, the obtained set S of sequences consisted of large amount of different sequences, often very similar. Such large amount of information is hard to clearly visualize and to present in well arranged way. Moreover, the comparison of users based on their behavior is computationally expensive with such dimension. Therefore, we present the identification of significant behavioral patterns based on the sequence similarity, which allows us to reduce amount of extracted sequences.

Following experiment is oriented to exploration, how the different methods for measurement of sequence similarity can influence finding of behavioral patterns. We have used LCSS a T-WLCS methods for the similarity measurement of sequences, described in Section 2.2, with comparison to the common one, cosine similarity. Cosine similarity [15] is well known method for similarity measurement in informational retrieval while working with vector model. Both methods LCSS and T-WLCS find the longest common subsequence α of compared sequences β_x and β_y, where $\alpha \subseteq \beta_x \wedge \alpha \subseteq \beta_y$, with relation to both methods, see Section 2.2. Similarity was counted by the Equation 1.

$$Sim(\beta_x, \beta_y) = \frac{(l(\alpha) * h)^2}{l(\beta_x) * l(\beta_y)}, \tag{1}$$

where $l(\alpha)$ is a length of the longest common subsequence α for sequences β_x and β_y; $l(\beta_x)$ and $l(\beta_y)$ are analogically lengths of compared sequences β_x and β_y, and

$$h = \frac{Min(l(\beta_x), l(\beta_y))}{Max(l(\beta_x), l(\beta_y))} \tag{2}$$

Table 3 shows cosine similarity and similarity counted by the methods LCSS and T-WLCS for selected sequences and sequence '4408,4409,4407,2181,2133'.

Sequence	Similarity		
	Cosine	**LCSS**	**T-WLCS**
'4408,4409,4407,2181,2133'	1	1 (5)	1 (5)
'4409,4407,4408,2181,2133'	1	0.64(4)	0.64 (4)
'4407,4409,4408,2181,2133'	1	0.36 (3)	0.36 (3)
'4408,4409,4407,2181'	0.991	0.512 (4)	0.512 (4)
'4409,4407,4408,2181'	0.991	0.288 (3)	0.288 (3)
'4409,4407,4408,2181,2133,225'	0.979	0.370 (4)	0.370 (4)
'4408,4409,4407,2181,225,2133'	0.979	0.579 (5)	0.579 (5)
'4407,4408,4409,2181,2133,225'	0.979	0.370 (4)	0.370 (4)
'4407,4408,4409,2181,2133,225,2133'	0.976	0.233 (4)	0.364 (5)
'4408,4409,4407,4408,2181,2133'	0.972	0.579 (5)	0.579 (5)

Table 3. Example: Similarity of '4408,4409,4407,2181,2133'to Selected Sequences

Numbers in the brackets present the length of the founded longest common sequence for each method. From Table 3 (for example from the second row of the table) is evident the significant disadvantage of cosine similarity: it does not take into consideration the ordering of events in the sequence, while the methods LCSS and T-WLCS do. However, cosine similarity supports weighted vector model, where frequency of attributes is taken into consideration. In our method, tf-idf weighting was used. From the 9th row we can see the difference between the methods LCSS and T-WLCS. T-WLCS method takes into consideration the recurrence of elements in one of the compared sequences.

On the basis of selected method for finding the similarity of sequences, we have constructed the similarity matrix for sequences ($|S| \times |S|$) which can be represented using tools of graph theory. For the visualization of network was constructed weighted graph $G(V,E)$, where weight w is defined as function $w : E(G) \rightarrow R$, when $w(e) > 0$. Set V is represented by set of sequences S, weights w are evaluated by the similarity of sequences, see Equation 1, depending on selected method. In Table 4 is more detailed description of weighted graphs of sequences, where weight is defined by cosine similarity and similarity counted on the basis of LCSS and T-WLCS method for selected threshold θ. The number of nodes for each graph is 5908.

From Table 4 we can see, that each graph consists of large amount of similar sequences. Moreover, they are dense and very large for further processing. Better interpretation of results is possible by finding the components, which can represent the

Cosine Measure				
θ	**Isolated Nodes**	**Edges**	**Avg. Degree**	**Avg. Weighted Degree**
0.1	0	13292202	2249.865	464.377
0.2	2	4651152	787.263	261.303
0.3	5	2040406	345.363	155.013
0.4	32	1050138	177.748	97.387
0.5	122	554278	93.818	60.066
0.6	395	290632	49.193	35.747
0.7	897	147984	25.048	20.181
0.8	1851	67584	11.439	10.034
0.9	3289	21966	3.718	3.524
LCSS				
θ	**Isolated Nodes**	**Edges**	**Avg. Degree**	**Avg. Weighted Degree**
0.1	113	1622206	274.578	45.563
0.2	672	352504	59.666	17.112
0.3	1711	85266	14.432	6.037
0.4	2586	39922	6.757	3.309
0.5	3812	13796	2.335	1.351
0.6	4891	3320	0.562	0.371
0.7	5697	390	0.066	0.049
0.8	5900	12	0.002	0.002
0.9	5908	0	0	0
T-WLCS				
θ	**Isolated Nodes**	**Edges**	**Avg. Degree**	**Avg. Weighted Degree**
0.1	31	5577366	944.036	179.431
0.2	143	1739042	294.354	88.883
0.3	606	534648	90.496	38.735
0.4	1200	271826	46.010	22.998
0.5	2465	103028	17.439	10.430
0.6	3781	29298	4.959	3.596
0.7	5038	8080	1.368	1.269
0.8	5517	5914	1.001	0.997
0.9	5568	5788	0.980	0.980

Table 4. Description of Sequence Graphs for Cosine Similarity, LCSS and T-WLCS

behavioral patterns. The graph reduction using only threshold θ leads to undesirable loss of information. Due to this reason, we have used spectral clustering by Fiedler vector and algebraic connectivity [7, 8]. More detailed description of finding components using this method was presented in our previous work [21, 14].

In table 5 are described graphs with different methods for computing similarity between sequences. The threshold was selected $\theta \geq 0.1$ or 0.2 for comparison between the largest components with similar size (bold numbers). We have analysed the largest connected components of each graph and we have obtained significant clusters after spectral clustering.

	Cosine $\theta \geq 0.1$	LCSS $\theta \geq 0.1$	T-WLCS $\theta \geq 0.1$	T-WLCSS $\theta \geq 0.2$
Connected Components	1	122	33	145
Size of the Largest Component	5908	**5778**	5875	**5763**
Clusters in the Largest Component	2	**19**	10	**20**
Size of Cluster 1	2969	2629	3472	1623
Size of Cluster 2	2939	1573	762	1247
Size of Cluster 3		1138	745	772
Size of Cluster 4		307	557	567
Size of Cluster 5		50	322	561

Table 5. Description of Components After Spectral Clustering

In Figure 1 we can see the weighted graph constructed for better visualization of the components with the similar sequences.

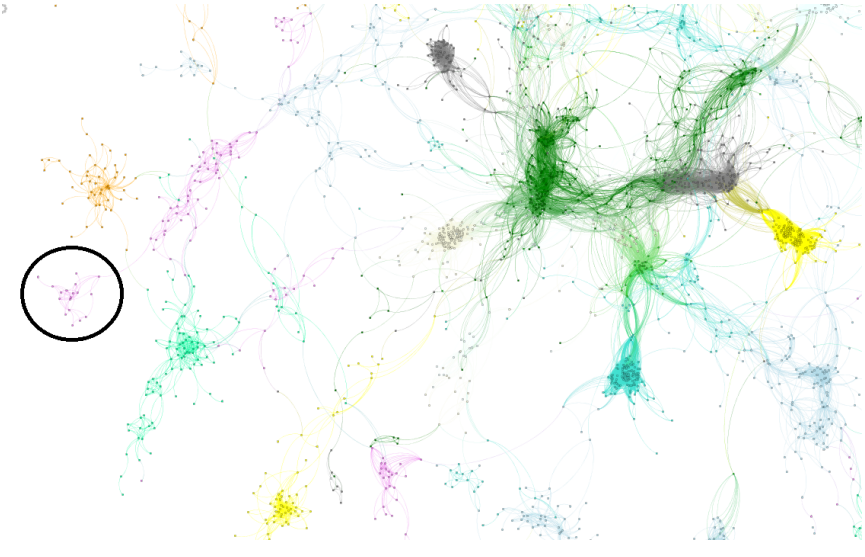


Fig. 1. Weighted Graph With Behavioral Patterns

The graph was constructed using an open souce software Gephi[3]. In Figure 1, the nodes in the graph represent the sequences, while the edges are weighted by their similarity using T-WLCS method. The graph was constructed using threshold θ=0.8. Each component in the graph can represent a behavioral pattern of similar sequences.

[3] http://gephi.org

Figure 2 shows the detailed view to the selected component from the graph on Figure 1. The component consists of 2940 sequences, each node is described by the sequence, while the actions in sequences have its own unique ID.

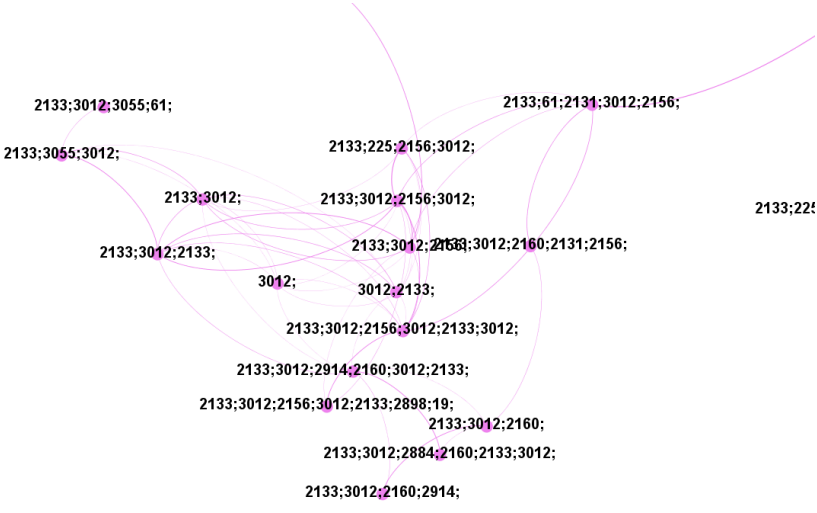


Fig. 2. Selected Component of Similar Sequences

It is possible to generate subgraphs relevant to selected activity, which can be in the area of our interest. The filtering by selected activities is performed by using vector model of sequences $\times$ activities.

4 Conclusion

The paper is oriented to finding the students' behavioral patterns performed in the e-learning system. The behavioral patterns were obtained using the methods of process mining and sequential mining, the patterns were visualized by the methods from graph theory. The authors focused on the comparison of the selected data mining methods suitable for the definition of the sequence similarity.

On the basis of previous experiments with suffix tree method and common vector model [17, 16] we have found, that the sequences are order dependent and it is better to respect this fact while comparing the sequence similarity. Due to this reason, the methods for finding the longest common subsequence were used.

In the experiments, the comparison of methods LCSS and T-WLCS with common vector model was described. Our results showed that each method has its unique characteristics. Vector model does not take into consideration ordering of actions inside the sequences, which is important disadvantage. On the other side, it allows weighting of activities on the basis of their frequency. LCSS and T-WLCS methods work with action

ordering and allow slight distortions in the sequence, while T-WLCS emphasizes the recurrence of elements in one of the compared sequences. These methods allowed to find the similarity between the two sequences more precisely.

On the basis of our experiments we have found that proposed method is usable for sequence extraction. Moreover, it can be effectively used for the reduction of sequence dimension. As we can see from presented results, we need to provide more precise division of extracted components to obtain more accurate behavioral patterns in some cases. The LCSS and T-WLCS methods are more time demanding than common cosine similarity. In our further work we intent to focus on their optimization. Another possible further work can be oriented to the definition of sequence similarity which will exploit the advantages from cosine measure and methods for finding the longest common subsequence.

Proposed method is suitable for finding the students' behavioral patterns in e-learning, which can be useful in providing feedback both to students and educators. Such type of information is valuable neither in e-learning sphere, nor in other areas like business process mining, finding behavior of users on the web, marketing etc.

Acknowledgment

This work was partially supported by SGS, VSB – Technical University of Ostrava, Czech Republic, under the grant No. SP2012/151 Large graph analysis and processing and by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

References

1. F. Castro, A. Vellido, A. Nebot, and F. Mugica. Applying data mining techniques to e-learning problems. *Studies in Computational Intelligence (SCI)*, 62:183–221, 2007.
2. B. Chen, C. Shen, G. Ma, Y. Zhang, and Y. Zhou. The evaluation and analysis of student e-learning behaviour. In *IEEE/ACIS 10th International Conference on Computer and Information Science (ICIS), 2011*, pages 244 – 248, 2011.
3. J. Chen, K. Huang, F. Wang, and H. Wang. E-learning behavior analysis based on fuzzy clustering. In *Proceedings of International Conference on Genetic and Evolutionary Computing*, 2009.
4. P. Dráždilová, G. Obadi, K. Slaninová, S. Al-Dubaee, J. Martinovič, and V. Snášel. Computational intelligence methods for data analysis and mining of elearning activities. In F. Xhafa, S. Caballe, A. Abraham, T. Daradoumis, and J. Perez, editors, *Studies in Computational Intelligence For Technology Enhanced Learning*, volume 273, pages 195–224. Heidelberg, Germany: Springer-Verlag, 2010.
5. P. Dráždilová, K. Slaninová, J. Martinovič, G. Obadi, and V. Snášel. Creation of students' activities from learning management system and their analysis. In A. Abraham, V. Snášel, and K. Wegrzyn-Wolska, editors, *IEEE Proceedings of International Conference on Computational Aspects of Social Networks CASON 2009*, pages 155–160, 2009.
6. A. El-halees. Mining students data to analyze learning behavior: a case study. 2008.
7. M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 1973.

8. M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25:619–633, 1975.
9. A. Guo and H. Siegelmann. *Time-Warped Longest Common Subsequence Algorithm for Music Retrieval*, pages 258–261. Universitat Pompeu Fabra, 2004.
10. D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.
11. A. Hershkovitz and R. Nachmias. Learning about online learning processes and students' motivation through web usage mining. *Interdisciplinary Journal of E-Learning and Learning Objects*, 5:197–214, 2009.
12. D. S. Hirschberg. Algorithms for the longest common subsequence problem. *J. ACM*, 24:664–675, October 1977.
13. M. Müller. *Information Retrieval for Music and Motion*. Springer, 2007.
14. G. Obadi, P. Dráždilová, J. Martinovič, K. Slaninová, and V. Snášel. Using spectral clustering for finding student's patterns of behavior in social networks. In *Proceedings of the Dateso 2010 Annual International Workshop on DAtabases, TExts, Specifications and Objects*, pages 118–130, 2010.
15. G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
16. K. Slaninová, R. Dolák, M. Miškus, J. Martinovič, and V. Snášel. User segmentation based on finding communities with similar behavior on the web site. In *Proceedings - 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT Workshops 2010*, pages 75–78, 2010.
17. K. Slaninová, J. Martinovič, T. Novosád, P. Dráždilová, L. Vojáček, and V. Snášel. Web site community analysis based on suffix tree and clustering algorithm. In *Proceedings - 2011 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT 2011*, pages 110–113, 2011.
18. V. Snášel, A. Abraham, J. Martinovič, P. Dráždilová, K. Slaninová, T. Daradoumis, F. Xhafa, and A. Martiínez-Monés. E-assessment of individual and group learning processes. *Journal of Computational and Theoretical Nanoscience*, 9(2):286–303, 2012.
19. W. M. P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Heidelberg, 1st edition, 2011.
20. W. M. P. van der Aalst, H. A. Reijers, and M. Song. Discovering social networks from event logs. *Comput. Supported Coop. Work*, 14(6):549–593, 2005.
21. L. Vojáček, J. Martinovič, K. Slaninová, P. Dráždilová, and J. Dvorský. Combined method for effective clustering based on parallel som and spectral clustering. In V. Snášel, J. Pokorný, and K. Richta, editors, *Proceedings of the 11th Annual Workshop DATESO 2011*, pages 120–131. VŠB - TU Ostrava, 2011.
22. F. Yang, R. Shen, and P. Han. Construction and application of the learning behavior analysis center based on open e-learning platform. 2002.
23. M. Zorrilla, E. Menasalvas, D. Marín, E. Mora, and J. Segovia. Web usage mining project for improving web-based learning sites. In *Computer Aided Systems Theory – EUROCAST 2005*, volume 3643/2005 of *Lecture Notes in Computer Science*, chapter Web Usage Mining Project for Improving Web-Based Learning Sites. Springer Berlin / Heidelberg, 2005.

The Bayesian Spam Filter with NCD*

Michal Prílepok[1], Jan Platoš[1], Václav Snášel[1], and Eyas El-Qawasmeh[2]

[1] Department of Computer Science, FEI, VSB - Technical University of Ostrava, 17. listopadu 15, 708 33, Ostrava-Poruba, Czech Republic
{michal.prilepok, jan.platos, vaclav.snasel}@vsb.cz
[2] King Saud University, Saudi Arabia
eyasa@usa.net

Abstract. Undesired e-mail (spam) becomes a big problem nowadays not only for users, but also for Internet providers. One of the main obstacles for elimination of this problem is a complicated security issues. In particular, the very low rate of falsely detected e-mails in practice. We tried to eliminate this problem by a sufficient similarity check of checked e-mails with e-mails marked as unrequested or spam. This paper uses Bayesian algorithm with many variations. For comparison purposes, we used a normalized compression distance which helped us reduce the rate of false detection of individual mails.

Keywords: e-mail, spam, Bayesian filter, data compression

1 Introduction

Spam senders are flooding us with increasing number of unrequested e-mails. Such practice makes us think about developing of defensive techniques. Currently, there are many developed techniques and approaches, which enable us to eliminate unrequested e-mails. These techniques and approaches can be divided into the following categories:

- Sender or mediator analysis
- E-mail content analysis
- Sponsor analysis

Neither of the given categories is separately effective enough. Nowadays, combination of various techniques and attitudes occurs in order to improve success of fighting against spam [10].

* This work was partially supported by the Grant Agency of the Czech Republic under grant no. P202/11/P142, SGS in VSB Technical University of Ostrava, Czech Republic, under the grant No. SP2012/58, and has been elaborated in the framework of the IT4Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 supported by Operational Programme 'Research and Development for Innovations' funded by Structural Funds of the European Union and state budget of the Czech Republic.

The organization of this paper will be as follows. Section 2 describes current related work. Section 3 contains description of the Bayesian spam filter. Section 4 describe normalized compression distance. Section 5 defines the interconnection of the Bayesian Spam filter an NCD. The results of our experiments are described in Section 6 and Section 7 contains conclusion of this paper.

2 Current Related Work

First techniques of fighting against spam came out of detection key words in the e-mail subject. These spam filters compared words contained in e-mail subject with the list of prohibited words. Spam authors started to avoid this way of spam detection by several modifications of e-mail subjects. Modification of e-mail subject is often used and makes us think that it was a forwarded e-mail from the third person, e.g. "RE: About us" [6, 10].

Later, filter techniques started to use body of the e-mail as well. In the same way, words from the e-mail body were compared to the list of prohibited words. This extension brought only a little improvement. Rate of success in detection of unrequested e-mail was dependent on the quality of the list of prohibited words. Individual prohibited words had to be chosen very carefully so that increasing in final false detection rate would not happen.

Filters use check-sums and marks for elimination of false detection rate. The check-sums are calculated over each received e-mail and resulting hash are compared with database of known spam e-mails. One-way transformation function also called hash function (MD5, SHA1, ...) are used as check-sum calculators.

Spam filtering based on key words [10] did not bring required success of unrequested e-mails detection. Analysis of recent e-mails trend seems to be a very effective method in fighting against unrequested mail. This technique reduces the rate of unrequested e-mail false detection. Heuristic filters are ranked to these techniques - rules-based filters and learning-based filters.

Rules-based filters [10] are looking for features which are characteristic for spam in the e-mails. These are some words (e.g. viagra), collocations and mistakes typical for spam. Example of such mistake can be sending e-mail with a future date, prohibited marks in the heading, incorrectly marked MIME type of e-mail, etc. Each detected feature has defined certain points score. Usually, points are summed up and if the sum exceed defined limit, the email is marked as a spam. Detected features are defined by the help of rules that have to be regularly updated and conformed to spamers practices.

Learning-based filters (often called bayesian) [7, 4, 2, 1] use tricks from the area of artificial intelligence. In the learning phase, the email is submitted into filter. Each email is marked as spam of ham (not spam). Filter extracts features from each email and stores them into database. Usually, the e-mail divided into words (eventually other text segments) and a probability of individual words is computed and statistically evaluated. The words probability for spam and ham emails is evaluated.

In detection phase, the filter uses collected information and computed the probability that the tested email is spam. The most often formula for probability calculation was suggested by mathematician Bayes. Learning filters are the most efficient if they are taught what is and what is not spam by end users according to their individual opinion. The Bayesian filters are also used in servers where the learning is made by all users together.

Even the Bayesian filter shows a high success of unrequested e-mail detection, false marking or non-detection still happens in some cases. This imperfection can be eliminated by comparison of the examined e-mail content with unrequested e-mails known in advance [6, 10].

3 The Bayesian spam filter

Bayesian spam filter is a statistic technique for filtering e-mails. It uses naive Bayesian classifiers for spam identification. The Bayesian classifiers work with relations between elements (typically words) from unrequested (spam) and requested e-mails. They calculate probability whether an e-mail is spam or not by the help of the Bayesian statistics. Particular words have a particular probabilities of occurrence in unrequested e-mails and legitimate e-mails. Filter does not recognize these probabilities in advance. It first has to learn them so that he could build upon them. Each new e-mail must be manually marked whether it is spam or is not. For all words in each e-mail the filter must adjust probability with which the given word occurs in spam or in legitimate e-mails in its database. For instance words "viagra" or "refinance" are often found in spam e-mails and names of friends or family members are often found in legitimate e-mails.

3.1 Calculation of probability that e-mail containing a word is spam

Probability of e-mail which contains a particular word can be calculated by the help of the following formula [7, 4, 2, 1]:

$$\Pr(S \mid W) = \frac{\Pr(W \mid S) \times \Pr(S)}{\Pr(W \mid S) \times \Pr(S) + \Pr(W \mid H) \times \Pr(H)}$$

where:

- $\Pr(S \mid W)$ is probability that e-mail is spam with knowledge that it contains the examined word.
- $\Pr(S)$ is total probability that e-mail is spam.
- $\Pr(W \mid S)$ is probability that the examined word occurs in spam
- $\Pr(H)$ is probability that the given e-mail is not spam (ham)
- $\Pr(W \mid H)$ is probability that the examined word occurs in ham e-mail

Probabilities for $\Pr(W \mid S)$ and $\Pr(W \mid H)$ will be determined in learning phase of the filter. By the help of $\Pr(H)$ and $\Pr(S)$, it may potentially affect partiality or impartiality of the filter against the checked mails. The more is

value of $\Pr(S)$ closer to 1.0, the more is filter partial against spam mails. The value $\Pr(H)$ adjust the filter's opposite. The more is this value higher, the less is the filter partial against spam mails. Summary of values $\Pr(H)$ and $\Pr(S)$ must be equal to 1.0. The statistics presents that the probability of spam is approximately 80%. On the basis of this statement we can determine values for $\Pr(\mathrm{s}) = 0.8$, $\Pr(\mathrm{h}) = 0.2$. In this case, the Bayesian filter anticipates that 80% of checked e-mails are spams and remaining 20% are legitimate ham mails. The majority of the Bayesian filters for detection uses a hypothesis that incoming e-mails contain less spam than legitimate e-mails (ham). Therefore, they have adjusted both probabilities to 50% ($\Pr(S) = 0.5$; $\Pr(H) = 0.5$)).

It can be said about the filters which use this hypothesis that they are impartial, they do not have any prejudice against incoming mails. This hypothesis enables simplification of the general formula to:

$$\Pr(S \mid W) = \frac{\Pr(W \mid S)}{\Pr(W \mid S) \times \Pr(W \mid H)}$$

This number is called *spamcity* or *spaminess* of the examined word. The value of $\Pr(W \mid S)$ that is used in this formula is rounded to frequency of e-mails containing the examined word in e-mails marked as spam during the learning phase. Similarly, the $\Pr(W \mid H)$ is rounded to frequency of e-mails containing the examined word in e-mails marked as ham during the learning phase. The collection of e-mails determined for learning has to be representative enough due to these approximations. Data files of ham and spam e-mails should be in accordance with a 50% hypothesis of the same size. Determination whether the e-mail is spam or ham, just on the basis of a single word, is prone to mistake. That is why the Bayesian filter tries to take into account several words and interconnect their spamcity in order to determine total probability.

3.2 Combination of individual probabilities

The Bayesian filter of unrequested e-mail assumes that words are independent. This is bad in natural languages where probability of detection of an adjective is affected, e.g. by probability of a noun. Considering this assumption we can deduce further formulas of the Bayesian theorem:

$$p = \frac{p_1 \times p_2 \ldots p_n}{p_1 \times p_2 \ldots p_n + (1 - p_1) \times (1 - p_2) \ldots (1 - p_n)}$$

where p is probability that the suspected e-mail is spam and p_i is probability that $\Pr(S \mid W_i)$ is spam containing the $i - th$ examined word.

The result p is compared to a specific value. If the result p is higher than the given limit, then the email is considered as a spam, otherwise it is a ham mail.

3.3 Use of rare words

In case that the word does not occur during the learning phase, numerator and denominator is equal to zero in general, but also in spamcity formula. Software

may leave out the words which do not provide any information. Words that occurred several times only in the phase of learning may cause a problem because it would be a mistake to believe to a blind information. Solution of this problem is to prevent of acceptance of such words into account. The Bayesian theorem is applied several times, and the division between spam and ham e-mails containing the examined word is a random quantity with beta distribution. Therefore, we may use a modified formula for calculation of probability:

$$\bar{\Pr}(S \mid W) = \frac{s \times \Pr(S) + n \times \Pr(S \mid W)}{s + n}$$

where

- $\bar{\Pr}(S \mid W)$ is corrected probability of spam e-mail with knowledge that it contains the examined word.
- s is a strength by which we give basic information about incoming unrequested mail.
- $\Pr(S)$ is probability that incoming e-mails are spams.
- n is a number of occurrences of the examined word in the course of the learning phase.
- $\Pr(S \mid W)$ is spamcity of the examined word.

This corrected probability is used instead of spamcity in combined formula. $\Pr(S)$ may equal to 0.5 in order to avoid too big distrust towards incoming mails. Three is a good value for s, which means that the examined word must exist three times within learning e-mails in order to increase trust of spamcity value as a default value. This formula may be extended in case when n is equal to 0 (and where spamcity is not defined). In this case, $\Pr(S)$ is evaluated.

3.4 Other heuristics

Neutral words like *the*, *and*, *some*, or *is* (in English), or their equivalents in other languages may be ignored. Generally said, some Bayesian filters ignore all words which has spamcity around 0.5, because they bring insufficiently good decision. Words taken into account should have spamcity close to 0.0 (distinctive mark of legitimate mail), or 1.0 (distinctive mark of spam mail). Method may look like this for example: 10 words that have the highest absolute value $|0.5 - p|$.

Some software products consider the fact that the given word occurs several times in the examined e-mail and others not.

Some software products use samples (word sequences) instead of separated words of natural language. For instance, they calculate the spamcity value of four words "Viagra is good for", instead of calculation of spamcity values for each word "Viagra", "is", "good" and "for". This method provides a higher sensitivity to context and leads to better elimination of the Bayesian noise to the detriment of a bigger database.

4 Normalized Compression Distance

Normalized Compression Distance (NCD) is a mathematical way for measuring similarity of objects. Measuring of similarity is realized by the help of compression where repeating parts are suppressed by compression. It is based on algorithmic difficulty of the Normalized Information Distance (NID) developed by Andrey Kolmogorov. NCD may be used for comparison of different objects, such as music, texts or gene sequences. We may use NCD for detection of plagiarism and visual data extraction [9].

Resulting rate of probability distance is calculated by the following formula:

$$NCD = \frac{C(xy) - \min(C(x), C(y))}{\max(C(x), C(y))}$$

Where:

- $C(x)$ is size of compressed file x.
- $C(y)$ is size of compressed file y.
- $C(xy)$ is size of compressed file created by interconnected files x and y.
- $\min\{x, y\}$ is minimum of values x and y.
- $\max\{x, y\}$ is maximum of values x and y.

The NCD is in the interval $0 \leq NCD(x, y) \leq 1$. If $NCD(x, y) = 0$, then files x and y are equal. They have the highest difference when the result value of $NCD(x, y) = 1$.

4.1 Implementation

In our approach we use a GZIP program for data compression. GZIP internally use a DEFLATE compression algorithm [3]. Deflate algorithm is based on the variant of LZ77 algorithm [11] called LZSS [8]. It also uses a semi-adaptive version of Huffman encoding [5]. LZ77 algorithm and its variants belong to the dictionary based compression algorithms which replace a symbol (bytes, characters, etc.) sequences by references into dictionary. LZSS algorithm uses two types of reference. The first type of reference in not reference at all because it represents one symbol. The second type of reference is a position and length of the same sequence in the already encoded text. All three parts - characters, positions and lengths are encoded by Huffman encoding, but each element has its own model, which increases the compression efficiency.

This algorithm is widely used in data compression. Because of popularity of this algorithm and its simplicity, it is very good choice for our purpose because its implementation is very efficient and fast. Therefore, involvment of NCD into spam detection will not reduce the speed of the decision engine.

5 Interconnection of the Bayesian spam filter and NCD

We may complete the Bayesian spam filter with further helping techniques that enable us to increase detection probability of unrequested email messages. One of the possibilities is comparison of the checked e-mail with a group of spam e-mails which were used for learning of individual probabilities of the Bayesian filter.

Additional check is applied in e-mails where spamcity is higher than 0.5. For each e-mail whose spamcity value is higher than 0.5, NCD-value of similarity, to the file which has approximately the same size after compression as the checked mail, is calculated.

Email with a similar size after compression is searched in the collection spam e-mails. We select the email according its size only. We take the email with the same or the size closest to the tested one. The size criterion helps in location of the possible similar one because messages with a similar size should be more similar to the checked email. Moreover, we may use a binary search algorithm for location of these emails. This step saves a lot of time and similarity values for files that have a higher or smaller size, will not be calculated. There is a small probability that files of a different size will have a similar content like the tested mail.

NCD results are values in the interval of 0 to 1, as was mentioned above. Whereas 0 stands for a maximum similarity of the tested files. In order to combine spamcity values from the Bayesian filter and NCD, it is necessary to modify NCD value by the help of the following simple formula:

$$p_{NCD} = 1 - (NCD)$$

By the help of this modification we can get probability with the same meaning like in the Bayesian filter. The more the files are similar, the closer is the value of p_{NCD} to 1. To get resulting spamcity value we have to combine probabilities from the Bayesian filter and NCD into one probability.

The combination proceeds by the help of the following Bayesian theorem:

$$P = \frac{p_B \times p_{NCD}}{p_B \times p_{NCD} + (1 - p_B) \times (1 - p_{NCD})}$$

where P is resulting probability, p_B is probability from Bayesian filter, and p_{NCD} is probability from NCD.

6 Results of unrequested e-mail detection

The Bayesian algorithm was implemented with many variations. These algorithms were tested in database of 270 045 e-mails, where 170 750 (63,23%) emails were marked as spam and 99 295 (36,77%) emails were marked as ham, i.e. legitimate mails. Test database comes from The Text REtrieval Conference (TREC) organized in years 2005, 2006 and 2007, co-sponsored by the National Institute

of Standards and Technology (NIST) and U.S. Department of Defense. In our experiments we tested three versions of the algorithm:

1. Classic Bayesian filter without any modification.
2. Classic Bayesian filter with NCD, all e-mails which reached spamcity higher than 0.5 were checked by means of NCD.
3. Classic Bayesian filter with NCD, all e-mails which had spamcity interval in the range of 0.5 to 0.75 were checked by means of NCD

The results are depicted in Table 1. The Classic Bayesian filter successfully indicated 162 894 (94.50%) spam e-mails, 7 856 (4.60%) spam e-mails were not recognized. The filter incorrectly marked as spam e-mails 9743 (9.81%) of 99 295 ham e-mails. Total number of incorrectly marked e-mails was 17 599 (6.52%). Average speed of e-mail checking process was 288 e-mails per second.

The Bayesian filter combined with NCD (with spamcity¿0.5) was able to successfully identify 169 886 (99.49%) spam e-mails. There were 864 (0,51%) of unidentified spam e-mails. Number of ham e-mails that were marked as spam was 12 575 (12.66%) emails. Total error rate of this algorithm modification was 13 439 (4.98%) of incorrectly marked e-mails. Average speed of e-mails checking process was 32.83 e-mails per second.

Last modification of the Bayesian filter in which additional testing by the help of NDC was limited on spamcity range from 0.5 to 0.75, successfully identified 169 886 (99.49%) of spam e-mails. Number of unidentified spam e-mails was 864 (0.51%). In examination of (legitimate) e-mails 12 852 (12.67%) of incorrectly marked e-mails was found out. Total error rate of the algorithm was 13 446 (4.98%) of incorrectly marked e-mails. Average speed of e-mails checking was reached in the level of 192 e-mails per second.

In comparison with the Classic Bayesian filter, versions completed with NCD show a higher efficiency in detection of spam e-mails. With the higher success rate of spam detection, the rate of incorrectly marked legitimate e-mails also increased. The Bayesian filter without NCD shows a very good filter permeability in the level of 288 e-mails per second. The version with NCD is slowed down by additional check of e-mails to 32.83 e-mails per second with NCD check, where spamcity is higher than 0.5 and 192 e-mails with restriction spamcity value in the interval of 0.5 to 0.75.

The difference in effectiveness and error rate of both Bayesian filter versions completed with NCD is not high. The only difference is in the speed of filtering process, where the version with limited usage of NCD was faster.

7 Conclusions

In this paper, a novel variant of Classic Bayesian filter with combination of Normaliced Compressed Distance was described. This combined filter was tested as filter for spam identification. In addition to Classical implementation of Bayesian filter, two versions of combination with NCD were implemented. The first version uses NCD for all emails which have spamcity higher than 0.5. The second version

Table 1. Results of the three tested algorithms

		Bayesian filter	Bayesian filter $NCD > 0.5$	Bayesian filter $0.5 > NCD < 0.75$
Spam	Success rate	95.40%	99.49%	99.49%
	Error rate	4.60%	0.51%	0.51%
Ham	Success rate	90.19%	87.64%	87.33%
	Error rate	9.81%	12.66%	12.67%
Total error rate		6.52%	4.98%	4.98%
Filter speed		288.36	32.93	192.59

uses NCD only, when the spamcity was in the interval from 0.5 to 0.75. Both filters have the same efficiency in detection of spam emails which was 99.49%. The second version is much faster than the first version and its speed is almost the same as speed of Classical Bayesian filter. Both new developed versions have worse efficiency in successfull marking of non spam emails. The overall efficiency of both new algorithm was better than the original filter.

References

1. T. Almeida, A. Yamakami, and J. Almeida. Evaluation of approaches for dimensionality reduction applied with naive bayes anti-spam filters. In *Machine Learning and Applications, 2009. ICMLA '09. International Conference on*, pages 517 –522, dec. 2009.
2. Y. Begriche and A. Serhrouchni. Bayesian statistical analysis for spams. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 989 –992, oct. 2010.
3. P. Deutsch. DEFLATE Compressed Data Format Specification version 1.3. RFC 1951 (Informational), May 1996.
4. B. C. Dhinakaran, D. Nagamalai, and J.-K. Lee. Bayesian approach based comment spam defending tool. In *Proceedings of the 3rd International Conference and Workshops on Advances in Information Security and Assurance*, ISA '09, pages 578–587, Berlin, Heidelberg, 2009. Springer-Verlag.
5. D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of IRE*, 40(9):1098–1101, 1952.
6. A. Khorsi. An overview of content-based spam filtering techniques. *Informatica (Slovenia)*, 31(3):269–277, 2007.
7. Y. Song, A. Kolcz, and C. L. Giles. Better naive bayes classification for high-precision spam detection. *Softw. Pract. Exper.*, 39:1003–1024, August 2009.
8. J. A. Storer and T. G. Szymanski. Data compression via textual substitution. *Journal of the ACM*, 26(10/82):928–951, 1982.
9. P. M. B. Vitányi. Universal similarity. *CoRR*, abs/cs/0504089:5, 2005.
10. P. Wolfe, C. Scott, and M. Erwin. *Anti-Spam Tool Kit.* McGraw-Hill Osborne Media, March 2004.
11. J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, IT-23(3):337–343, 1977.

eXolutio: Tool for XML Schema and Data Management*

Jakub Klímek, Jakub Malý, Irena Mlýnková, and Martin Nečaský

XML and Web Engineering Research Group, Department of Software Engineering
Faculty of Mathematics and Physics, Charles University in Prague
Malostranské náměstí 25, 118 00 Praha 1, The Czech Republic
{klimek, maly, mlynkova, necasky}@ksi.mff.cuni.cz

Abstract. Recently XML has achieved the leading role among languages for data representation and, thus, the amount of related technologies and applications exploiting them grows fast. However, only a small percentage of applications is static and remains unchanged since its first deployment. Most of the applications change with newly coming user requirements and changing environment. In this paper we describe a tool for evolution and change propagation of XML applications called *eXolutio*, which has been developed and improved in our research group during last few years. The text should help the reader to get acquainted with the tool and its theoretical background.

Keywords: XML schema, conceptual modeling, tool, evolution

1 Introduction

The eXtensible Markup Language (XML) is currently a de-facto standard for data representation and together with accompanying standards, such as XML Schema, XPath, XQuery, XSLT, etc., it becomes a powerful tool. Consequently, the amount and complexity of software systems that utilize XML and/or selected XML-based standards and technologies for information exchange and storage grows very fast. The systems represent information in a form of XML documents. One of the crucial parts of such systems are *XML formats* which describe the syntax of the XML documents in a form of *XML schemas* expressed in some XML schema language, e.g. DTD or XML Schema. Usually, a system does not use only a single XML format, but a set of different XML formats, each in a particular logical execution part. The XML formats represent particular views of the application domain of the software system. We can, therefore, speak about a *family of XML formats* utilized by a software system.

Having a system which exploits a family of XML formats, we face the problem of *XML format evolution* as a specific part of evolution of the software system as a whole. The XML formats may need to be evolved whenever user requirements or surrounding environment changes. Each such change may influence many different XML formats in the family. Without a proper technique, we have to identify the XML formats affected by the change manually and ensure that they are evolved coherently with each

* This work was supported in part by the Czech Science Foundation (GAČR), grant numbers P202/10/0573 and P202/11/P455 and in part by the grant SVV-2012-265312.

other. Such evolution brings a challenge for research, so that the user interaction and, hence, the expensive and error-prone work can be minimized. We cannot leave the user out completely, there still remain cases when user decision is unavoidable; however, automatic management of evolution enables to identify all the affected parts of the application and perform the user-selected changes correctly and efficiently and, possibly, exploit them in further automatic processing.

In our research group we have focused on the area of efficient and correct management of a family of XML formats for several recent years. Starting with a simple idea of propagation of changes among related XML formats, we have gradually extended our effort towards a robust framework and its implementation in a tool called *eXolutio*. It currently supports the original idea of designing XML formats using the principles of *Model Driven Architecture* (MDA) [18], their evolution, and integration of new XML formats into the framework.

Contributions The aim of this paper is to provide a covering overview of our research in the area of XML evolution, to describe architecture and implementation of the *eXolutio* tool, to present results of our experiments with the tool proving the concept and efficiency and a comparison of the tool with similar tools.

Outline The rest of the paper is structured as follows: In Section 2 we focus on the background theoretical aspects – our conceptual model for XML. In Section 3 we introduce *eXolutio*, our tool in which we implement our research results. In Section 4 we provide the proof of the concept using a set of experiments. In Section 5 we discuss the related work. Finally, in Section 6 we conclude.

2 Conceptual Model for XML

In this section, we introduce our conceptual model for XML and its inheritance extension. It has two levels, PIM and PSM, which is inspired by MDA.

A PIM schema is based on UML class diagrams and models real-world concepts and relationships among them. It contains three types of components: classes, attributes and associations. A sample PIM schema is depicted in Figure 1. Where association cardinality is not explicitly stated, default cardinality `1..1` applies. A part of the PIM are also integrity constraints which we are currently working on, but which are not in the scope of this paper.

Definition 1. *A* platform-independent (PIM) schema *is a triple* $\mathcal{S} = (\mathcal{S}_c, \mathcal{S}_a, \mathcal{S}_r)$ *of disjoint sets of* classes*,* attributes*, and* associations*, respectively.*

- Class $C \in \mathcal{S}_c$ *has a name assigned by function* name*. For inheritance purposes, function* isa *assigns a parent class to a child class (UML generalization) and must not form a cycle. Furthermore, functions* abstract *and* final *determine whether the class can have instances in data and whether this class can be inherited from, respectively.*
- Attribute $A \in \mathcal{S}_a$ *has a name, data type and cardinality assigned by functions* name*,* type*, and* card*, respectively. Moreover,* A *is associated with a class from* $\mathcal{S}_c$ *by function* class*.*

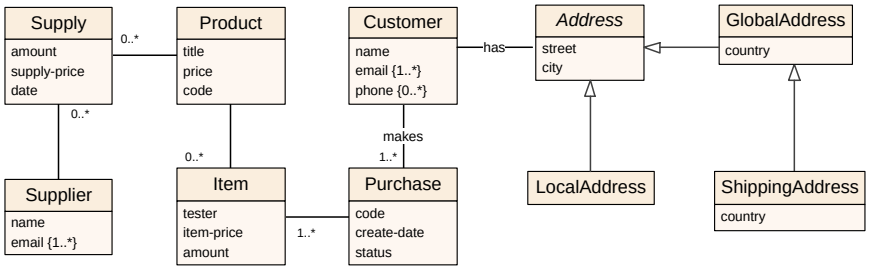


Fig. 1. Example of a PIM diagram

– Association $R \in \mathcal{S}_r$ *is a set* $R = \{E_1, E_2\}$*, where* E_1 *and* E_2 *are called* association ends *of* R*.* R *has a name assigned by function* name*. Both* E_1 *and* E_2 *have a cardinality assigned by function* card *and are associated with a class from* $\mathcal{S}_c$ *by function* participant*. We will call* participant(E_1) *and* participant(E_2) participants *of* R*.* name(R) *may be undefined, denoted by* name$(R) = \lambda$*.*

For a class $C \in \mathcal{S}_c$*, we will use* attributes(C) *to denote the set of all attributes of* C*, i.e.* attributes$(C) = \{A \in \mathcal{S}_a : \text{class}(A) = C\}$*. Similarly,* associations(C) *will denote the set of all associations with* C *as a participant, i.e.* associations$(C) = \{R \in \mathcal{S}_r : (\exists E \in R)(\text{participant}(E) = C)\}$*. For a given association* $R = (E_1, E_2)$*, we will use notation* (C_1, C_2) *as an equivalent of* $(participant(E_1), participant(E_2))$ *if there are no more associations connecting* C_1 *and* C_2*.*

The *platform-specific model (PSM)* specifies how a part of the reality is represented in a particular XML schema in a UML-style way. We introduce it formally in Definition 2. We view a PSM schema in two perspectives. From the *grammatical perspective*, it models XML elements and attributes. From the *conceptual perspective*, it delimits the represented part of the reality. Its advantage is that the designer works in a UML-style way which is more comfortable then editing the XML schema. Formally, there is a mapping from each PSM schema to the PIM schema.

Definition 2. *A* platform-specific (PSM) schema *is a tuple* $\mathcal{S}' = (\mathcal{S}'_c, \mathcal{S}'_a, \mathcal{S}'_r, \mathcal{S}'_m, \mathcal{C}'_{\mathcal{S}'})$ *of disjoint sets of* classes*,* attributes*,* associations*, and* content models*, respectively, and one specific class* $\mathcal{C}'_{\mathcal{S}'} \in \mathcal{S}'_c$ *called* schema class*.*

– Class $C' \in \mathcal{S}'_c$ *has a name assigned by function* name*. For inheritance purposes, function* isa *assigns a parent class to a child class and the relation must not form a cycle. Furthermore, functions* abstract *and* final *determine whether the class can have instances in data and whether this class can be inherited from, respectively.*
– Attribute $A' \in \mathcal{S}'_a$ *has a name, data type, cardinality and XML form (whether it models an XML attribute or an XML element) assigned by functions* name*,* type*,* card *and* xform*, respectively.* xform$(A') \in \{\text{e}, \text{a}\}$*. Moreover, it is associated with a class from* $\mathcal{S}'_c$ *by function* class *and has a position assigned by function* position *within the all attributes associated with class*(A')*.*

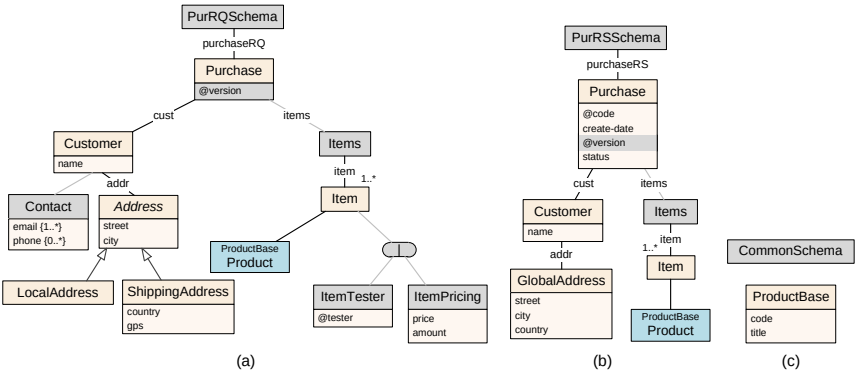


Fig. 2. Examples of PSM schemas

- Association $R' \in \mathcal{S}'_r$ *is a pair* $R' = (E'_1, E'_2)$*, where* E'_1 *and* E'_2 *are called* association ends *of* R'*. Both* E'_1 *and* E'_2 *have a cardinality assigned by function* card *and each is associated with a class from* $\mathcal{S}'_c$ *or content model from* $\mathcal{S}'_m$ *assigned by function* participant*, respectively. We will call* participant(E'_1) *and* participant(E'_2) parent *and* child *and will denote them by* parent(R') *and* child(R')*, respectively. Moreover,* R' *has a name assigned by function* name *and has a position assigned by function* position *within the all associations with the same* parent(R'). name(R') *may be undefined, denoted by* name$(R') = \lambda$.
- Content model $M' \in \mathcal{S}'_m$ *has a content model type assigned by function* cmtype. cmtype$(M') \in \{\texttt{sequence}, \texttt{choice}, \texttt{set}\}$.

The graph $(\mathcal{S}'_c \cup \mathcal{S}'_m, \mathcal{S}'_r)$ *must be a forest*[1] *of rooted trees with one of its trees rooted in* $\mathcal{C}'_{\mathcal{S}'}$*. For* $C' \in \mathcal{S}'_c$*, attributes* (C') *will denote the sequence of all attributes of* C' *ordered by* position*, i.e.* attributes $(C') = (A'_i \in \mathcal{S}'_a : \text{class}(A'_i) = C' \wedge i = \text{position}(A'_i))$*. Similarly,* content (C') *will denote the sequence of all associations with* C' *as a parent ordered by* position*, i.e.* content $(C') = (R'_i \in \mathcal{S}'_r : \text{parent}(R'_i) = C' \wedge i = \text{position}(R'_i))$*. We will call* content (C') content of C'.

A sample PSM schema is depicted in Figure 2. PSM uses similar constructs to PIM: classes, attributes and associations. The PSM-specific constructs have precisely defined semantics. A class models a complex content. The complex content is specified by the attributes of the class and associations in its content (their ordering is given by functions *attributes* and *content*). An attribute models an XML element declaration with a simple content or XML attribute declaration depending on its XML form (function *xform*). An association models an XML element declaration with a complex content if it has a name. Otherwise, it models only that the complex content modeled by its child is nested in the complex content modeled by its parent. Type of the modeled content (set,

[1] Note that since $\mathcal{S}'$ is a forest, we could model R' directly as a pair of connected components. However, we use association ends to unify the formalism of PSM with the formalism of PIM.

choice, sequence) can be specified by a special construct that can be, for example, seen in Figure 2(a) under the `Item` class.

2.1 Interpretation of PSM schema against PIM schema

A PSM schema represents a part of a PIM schema. A class, attribute or association in the PSM schema may be mapped to a class, attribute or association in the PIM schema. In other words, there is a mapping which specifies the semantics of classes, attributes and associations of the PSM schema in terms of the PIM schema. The mapping must meet certain conditions to ensure consistency between PIM schemas and the specified semantics of the PSM schema. The interpretation of a PSM schema against a PIM schema is what we call the mapping. It is the core feature of our conceptual model. It interconnects constructs on the platform-specific level with those on the platform-independent level and allows for interesting use cases for the conceptual model like XML schema evolution and integration [13, 14, 19, 20]. Its definition is, however, not trivial and is beyond the scope of this paper.

3 eXolutio architecture

The implementation of our research results is a tool called *eXolutio* [12]. There exists also an older version of our conceptual model and its implementation called *XCase* [11], which is the predecessor of *eXolutio*. For simplicity, we will stick to the current name. *eXolutio* allows the user to model a PIM schema and multiple PSM schemas with interpretations against the PIM schema. The user can then evolve the whole set of schemas coherently, because his operations are propagated to all affected places by a mechanism described in [19].

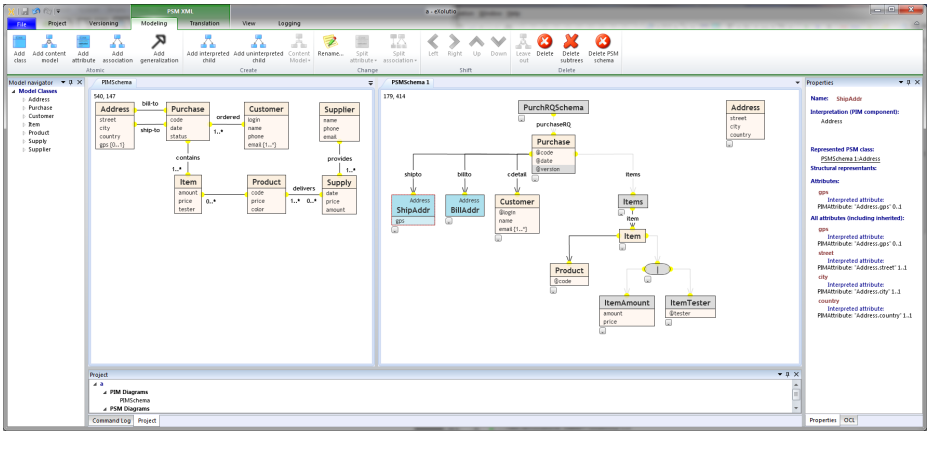
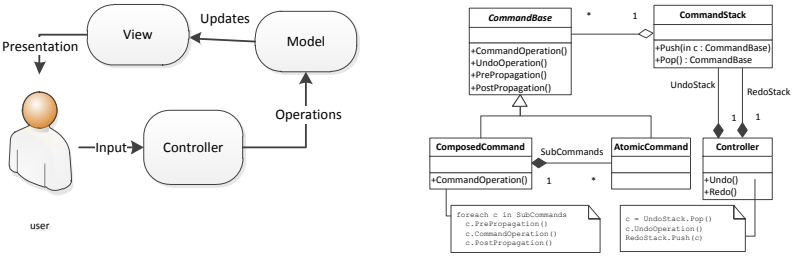


Fig. 3. *eXolutio* screenshot



(a) Overall architecture (b) Controller

Fig. 4. eXolutio – main MVC components

The architecture of *eXolutio* is based on the Model–View–Controller (MVC) design pattern (Figure 4(a)). This means that we hold all the project data in the model part, neither mixing it with operations, nor visualization. Whenever a user issues a command, it is handled by the controller part. The controller makes all the necessary changes in the model. The view part observes that the model has changed and updates the visualization. The connections between individual parts are loose enough so it is possible to, e.g., use multiple visualizations. In particular, we have a Windows Presentation Foundation [17] visualization (a desktop application) a Silverlight [16] visualization (a web application) and a no-visualization (a console application) versions of *eXolutio* which all share the same model and the same controller.

Model The model part of the tool based on our conceptual model [21] consists of classes for each modeled component, such as a class, an association or an attribute on each of the modeled levels (PIM and PSM), a class for PIM and PSM schemas and a class for the whole project. Besides the obvious properties of components like a name or a collection of attributes of a class, each component class implements methods for serialization and deserialization of the component to and from XML. Therefore, when we save and load a project, we simply call a serialization or a deserialization method on all found objects in a certain order. Finally, each schema contains lists of all the components of individual types in that schema, so we can easily go through, e.g., all associations in a certain schema. Since one of the main features of our tool is the visualization of connections between the two levels of abstraction, one of the most common queries is “Give me all PSM classes which have this PIM class as their interpretation”. We basically go through each PSM schema in the project and through each PSM class in that schema and check whether its interpretation is the given PIM class. In addition, the model contains methods for easy traversal of both the PIM and PSM schemas. An example can be a method for retrieval of all attributes of a PSM class including those inherited by the structural representative constructs. Another example can be a method that gets all uninterpreted descendants of an interpreted PSM class. When a certain method representing a query over the model is needed by the controller more than once, we make it a model method so that everyone can use it.

View The view component serves for two purposes: it visualizes the model for the user and provides user-friendly interface to run the controller commands. PIM schemas are depicted as UML diagrams and the layout of the diagram is left up to the user preference, for PSM schemas we use automatic hierarchical layouting to emphasize the fact that a PSM schema is a tree/forest. Besides the visualizations of the schemas, view component provides several windows and controls that help the user to navigate the modeled project, see the connections between individual concepts and follow the various links (e.g. find interpretation of an attribute or a class referred from a structural representant). The view component can be run either as a desktop application or inside a web browser using Silverlight plugin technology. This browser view can be used to accompany a documentation of published XML schema standards (e.g. [1]). An interactive visualization of a family of schemas joined by a common model can benefit greatly every system designer, who wants to adopt a third party standard and needs a clear overview of the whole problem domain and its individual schemas.

Controller The controller is the core of the tool. It contains all the operations and algorithms that make the tool unique. Also, it contains the usual command and undo/redo management. Whenever a user issues a command from view, it is handled by the controller. The controller (depicted in Figure 4(b)) gets all the necessary parameters from view such as what command is requested, the currently selected components, the new name for a component, etc. The controller creates the appropriate command, which in most cases will be one of our composite operations (described later in this section) and passes all the required parameters. The operation executes and updates the model accordingly. Then it places the command on the undo stack. The command itself contains all the information it needs to change the model back to the state it was in before the command was executed. In other words, we can simply call undo and the command knows what it needs to do and whether it is possible. This way, we can stack the executed commands and perform undo and redo operations as needed and as usual. In [19] we have described a theoretical background for atomic (simple, well defined) and composite (user-friendly) operations, which we will now describe from the implementation point of view. One of our goals was also to make the two levels of abstraction (PIM and PSM) work as independently of each other as possible while maintaining consistency when there are connections between the levels. Therefore, the operations need to work at their respective levels and be propagated only when there is an interpretation. Since we have a quite complex system of operations, we had to break it down into simpler parts. This means that among our atomic operations one can find for example an operation that creates an attribute. But it does not do anything else than that. Specifically, it does not give a name to the attribute, it does not set its datatype, etc. For that, we have other atomic operations. Having the atomic operations, we can compose more complex and user-friendly ones. A basic composite operation can be the already mentioned creation of an attribute, which this time is user friendly. It is composed of the creation of the attribute, renaming the attribute, setting its cardinality and its datatype. If it was a PSM attribute, the operation would also set its *xform* (Definition 2). So this is basically a predefined sequence of 4 or 5 operations, which is quite simple. Another simple example can be deletion of an attribute. This means setting its cardinality, name and datatype to default values and then deleting it. The reason for this is that when we undo

this operation, we want the name and the other values of the attribute to recover, so it is not correct to just delete the attribute. Let us have a look at a more advanced example.

So far we have described how to compose atomic and composite operations. However, these worked on their respective levels of abstraction. Now we have to make sure that when there is an interpretation of a PSM schema against a PIM schema, we keep the model in a consistent state and save the user's time by propagating the changes between the levels. This is achieved by the propagation. Before each atomic operation is executed, a method implementing its propagation to the other level is called. It determines whether there is an interpretation and therefore the need to propagate. If so, it creates a (possibly) composite operation on the other level of abstraction and integrates it to the currently running operation. Only when the propagation succeeds, the original atomic operation that caused it is executed. This way, the propagation actually becomes a part of the currently running operation. This is convenient because when it finishes, it can be undone and redone like any other operation.

4 Experiments

To provide the proof of the whole concept and show the advantages of our tool, we evaluate our approaches using experiments based on a real-world family of XML schemas. We experiment with the *National Register for Public Procurement System* (NRPP)[2]. It is a governmental information system where public authorities in the Czech Republic publish data about their public contracts. Authorities send contract information to the information system formatted in one of the 17 XML formats accepted by the NRPP. This includes, e.g. XML format for contract notifications, supplier selection notifications, etc. The information is then published by the system in the form of HTML pages. The goal of the experiment is to show how our approach would save time if the authors of the NRPP XML formats used *eXolutio* to design the XML formats and evolve them according to changing legislation instead of their manual editing and adaptation.

Currently, the NRPP only provides a textual documentation for the XML formats and a set of sample XML documents. Therefore, our first goal is to design a conceptual schema in a form of a PIM schema which models the domain of public contracts and design PSM schemas of the XML formats mapped to the PIM schema.

The PIM schema contains classes which model public contracts and their procurers and suppliers. There are also some additional concepts modeled – i.e. prices and contact information. A supplier is associated with a contract, a procurer is associated with a contract by a path of associations *has_contact* and *main*. Each contract has additional contact information – where documentation for the contract is provided and where bids to the contract are collected. Finally, there are four different prices – expected price, the best offered price, price agreed by a selected supplier and procurer, and a final real price known after finishing the contract. The PSM schema depicted in Figure 5 (a) models an XML format which a public authority uses to send a notification about a new public contract to NRPP. The PSM schema depicted in Figure 5 (b) models an XML format for notifications about the supplier selected for the contract.

[2] `http://www.isvz.cz` (in Czech only)

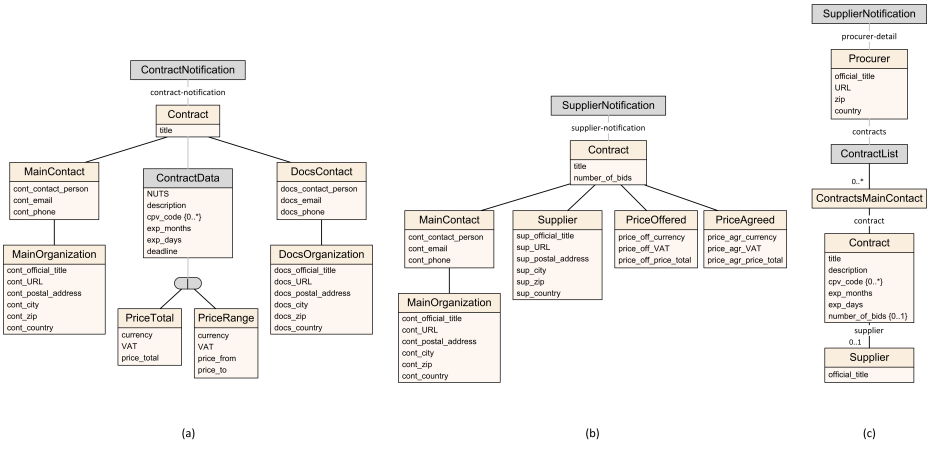


Fig. 5. PSM schemas modeling XML formats for (a) sending contract notifications to NRPP, (b) reporting on contract supplier selection to the NRPP, and (c) representing procurer detail

We can measure the amount of manual work required to design the PIM and PSM schemas in terms of numbers of executed atomic operations. The numbers of atomic operations executed to create the PIM and PSM schemas are depicted in Figure 6 (a). It shows that only creation and update operations were used. Here, manual creation of the schemas is necessary so there is no direct advantage in comparison to writing the XML schemas of the XML formats directly. However, *eXolutio* saves time because for each performed atomic operation it checks whether it does not break the consistency between the XML formats. When the designer codes the XML schemas of the XML formats directly no such control is performed automatically and (s)he must do it manually in each step during coding.

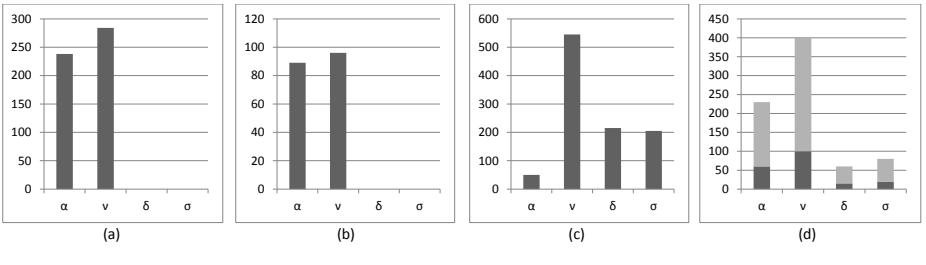


Fig. 6. Numbers of atomic operations performed manually by the designer (dark grey) and automatically by the propagation mechanism (light grey)

Having the PIM schema and a set of PSM schemas of the XML formats used by NRPP we set ourselves three goals. The first goal is to show how *eXolutio* facilitates creating new XML formats on the basis of an existing PIM schema. A PSM schema of

a new XML format for public procurer details is depicted in Figure 5 (c). The numbers of the atomic operations executed at this step are depicted in Figure 6 (b). Again, only the creation and update operations were performed. Even though the designer needs to design the PSM schemas for the new XML formats manually, the experiment shows that our approach saves him/her a great deal of work and prevents him/her from making unnecessary errors. This is because our technique enables us to create the PSM schemas on the basis of the PIM schema (which is faster than creating PSM schemas separately) and ensures that the designer creates the PSM schemas coherently with the PIM schema (as it preserves the consistency of the interpretation).

The second goal is to improve the quality of the NRPP XML formats, which is low. The designers of the XML formats did not follow basic XML design principles (e.g. exploiting the hierarchical nature of XML). For example, contact information is modeled by XML elements with names prefixed with `cont_`, `docs_`, etc. It would have been better to remove the prefixes and enclose the semantically related XML elements into separate XML elements (e.g. enclose contact XML elements to XML element `contact` structured to `main`, `doc`, etc. or enclose all information related to the supplier into XML element `supplier`). We have made these adaptations in the present XML formats. The numbers of the executed atomic operations are depicted in Figure 6 (c). In this step, synchronization and removal operations were also used, because some of the old parts of the PSM schemas were replaced by new ones. Again, the experiment demonstrated that our approach saves a lot of work as it preserves the consistency of PSM schemas against the PIM schema when changes are performed.

The third goal was to show how the set of schemas can be evolved coherently. We implemented various changes which resulted from new requirements on the NRPP functionality and from new legislation. In both cases, changes to the PIM schema needed to be done.

The new legislation required to report not only the number of bids received for each contract, but also particular bids including the bidding supplier and offered price.

Finally, there was a requirement to update the XML format for contract notifications (Figure 5 (a)) so that it is possible to give notification not only on the expected months and days in which the contract should be finished, but also on the exact date. This change was correctly propagated to the PIM schema, because it is a conceptual change. From here, it was propagated to the other PSM schemas.

The numbers of the atomic operations executed during the last two steps are depicted in Figure 6 (d). The darker part shows the numbers of manually executed operations. The lighter part shows the numbers of operations executed automatically by the propagation mechanism. α are additions, υ are changes, δ are deletions and σ are synchronizations - statements that two modeled sets of attributes or associations are semantically equivalent. The synchronizations are very useful in our change propagation mechanism, for details refer to [19, 20].

5 Related work

The current approaches towards evolution management can be classified according to distinct aspects [15, 8]. The changes and transformations can be expressed [22, 4] as

well as divided [6] variously too. However, to our knowledge there exists no general framework comparable to our proposal; particular cases and views of the problem have previously only been solved separately, superficially and mostly imprecisely without any theoretical or formal basis.

XML View We can divide the current approaches to XML schema evolution and change management into several groups. Approaches in the first group consider changes at the schema level and differ in the selected XML schema language, i.e. DTD [2, 7] or XML Schema [24, 5]. The changes are expressed variously and more or less formally. Approaches in the second and third group are similar, but they consider changes at an abstraction of logical level – either visualization [10] or a kind of UML diagram [9]. Both cases work at the PSM level, since they directly model XML schemas with their abstraction. No PIM schema is considered. All approaches consider only a single separate XML schema being evolved.

In all the papers cited the authors consider only a single XML schema. In [23] multiple *local* XML schemas are considered and mapped to a *global* object-oriented schema. Then, the authors discuss possible operations with a local schema and their propagation to the global schema. However, the global schema does not represent a common problem domain, but a common integrated schema; the changes are propagated just upwards and the operations are not defined rigorously. The need for well defined set of simple operations and their combination is clearly identified in Section 6 of a recent survey of schema matching and mapping [3].

6 Conclusion

In this paper, we introduced *eXolutio*, our tool for XML schema and data management. We surveyed related work and we showed the theoretical background behind our tool and evaluated it on real-world XML schemas.

References

1. *OpenTravel.org*.
2. L. Al-Jadir and F. El-Moukaddem. Once Upon a Time a DTD Evolved into Another DTD... In *Object-Oriented Information Systems*, pages 3–17, Berlin, Heidelberg, 2003. Springer.
3. Z. Bellahsene, A. Bonifati, and E. Rahm. *Schema Matching and Mapping*. Data-Centric Systems and Applications. Springer Berlin Heidelberg, 2011.
4. A. Boronat, J. A. Carsí, and I. Ramos. Algebraic Specification of a Model Transformation Engine. In *FASE '06: Proc. of the 9th Int. Conf. Fundamental Approaches to Software Engineering, Vienna, Austria*, volume 3922 of *LNCS*, pages 262–277. Springer, 2006.
5. F. Cavalieri. EXup: an Engine for the Evolution of XML Schemas and Associated Documents. In *EDBT '10: Proc. of the 2010 EDBT/ICDT Workshops*, pages 1–10, New York, NY, USA, 2010. ACM.
6. A. Cicchetti, D. D. Ruscio, and A. Pierantonio. Managing Dependent Changes in Coupled Evolution. In *Proc. of the 2nd Int. Conf. on Model Transformations, ICMT 2009, Zurich, Switzerland*, volume 5563 of *LNCS*, pages 35–51. Springer, 2009.
7. S. V. Coox. Axiomatization of the Evolution of XML Database Schema. *Program. Comput. Softw.*, 29(3):140–146, 2003.

8. K. Czarnecki and S. Helsen. Feature-Based Survey of Model Transformation Approaches. *IBM Syst. J.*, 45(3):621–645, 2006.
9. E. Domínguez, J. Lloret, A. L. Rubio, and M. A. Zapata. Evolving XML Schemas and Documents Using UML Class Diagrams. In *DEXA'05: Proc. of the 16th Int. Conf. on Database and Expert Systems Applications*, volume 3588 of *LNCS*, pages 343–352. Springer, 2005.
10. M. Klettke. Conceptual XML Schema Evolution – The CoDEX Approach for Design and Redesign. In M. Jarke, T. Seidl, C. Quix, D. Kensche, S. Conrad, E. Rahm, R. Klamma, H. Kosch, M. Granitzer, S. Apel, M. Rosenmüller, G. Saake, and O. Spinczyk, editors, *BTW'07*, pages 53–63, Aachen, Germany, March 2007.
11. J. Klímek, L. Kopenec, P. Loupal, and J. Malý. XCase - A Tool for Conceptual XML Data Modeling. In *Advances in Databases and Information Systems*, volume 5968/2010 of *Lecture Notes in Computer Science*, pages 96–103. Springer Berlin / Heidelberg, March 2010. `http://www.springerlink.com/content/v45198r1v783xu13`.
12. J. Klímek, J. Malý, and M. Nečaský. eXolutio – A Tool for XML Data Evolution, 2011. `http://exolutio.com`.
13. J. Klímek and M. Nečaský. Integration and Evolution of XML Data via Common Data Model. In *Proceedings of the 2010 EDBT/ICDT Workshops, Lausanne, Switzerland, March 22-26, 2010*, New York, NY, USA, 2010. ACM.
14. J. Klímek and M. Nečaský. Generating Lowering and Lifting Schema Mappings for Semantic Web Services. In *25th IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2010, Biopolis, Singapore, 22-25 March 2011*. IEEE Computer Society, 2011.
15. T. Mens and P. Van Gorp. A Taxonomy of Model Transformation. *Electron. Notes Theor. Comput. Sci.*, 152:125–142, 2006.
16. Microsoft. Silverlight. `http://www.microsoft.com/silverlight/`.
17. Microsoft. *Windows Presentation Foundation (WPF)*. December 2010. `http://msdn.microsoft.com/en-us/library/ms754130.aspx`.
18. J. Miller and J. Mukerji. *MDA Guide Version 1.0.1*. Object Management Group, 2003.
19. M. Nečaský, J. Klímek, J. Malý, and I. Mlýnková. Evolution and Change Management of XML-based Systems. *Journal of Systems and Software*, 85(3):683 – 707, 2012.
20. M. Nečaský, I. Mlýnková, and J. Klímek. Model-Driven Approach to XML Schema Evolution. In R. Meersman, T. S. Dillon, and P. Herrero, editors, *OTM Workshops*, volume 7046 of *Lecture Notes in Computer Science*, pages 514–523. Springer, 2011.
21. M. Nečaský, I. Mlýnková, J. Klímek, and J. Malý. When conceptual model meets grammar: A dual approach to XML data modeling. *Data & Knowledge Engineering*, 72(0):1 – 30, 2012.
22. OMG. *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification Version 1.0*. Object Modeling Group, April 2008. `http://www.omg.org/spec/QVT/1.0/`.
23. K. Passi, D. Morgan, and S. Madria. Maintaining Integrated XML Schema. In *IDEAS '09: Proc. of the 2009 Int. Database Engineering, Applications Symp.*, pages 267–274, New York, NY, USA, 2009. ACM.
24. M. Tan and A. Goh. Keeping Pace with Evolving XML-Based Specifications. In *EDBT'04 Workshops*, pages 280–288, Berlin, Heidelberg, 2005. Springer.

Unsupervised Algorithm for Post-Processing of Roughly Segmented Categorical Time Series

Tomáš Kocyan[1], Jan Martinovič[2], Štěpán Kuchař[2], and Jiří Dvorský[1]

[1] VŠB - Technical University of Ostrava,
Faculty of Electrical Engineering and Computer Science,
17. listopadu 15/2172, 708 33 Ostrava, Czech Republic
{tomas.kocyan,jiri.dvorsky}@vsb.cz
[2] VŠB - Technical University of Ostrava,
IT4Innovations,
17. listopadu 15/2172, 708 33 Ostrava, Czech Republic
{jan.martinovic,stepan.kuchar}@vsb.cz

Abstract. Many types of existing collections often contain repeating sequences which could be called as patterns. If these patterns are recognized they can be for instance used in data compression or for prediction. Extraction of these patterns from data collections with components generated in equidistant time and in finite number of levels is now a trivial task. The problem arises for data collections that are subject to different types of distortions in all axes. This paper discusses possibilities of using the Voting Experts algorithm enhanced by the Dynamic Time Warping (DTW) method. This algorithm is used for searching characteristic patterns in collections that are subject to the previously mentioned distortions. By using the Voting Experts high precision cuts (but with low level of recall) are first created in the collection. These cuts are then processed using the DTW method to increase resulting recall. This algorithm has better quality indicators than the original Voting Experts algorithm.

Keywords: Voting Experts, Dynamic Time Warping, Time Series

1 Introduction

This paper is focused on processing of semistructured data such as text. It is commonly know fact that the amount of this data rapidly grows so that there are two subproblems: how to store this kind of data and retrieve any piece of information from the data. To efficiently store the data it is common to use data compression. Data compression methods [11] can treat the data as sequence of bytes, or they can use additional knowledge about the data, such as knowledge of the language of the data. With this additional knowledge data compression methods can improve their results for example by moving from byte oriented alphabet to alphabet of words. The word alphabet is connection to the second subproblem – information retrieval. Information retrieval algorithms usually do

not process the input data as sequence of bytes, but they use even bigger pieces of the data, say words or generally some chunks of the data. This is the main motivation of the paper. How to split the input data into smaller chunks without a priori known structure of the input data? To do this, we use Voting Experts Algorithms in our paper. For test purposes, the Czech and English text was used as test bed for the segmentation algorithm, because the segmentation into words is known without any doubts for Czech or English text so that results of the Voting Experts Algorithm can be easily checked. During the future research, text inputs will be substituted by quantitative time series, such as river measured dicharge volume, and the typical patterns will be searched. These patterns will be further used in Case-Based Reasoning methodology as a input step for prediction.

The paper is organized as follows: in Sect. 2 a brief introduction of Voting Experts algorithm is given. Section 3 describes Dynamic Time Warping post-process of Voting Experts. Experimental results are provided in Sect. 4, and conclusion is given in Sect. 5.

2 Voting Experts

The *Voting Expert Algorithm* is a domain-independent unsupervised algorithm for segmenting categorical time series into meaningful episodes. It was first presented by Cohen and Adams in 2001 [3]. Since this introduction, the algorithm has been extended and improved in many ways, but the main idea is always the same. The basic Voting Experts algorithm is based on the simple hypothesis that natural breaks in a sequence are usually accompanied by two statistical indicators [4]: low internal entropy of episode and high boundary entropy between episodes.

The basic Voting Experts algorithm consists of following three main steps[3]:

- Build an nGram tree from the input, calculate statistics for each node of this tree (internal and boundary entropy) and standardize these values in nodes at the same depth.
- Pass a sliding window of length n over the input and let experts vote. Each of the experts has its own point of view on current context (current content of the sliding window) and votes for the best location for the split. The first expert votes for locations with the highest boundary entropy, the second expert votes for locations with a minimal sum of internal split entropy. By this way, the votes are counted for each location in the input.
- Look for local maximums which overcome selected threshold. These points are adepts for a split of sequence.

Tests showed that the algorithm is able to segment selected input into meaningful episodes successfully. It was tested in many domains of interest, such as looking for words in a text [3] or segmenting of speech record [8].

There are several ways how to improve the basic Voting Experts algorithm. Simply we can divide these improvements into the two main groups. On the

[3] For detailed explanation of each of mentioned steps see [4].

one hand, custom "expert" can be added to voting process (for example Markov Expert in [2]) and receive additional point of view on your input. On the other hand, there are methods based on repeated or hierarchical segmenting of the input [5, 9].

One of the simplest ways how to slightly improve performance of segmenting is two-way passing of the sliding window. It means using classic voting algorithm supplemented by segmenting of reversed input. This idea was outlined in [5] which showed the way to make high-precision cut points by selection of higher values of the threshold. Additionally, reversing the corpus and segmenting the reversed input with Voting Experts generates a different set of backward cut points. The subsequent intersection of sets of cut points offers high precision segmenting. However, on the other hand, this high precision causes loss of recall.

3 Voting Experts Post-Process

Proposed solution for Voting Experts improvement takes the task of using Dynamic Time Warping algorithm (introduced below) and high precision cuts as a starting point for looking for typical patterns located in the input. Basic idea is to refine the sparse set of high precision cuts into regular sequences as correctly as possible. The mentioned refinement will be done by several types of post-processing methods and the results will be compared.

Methods will differ, but they share a common principle (as shown in Fig. 1). If there are high precision cuts in the input (such as cuts A, B, C a D in Fig. 1) and if the shorter sequence (bounded by cuts C and D) is subsequence of the longer one (bounded by cuts A and B), we can deduce new boundaries E and F by projecting the boundaries of common subsequence to the longer sequence. In this very simplified example the sequences were composed by definite number of values and limited length, so the evaluation is quite straightforward.

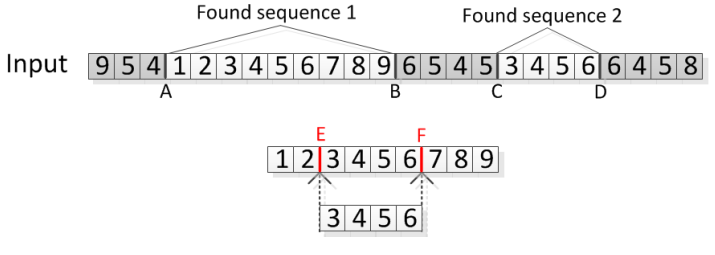


Fig. 1. Refinement of high precision cuts

3.1 Voting Experts DTW Post-Process

Dynamic time warping (DTW) is a technique to find an optimal alignment between two given sequences under certain restrictions [10]. The sequences are

warped in a nonlinear fashion to match each other. First DTW was used for comparison two different speech patterns in automatic speech recognition. In information retrieval it has been successfully applied to dealing with time deformations and different speeds associated with time-dependent data. For our purposes the DTW algorithm will be used as a tool for finding the longest common subsequence of two sequences.

In the case of application of previously mentioned process on distorted data, it is necessary to slightly modify it. Typical episodes of measurement of natural phenomena (such as precipitations, measured discharge volume etc.) are, unfortunately, subject to distortion in both time and value axes. For this reason, it is necessary to find out suitable mechanism that is able to deal with this deformation. The Dynamic Time Warping algorithm can be used for this purpose. The main idea of the Voting Experts DTW post-process is summarized into the following steps:

1. First of all, the high precision (but not complete) cuts are created by splitting the input with high level of threshold by the Two-Way Voting Experts method.
2. Let's suppose that there are m unique sequences which have been created according to cuts from step 1.
3. A $m \times m$ distance matrix is build.
4. For each pair in this matrix, where the length of sequence s_1 is bigger than length of sequence s_2:
   (a) The optimal mapping of shorter sequence s_2 to longer sequence s_1 is found by using DTW modified for searching subsequences.
   (b) If the mapping cost does not overcome selected threshold, the longest sequence s_1 stores the shorter sequence s_2 into its own list of similar sequences. By this way, every sequence gets its own list of the most similar shorter sequences.
   (c) Each of the shorter sequences points to positions in the longer sequence, where it should be splitted. Because there are usually more than one similar shorter sequences, it is pointed to several locations whereas many of these locations are duplicated. For this reason, the votes are collected into internal vote storage.
   (d) After these votes are collected, the local maximums are detected. These places are suggested as new cuts in original input.
5. The granted votes from step 4d are summed with votes of frequency and entropy experts in the input. Subsequently, the local maximums of votes are searched again. The cuts are made in locations where the number of granted votes is higher than the specified threshold.
6. Algorithm ends or it can continue with step 2 for further refinement.

3.2 Variations of the proposed algorithm

For our algorithm improvement, several variants of each particular step were proposed and then their influence were tested on final results. The most important variants of the algorithm will be introduced in the following paragraphs.

Method for finding similar sequences. The algorithm works only with sequences that do not overcome specified cost threshold of mapping (see step 4b in Sect. 3.1). For this reason, it is necessary to specify this threshold and limit for splitting of long sequences only to reasonable number of subsequences. Test showed that when the length of subsequences is not limited to reasonable length with respect to the length of longer sequence, the input is broken into large number of short sequences. On one hand, it increases the recall, but on the other hand, it rapidly decreases the precision.

To avoid unwanted splitting we used two parameters – divisor and offset. Length of the longer sequence is divided by the *divisor* and then the $offset$ is added. In order to avoid removing the shorter sequence from list of similar sequences, its length has to be longer than the resulting number. Formally we can specify the necessary condition as:

$$\|shorter\ sequence\| > \frac{\|longer\ sequence\|}{divisor} + offset \tag{1}$$

Method for voting. Method for granting votes based on mapping shorter sequence to longer one was introduced in Sect. 3.1. Additional three modifications were proposed for this experiment. These methods and their modifications will be further called as $VotingMethod_1$, $VotingMethod_2$, etc. Their principles are described as:

- $VotingMethod_1$: Post-process runs as well as described in Sect. 3.1.
- $VotingMethod_2$: Post-process is the same as in the $VotingMethod_1$, but it is supplemented by boundary condition. This condition limits voting on boundary positions in found sequences. This restriction should avoid situations, in which the high precision cuts have some errors and whole pattern is longer than the area bounded by cuts. Automatic cuts on sequence boundaries may distort further computation, so they are ommited.
- $VotingMethod_3$: First of all, the algorithm groups all similar found sequences and then find the longest common prefixes and suffixes in original input. For example, if the three sequences of value '3456' are found (see Fig. 2), the common prefix is '2' and the longest common suffix is '78'. The resulting common sequence is '2345678'. Subsequently, the same procedure as in the step 1 is performed.
- $VotingMethod_4$: The last method of post-process also looks for common prefix and suffix. But in this case the DTW is substituted by the longest common substring method.

Method for determining local threshold. All subsequences that satisfy the condition (1) subsequently vote for places in which they should split the longest (parent) sequence. In this way, several potentional places for cuts are created. Now it is necessary to decide in which locations will be the input really cut. In our experiments, the required threshold was computed in two ways:

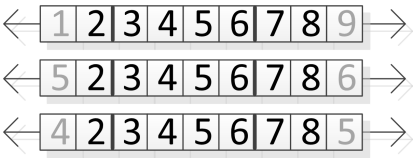


Fig. 2. Common prefix and suffix

1. Threshold was chosen as a multiple of maximum of votes (from 0.2 to 0.8).
2. Threshold was chosen as a multiple of nonzero votes average (from 1 to 2).

Method for determing incrementing value. Locations in which number of local votes overcomes the threshold are new proposals for cuts. This locations increment votes in original input. The question is how many votes should be these locations incremented. Three various types were suggested:

1. Incrementation of specified constant.
2. Incrementation of frequency with which the sequence appears in input and multiplied by specified constant.
3. Incrementation of multiple by which the threshold was overcome.

4 Experiments

Typical test of algorithm's verification of Voting Experts algorithm's performance is searching words in continuous text. In this text, spaces and punctuations (dots, dashes, new lines etc.) are removed and the goal of the algorithm is to put spaces back into correct places. Because the correct placement of spaces in the original text is known, it is very easy to quantify the algorithm's accuracy. To objectively compare the accuracy of suggested improvement with the basic method, experiments were performed on the same type of data, specifically on Jaroslav Hasek's novel named *Good Soldier Svejk*. To compare performance on different languages same text written in English and Czech language was chosen. English version was choosen as a default language, because original algorithm was tested on George Orwell's novel *1984*. Czech version was selected as a different type of language, which is characterized by large amount of possible word suffixes.

4.1 Evaluation

For the evaluation of proposed algorithm performace, precision and recall coefficients were defined. *precision coefficient* P and *recall coefficient* R rank among the most often used for the methods that are able to provide relevant documents in the information system. The precision coefficient is understood as the ratio of the amount of relevant documents returned to the entire number of returned

documents. Recall represents the ratio of the amount of relevant documents returned to a given query to the entire amount of documents relevant to this query. In our case, the precision coefficient will be understood as the ratio of the amount of correct spaces induced by algorithm to the entire number of induced spaces. Recall will represent the ratio of the amount of correct induced spaces to the entire amount of spaces in input. In order to simplify information about system effectivity, methods have been created to display precision and recall measured in a 1-dimensional space. One of these methods is Van Risjbergen's *F-measure*[10, 6]:

$$F_\beta = \frac{1+\beta^2}{\frac{\beta^2}{R}+\frac{1}{P}} = \frac{\left(1+\beta^2\right) R\ P}{\beta^2 P + R} \tag{2}$$

where β indicates the ratio of significance between precision and recall. For example, when β is an even 0.5, it means that the user is twice as interested in precision than in recall and when β is an 2, the users interest is vice versa. β was set to 1 in our experiment.

Each of the parameter combinations mentioned above were tested and the results were compared with basic algorithm. In all cases we observed percentage improvement of qualitative indicators.

4.2 Czech version results

The best configuration of input parameters for the Czech version is described in Table 1.

Table 1. The best parameter configuration for Czech version

Method for voting	Name	$VotingMethod_1$
Method for finding similar sequences	Threshold	0
	Delimiter	3
	Offset	0
Method for determining local threshold	Name	Nonzero votes avg.
	Multiplier	0.6
Method for determining incrementing value	Name	Increment of constant
	Value	2
Votes threshold	Value	3

Results of the best configuration applied on various input lengths are summarized in Table 2 and graphically in Fig. 3. It is evident that modified algorithm slightly decreases precision P but considerably increases recall R (up to 17.5%). Observed F-measure has an average improvement about +5.5%.

4.3 English version results

English input version reached the best results with configuration listed in Table 3 and its concrete qualitative indicators are specified in Table 4. In comparison

Table 2. Algorithm improvement for various input length of Czech input

Input length	F-measure	Precision	Recall
1000	7.02	-1.59	13.33
2000	8.40	-0.23	17.59
4000	3.81	-3.67	11.84
5000	5.04	-4.00	15.14
6000	5.88	-3.50	16.57
8000	5.65	-4.53	17.18
10000	4.25	-5.78	15.54
15000	4.23	-5.47	15.46

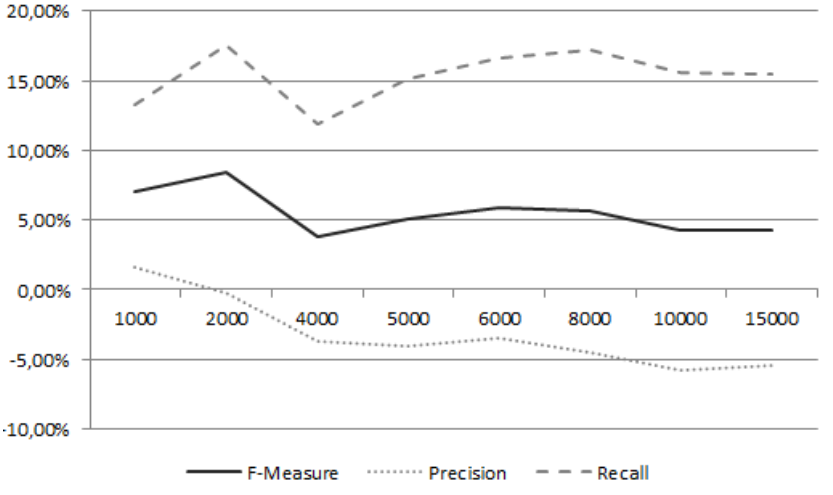


Fig. 3. Graphical representation of the best Czech configuration

with the Czech version, the English results are slightly worse. However, the average algorithm improvement of *F-measure* reaches 4.8%.

4.4 Overal methods evaluation

In previous sections, only the best configuration of parameters for each language were introduced. Now, overal success of each splitting method ($VotingMethod_1$, $VotingMethod_2$, etc.) regardless of remaining parameters and used language will be compared.

From each of the particular input size results, the top 300 configurations were selected and then the frequency, average frequency and relative frequency of concrete methods in this list were observed. These values are presented in Table 5.

The $VotingMethod_4$ reached the best result in average. This method appears in 42.38% of all selected configurations. It is probably caused by the fact that the

Table 3. The best parameter configuration for English version

Method for voting	Name	$VotingMethod_4$
Method for finding similar sequences	Threshold	0
	Delimiter	4
	Offset	1
Method for determining local threshold	Name	Nonzero votes avg.
	Multiplier	0.8
Method for determining incrementing value	Name	Increment of constant
	Value	2
Votes threshold	Value	3

Table 4. Algorithm improvement for various input length of English input

Input length	F-measure	Precision	Recall
1000	0.15	-4.40	5.30
2000	6.66	-2.05	15.33
4000	4.48	-3.61	12.73
5000	6.78	-2.23	16.15
6000	6.10	-3.02	15.81
8000	4.48	-3.97	13.93
10000	5.34	-3.91	15.34
15000	3.87	-5.45	14.21

longest common substring algorithm used in this method is designed specifically for strings. The second-best was the $VotingMethod_3$ with 29%. However, we hope that this method using DTW will overcome the $VotingMethod_4$ while testing on quantitative time series, because it is more robust against distortion in time axis.

Table 5. Algorithm improvement for various input length of English input

Used method	*Frequency*	*Average frequency*	*Relative frequency*
$VotingMethod_1$	387	48.38	16.13%
$VotingMethod_2$	300	37.50	12.50%
$VotingMethod_3$	696	87.00	29.00%
$VotingMethod_4$	1017	127.13	42.38%

4.5 Influence of algorithm configuration to results

During the tests we have not observed only the relative increment of F-measure, Presion and Recall, but we have also evaluated the influence of values of indu-

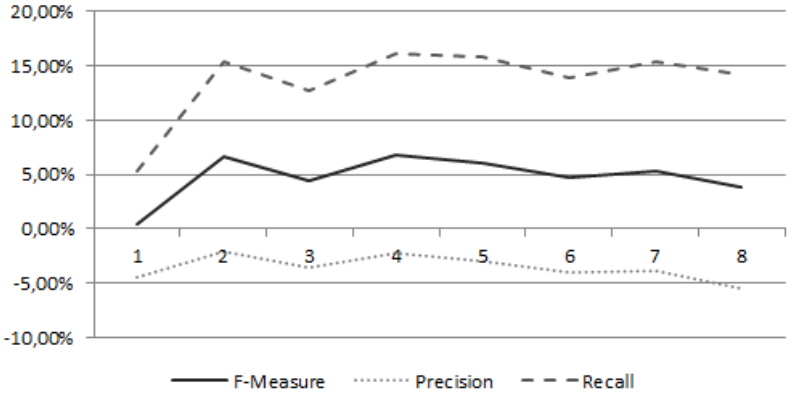


Fig. 4. Graphical representation of the best English configuration

vidual parameters to the results. This information will be further important for design algorithm parameters run on quantitative time series.

Test ran on grouped configurations by all parameters without the monitored one and the dispersion of resulting F-measure (caused by the ommited parameter) was observed. The dispersions of particular parameters are displayed in Fig. 5.

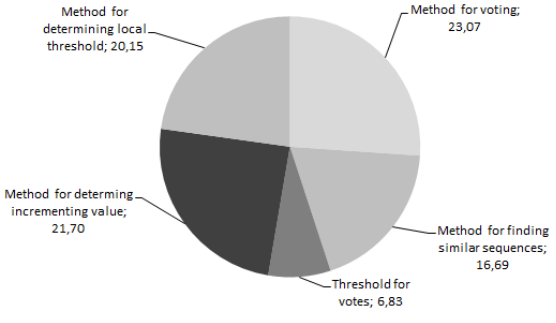


Fig. 5. Dispersions of particular parameters

It is evident that the proposed method is sensitive to the choise of the voting method, method for determining local threshold and incremental method. The algorithm is little less sensitive to the method for searching similar sequences and it is insensitive to the value of the threshold.

4.6 Reusing on other data collections

Once the best configuration for specific colection type has been found, it can be further reused on this type of input data (text data in our experiments). We verified this idea with the best English configuration on the another two English texts – Gorge Orwell's novel *1984* and Mark Twain's *Adventures of Huckleberry Finn*. In Fig. 6 you can see the results. Both inputs reached better results than the basic algorithm.

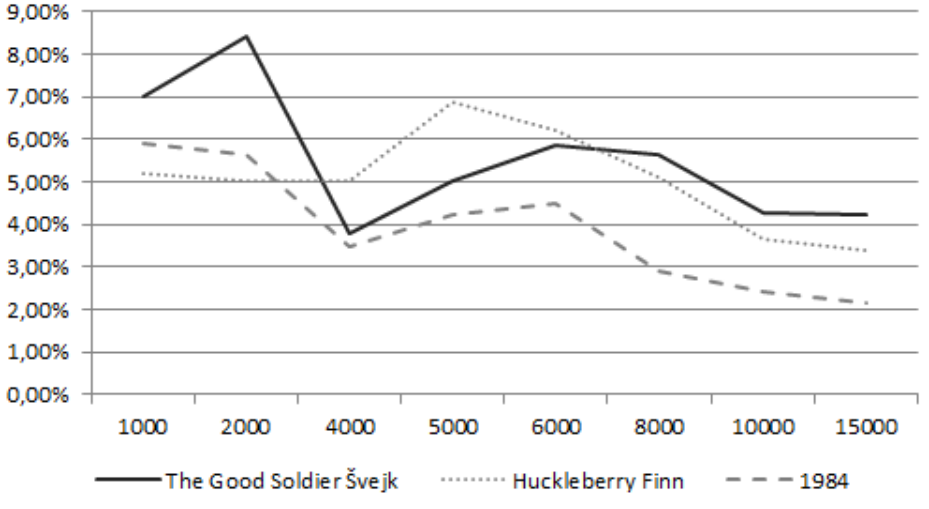


Fig. 6. Reusing the best configuration to other data collections

5 Conclusion and Future Work

Practical applications of the Voting Experts algorithm showed that it can be used in many domains in which we want to look for some meaningful episodes. Proposed solution overcomes qualitative indicators of original Voting Experts algorithm and offers different point of view to the solution of searching meaningful episodes. Experiments showed that if the proposed algorithm modification is trained to a specific type of data (such as English text, Czech text etc.) it can be further used on various inputs and it should always overcome the basic version of Voting Experts.

Our future work will be focused on searching typical patterns in measured and distorted time series, specifically on searching typical patterns in measured river discharge volumes. This found patterns will be further used for prediction using the Case-Based Reasoning method. This method requires suitable mechanism that is able to extract the most similar patterns from the input.

Acknowledgement

This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

References

1. G. Altmann. Prolegomena to Menzerath's law. Glottometrika 2 (1980). P. 1-10.
2. J. Cheng, and M. Mitzenmacher. Markov Experts. Proceedings of the Data Compression Conference (DCC). 2005.
3. P. R. Cohen, and N. Adams. An Algorithm for Segmenting Categorical Time Series into Meaningful Episodes. Proceedings of the Fourth Symposium on Intelligent Data Analysis, Lecture Notes in Computer Science. 2001.
4. P. R. Cohen, N. Adams, and B. Heeringa. Voting Experts: An Unsupervised Algorithm for Segmenting Sequences. In Journal of Intelligent Data Analysis. 2007.
5. D. Hewlett, and P. Cohen. Bootstrap Voting Experts. Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI). 2009.
6. T. Ishioka. Evaluation of criteria on information retrieval. *Systems and Computers in Japan*, 35(6):42–49, 2004.
7. T. Kocyan, J. Martinovic, J. Dvorsky, V. Snasel. *Czech Text Segmentation Using Voting Experts and Its Comparison with Menzerath-Altmann law.* Computer Information Systems Analysis and Technologies, 2011, 978-3-642-27245-5, pages 152-160.
8. M. Miller, P. Wong, and A. Stoytchev. Unsupervised Segmentation of Audio Speech Using the Voting Experts Algorithm. Proceedings of the Second Conference on Artificial General Intelligence (AGI). 2009.
9. M. Miller, and A. Stoytchev. Hierarchical Voting Experts: An Unsupervised Algorithm for Hierarchical Sequence Segmentation. Proceedings of the 7th IEEE International Conference on Development and Learning (ICDL). (Best Paper Award, ICDL 2008)
10. M. Muller. *Dynamic Time Warping.* Information Retrieval for Music and Motion, Springer, ISBN 978-3-540-74047-6, 69–84, 2007.
11. D. Salomon. Data Compression: The Complete Reference. Springer-Verlag. 2007.
12. B. E. Swartz, and E. S. Goldensohn. Electroencephalography and Clinical Neurophysiology, in Electroencephalography and Clinical Neurophysiology, 106(2), 173 - 176, 1998.
13. C. J. Van Rijsbergen. *Information Retrieval, Second Edition.* Department of Computer Science, University of Glasgow, 1979.

Using OCL in Model Validation According to Stereotypes*

Zdenek Rybola[1] and Karel Richta[2,3]

[1] Faculty of Information Technology, Czech Technical University in Prague
rybolzde@fit.cvut.cz
[2] Faculty of Mathematics and Physics, Charles University in Prague
richta@ksi.mff.cuni.cz
[3] Faculty of Electrical Engineering, Czech Technical University in Prague
richta@fel.cvut.cz

Abstract. Model-Driven Development approach became popular in past years. Domain-specific profiles are defined for various domains and tools are used to transform models using these profiles to source code artifacts. However, rules need to be defined for the profile elements usage so the transformation can be effective and reliable.
This paper deals with an approach of expressing these rules using special type of metamodel with UML class diagrams with the stereotypes defined in the profile – we call them constraint diagrams. Each class in this metamodel represent all classes in the model with the same stereotype. Using stereotyped associations, we can link classes with different stereotypes and restrict the usage of such stereotype only to relations between specific stereotyped classes in the model. OCL constraints can be generated from the constraint diagram to enable validation of the model according to the rules in the metamodel. This paper deals with the description of the constraint diagram creation and OCL constraints generation.

Keywords: UML, OCL, constraint, stereotype, validation

1 Introduction

Model Driven Development [10] is a modern and popular software development approach. It is based on creation of model of different abstraction levels and transformations between those models. It also includes forward and reverse engineering methods. Forward engineering becomes especially popular for generation of source code from models.

Models of software systems are usually created using UML and transformed to source code using a tool that generates all required artifacts from the model. For instance, many various artifacts for J2EE application such as Entities, Session beans, Message-driven beans and many others can be generated from an

* This research was partially supported by Grant Agency of CTU No. SGS12/093/OHK3/1T/18 and Czech Grant Agency (GAČR) No. GA201/09/0990

UML class model. Because of various domains of software systems, domain-specific UML profiles are usually defined. UML profile [9] is a meta-model that allows the definition of stereotypes and tagged values for model elements. If a profile is defined and used, model transformation can be adapted according to the specified stereotypes and tagged values. However, special rules usually come with the profile to restrict usage of various profile artifacts that needs to be satisfied by any model based on the profile. To make those adapted transformations effective, model validation against the defined rules is required.

In our current research, we deal with an approach to generate OCL constraints for domain-specific profiles using an instance of the meta-model defined in the profile. Creating a constraint diagrams using artifacts of the profile allows us to graphically express domain-specific rules. These diagrams can be easily transformed to OCL constraints that can be used for model validation using various CASE and OCL tools. Whole process is shown in Fig. 1.

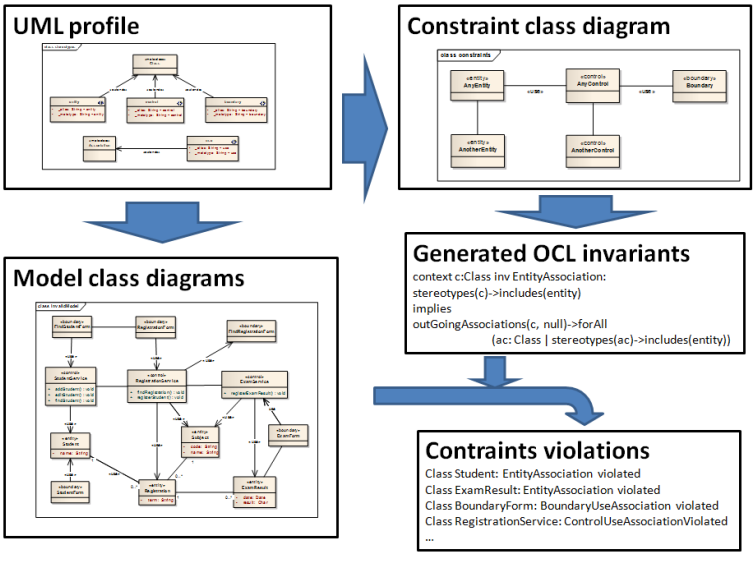


Fig. 1. Graphical description of the model validation process - model diagrams and a constraint diagram are created using the profile; the constraint diagram is transformed to OCL constraints; model diagrams are validated using OCL constraints resulting in a set of constraints violation messages

In this particular paper, we deal with generation of OCL constraints for rules restricting connections between various stereotypes. An example is based on fundamental analysis class model specification [2]. In this specification, only classes with Entity, Control and Boundary stereotypes can be used with some

restrictions for connections between various stereotypes. However, our approach can be used for any profile defined for any domain.

The paper is structured as follows: Section 2 presents related work and their difference from our approach. In section 3, we show an example of model with a profile and definition of rules for stereotype usage. In section 4, we describe the constraint diagram for expressing the rules. Generation of OCL invariants is explained in section 5. Conclusions and further work plans are given in section 6.

2 Related work

There have been done some research on model checking and model validation using OCL. Richters and Gogolla [11] presented an approach of animating model snapshots and validating it against OCL constraints. The authors use USE tool [12] for model animation and validation. The authors also introduced a method of automatic model snapshot generation in USE tool [5]. However, the authors only generate snapshot of the model and the constraints must be defined as required directly in OCL.

Some research was made on model validation against fundamental properties of models defined in UML. Chae et al. [2] focuses on analysis class model used in many standard object-oriented processes where three basic stereotypes are defined for analysis classes - boundary, control and entity. The authors define a set of constraints in OCL that any analysis class model should satisfy including constraints for associations between classes of particular stereotypes. The authors also demonstrate a case study of validation of analysis models against defined OCL constraints using OCLE tool [3]. However, the authors only define particular constraints for these three basic stereotypes of analysis classes.

In [6], Guizzardi defines ontological extension of UML class diagram with ontology stereotypes and constraints called OntoUML. He presents a fine-grained approach for domain modelling using stereotypes to distinguish between various types of domain artefacts and relationships. In [1], Benevides and Guizzardi present a graphical editor for OntoUML and validation of OntoUML model using OCL constraints. These constraints are based on stereotypes of model elements and defined according to the specification of OntoUML.

In comparison to the approaches mentioned above, in our approach we do not define any particular constraint for any particular domain. Instead, we propose a general approach to create a constraint diagram using domain-specific and user-defined profile to model domain-specific business rules. This diagram can be used to generate OCL constraints for validation of any model based on the user-defined profile. We deal with generating OCL constraints for UML class diagrams in this particular paper.

There is also a lot of tools that support model transformation and validation against OCL constraints. Dirigent [7] is an open-source MDA tool for generation of source code from model. The tool reads model stored in XMI format and generate source code files according to specified patterns in Apache Velocity.

These patterns can be based on model element stereotype and generate a set of source files. If extended appropriately, Dirigent would be able to parse the proposed constraint diagram and generate OCL constraints for validation of a model based on the same profile.

Dirigent could also be extended to validate the model using the generated OCL constraints itself. However, there are also many other tools that support model validation using OCL constraints. OCLE tool [3] can be used as mentioned above. DresdenOCL [4] is a toolkit for creating and validating OCL constraints for specified model. It also supports model validation and model transformation together with the constraints to SQL and Java with AspectJ.

3 Defining rules

To make transformations of domain-specific model in UML to source code files effective, domain-specific UML profile should be defined. This profile includes mostly stereotypes of classes and associations for class diagrams and can define tagged values of such stereotypes as well. In the model, various defined stereotypes are used to model various types of artifacts using classes and their associations. During transformation to source files, various stereotyped classes can be transformed to various source code artifacts such as J2EE entities, session beans or servlets.

Lets imagine an example shown in Fig. 2 where a class model of a part of a university information system is shown. The model is created using analysis model profile with stereotypes entity, control and boundary to distinguish between three kinds of analysis classes. We use only a part of the analysis model profile required for our model. The original analysis model profile with full set of rules is defined in [2].

For each of these stereotypes, various source code artifacts can be generated during transformation. For instance of a J2EE application:

- an entity class, a data access object class and a SQL creation script for each entity-stereotyped class,
- a session bean local interface and implementation for each control-stereotyped class,
- and a servlet for each boundary-stereotyped class can be generated.

With generation of source code artifacts for each model element, references between those generated artifacts can be also generated. For instance:

- relationships between entities for each association between entity-stereotyped classes,
- reference between session beans,
- and reference to session bean from servlets can be generated.

However, to generate such artifacts and connections effectively and correctly, the model must satisfy specific rules. These rules must be defined and obeyed during

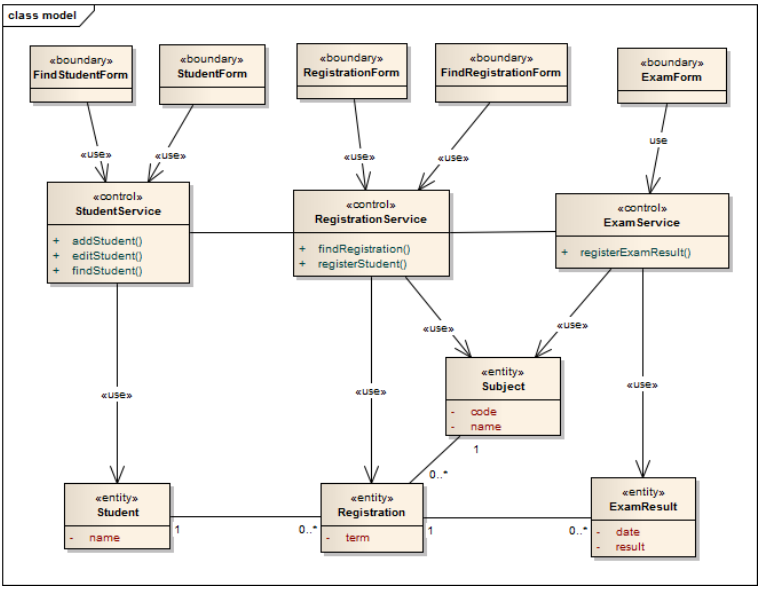


Fig. 2. Model of a part of a university information system with analysis model stereotypes

modeling. The rules for analysis class model are defined in [2]. However, for our example with simplified profile, only some of these rules are important and they can be expressed as follows:

1. Each entity class can be related by stereotype-free association only to another entity class.
2. Each control class can be related by use-stereotyped association to entity class.
3. Each control class can be related by stereotype-free association only to another control class.
4. Each boundary class can be related by use-stereotyped association to control class.

Notice, that the definition is always defined in the direction from the source class in the relation. This is to eliminate redundancy and to make each relation checked only once. If these rules are obeyed in the model, the transformation to the source code artifacts will be correct and no error shall appear.

4 Modeling constraints

Unfortunately, developers and analysts make mistakes so we need to validate and check the model before transformation and source code artifacts generation.

Therefore, the rules must be defined in some formal way so a tool such as OCLE [3] or DresdenOCL [4] can be used to validate the model against the defined rules for stereotype usage and relations restrictions.

Object Constraint Language (OCL) [8] is the part of UML used to define model constraints in a form of invariants that each instance of the element in the context must satisfy. However, definition of the required OCL invariants can be a bit tricky for unfamiliar developers. Not many developers or even analysts posses the knowledge of OCL. Therefore, we propose an approach of creating a special class diagram – we call it *constraint diagram* – to model the rules using stereotypes used in the analysis model and to generate the OCL constraints for model validation.

Definition 1. *A* constraint diagram *is a UML class diagram with classes and relations using domain-specific stereotypes used to define domain-specific rules for relations allowed between various used stereotypes and to generate OCL invariants to validate domain-specific UML class diagrams.*

In the constraint diagram, we have to model all relations allowed in the model between classes with particular stereotypes. Therefore, we add classes with defined stereotypes as required and add relations with defined stereotypes and direction. Names of the classes and multiplicity values are not important in this diagram, they just represent any instance of a particular stereotype and any relation of a particular type and stereotype in the model. Each relation in the constraint diagram stands for a rule that restricts relations between particular stereotypes. Each of them can be transformed to OCL constraint as described later in section 5.

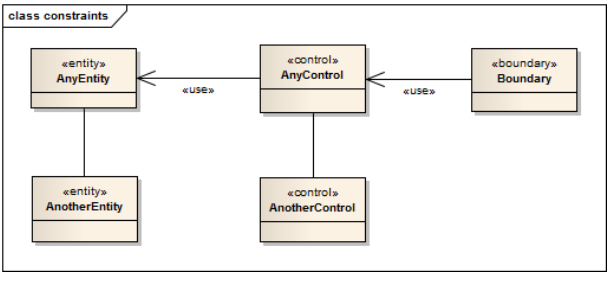


Fig. 3. Constraint diagram for the used part of analysis model profile

In Fig. 3, constraint diagram is shown for the rules defined in section 3. We have created two classes with stereotype *entity*. These represent any class with that stereotype used in the model. We add an association between *AnyEntity* and *AnotherEntity* – in this direction – with no stereotype to express that any class with stereotype *entity* can be related by stereotype-free association to any other class with stereotype *entity*. This relation stands for the rule 1.

To define rule 2 in the constraint diagram, we add a class with stereotype *control*. Then we connect it with a class with stereotype *entity* by a use-stereotyped association to express that each control class can be related to any entity by a use-stereotyped association. In our case on Fig. 3 we connected *AnyControl* to *AnyEntity* class.

Similar to the rule 1, to define rule 3 in the constraint diagram, we add another class with stereotype *control* and connect it from the class *AnyControl* to express the allowed stereotype-free association between control-stereotyped classes. Finally, similar to rule *rule:control-entity*, a class with stereotype *boundary* is added to the diagram with a connection to class *AnyControl* by a use-stereotyped association to define the rule *rule:boundary-control*.

Notice that for correct rules definition and constraint generation, some constraints for the constraint diagram must be obeyed as well. For instance, only one relation of each partial type and stereotype can be connected from each particular-stereotyped classes, otherwise there would be redundant rules defined in the diagram for the same relation and the same source artifact. Also, no directed relations can be used in the diagram. As mentioned in section 3, only outgoing relations stand for a rule to eliminate redundancies. If a relation of both directions between two various-stereotyped classes can be used in the model, two distinct relations must be created in the constraint diagram, each with the different direction – source and target classes – of the relation.

Although we have to check the constraint diagram for these rules satisfaction, we do this only for this single constraint diagram, while all the model diagrams based on the same profile with the rules defined in the constraint diagram can be validated and checked automatically by the generated constraints.

5 Generating OCL constraints

A set of OCL constraints can be generated from a constraint diagram using a tool such as Dirigent [7] to parse the model diagrams. The tool traverses the model and for each relation in the diagram, an OCL invariant is generated. To explain the constraint structure, let us define several terms used in the constraint first.

Definition 2. Source Class Stereotype *is the stereotype of the source class of the relation in the constraint diagram just being processed by the generation tool.* Target Class Stereotype *is the stereotype of the target class of that relation.* Relation Stereotype *is the stereotype of that relation.*

The structure of such constraint is shown in Fig. 4. The invariant is defined in the context of Class with its name generated from the *Source Class Stereotype*, the *RelationStereotype* and the type of the relation. Then, two functions are defined – *sts()* return a set of names of stereotypes of the classifier parameter – i.e. a class or an association –; *associationSet()* return a set of associations – that have no stereotype or include the given stereotype according to the stereotype given as parameter – from the given classifier. Then, the main constraint body is

defined – if the class in context includes the *Source Class Stereotype* then target classes of all associations with the same stereotype as the *RelationStereotype* must include the *Target Class Stereotype*.

```
context c:Class inv <SourceClassStereotype><RelationStereotype>Association:
let sts(c:Classifier) : Set(String) =
  c.stereotype->collect(name)->asSet()
let associationSet(c:Classifier, s:String) : Set(Association) =
  c.association.association.connection->
  select(isNavigable = true)->
  select(a:Association | (s = "" and sts(a)->empty())
    or sts(a)->includes(s))->asSet()
sts(self)->includes(<SourceClassStereotype>) implies
  associationSet(self, <RelationStereotype | "">)->collect(participant)->
  select(ac:Classifier | ac <> self)->
  forAll(ac:Classifier | sts(ac)->includes(<TargetClassStereotype>))
```

Fig. 4. General form of OCL invariant generated from a constraint class diagram with wildcards for *Source Class Stereotype*, *Target Class Stereotype* and *RelationStereotype*

When parsing the constraint diagram shown in Fig. 3, the tool will find four associations to generate OCL invariants – stereotype-free associations between two entities and two control classes, respectively, a use-stereotyped association from *AnyControl* class to *AnyEntity* class and a use-stereotype association from *AnyBoundary* class to *AnyControl* class. Therefore, four OCL invariants are generated as shown in Fig. 5. All generated invariants use the same functions *sts()* and *associationSet* as defined in Fig. 4, however, they are not displayed in the figure because of limited space in the paper.

Now, having these OCL invariants describing rules for the use of the profile stereotypes, we can use any tool supporting model validation using OCL constraints such as OCLE or DresdenOCL Toolkit. Using such a tool, we can validate any part of a model of a system using the same profile against the defined rules. The tool will find any classes that does not satisfy any of our invariants pointing the class is connected to some other class using wrong-stereotyped association or the target class have wrong stereotype attached.

6 Conclusions

Model-Driven Development approaches for software development became popular in last years. Many software development processes use a tool to transform models and to generate source code artifacts. Many domain-specific UML profiles are also created for various domains or software projects and generation tools are adapted to utilize these profiles during transformation and generation. However, this approach brings the need of domain rules definition of how the profile should be used.

```
-- rule 1
context c:Class inv EntityAssociation:
sts(self)->includes("entity") implies
  associationSet(self, "")->collect(participant)->
  select(ac:Classifier | ac <> self)->
  forAll(ac:Classifier | sts(ac)->includes("entity"))

-- rule 2
context c:Class inv ControlUseAssociation:
sts(self)->includes("control") implies
  associationSet(self, "use")->collect(participant)->
  select(ac:Classifier | ac <> self)->
  forAll(ac:Classifier | sts(ac)->includes("entity"))

-- rule 3
context c:Class inv ControlAssociation:
sts(self)->includes("control") implies
  associationSet(self, "")->collect(participant)->
  select(ac:Classifier | ac <> self)->
  forAll(ac:Classifier | sts(ac)->includes("control"))

-- rule 4
context c:Class inv EntityAssociation:
sts(self)->includes("boundary") implies
  associationSet(self, "use")->collect(participant)->
  select(ac:Classifier | ac <> self)->
  forAll(ac:Classifier | sts(ac)->includes("control"))
```

Fig. 5. OCL invariants for the rules to check and validate the model

In this paper, we presented an approach of modeling these domain rules for the use of user-defined stereotypes and relations between each other using UML class diagram. We presented a method how such diagram can be created for a set of example rules. We also presented a technique how to generate OCL invariants from the diagram to be used for model validation. Whole process was illustrated on an example using analysis model profile with standard class stereotypes *entity*, *control* and *boundary*.

In our further research, we would like to extend our approach to enable to model directed association constraints to validate usage of directed associations in the model, other types of relations such as generalization or dependency. Some research can also be done to define multiplicity constraints for the number of related classes using each particular stereotyped relation. Finally, extending the approach by other types of model artifacts such as actors, components or use cases can be researched.

References

1. Benevides, A.B.: A Model-based Graphical Editor for Supporting the Creation, Verification and Validation of OntoUML Conceptual Models. Ph.D. thesis, Federal University of Espírito Santo (UFES), Vitória, E.S., Brazil (Feb 2010)
2. Chae, H.S., Yeom, K., Kim, T.Y.: Specifying and validating structural constraints of analysis class models using OCL. Information and Software Technology 50(5), 436–448 (Apr 2008), http://www.sciencedirect.com/science/article/B6V0B-4NVH7T5-1/2/02217cd36c68c34c93fc63253c28bf62
3. Chiorean: OCLE 2.0 - object constraint language environment. http://lci.cs.ubbcluj.ro/ocle/index.htm (Feb 2012), http://lci.cs.ubbcluj.ro/ocle/index.htm
4. Demuth, B.: DresdenOCL. http://www.reuseware.org/index.php/DresdenOCL (Jan 2011)
5. Gogolla, M., Bohling, J., Richters, M.: Validating UML and OCL models in USE by automatic snapshot generation. Software and Systems Modeling 4(4), 386–398 (2005)
6. Guizzardi, G.: Ontological foundations for structural conceptual models. PhD thesis, University of Twente, Enschede, The Netherlands (Oct 2005), http://eprints.eemcs.utwente.nl/7146/
7. Hubl, K.: Dirigent (Jan 2012), code.google.com/p/dirigent/
8. OMG: Object constraint language, version 1.3. http://www.omg.org/spec/OCL/2.2/PDF (Feb 2010)
9. OMG: UML 2.4. http://www.omg.org/spec/UML/2.4/ (Aug 2011), http://www.omg.org/spec/UML/2.4/
10. OMG, Miller, J., Mukerji, J.: MDA guide version 1.0.1. http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf (Jun 2003)
11. Richters, M., Gogolla, M.: Validating UML models and OCL constraints. UML 2000 - THE UNIFIED MODELING LANGUAGE, PROCEEDINGS - ADVANCING THE 1939, 265–277 (2000)
12. Richters, M., Buettner, F., Gutsche, F., Kuhlmann, M.: USE - a UML-based specification environment. http://www.db.informatik.uni-bremen.de/projects/USE/ (Jan 2011)

Models for Efficient Semantic Data Storage Demonstrated on Concrete Example of DBpedia

Ivo Lašek and Peter Vojtáš

[1] Czech Technical University in Prague, Faculty of Information Technology, Prague, Czech Republic, lasekivo@fit.cvut.cz
[2] Charles University in Prague, Faculty of Mathematics and Physics, Prague, Czech Republic, vojtas@ksi.mff.cuni.cz

Abstract. In this paper, we introduce a benchmark to test efficiency of RDF data model for data storage and querying in relation to a concrete dataset. We created Czech DBpedia - a freely available dataset composed of data extracted from Czech Wikipedia. But during creation and querying of this dataset, we faced problems caused by a lack of performance of used RDF storage. We designed metrics to measure efficiency of data storage approaches. Our metric quantifies the impact of data decomposition in RDF triples. Results of our benchmark applied to the dataset of Czech DBpedia are presented.

Keywords: Semantic Web, Linked Data, Storage, RDF

1 Introduction

DBpedia is a community effort to extract structured information from Wikipedia and to make this information available on the Web. In this paper, we introduce a Czech branch of this effort. The main DBpedia development gathers around English DBpedia. However, couple of local clones have emerged lately. To name some of them, we mention Greek, Russian or German local DBpedias. The complete list may be found in [17]. To the best of our knowledge, there has not been any other Czech DBpedia clone, apart from our work. We aim to extract structured information from Czech Wikipedia and provide the data for free use. In this paper, we show some use cases in order to demonstrate what can be such a huge database of machine readable information good for (focused particularly on Czech users).

We describe the data, we have extracted so far. We show basic statistics that can provide also a new point of view on the information available at Czech Wikipedia - not limited to DBpedia.

Described data are accessible via the SPARQL endpoint[3], so anyone can query them and use them. Also anyone can contribute to the community driven extraction effort via Mappings Wiki [15].

[3] The SPARQL endpoint is accessible on http://cs.dbpedia.org/sparql.

1.1 The Problem

Though datasets such as DBpedia can be very useful as pointed out in section 4, the adoption of these datasets is often limited to semantic web community. One of the main drawbacks is lack of performance. Data are usually stored as RDF triples. The performance of main RDF stores (or triple stores) increases as indicated by several benchmarks [1, 2]. But ordinary operations that are effectively executed in relational databases remain very demanding in the case of RDF stores.

The main factor that causes RDF stores to be inefficient for certain types of queries is a number of joins the store has to perform in order to evaluate a query. However, in many cases on the web, data is presented in a joined form (e.g. description of one entity and its properties on one page like in a Wikipedia article) and it is split in order to represent it as RDF. We elaborate this process in detail in Section 5. The opened question is, whether this split is necessary. We evaluate this phenomenon and quantify its influence on the dataset performance. Czech DBpedia is used as a representative dataset.

1.2 Contributions

- We introduce a comprehensive dataset of machine readable information extracted from Czech Wikipedia.
- The presented dataset covers wide variety of areas and preserves the connection of data to corresponding Wikipedia articles. As such can serve as a data integration hub and a source of common identifiers facilitating data integration of diverse data sets.
- We show possible applications of this dataset. We focus particularly on specific Czech use cases, where Czech DBpedia is worth than the global one.
- We point out the problem of ineffective data representation in case of RDF and quantify its influence on a dataset.
- We provide a SPARQL benchmark for testing data stored in triple stores.

1.3 Organization of the Paper

The rest of the paper is organized as follows. Related work is recalled in Section 2. Section 3 introduces the process of our dataset creation and Section 4 describes possible use cases of the dataset. Section 5 describes methods used to store RDF data and introduces our metrics measuring efficiency on a concrete dataset. Experimental results are presented in Section 6. The paper is summarized in Section 7.

2 Related Work

The whole philosophy of DBpedia and technical details are provided in [3, 4]. Both papers provide a broad overview of the DBpedia background. In our paper

we focus on a specific Czech environment and describe the dataset based on data from Czech Wikipedia, which might have slightly different characteristics (i.e. more fine grained information about locally specific entities).

Additionally, based on our dataset, we introduce metrics to measure the performance of data storage. There are several benchmarks measuring the performance of RDF data stores using SPARQL queries, similarly to our work. The suite of benchmarks for comparing the performance of SPARQL endpoints - Berlin SPARQL Benchmark [1] - tests the performance of SPARQL engines in prototypical e-commerce scenarios. Another recent benchmark built on top of English DBpedia dataset is DBpedia SPARQL Benchmark [2]. Contrary to Berlin Benchmark, that is composed of artificially designed queries, DBpedia Benchmark uses real world queries of real users. The queries are extracted from query logs of DBpedia SPARQL endpoint.

All the mentioned benchmarks use same metrics to measure the performance. The basic metrics are Queries per Second (QpS), Query Mixes per Hour (QMpH) and Overall Runtime (oaRT). But all these metrics are heavily dependent on a hardware configuration and overall system conditions of a test run. In Section 5.2, we introduce metrics that are more qualitative and quantitative characteristics of the tested dataset according to the used storage approach. Thus the results of our benchmark are completely independent of a concrete hardware configuration.

In order to develop our metrics, we considered various approaches to RDF data storage presented by major triple stores Jena [5], Virtuoso [9], Sesame [7], Oracle [6] and 3store [8]. Additionally, we consider vertical partitioning presented in [10]. These approaches are described and compared in Section 5. A similar problem of transformation of an ontology into a relational database is discussed in [11].

3 DBpedia Extraction Framework

In order to obtain raw data from Wikipedia, we use the DBpedia Extraction Framework [12]. This is a module based framework maintained by the international DBpedia team. Wikipedia provides freely accessible dumps of the whole article database [13]. The framework thus downloads recent dumps of all Wikipedia pages covering all topics described on Wikipedia. The pages are downloaded in the source format marked by Wiki markup [14]. These source files are parsed. Data are extracted from parsed pages using various extractors. An extractor is a mapping from a page node to a graph of statements about it.

Various information can be obtained from Wikipedia pages. It is quite easy to get labels of entities and extract their connections by analysis of links between corresponding Wikipedia articles. However, the core of the extraction process is the retrieval of information contained in so called infoboxes. An example of such an infobox in a Wikipedia article is shown in Figure 1.

In case of the Czech DBpedia, we use following extractors provided by the extraction framework: Label Extractor (extracts labels of entities from titles of corresponding articles), Geo Extractor (extracts geographic coordinates), Page

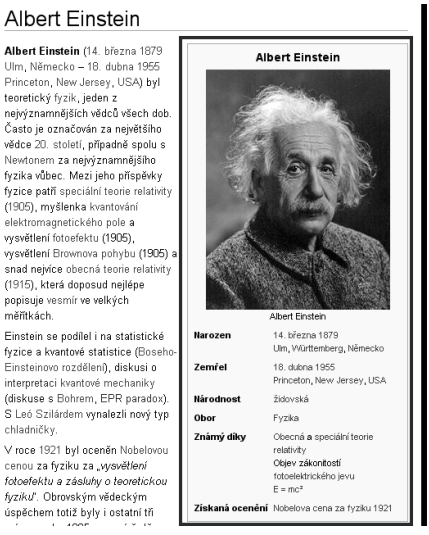


Fig. 1. Infobox example taken from Czech Wikipedia. On the right side, there is a source code of this Wikipedia infobox written using Wiki markup.

Links Extractor (extracts internal links between DBpedia instances from the internal pagelinks between Wikipedia articles), Wiki Page Extractor (extracts links to corresponding articles on Wikipedia), Infobox Extractor (extracts all properties from all infoboxes) and Mapping Extractor (extracts structured data based on hand-generated mappings of Wikipedia infoboxes to the DBpedia ontology).

It is important to note the difference between Infobox Extractor and Mapping Extractor. Consider the source code from Figure 1. The Infobox Extractor extracts this information as it is written in the source code. Property names are not cleaned, there is no consistent ontology for the infobox dataset. Thus generating RDF triples like:

```
<http://cs.dbpedia.org/resource/Albert_Einstein>
     <http://cs.dbpedia.org/property/misto_narozeni>
     <http://cs.dbpedia.org/resource/Ulm> .
```

Unfortunately, infoboxes on Wikipedia are inconsistent and it is usual that same property is described in many different ways. Someone can call the same property misto_narozeni (placeOfBirth), whereas someone else might use puvod (origin), or narozeni_misto (birthPlace).

The answer to these difficulties is the Mapping Extractor which uses hand written rules that map different patterns used in Wikipedia infoboxes to a consistent ontology of DBpedia.

For our dataset we generated rules covering 60 most important entities (e.g. cities, politicians, actors, writers). Totally, there are currently about 109 Infobox

templates on Czech Wikipedia that can be potentially mapped to DBpedia ontology. This means so far we have mapped more than half of them. Mappings can be edited via the Mappings Wiki [15]. As the mapping effort is community driven, everyone can join and help creating and maintaining mapping rules.

4 Use Case Scenarios

In this section, we want to point out the advantages of a local clone of DBpedia compared to the English one.

Similarly to local Wikipedias, local DBpedias usually provide more comprehensive information about specific local entities, like geographical data (smaller Czech cities, mountains, rivers, lakes), data about important persons (Czech politicians, movie directors, writers, etc.).

As such, this dataset may serve as a base for various mashup application or automatic data processing tools. Apart from the range of locally specific data, the language of entries and their direct connection to Czech Wikipedia pages might be an advantage too.

Thanks to tens of manually created mapping rules, Czech DBpedia is a good ontology mapping dictionary as well. It may serve as a central hub providing a set of common identifiers together with basic properties of identified entities.

All machine readable data on DBpedia have a direct connection to corresponding Wikipedia articles, where the information is presented in an unstructured way, usually as a plain text. Thus Czech DBpedia might serve as a testing dataset for various natural language processing and information retrieval approaches focusing on Czech language.

5 Efficiency of Data Storage

5.1 Data Representation

The RDF data model represents data as statements about resources using a graph connecting resource nodes. An RDF statement has the form of a triple consisting of subject, predicate and object. In the following text, unless otherwise stated, under subject, predicate and object, we understand the appropriate part of an RDF triple.

The majority of RDF data storage solutions including Jena [5], Virtuoso [9], Sesame [7], Oracle [6] and 3store [8] use relational databases to store the data. The general idea is to create one giant triples table with three columns (corresponding to RDF triples: subject, predicate, object) containing one row for each RDF statement. This data representation results in many self-joins on triples table even to execute simple SPARQL queries. Consider following query that returns names of movies directed by Jan Svěrák and filmed in 1996[4]:

[4] Usually, when querying real SPARQL endpoints, queries involve use of identifiers in similar form to URLs. For clarity of presented SPARQL queries, we omit these long identifiers as well as declaration of appropriate prefixes. Instead we use shorter identifiers in our examples (e.g. director, filmed, name).

```
SELECT ?name
WHERE { ?movie director "Jan Svěrák" .
?movie filmed "1996" .
?movie name ?name . }
```

This query results in following SQL query over triples table, invoking three joins:

```
SELECT T3.object
FROM triples_table AS T1,
     triples_table AS T2,
     triples_table AS T3
WHERE T1.subject = T2.subject AND T1.property = 'director'
  AND T1.object = '"Jan Svěrák"' AND T2.subject = T3.subject
  AND T2.property = 'filmed' AND T2.object = '"1996"'
  AND T3.property = 'name'
```

Instead of storing whole identifiers and literals directly in the triples table, Oracle, Virtuoso and Sesame replace strings with integer identifiers, so that the data is normalised in two tables. Whereas Jena takes the trade off - space for speed - and keeps strings directly in the triples table.

In order to minimize the count of inefficient join operations that are invoked by each statement used in SPARQL query, various optimizations were proposed.

Property tables first introduced by authors of Jena [5] optimize data organisation in following way:

- Create clusters of properties that often occur together.
- Based on identified clusters, create tables having properties occurring together as columns.
- The primary key of tables are subjects, in columns are stored objects that are connected with a particular subject by property represented by the column.

A variant of property tables are property-class tables, where clusters are created based on RDF types of subjects.

A similar approach to the property table data structure was introduced in [11] in order to store ontologies in relational databases.

A different approach is vertical partitioning described in [10]. The basic principle is to create a table for each property in the dataset. The table has two columns: subject and object. Each row thus corresponds to an RDF statement in that the particular property (represented by the table) connects subject and object stored in the same row. The performance optimization is achieved by storing data in a column oriented storage.

5.2 Metrics

The aim to minimize the impact of joins on the query execution leads us to define metrics that quantify this impact on a particular dataset.

In the following text, we use the term **shallow properties** to label properties such that there is at least one resource appearing with them in an object position that does not appear in a subject position in another statement.

Contrary, by **deep properties** we mean properties such that there is at least one resource appearing with them in an object position that appears in a subject position in at least one another statement.

Joinability measures an average count of shallow properties on a subject. If shallow properties were stored in one table (as columns of the table), there would not be a need to join the data at all. Shallow properties result in an avoidable subject-subject join in that case. Joinability is defined as follows:

$$j = \frac{c_s}{|S|}$$

Where c_s is the count of statements, where a property plays a role of a shallow property and $|S|$ is the count of distinct subjects in the dataset.

Linkability is an average count of deep properties on a subject. The occurrence of deep properties may result in an object-subject join, if we query for properties of an object as well. Linkability is defined in the following way:

$$l = \frac{c_d}{|S|}$$

Where c_d is the count of statements, where a property plays a role of a deep property and $|S|$ is the count of distinct subjects in the dataset.

Finally, we define**Weighted Joinability** as:

$$w = \frac{j}{l}$$

Another interesting characteristic of a dataset is an average **indegree** of an object and **outdegree** of a subject. Especially outdegree of a subject indicates, how many joins could be potentially saved, if all the properties of a particular subject were stored in one row. Note that sum of an average joinability and linkability gives an average outdegree of subjects in a dataset.

6 Evaluation

6.1 Data Characteristics

For the evaluation, we used the dataset of Czech DBpedia. It is smaller than the dataset of English DBpedia, so it is feasible to run some more demanding queries that are difficult to run in reasonable time on the English one. However, still Czech DBpedia represents a comprehensive dataset, with similar characteristics to the English one. In this section, we provide some basic characteristics of data that can be extracted from Czech Wikipedia and compare it to the English Wikipedia.

Czech DBpedia contains totally 12 402 513 RDF statements. About 251 877 RDF statements are extracted based on our hand written mappings, whereas the English DBpedia composes of about 17,5 million RDF statements based on mappings (according to the last dataset release report [16]).

Data is extracted from 220 000 articles on Czech Wikipedia. English Wikipedia has about 3 800 000 articles which is more than 17 times bigger dataset.

There are 704 760 distinct subjects that have some property in the current Czech dataset.

Additionally, we counted entities. Under entities, we mean real world concepts. Usually, each entity corresponds to a Wikipedia article that describes it. Thus total count of entities should correspond to 220 000 Wikipedia articles. But in reality the count of identified entities is heavily dependent on a coverage of hand written mapping rules. Entities have commonly a type. Counts of most common entities compared to the English dataset are provided in Table 1. It is remarkable that in case of cities, the count of entities in Czech dataset is closer to the English dataset than in other cases (it is more than 34% of the count of cities in English dataset). This points to the fact that information about cities are well elaborated on Czech Wikipedia and also good mappings are provided to transform it to DBpedia. The English DBpedia presents actually much more countries than their real count in the world. This fact is caused by a vague definition of a country on Wikipedia. For example a self-governing British Overseas Territory Falkland Islands is considered to be a country as well.

Table 1. Comparison of Czech and English DBpedia. Count of entities of certain types. In the column Size Comparison the sizes of both datasets are compared in percent.

	Count of Entities		
Entity Type	**Czech DBpedia**	**English DBpedia**	**Size Comparison**
Person	8478	416079	2,0%
Company	964	40132	2,4%
Country	287	2531	11,3%
City	4730	13790	34,3%

6.2 Measurements and Benchmarks

In this section, we provide some characteristics, we measured on the dataset of Czech DBpedia with respect to metrics introduced in Section 5.2. Due to the lack of space in this paper, we present the actual benchmark on a separate web page[5].

In Figure 2, we compare count of distinct subjects with at least one shallow property and at least one deep property. We can see that almost all subjects

[5] The whole benchmark used to obtain presented results composes of SPARQL queries and basic scripts to parse results of these queries. It is available for download at `http://research.i-lasek.cz/semantic-web/joinability-benchmark/`

have a shallow property as well as a deep property. There were less than 6 000 subjects having all properties shallow. These were in most cases operational and meta data information rather than real entities such as cities, persons etc.

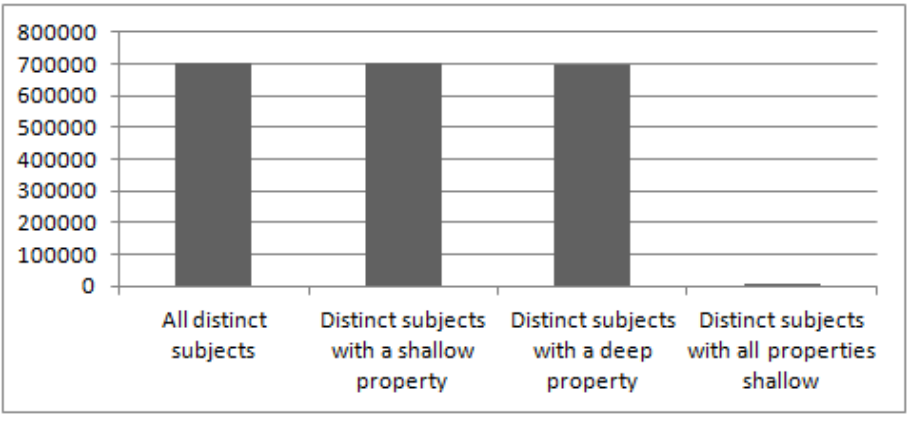


Fig. 2. Counts of distinct subjects with at least one shallow property and at least one deep property compared to the total count of distinct subjects in the dataset. Additionally subjects with all properties shallow are displayed.

In Figure 3, we further investigated overall count of distinct shallow properties and distinct deep properties. It is remarkable that both sets have a common intersection. According to the results, almost all properties are in some cases shallow. While some of them play a role of a deep property as well. This is legal, because a property might connect a subject with an object, that is not further described in the dataset, while in another case another connected object is a subject in another statement. Also sometimes, data is messy and same property connects subject with a literal value and at the same time connects it with an object with further properties. 1 697 out of 6 714 properties played a role of deep properties in at least one statement.

In Figure 4, we compare counts of objects that do not play a role of a subject in any other statement (result in a shallow connecting property), with objects that play a role of subjects in at least one another statement (result in a deep connecting property).

Afterwards, we measured counts of non distinct shallow and deep properties and counted Linkability and Joinability. For Czech DBpedia, these are: $j = 8,92$ and $l = 8,67$.

Finally, we evaluated average outdegree of subjects (for Czech DBpedia it is approximately 17,6) and average indegree of objects (approximately 5,0). For the quantification of join impact, the outdegree of subjects is especially important. This means that, while querying DBpedia for all properties of an entity, a query results in average in almost 18 self-joins, if we consider the triples table approach

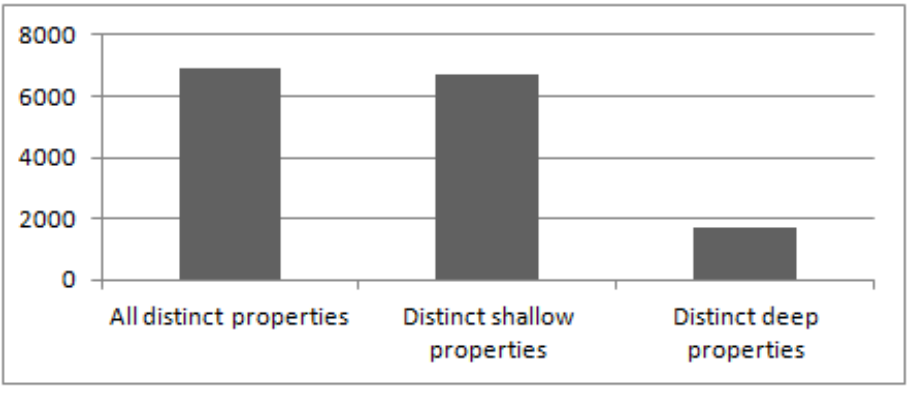


Fig. 3. Counts of distinct shallow properties and distinct deep properties compared to overall count of distinct properties.

Fig. 4. Counts of distinct objects that have at least one property in the dataset (are in the role of a subject in another triple) and distinct objects that do not have any property within the given dataset.

described in Section 5.1. The distribution of subjects outdegree is displayed in Figure 5.

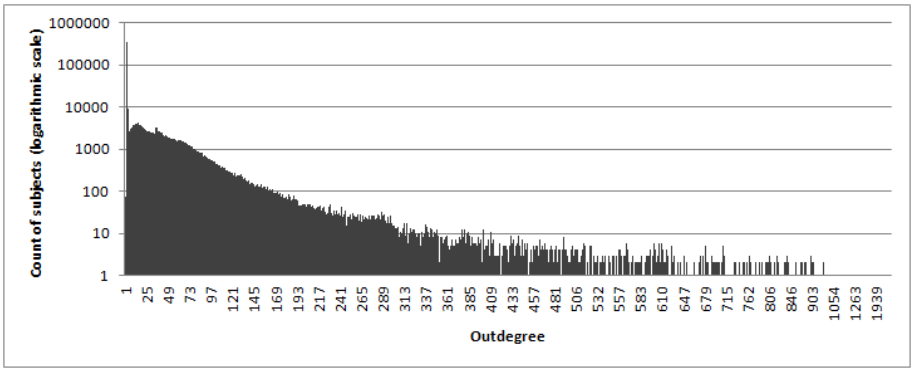


Fig. 5. Subjects outdegree distribution. Counts of occurences are displayed in a logarithmic scale.

7 Conclusion

We introduced Czech DBpedia and briefly described the process of creation of this dataset. The main idea was introduced: On the web, huge amount of data is presented in a joined from (e.g. infobox tables on Wikipedia). However, often this data is split in order to present it as RDF. Afterwards it has to be joined again, while querying RDF data storage, which is a demanding operation. We designed metrics to measure efficiency of data storage approaches, according to the join demands. We introduced a benchmark to test efficiency of RDF data model for data storage and querying in relation to a concrete dataset. Finally, we presented results of this benchmark applied to a dataset of Czech DBpedia. The benchmark is publicly available on the web, so anyone can use it to test any dataset accessible via a SPARQL endpoint.

7.1 Future Work

In our future work, we plan to design an efficient RDF repository. Based on our current findings, we intend to use property tables. Our modification is that we use a variant of the algorithm for creating concepts (in the sense of Conceptual lattices (see [18]) on properties (columns of tables). Our optimisation is driven by requirement to minimize number of joins in most frequent expected queries (conjunctive queries looking for subject with conjunction of conditions on property values). Secondary optimization is devoted to minimization of number of NULL values (which occurs when a subject does not have some property).

Acknowledgments. This work has been supported by the grant of Czech Technical University in Prague (SGS12/093/OHK3/1T/18) and by the grant GACR P202/10/0761.

References

1. Bizer, C., Schultz, A.: Benchmarking the Performance of Storage Systems that expose SPARQL Endpoints. In International Journal On Semantic Web and Information Systems, 2009.
2. Morsey, M., Lehmann, J., Auer, S., Ngonga Ngomo: DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data. In The International Semantic Web Conference – ISWC 2011, Springer Berlin / Heidelberg, 2011, 7031, Pages 454–469.
3. Bizer, Ch., Lehmann, J., Kobilarov, G., Auer, S., Becker, Ch., Cyganiak, R., Hellmann, S.: DBpedia - A crystallization point for the Web of Data. In Web Semantics: Science, Services and Agents on the World Wide Web, Volume 7, Issue 3, September 2009, Pages 154–165, ISSN 1570-8268.
4. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data The Semantic Web. Springer Berlin / Heidelberg, 2007, 4825, Pages 722–735.
5. Wilkinson, K., Sayers, C., Kuno, H., Reynolds, D.: Efficient RDF Storage and Retrieval in Jena2. In SWDB, Pages 131–150, 2003.
6. Chong, E. I., Das, S., Eadon, G., Srinivasan, J.: An Efficient SQL-based RDF Querying Scheme. In VLDB, Pages 1216–1227, 2005.
7. Broekstra, J., Kampman, A., Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In ISWC, Pages 54–68, 2002.
8. Harris, S., Gibbins, N.: 3store: Efficient bulk RDF storage. In In Proc. of PSSS 03, Pages 1–15, 2003.
9. Erling, O., Mikhailov, I.: RDF Support in the Virtuoso DBMS. In Networked Knowledge - Networked Media, Studies in Computational Intelligence, Springer Berlin / Heidelberg, isbn: 978-3-642-02183-1, 2009, Pages 7–24.
10. Abadi, D. J., Marcus, A., Data, B.: Scalable semantic web data management using vertical partitioning. In VLDB 2007, Pages 411–422, 2007.
11. Pokorný, J., Pribolová, J., Vojtáš., P.: Ontology Engineering Relationally. In DATESO 2009, Špindlerův Mlýn, Czech Republic, 2009, Pages 44–55.
12. The DBpedia Information Extraction Framework, `http://wiki.dbpedia.org/Documentation`
13. Wikipedia Database Download `http://en.wikipedia.org/wiki/Wikipedia:Database_download`
14. Wiki markup `http://en.wikipedia.org/wiki/Help:Wiki_markup`
15. Czech Mappings on DBpedia Mappings Wiki `http://mappings.dbpedia.org/index.php?title=Special%3AAllPages&from=&to=&namespace=224`
16. Dataset Release Report `http://blog.dbpedia.org/category/dataset-releases/`
17. DBpedia Internationalization Committee `http://dbpedia.org/Internationalization`
18. Ganter, B., Stumme, G., Wille, R., eds. (2005): Formal Concept Analysis: Foundations and Applications. In Lecture Notes in Artificial Intelligence, no. 3626, Springer-Verlag, ISBN 3-540-27891-5.

Top-*k* Search Over Grid File

Martin Šumák and Peter Gurský

Institute of computer science, Faculty of Science, P. J. Šafárik University in Košice
Jesenná 5, 040 01 Košice, Slovakia
martin.sumak@student.upjs.sk, peter.gursky@upjs.sk

Abstract. In the era of huge datasets, the top-*k* search becomes an effective way to decrease the search time of top-*k* objects. Since we suppose locally accessible data only, the multidimensional indexes containing all attributes together seem to be more effective than a distribution of each attribute to a separate index. Therefore we introduce the top-*k* search algorithm over grid file – the multidimensional index not used for the top-*k* search yet. Grid file does not require computation with all attribute values together like R-tree, R*-tree (i.e. computation of area, perimeter) nor a metric like M-tree. Grid file can be used directly for indexing any type of attributes with natural ordering. Our experiments show that grid file, R-tree and R*-tree offer much better performance of the top-*k* search than separated B^+-trees and table scan.

1 Introduction

In our research we deal with the problem of searching top-*k* products in e-shops according to user preferences. Current e-shops typically provide fulltext search, menu of product domains, single attribute value specification and products sorted usually according to price or product name. We are not aware of any e-shop with more complex user preferences model e.g. a combination of selection techniques mentioned above.

Our model of user preferences [6] consists of preferences to values of several attributes in the form of fuzzy functions (see Figure 1) and a monotone combination function. Such complex user preference model approaches real life preferences and leads to more precise results than standard selection techniques. The preferences can be obtained implicitly by tracking user actions in the e-shop or explicitly by user specification. The top-*k* search with a query based on such preferences can be computed over different index structures – a set of B+-trees [5, 6], MDB-tree [12] and R-tree [13]. In this paper we introduce a top-*k* search algorithm over the next multidimensional index – Grid file.

In [12, 13] it was shown that the top-*k* search over multidimensional indexes (MDB-tree, R-tree, R*-tree) is faster over local data than over a combination of multiple indexes. The MDB-tree can hold all types of ordered attributes but the query cannot hold any subset of attributes. The R-tree structure allows a query to contain any subset of attributes but the metrics used in the R-tree requires having the numbered attributes only. The reason why we employed a grid file for the top-*k* search was the elimination of the limitations along with the preservation of the advantages of the mentioned indexes.

© J. Pokorný, V. Snášel, K. Richta (Eds.): Dateso 2012, pp. 115–126, ISBN 978-80-7378-171-2.
Charles University in Prague, MFF, Department of Software Engineering, 2012.

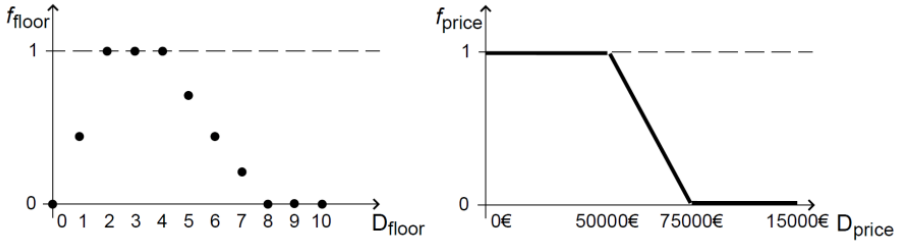


Fig.1. User's local preferences to a floor and a price of a flat

This paper is organized as follows. Section 2 presents the related work. Section 3 formalizes the problem of the top-*k* search over our user preference model. Section 4 presents the main contribution of this paper – the top-*k* search over grid file. Section 5 reports the experiments results comparing the top-*k* search performance over several index structures. Section 6 concludes this paper.

2 Related work

The top-*k* search was introduced by R. Fagin [4] as a problem of finding *k* best objects according to a monotone aggregation function over distributed ordered lists of attribute values. The original Threshold algorithm (TA) [4] has many improvements and modifications for the similar distributed environment, e. g. [1, 2, 4, 6]. We call them the TA-like algorithms.

The idea of the top-*k* search over data stored in several indexes was considered also for local data, especially inside a RDBMS, e. g. [10]. These approaches are concerned with augmenting the query optimizer to consider rank-joins (similar to TA) during a plan evaluation. Optimization can be effective especially in case of very selective attributes. The rank-join algorithm requires ordered data on input similarly to the middleware algorithms.

The idea of using R-tree for top-*k* search is already presented in [14] where the algorithm incremental nearest neighbour is exploited for that purpose. Nevertheless, this approach does not offer a query as complex as we offer in our query model.

Originally in the top-*k* query, the simple monotone aggregation function was considered only [4]. The query composed of local preferences and monotone combination function (resulting in non-monotone aggregation function) was introduced in [5]. In [6] it was shown that the simulations of sorted accesses using separated indexes for each attribute allow using TA-like algorithms.

The algorithm in [16] does a top-*k* search with an arbitrary non-monotone query analyzing the aggregation function with numerical methods. The algorithm supposes that the numerical methods can analyze any aggregation function over any domain sub-region (to find the maximum and possibly recognize monotonicity). In our opinion this analysis is rather difficult to do in a reasonable time. Note that this approach uses multiple indexes.

The grid file was introduced in [11]. In many papers the grid file is considered to be a dynamic index with a directory structure mapping grid windows to the disk pages

[7, 11, 15]. In our pilot grid file implementation we considered static data only, thus we made some simplifications (see Section 4). First, we made the numbering of grid windows that can substitute the presence of directory structure and dramatically decrease the number of accesses, thus making most objects accessible in 1 I/O. Second, unlike the original grid file we employed the overflow pages to avoid dense grid structure with many empty windows over possibly skew data. We analyzed several bulk loading techniques [3, 8, 9]. The STR algorithm [9] has the best results for our top-k search.

3 Top-k search problem definition

For a given set $\boldsymbol{S}$ of objects we have to find k most preferred objects for the user. Each object $O \in \boldsymbol{S}$ has the same m attributes with values $v_1(O), \ldots, v_m(O)$ from attribute domains $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m$ respectively (i.e. $v_i: \boldsymbol{S} \to \boldsymbol{A}_i$ for all $i \in \{1, \ldots, m\}$). Query, i.e. input obtained from the user, consists of m fuzzy functions $f_1, \ldots, f_m$ (or less if user does not consider all attributes) and a monotone combination function C. The overall value of object O is $C(f_1(v_1(O)), \ldots, f_m(v_m(O)))$. For example, if C is a weighted sum, user is expected to specify only nonnegative weights – one for each considered attribute to specify a non-descending combination function. Then we have:

$$C(f_1(v_1(O)), \ldots, f_m(v_m(O))) = w_1 * f_1(v_1(O)) + \cdots + w_m * f_m(v_m(O))$$

where $w_1, \ldots, w_m$ are the weights. The bigger the overall value, the more preferred the object O is to user. The output is a list of k objects from $\boldsymbol{S}$ ordered from the most preferred objects to the less preferred ones.

4 Grid file

The grid file [9] is an index structure for multidimensional points designed to store the data on disk pages. Grid file is based on slicing space in each dimension, i. e. an attribute domain, to get a multidimensional grid. For the formal description we introduce the following notation. Partition $P = (U_{1,1}, \ldots, U_{1,n_1}) \times \ldots \times (U_{m,1}, \ldots, U_{m,n_m})$ is determined by a sequence of intervals in each dimension. Each (i-th) sequence consists of disjunctive intervals such that $U_{i,1} \cup \ldots \cup U_{i,n_i} = \boldsymbol{A}_i$. Picking one interval in each dimension specifies a window $U_{1,j_1} \times \ldots \times U_{m,j_m}$. Each window is mapped to one data page on disk (these pages are called primary pages). The grid file contains as many primary pages as windows. All data pages have the same fixed size (i.e. the same fixed capacity). Overfilled windows are handled by creating a linked chain of overflow pages.

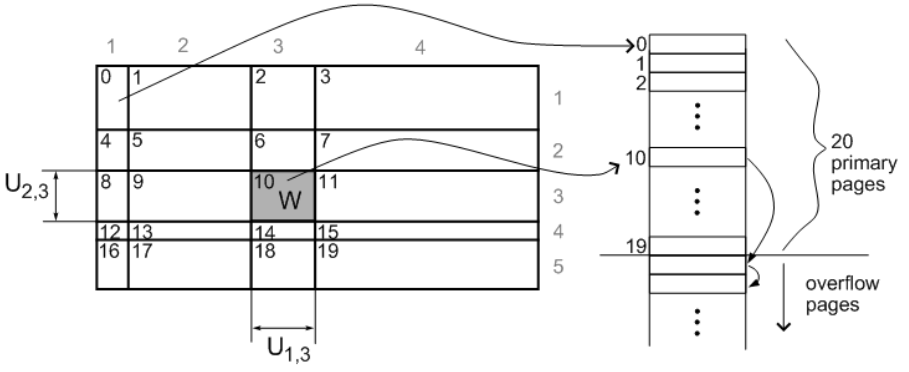


Fig. 2. An example of a two-dimensional grid with 20 windows and the overfilled window W. Window W is determined by the third interval in both dimensions.

On the other hand, the grid file may contain empty windows. Each empty window refers to an empty primary page. Reading empty pages (e. g. during query evaluation) is avoided by a set of numbers of non-empty windows held in memory. The windows are numbered in a fashion shown on Figure 2. Since we have a static grid, the mapping between a window and its number is easy to compute without the need of a directory structure. The extension of this idea to more dimensions is straightforward.

We create the grid file with bulk loading algorithm STR [9]. Page capacity and the number of objects in $\boldsymbol{S}$ determine the final number of windows (primary pages). In a m-dimensional space the m-th root of the number of windows determines the number of intervals in each dimension. Each dimension is partitioned into intervals with the same number of objects falling in them. Since real data is rarely distributed uniformly we reduce the number of overfilled windows by increasing the number of windows by the multiplication with appropriate filling factor (we use filling factor 1.3). After the bulk loading of input data we are not restricted from adding more objects – it simply leads to higher utilization of pages and possibly to some new overflow pages.

4.1 Top-k search over grid file

A contribution of this paper is a top-k search algorithm over grid file. As shown in the experiments, this approach is much more effective than B^+-trees based approach and it is comparable with the top-k search over R-tree [13].

Since each object O can be represented by the point $p(O) = (v_1(O), \dots, v_m(O))$ in m-dimensional space (note that $p: \boldsymbol{S} \rightarrow \boldsymbol{A}_1 \times \dots \times \boldsymbol{A}_m$ is the function mapping objects to m-dimensional points), the set $\boldsymbol{S}$ of objects can be stored in multidimensional index such as grid file [8, 9]. Grid file does not require attribute domains to be sets of numbers. It can handle different types of attributes at once. For example the first attribute can be price represented by decimal numbers while the second attribute can be a manufacturer represented by strings (with alphabetical ordering). Grid file treats attribute domains separately therefore they are not required to have any common property. Attribute domain just needs to be an ordered set.

For searching top-k objects over a grid file we developed algorithm similar to the breadth first search in graphs. For formal description of our algorithm some concepts need to be defined first.

Definition 1: Point Z is m-dimensional vector $Z = (Z_1, \dots, Z_m) \in \boldsymbol{A}_1 \times \dots \times \boldsymbol{A}_m$.

Having $O \in \boldsymbol{S}$ and point Z such that $Z = p(O)$ then $Z_i = v_i(O)$ for all $i \in \{1, \dots, m\}$.

Definition 2: A window is defined as an m-tuple of intervals $(U_1, \dots, U_m)$ where U_i is an interval within $\boldsymbol{A}_i$ for all $i \in \{1, \dots, m\}$.

Object O belongs to a window $W = (U_1, \dots, U_m)$ if $v_i(O) \in U_i$ for all $i \in \{1, \dots, m\}$.

Definition 3: We say that window $V = (U_{1,i_1}, \dots, U_{m,i_m})$ is a neighbour to window $W = (U_{1,j_1}, \dots, U_{m,j_m})$ iff there is a $q \in \{1, \dots, m\}$ such that $|i_q - j_q| = 1$ and for all $r \in \{1, \dots, m\} \backslash \{q\}$ holds $i_r = j_r$.

Note that window in a 2-dimensional grid has at most 4 neighbour windows – top, bottom, left and right. Window in the corner of the gird has two neighbour windows.

For the top-k search over a grid file we need to know how to evaluate objects and also grid windows. For this purpose we define aggregation function h giving the overall value for an object and the maximal possible overall value for any object in a grid window.

Definition 4: Function $h: \boldsymbol{S} \cup \boldsymbol{W} \to \boldsymbol{R}$ where $\boldsymbol{W}$ is a set of windows of grid file is defined as follows: $h(E) = C(y_1, \dots, y_m)$ where

$y_i = f_i(v_i(E))$ for all $i \in \{1, \dots, m\}$ if $E \in \boldsymbol{S}$ OR
$y_i = max\{f_i(x): x \in U_i\}$ if $E = (U_1, \dots, U_m) \in \boldsymbol{W}$.

Lemma 1: If $W = (U_1, \dots, U_m)$ is a window and O is an object such that $p(O) \in U_1 \times \dots \times U_m$ (i.e. O belongs to a window W) then $h(O) \leq h(W)$.

Proof: From definition 4 we have $h(O) = C(f_1(v_1(O)), \dots, f_m(v_m(O)))$ and $h(W) = C(max\{f_1(x): x \in U_1\}, \dots, max\{f_m(x): x \in U_m\})$. Moreover for all $i \in \{1, \dots, m\}$ holds: $f_i(v_i(O)) \leq max\{f_i(x): x \in U_i\}$ because $v_i(O) \in U_i$. Since combination function C is monotone (non-descending in each parameter) we get $h(O) \leq h(W)$.

Lemma 2: If $h(O) \geq h(W)$ (where O is an object and W is a window) then for any object Q within W holds $h(O) \geq h(Q)$.

Proof: directly from Lemma 1 and the transitivity of relation $\geq$.

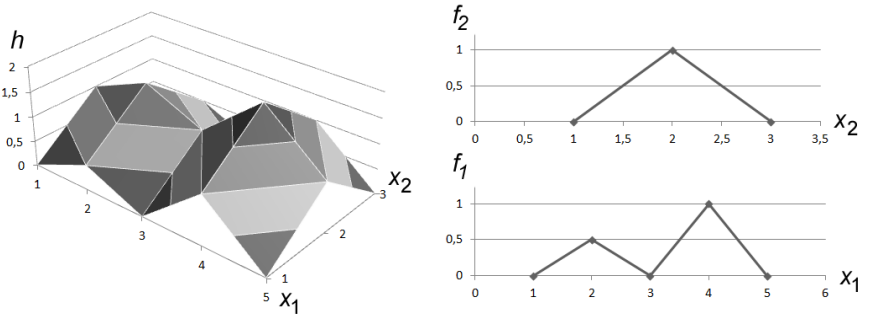


Fig. 3. Graphical representation of the aggregation function $h(E) = y_1 + y_2$ giving the overall value $f_1(x_1) + f_2(x_2)$ for a 2-dimensional data with user-defined fuzzy functions f_1 and f_2 on the right.

Preferential top-k search algorithm over grid file:

```
Input: grid file containing objects from S, fuzzy
  functions f1,...,fm, combination function C and number k
Output: ordered list of k objects with the highest value
    of the h function
1. queue = empty priority queue ordered by the value of
   the h function of its elements in descending order
2. result = empty list of objects
3. for each local extreme of the function h do
   a. choose arbitrary one window W containing the
      extreme
   b. if queue does not contain W then put W into queue
      and label W as visited window
4. while the result does not contain k objects do
   a. let E be the first element of the queue, remove E
      from the queue
   b. if E is a window then
      i. add all objects within E to the queue
      ii. add all not visited neighbour windows of E to
          the queue and label them as visited windows
   c. if E is an object then add it at the end of the
      result
5. return result
```

The estimation of time and space complexity can be reduced to an estimation of the number of visited windows, which is highly dependent on the grid partition, data distribution and query. The only estimation we can make are the lower and upper bounds, which is not very rewarding. In the best case we get the top-k objects after processing one window – the first one in the queue containing global extreme. In the worst case all the windows must be read – typically when a high k or low discriminating fuzzy functions or fuzzy functions containing many local extremes are obtained

from user. Fuzzy functions with many local extremes are not common in a queries made by people.

Labeling visited windows can be realized by maintaining a set of numbers of visited windows starting with an empty set. Avoiding repetitive reading of visited windows is implemented the same way as avoiding reading of empty windows – by maintaining a set of window numbers in memory.

4.2 Correctness of the top-k search over grid file

The proof of correctness of presented algorithm can be reduced (without impact on generality) to the situation with just one local extreme of function h. If we prove that it works for the case of one local extreme then the generalization to more extremes can go as follows: we can prepare as many priority queues as windows with local extremes in step 3. Then in step 4.a we can pick the priority queue with the highest value of its first element and continue without any other changes. Using separated priority queues for each starting window picked in step 3 is not necessary. The same effect can be achieved by one priority queue managing windows of all local extremes because on the top there is always an element with the highest value from all top elements in imaginary separated priority queues.

Let us focus on one local extreme of the function h. First of all we will show that the value of the first element in priority queue is non-ascending. Using mathematic induction we will show that each time the first element is removed from priority queue, the new top element of the priority queue (i.e. in the next iteration of while cycle) has lower or equal value to the previous top element.

If the top element in the priority queue is an object then the condition holds trivially, since no new element is added into priority queue and the next top element was already in the priority queue in previous iteration.

Let's assume that there is a window at the top of the priority queue. We have to show that all new elements added into priority queue in steps 4.b.i and 4.b.ii have the overall value lower or equal to the window just removed from the top. For better imagination we will use the example on Figure 4.

The first induction step is as follows: at the beginning, the priority queue contains just one window W – the one containing a local extreme of function h. Window W is to be removed from the queue directly in the first iteration of while cycle (step 4). After that, the algorithm inserts the objects within window W and its neighbour windows A, B, C and D to the priority queue. In Lemma 1 we showed that objects belonging to window W have the overall value lower or equal to the overall value of window W. Trivially, the neighbour windows A, B, C, D do not have the overall value greater than window W because the algorithm started with window containing the only local extreme.

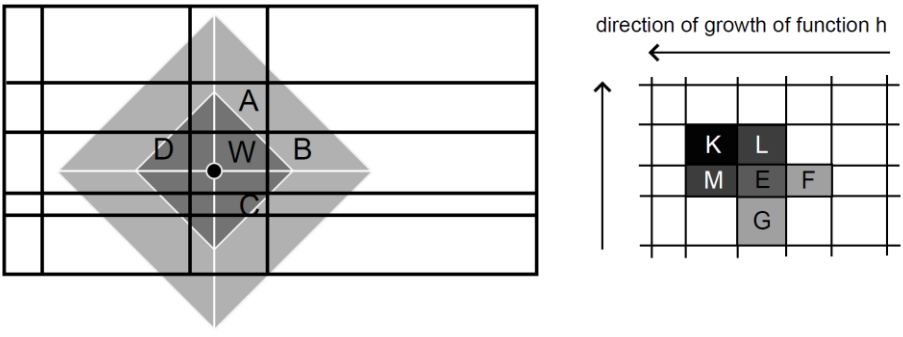


Fig. 4. On the left, there is a window W containing local extreme and its neighbour windows A, B, C and D. On the right, there is an example of the second inductive step with local extreme somewhere in left top corner. The darker shade means higher overall value.

For the second induction step we assume the following induction assumption: in each of previous iterations of while cycle, the overall value of the top element in the priority queue decreases or does not change. Let window E (Figure 4 on the right) be the first element in the queue. After removing the window E from the priority queue, the objects from E and not visited neighbour windows (L, M, F, G) are inserted into the priority queue. In each dimension there is one direction in which the respective fuzzy function is non-descending and the opposite direction oriented off the local extreme in which the fuzzy function is non-ascending. In the example on Figure 4 the local extreme is somewhere in the top left corner. Therefore we can trivially say that $h(K) \geq h(L)$, $h(K) \geq h(M)$, $h(L) \geq h(E)$, $h(M) \geq h(E)$, $h(E) \geq h(F)$ and $h(E) \geq h(G)$. We are left to show that windows L and M have been already visited and therefore they are not to be inserted to priority queue now. From the induction assumption we get that window E was added to priority queue as neighbour when either window L or M was removed from the top. Without impact on generality let us suppose that window L is the removed window. Since we know that window L has been already visited we are left to discuss window M. Since window K has higher value than window L and L is its neighbour, from the induction assumption we get that window K must have been visited before L. Moreover only top windows from the priority queue are processed. Therefore window K must have been processed before window L and window M must have been added into priority queue when window K was being processed. Hence windows L and M had been visited before window E was processed and are not inserted into the priority queue.

Although we described the second induction step on an example in a 2-dimensional space the generalization to more dimensions is straightforward. Even in 2D we can imagine a situation slightly different from the one drawn on Figure 4 on the right. Let us imagine the following change: $h(L) \leq h(E)$. In this situation window E has only one neighbour with higher overall value – window M. The discussion for this case is even simpler. From the induction assumption we know that window M must have been visited prior to window E.

Since value of the element at the top of the priority queue is non-ascending the first object that appears at the top is the best object of all. Each object which would appear

in the priority queue later will be at most as good as any object in the result set. Since we look at all neighbour windows, processing the rest of the priority queue leads to acquiring all objects within grid file in order from the best to the worst.

Note that the presented grid file expansion strategy in the top-k search works correctly for our user preferences model, however it cannot be used for arbitrary aggregation function in general. For arbitrary aggregation function it requires a modification of the neighbour windows definition to two windows with a common point and it leads to exponentially more priority queue insertions according to number of dimensions than in the presented algorithm.

5 Experiments

Average time of top-k query evaluation is the basic measure we surveyed. We used a real data set containing approximately 27 000 flat or house advertisements in Slovakia having 6 attributes: price, area, floor, the highest floor of building, year of approbation and the number of rooms. Since the real data set was small we generated bigger pseudo real sets by generation of several similar objects for each one from the original set. This way we generated two sets, one with about 550 000 objects (the 20-multiple set) and second one with about 2 700 000 objects (the 100-multiple set).

We compared the following approaches for top-k search problem: grid file based approach, R-tree and R*-tree based approach [13], local TA on B^+-trees [6] and table scan (on heap file).

There are several algorithms based on ordered lists presented in [6]. All of them work with distributed data and sorted access. Moreover the original TA requires also the random access. Since we presuppose only locally accessible data (not distributed) we slightly adapted the TA in the following way: each B^+-tree which represents one ordered list (providing sorted access for one attribute) will contain all the data i.e. values of all attributes. Thus no random access is necessary because one sorted access to any ordered list provides complete information about all attribute values of one object. Such version of TA does not longer suffer from the handicap of distributed data. We made a small experiment which showed that algorithms NRA [6] and original TA are significantly less efficient than the local version of TA. Due to this handicap we have not involved algorithms NRA and original TA to the tests.

Each of the tests consists of the same set of 1100 random queries (i.e. about 200 random queries containing gradually 2, 3, 4, 5 and all 6 attributes). Not all n-attribute queries consist of the same attributes.

All of the compared approaches manage data file on disk differently. The page size is the only adjustable parameter common to all of them. Since we wanted to compare them objectively we had to find the most suitable page size for each one. For the brevity we do not present graphs showing the time dependency on page size. We found out that the best page sizes for top-k search over 20-multiple set are the following: 1 kB for R-tree, 2 kB for R*-tree, 8 kB for grid file, 4 MB for heap file (table scan) and 64 kB for B^+-trees (local TA). For the 100-multiple set we found out the following: 2 kB for R-tree, 1 kB for R*-tree, 4 kB for grid file, 2 MB for heap file (table scan) and 128 kB for B^+-trees (local TA). We strongly recommend to do such a

survey for all application domains and not to consider these results to be universal. Different size of objects leads to a different capacity of page sizes and probably a different best page sizes. We compared average time of top-k query just for the best page sizes.

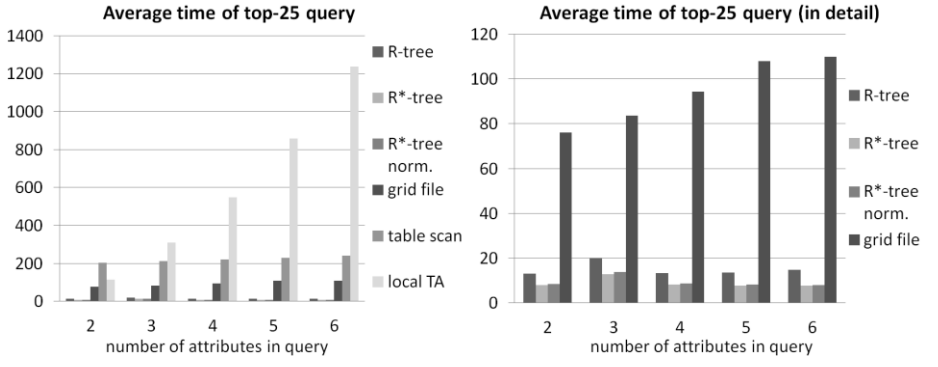


Fig. 5. The average time of top-25 query evaluation in milliseconds over 20-multiple data set (about 550 000 objects). On the right graph the table scan and the local TA are omitted for detailed comparison.

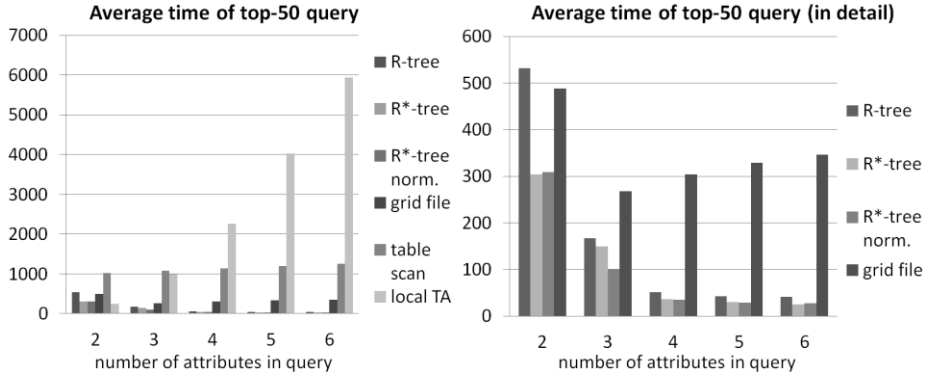


Fig. 6. The average time of top-50 query evaluation in milliseconds over 100-multiple data set (about 2 700 000 objects). On the right graph the table scan and the local TA are omitted for detailed comparison.

On the left graphs we see that number of attributes in query has a very low impact on time of table scan and very high impact on time of local TA. Moreover we can say that table scan is significantly less efficient than R-tree, R*-tree and grid file based approaches. The local TA is quite efficient for queries with only 2 attributes. Local TA loses its efficiency when 3 or more attributes are required.

R-tree, R*-tree and grid file based approaches seem to be faster therefore the graphs on the right bring the detailed look just on them. We can see that R*-tree offers a better efficiency than grid file in all cases. Moreover R*-tree with normalized data does not offer better search performance than R*-tree with original data. We did

not use R-tree with normalized data because normalization of data has no effect when quadratic split algorithm is used [13].

6 Conclusion

In this paper we introduced the top-*k* search algorithm over grid file. Grid file is a multidimensional index structure in which we can store objects with arbitrary ordered attributes (numbers, strings, hierarchies) and which allows using a query with any subset of attributes.

Grid file organizes data by means of multidimensional intervals (windows) which are used also in R-tree as hyper-rectangles of nodes. In grid file there is no hierarchy or overlaps as it is in case of nodes of R-tree. Hence there was a question: can grid file offer better top-*k* search performance than R-tree or R*-tree? It would be premature to say no just because our introductory tests showed that the top-*k* search over R*-tree is faster. Our grid file implementation is quite simple. We have found many overflow pages and many empty windows because of the real data distribution. The results are quite promising and encourage us to look for more sophisticated ways of creating and organizing grid file.

Acknowledgement

This work was partially supported by VEGA 1/0832/12.

References

1. Akbarinia, R., Pacitti, E., Valduriez, P.: Best Position Algorithms for Top-k Queries. In VLDB, (2007)
2. Bast, H., Majumdar, D., Schenkel, R., Theobald, M., Weikum, G.: IOTop-k: Index-Access Optimized Top-k Query Processing. In VLDB, (2006)
3. Bercken, J., Seeger, B.: An Evaluation of Generic Bulk Loading Techniques. Proceedings of the 27th International Conference on Very Large Data Bases, ISBN:1-55860-804-4, pp.461-470, (2001)
4. Fagin, R., Lotem, A., Naor, M.: Optimal Aggregation Algorithms for Middleware. Proc. ACM PODS, 2001
5. Gurský, P.: Towards better semantics in the multifeature querying. Proceedings of Dateso 2006, ISBN 80-248-1025-5, pages 63-73 (2006)
6. Gurský, P., Pázman, R., Vojtáš, P.: On supporting wide range of attribute types for top-k search. Computing and Informatics, Vol. 28, no. 4, 2009, ISSN 1335-9150, p. 483-513.
7. Kumar, A.: G-tree: a new data structure for organizing multidimensional data. IEEE Transactions on knowledge and data engineering, ISSN: 1041-4347, pp. 341 - 347, vol. 6, issue 2, (1994)
8. Leutenegger, S. T., Nicol, D. M.: Efficient Bulk-Loading of Grid files. IEEE Transactions on knowledge and data engineering, ISSN: 1041-4347, vol. 9, no. 3, (1997)

9. Leutenegger, S.T.; Lopez, M.A.; Edgington, J.: STR: a simple and efficient algorithm for R-tree packing. Proceedings of the 13th International Conference on Data Engineering, ISBN: 0-8186-7807-0, pp. 497-506, (1997)
10. Li, C, Chang, K., Ilyas, I.F., Song, S.: RankSQL: Query Algebra and Optimization for Relational Top-k Queries. SIGMOD (2005)
11. Nievergelt, J., Hinterberger, H., Sevcik, K. C.: The Grid File: An Adaptable, Symmetric Multikey File Structure. ACM Transactions on Database Systems, pp. 33-71, vol. 9, issue 1, (1984)
12. Ondreička M., Pokorný J.: Efficient Top-K Problem Solvings for More Users in Tree-Oriented Data Structures. Proceedings of Signal-Image Technology & Internet-Based Systems, ISBN: 978-1-4244-5740-3, pp. 345-354 (2010)
13. Šumák, M., Gurský, P.: Top-k Search in Product Catalogues. Proceedings of Dateso 2011, ISBN 978-80-248-2391-1, pp. 1-12 (2011)
14. Tsaparas, P., Palpanas, T., Kotidis, Y., Koudas, N., Srivastava, D.: Ranked Join Indices. ICDE, pp.277-288 (2003)
15. Whang, K.-Y., Krishnamurthy, R.: The Multilevel Grid File - A Dynamic Hierarchical Multidimensional File Structure. Proceedings of Database Systems for Advanced Applications, ISBN 981-02-1055-8, pp. 449-459, (1992)
16. Xin, D., Han, J., Chang, K.: Progressive and Selective Merge: Computing Top-K with Ad-Hoc Ranking Functions. SIGMOD (2007)

On Indexing in Native XML Database Systems*

Pavel Loupal[1], Aleš Kantor[1], Ondřej Macek[2], and Pavel Strnad[2]

[1] Department of Software Engineering
Faculty of Information Technology, Czech Technical University in Prague
Czech Republic
loupalp@fit.cvut.cz, kantoale@fit.cvut.cz
[2] Department of Computer Science and Engineering
Faculty of Electrical Engineering, Czech Technical University in Prague
Czech Republic
macekond@fel.cvut.cz, pavel.strnad@fel.cvut.cz

Abstract. Database indices are fundamental data structures that improve the speed of data retrieval operations. In this paper, we focus on native XML database systems and provide an elementary survey of existing approaches for indexing semistructured data employed in selected academic open-source systems. Considering the requirements set for a particular system, ExDB, and the results of the accomplished research, we provide a design proposal of the indexing facility and discuss the properties of the solution we plan to subsequently realize.

1 Introduction

Native XML database management systems (NXDs) are nowadays a promising sort of document-based systems oriented on semistructured data. With the growing amount of XML data available it is essential to provide systems that can still process increased workloads efficiently. It is a fairly obvious challenge that is addressed by many research teams working on various aspects of data management. As a consequence of this situation there is a huge variety of existing algorithms and their prospective implementations in production-quality systems. To clarify the purpose and contribution of this submission let us first identify our position in this space and depict the issues we aim to address.

Our effort is driven by the endeavour to design and develop an indexing module in the ExDB system [7] that is being developed within our research group. Thus, this paper reflects the approach how to achieve this goal as a software engineering task. First, we depict here the theoretical background related to indexing (naturally only in a conceptual overview) to get acquainted with existing methods. The next step is to identify some of existing systems that might offer a useful real-world experience. The selection of presented systems we have made

* This work was partially supported by the Czech Technical University in Prague, grant no. SGS10/226/OHK3/2T/18 and by the grant project of the Czech Grant Agency (GAČR) No. GA201/09/0990.

is not random; we have decided to include those claiming to offer distinct indexing facilities and which are regarded as stable products. We have already had a positive experience with some of them from our past experiments. An additional condition was also the source code availability for potential detailed exploration.

Upon the comparison of existing open-source products we can then provide a design proposal how to construct the indexing module in ExDB according to the requirements we have set. Subsequently, we discuss potential influence of this newly built module to operation of the database system. The final evaluation of the proposal will be naturally available after the implementation and adequate benchmarking.

Related Work. There are loads of papers and books focused on database systems and on related particular problems. Here we highlight only the most important resources for us. The general theoretical foundations required for the work are sufficiently covered by well-known "database Bibles", by Date [3], and Ramakrishnan [10]. Some internals of the systems we discuss later in this paper can be found on respective project homepages (i.e., for BaseX [4], eXist [9], Sedna [1], and CellStore [11]) – either by reading the documentation provided or by accessing their source codes.

2 Native XML Database Systems

Apparently, the most natural way of storing XML documents is to employ a native XML database system (NXD). The term itself is nevertheless understood differently by various groups. For our purposes we consider the XML:DB initiative definition [13]: a NXD database utilizes an (arbitrary) logical model for an XML document, as opposed to the data in that document, and stores and retrieves documents according to that model. At a minimum, the model must include elements, attributes, PCDATA, and document order. The system then considers such XML document as its fundamental unit of (logical) storage (but, obviously, may employ an arbitrary physical storage model). To distinguish from so called XML-enabled databases, we require an NXD to be freshly grown-up upon the XML technology and not to benefit from facilities available in an existing (e.g., either relational or object) database system.

2.1 Selected Current NXDs & Feature Survey

To gain some experience with existing products we have selected few products that seem to offer appropriate and helpful view into the world of production-quality open-source systems and are initially originated in the academic environment. We try to study their internals and assess the particular findings to learn the best from it. There are two key decision points to be made in order to obtain valuable and beneficial information – *which systems* to examine and *what criteria* to consider – from such comparison.

Into the list of investigated systems we have selected those that we consider as potential competitors to our systems (CellStore, ExDB), i.e. open-source products grown-up in academic environment that are in active development and have a certain track of public releases. Hence, we have picked *BaseX*, *eXist* and *Sedna*.

To select the criteria most relevant to indexing is a more difficult problem with respect to the complexity of database management systems (in general). For our purposes we focus mainly on the following areas: supported types of indices along with their configuration options, utilized numbering schemas, involvement of available indices in query processing and space consumption (either by database or index). If any additional and beneficial properties have been identified then they are naturally included in this section, too.

BaseX [4] claims to be a light-weight, high-performance and scalable NXD. It is written completely in Java and shall be thus available on all supported platforms. The system supports XPath and XQuery query languages with almost complete coverage of the XQuery Test Suite (99.9 %). For client applications, provides the most of the APIs utilized nowadays – REST, WebDAV, XML:DB and XQJ.

The product package contains both server part and GUI client. There are two ways how to utilize the suite – either in client/server architecture (the most common deployment scenario) or (locally) as an embedded database. Undoubtedly, the supplied GUI client is the best one from all systems mentioned in this paper. It is user-friendly and offers many ways how to look on data stored at the server. Moreover, it provides also valuable statistical reports exposing interesting internals such as index configuration parameters, index size or detailed query execution plans.

Internally, the system supports a (1) structural index (Path Summary Index) and (2) value indices (text and attribute indices) and a (3) full-text index. All of these can be independently turned on/off and (3) can be moreover configured in detail. Particular employment of these indices can be tracked in execution plans for queries executed within the GUI client.

eXist [9] is another Java-based NXD that, in contrast to other products, depends heavily on several external components, e.g. from the Apache Foundation, such as Xerces and Xalan. The system supports almost all relevant query languages – XPath 2.0, XQuery 1.0 and XSLT (1.0 + 2.0). The XQuery compliance is slightly lower than for BaseX (99.4 %).

Although the documentation of the system's internals is very sparse we can observe that the vast majority of work has been done (at least in the recent time) in the field of numbering schemas and indexing concepts. According to [8] there are two node ID identification schemes implemented – Level-Order Numbering (LON) and preferred Dynamic Level Numbering (DLN). The LON uses a simple arithmetic computation to determine the relationship between two given nodes, therefore the algorithm works well for all XPath axes (on the contrary, such algorithm is not update friendly and there exists a document size limit due to

existing number of available IDs). The DLN is based on decimal classification and removes thus the disadvantages of the former one.

Using these schemas there are various (built-in or optional) indices available. The modularized design of the indexing subsystem easily allows to plug in a new index and attach it to the indexing pipeline. Supported built-in indices are basically a B+-Tree based *Structural Index* that is created by default for each element or attribute in a document and a *Range Index* (able to directly select nodes based on their typed values and applied when comparing nodes by way of standard XPath operators and functions, e.g. $=, >, <$). Pre-packaged optional indices are the *Spatial Index*, *N-Gram Index* and a *Full-text Index* (realized by the Apache Lucene engine).

Configuration of indices in eXist is accomplished by collection-specific configuration files (stored in special path with `.xconf` extension). The system does not index any element or attribute values by default therefore the configuration is needed. Subsequently all indices are automatically maintained and updated as necessary (according to all modification operations performed).

Sedna [1] is an NXD written in C++ aiming to provide "schema-based clustering storage efficient for querying and updating". This motto has been only partially confirmed by our benchmark (will be published in detail at the conference) as the storage size grew too steeply and the execution times did not overcome its competitors considerably. Such results might even re-swirl a discussion on C++ vs. Java environment efficiency.

The system is available on all major operating systems (Windows, Linux, FreeBSD, MacOS) and comprises of several command-line programs. This approach differs from all the other systems and makes the use of provided tools a bit more difficult (at least at the first glance). On the other side, there are API drivers available for a really wide variety of languages (Java, C, PHP, Python, Ruby, Perl, Delphi, C#) and XQJ and XML:DB drivers for Java. According to documentation provided the only query language supported is XQuery 1.0 (with the coverage confirmed by the XQuery Test Suite to 98.8 %).

Sedna provides two kinds of indices – value (to index XML element content and attribute values) and full-text index. In the current version, however, the query executor *does not* use these indices automatically, but it is necessary to explicitly use the respective XQuery functions `index-scan`, `index-scan-between`, and `ftindex-scan`. Value indices can be stored is either B+-tree or Block String Trie (BST). The latter option is an experimental feature that should provide more space-efficient alternative to B+-tree with the same search speed.

CellStore The main goal of the CellStore project [11] is to develop an NXD for both educational and research purposes. It is meant rather as an experimental platform than an in-box and ready-to-use database engine. We planned such an engine because the students can easily look inside it, understand and create new components for this engine as, e.g., a built-in XSLT engine, a query opti-

mizer, an index engine or an event-condition-action (ECA) processing. CellStore is developed in Smalltalk/X.

System's *architecture* is depicted in Figure 1. It can be approached through several interfaces at different levels of services. The lowest layer, low level storage, consists of several cooperating modules. Modules depicted in solid boxes are already implemented, whereas modules in dotted boxes are not ready yet.

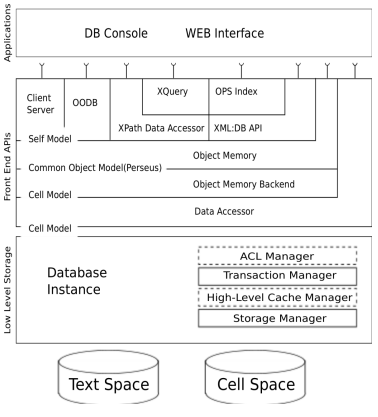


Fig. 1. *CellStore* Architecture

The system currently uses only the Path Summary Index (particular design and benchmarking depicted in [2]), which is a structural index very similar to the C–Tree index structure. As the CellStore has a specific internal structure based on the Self Model [12] the common indexing structures are not directly applicable. It is an issue to be addressed by future developments.

3 ExDB

ExDB [7] is an NXD being developed as a student research project at our university. Its primary goal is to prototype a working database environment in Java based upon the XML-λ Framework, a functional framework for XML, and thus confirm its suitability for such use case. The framework and related research activities are described in detail in [6].

Currently, it allows to persist data in either filesystem-based or native storage. XML data can be queried by XPath, XQuery and XML-λ languages. The weak point of the present solution is the non-existence of any indexing facility. Although we have investigated some potential options and proposed a solution ([5]) yet there is no indexing support available. Such shortcoming naturally prevents the system from performing efficiently for any query-based workload.

3.1 Indexing-related Requirements

The overall goal of our effort is to design and develop a configurable and extensible indexing module for the ExDB system. This module should be in charge of all documents indices and should provide access to them.

According to the previous survey we have identified fundamental requirements listed in short as follows. The design proposal should be

- *configurable* in various directions
  - enabling general setup of the indexing subsystem
  - supporting database, collection or document level configuration
  - allowing alternative physical storage structure approach – either clustered by index type or by collection hierarchy
- *extensible* for future improvements and modifications
- supporting *multiple index types*, generally both value, structure and full-text
- offering *automatic index update*
- open to *multiple query languages* – actually for XQuery and XML-λ
- helping with *query cost estimation*

Moreover, with respect to the nature of the ExDB project, we need to be able to select some of these requirements and build an prototype within a few months (till the end of current term). The remaining part is to be done subsequently.

3.2 Analysis

The requirements stated in the previous section cover a very wide range of particular sub-items that shall be analyzed in detail. In order to cover the most important ones and due to limited space available we focus here only on the major issues. Basically, with all these requirements in mind we propose a design addressing the key areas and furthermore attempt to take into account also those that remain for the future work. There are three important parts that need to be examined at first – module *configuration*, its *storage strategy* and *query interface*.

The configuration of the indexing module will obviously control its behavior and the scope of features available. Technically, there already is a configuration facility within the system and thus no additional extension is necessary. So far, we have identified about 20 parameters that could be used for setting up the module and its activities. For its length we do not publish it here in depth.

For persisting indices we need to extend current Storage module. Principally, retrieving and storing does not differ much from working with XML data and is thus not too complicated. The only doubt is the physical structure of the data – there are a few different approaches that might affect the efficiency of read/write operations – particular indices (of distinct types) can be stored in separate operating system files according to database collection hierarchy or in one "big" file all-together. One might consider also a hybrid approach when, for example, the full-text index for all documents in database is stored in one file

and all the remaining indices are stored separately and organized in collection-like directory hierarchy. Each approach has its pros and cons (chiefly clashing memory consumption with disk look-up time overhead) and we are not aware of any study with general and clear results. Thus, presuming the first approach – an individual per-document, per-index file – to be sufficiently suitable (i.e. still efficient and easily implementable) for our prototype.

Regarding the index-lookup interface we are confronted with a problem of using two implementations of distinct query language (XPath/XQuery and XML-λ) using different internal data model. There is a planned work on transforming XQuery queries into their XML-λ equivalent but a working transformation library seems to be still too distant. There are two alternatives how to face this problem – we can either speed up the development of the transformation tool or create a more general interface with adapters to both implementations. From our programmers experience we prefer to invest the time in the later option and put some additional effort into developing an adapter layer between indexing and querying modules.

3.3 Design Proposal

From the requirements and analytical notes stated above we have derived an outline of module design as shown in Figure 2. The diagram depicts the separation of the functionality into two logical parts, the first is the manager aiming to administer particular document indices. Its task is the creation, modification, deletion, storage and loading of indices. This component is connected to the system core from where the commands arrive. Moreover, upon its configuration, the component may automatically reindex or drop obsolete indices when necessary.

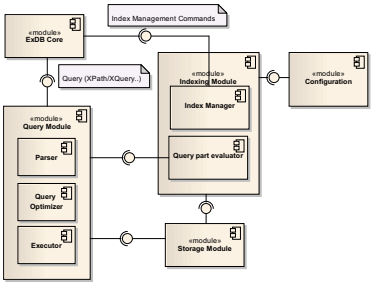


Fig. 2. Indexing Module Component Diagram

Parsing the query and consequent preparation of execution plans will be naturally the task for particular implementation of the query language. These plans are sent to the indexing module for estimating their costs. The query module then chooses the most convenient execution plan and starts to evaluate its steps it accessing either the storage or the indexing module.

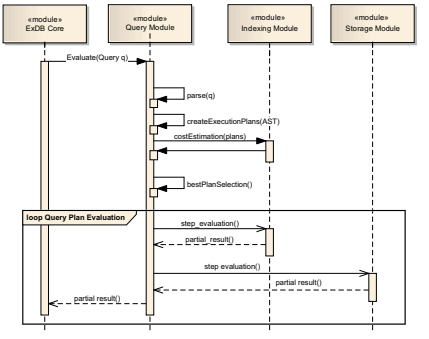


Fig. 3. Query Evaluation Process Model

4 Conclusion and Future Work

The aim of this paper was to provide an overview of existing approaches for indexing XML data available in open-source NXDs and to describe a particular design proposal for a configurable indexing module inside the ExDB system. Although we have not dipped into the problem in full detail we suppose that the text sufficiently covers the judgment of our proposal.

Our future work will include the implementation and benchmarking of the indexing module. These activities are scheduled for the following months along with implementation of a new transaction module.

References

1. K. Antipin. Sedna project homepage. `http://www.sedna.org`, 2012.
2. K. Beyr. Index implementation in CellStore project. Master's thesis, Dept. of Computer Science and Engineering, FEE CTU, Prague, 2008.
3. C. J. Date. *An Introduction to Database Systems.* Addison-Wesley, 1995.
4. C. Grün. BaseX project homepage. `http://www.basex.org`, 2012.
5. M. Janek. Indexing techniques for native XML database systems. Master's thesis, Dept. of Computer Science and Engineering, FEE CTU, Prague, 2011.
6. P. Loupal. *XML-λ : A functional framework for XML.* PhD thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, February 2010.
7. P. Loupal. ExDB project homepage. `http://exdb.fit.cvut.cz`, 2012.
8. W. Meier. Index-Driven XQuery Processing in the eXist XML Database. `http://www.xmlprague.cz/2006/slides06/meier.pdf`, 2006.
9. W. Meier. eXist project homepage. `http://exist.sourceforge.net`, 2012.
10. R. Ramakrishnan and J. Gehrke. *Database Management Systems.* McGraw-Hill Science/Engineering/Math, 3rd edition, 2002.
11. M. Valenta. CellStore project homepage. `http://swing.fit.cvut.cz/projects/cellstore`, 2012.
12. J. Vraný. CellStore - the vision of pure object database. In *DATESO*, 2006.
13. XML:DB. What is a XML database? `http://xmldb-org.sourceforge.net`, 2003.

Inter-Project Dependencies in Java Software Ecosystems

Antonín Procházka[1], Mircea Lungu[2], Karel Richta[3]

[1]Czech Technical University in Prague, [2]University of Bern,[3]Charles University in Prague

Abstract Understanding the legacy of code in a software ecosystem is critical for the organization that is the owner of the ecosystem as well as for individual developers that work on particular systems in the ecosystem. Model driven development (MDD) and model driven architecture (MDA) techniques for describing inter-project dependencies are rarely used or they're not updated by anyone during software evolution process. Describing the dependencies by hand can be painful and error prone process. Another solution is recovering the dependencies using some reverse-engineering process. There are some existing technologies today. One of them is an Ecco model of inter-project dependencies with a set of methods for recovering the dependencies from Smalltalk based software ecosystems developed by Lungu et al. Aim of our research is applying this model with its methods on Java based software ecosystem.

Keywords

Model Driven Development, Software Ecosystems, Inter-Project Dependencies, Java, Reverse Engineering

1 Introduction

Software engineering is concentrated mostly on individual projects nowadays. We've got sophisticated methods and tools for project management, version management, refactoring, testing, deployment and so on. But projects are rarely developed individually. They coexist together, evolve together and benefit from each other. We call these systems of projects *software ecosystems*. Like other terms connected to computers, the term ecosystem comes from biology. In nature we define an ecosystem as *the complex of a community of organisms and its environment functioning as an ecological unit*[1]. In the context of software engineering the ecosystem is defined as *a collection of software projects which are developed and co-evolve in the same environment [2].* Example of such software ecosystem can be a company developing software, an open-source community or a research group. As every project is located in its version control repository, we define

[1] Webster's Dictionary definition.

a *super-repositories* as *a collection of all the version. control repositories for multiple software projects [3].*

Looking at the software from a point of view of software ecosystems uncovers wide range of important information which help managers to manage their teams and projects and also help individual developers to better understand their work. Analysis of software at the abstraction level of software ecosystems can be either focused on the projects or on the developers in the ecosystem. Our work is currently focused on projects and their relationships inside a software ecosystem. We extend previous work of Lungu et al. [4] focused on recovering inter-project dependencies in Smalltalk ecosystems. In their work they argued for importance of raising abstraction of view on software products from individual projects to whole software ecosystems. They presented several viewpoints at this abstraction level including the inter-project dependency viewpoint. Each viewpoint, including this one, provides two areas of research. One is own visualization Having an interesting information is not enough - we also need to know how to present it to the user. The second area is information retrieval. Before we can present some information, we need to get it by some technique from some source. At first we focus on inter-project information retrieval from java based software ecosystems.

Structure of this paper is following: In section 2 we describe a model used to store retrieved information. Section 3 summarizes information specific about inter-project dependencies specific for Java base software ecosystems. Evaluation of different methods for dependency information retrieval is described in section 4. In section 5 we discuss contribution of this work and outline our further research to be performed on this topic.

2 Ecco model

Lungu et. al presented in their work a lightweight model describing inter-project dependencies called *Ecco*. They defined the model and filled it up with information about inter-project dependencies present in selected Smalltalk based software ecosystems.

The Ecco model consist of four main elements.

Ecosystem. In relation to the Ecco model the ecosystem means a set of software projects and dependencies between them.

Project. Every software ecosystem consists of one or more projects. Modules of each project call some methods and define another. A project can call a method which is defined in another project. Methods like this are called requirements.

Dependency. When one project require some method and another defines it, we call this relationship a dependency. The dependency consists of a client project, which requires the methods, and of a provider project, which provides the required methods. The methods making the dependency between two projects are called elements of dependency.

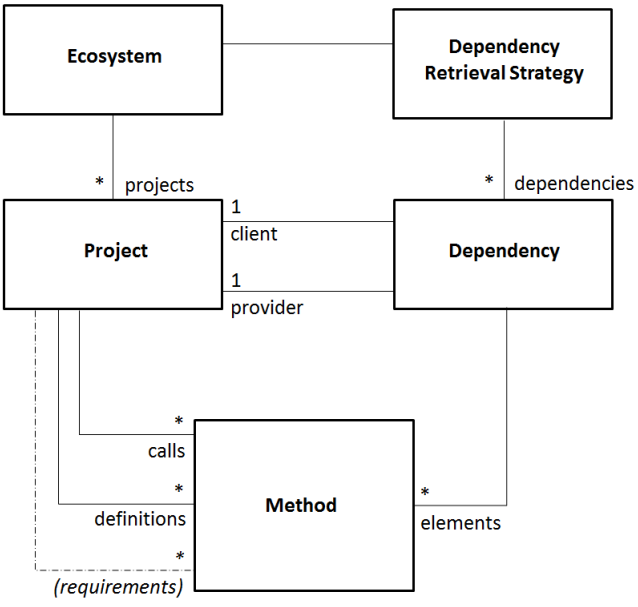


Fig. 1. Ecco is a very lightweight model aimed at extracting dependencies between projects in an ecosystem [4]

Dependency Extraction Strategy. There are several existing techniques for gathering information about inter-project dependencies and others can be defined in future. Techniques like this are called dependency extraction strategies. We include them in the model to be able to compare them during our research process.

3 Java Dependencies

In general we have two types of dependency extraction strategies. The first type reuses information existing explicitly in software super-repositories. The disadvantages of such sources are limited availability in different ecosystems and error-prone and time-wasting maintenance. On the other hand, this source is very important during research because it tells us what results to expect during evolution of the second type of dependency extraction strategies.

The second type is base on reverse-engineering of source code. In contrast to the first type, this one can be used on any kind of super-repository and doesn't need any maintenance at all. However it is harder to retrieve the information this way.

3.1 Project Object Model

If we'd like to find some reverse-engineering strategy for recovering inter-project dependencies in Java based software ecosystems, we first need to find proper source of data. We need to have a super-repository which will provide us both the explicit data and source code which we'll reverse-engineer.

Looking for such super-repository we found Apache Maven best suits our needs. Maven is a project-centric tool for software development. Its data structures contain different information about each project enabling to manage project's build, reporting and documentation. Whole Maven stands on technology called Project Object Model (POM) [1]. Every project has its own so-called POM-file, which is an XML file containing all the information relevant to this project like the developers working on it, the path of its sources, required binaries, the builder, the documentation manager, the bug tracking system and much more. It includes the explicit information about the inter-project dependencies. This information has to be compounded from four inter-project relationships described in the POM: dependencies, exclusions, inheritance and aggregation. There's also a file called Super-POM which defines value common for all project in the Maven repository unless they are redefined. A simple POM with one dependency can look like this:

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>cz.cvut.fit.swing</groupId>
  <artifactId>my-project</artifactId>
  <version>1.0</version>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.0</version>
      <type>jar</type>
      <scope>test</scope>
      <optional>true</optional>
    </dependency>
  </dependencies>
</project>
```

Dependencies. If one project depends directly on another then the information is described in a dependencies section. This section is located in POM file of the project which requires these dependencies - the Client Project from the Ecco's point of view. These dependencies can also be transitive. Transitive dependency means that if a client project A requires a project B which requires a provider project C, C becomes common requirement for both A and B. Dependencies here are divided into 5 scopes:

A Compile Scope is a default scope representing group of regular projects which are available with their source code and are necessary for successful build of a Client Project. The Compile Scope dependencies *are transitive*.

A Provided Scope represents a group of precompiled projects expected to be given at compile time by Software Development Kit (SDK), container or another way. The Provided Scope dependencies *are not transitive*.

A Runtime Scope is much like the Provided Scope but represents projects expected to be given at runtime. The Runtime Scope dependencies *are not transitive* as well.

A Test Scope is like the Compile Scope but represents projects needed for testing purposes. The Test Scope dependencies *are transitive* as well as the Runtime Scope.

A System Scope is similar to the Provided Scope but requires a developer to provide its dependencies explicitly. The System Scope dependencies *are not transitive* as well as the Provided Scope.

As we'll be examining only projects contained in a given ecosystem, we are interested only in the Compile Scope dependencies. Possibly we can be also interested in the Test Scope dependencies if we'll extend our analysis to project's used for testing purposes.

Exclusions. Transitive dependencies can produce unwanted behavior. If a developer needs to exclude some project from the dependency list she includes it into the exclusions section of the dependency which causes the problem. The meaning of the exclusions during populating the Ecco model is obvious. We should respect these exclusions and throw away dependencies excluded by them.

Inheritance. The Project Object Model brings a feature which enables us to make an inheritance tree of projects. From the view of POM this means that if we define something in an ancestor project's POM file, all its child project inherit these definitions unless they are redefined in a child project's POM files. There are two points important for us. First, the inheritance relationship itself represents a dependency and we have to to think about it this way. Second, dependencies of ancestor client projects become dependencies of child client projects since these two projects are in inheritance relationship.

Aggregation. If a project is made of a modules, Maven thinks about the modules as about separated projects which are aggregated into another project called multi-module project. This relationship is described in the multi-module project's POM file in a modules section. As the modules are expected to belong to the same group as their multi-module project, they are defined only by their project names. From our point of view, the aggregation relationship represents another way to express the inter-project dependencies between the modules and the multi-module project.

3.2 Java Bytecode

When we think about a reverse-engineering of a Java software, we are not limited only to a Java language. We can think of any language which can be compiled to a Java Bytecode. The original information can be simply disassembled from the byte-code [6]. Consider this simple class definition written in the Java language:

```
import java.awt.*;
import java.applet.*;

public class DocFooter extends Applet {
  String date;
  String email;

  public void init() {
    resize(500,100);
    date = getParameter("LAST_UPDATED");
    email = getParameter("EMAIL");
  }

  public void paint(Graphics g) {
    g.drawString(date + " by ",100, 15);
    g.drawString(email,290,15);
  }
}
```

If we call javap DocFooter to disassemble a DocFooter.class, we get this output:

```
Compiled from DocFooter.java
public class DocFooter
    extends java.applet.Applet {
  java.lang.String date;
  java.lang.String email;
  public DocFooter();
  public void init();
  public void paint(java.awt.Graphics);
}
```

Passing some arguments will give us also a disassembly of a behavior, but this interface declaration is all what we need. We've got fully qualified name of every class and method used in the compiled code.

This is how our reverse-engineering dependency extraction strategies will look like. At first we take a Java Archive. Every java project is distributed as a Java Archive. The archive is a regular compressed package of data containing a Class Files. Every Class File contains a byte-code of one Java class. We open the archive, disassemble every class file and see which methods are called and which are defined. We fill this information into the Ecco model. Information

gathered this way needs some more processing before we'll get reliable result. This post-processing is topic of our further research.

4 Evaluation of Results

To let us compare different inter-project dependency retrieval techniques we need to have a measuring method to let us assign a value to each technique. For this purpose we'll use well-known information retrieval metrics - *a precision*, *a recall* and *an F-measure* [5] adopted for our case by Lungu et al. [4]. To use them we first need a "golden standard" or an oracle. This is the information we retrieve from Maven's POM. Thanks to this information we are able to distinguish *a Relevant* dependencies which are present in the oracle and *a Nonrelevant* which are not present in the oracle. Besides this we can divide the dependencies to those which *have* or *have not been* retrieved by a concrete reverse-engineering technique. In common we get four different statistical sets of dependencies which can be seen in table 1.

Table 1. Statistical sets of retrieved inter-project dependencies [5]

	Relevant ($TP \cup FN$)	Nonrelevant ($FP \cup TN$)
Retrieved ($TP \cup FP$)	True Positives (TP)	False Positives (FP)
Not Retrieved ($FN \cup TN$)	False Negatives (FN)	True Negatives (TN)

The metrics are then defined as follows. The Precision (P) is a fraction of retrieved dependencies that are relevant. The Recall (R) is a fraction of relevant documents that are retrieved. The F-measure (F) is the weighted harmonic mean of precision and recall. The F-measure represents a single measure that trades off the precision versus the recall and thus indicates an overall accuracy of the measured technique.

$$P = \frac{|TP|}{|TP \cup FP|} \qquad R = \frac{|TP|}{|TP \cup FN|} \qquad F_1 = \frac{2PR}{P+R}$$

We use a default balance F-measure (F_1) which equally weights the precision and the recall because we don't want to emphasize the recall nor the precision.

During evaluation of our reverse-engineering techniques we'll calculate these values for each technique and compare them. This comparison will give us the required information about the technique's effectivity.

5 Conclusion

The information summarized in this paper gives us excellent base for our further research aimed on different reverse-engineering techniques for retrieval of inter-project dependencies in the Java based software ecosystems. We have an excellent source of data which will help us with a development of the techniques. Using the explicitly given information about the dependencies and using the mentioned metrics we are able to compare every techniques and tell which one better suits our needs. We found a way which lets us to retrieve the dependencies from any language which can be compiled to the Java byte-code. In connection with the work done by Lungu et al. on the Smalltalk based software ecosystem we'll be also able to summarize differences between a dependency retrieval from statically and dynamically typed languages.

6 Acknowledgments

We would like to thank for financial support of Student Grant Competition of CTU in Prague, grant number SGS12/093/OHK3/1T/18.

References

1. Apache. Maven project, 2002.
2. Lungu, M. *Reverse Engineering Software Ecosystems.* PhD thesis, University of Lugano, 2009.
3. Lungu, M., Lanza, M., Girba, T., and Heeck, R. Reverse engineering super-repositories. In *Proceedings of the 14th Working Conference on Reverse Engineering* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 120–129.
4. Lungu, M., Robbes, R., and Lanza, M. Recovering inter-project dependencies in software ecosystems. In *Proceedings of the IEEE/ACM international conference on Automated software engineering* (New York, NY, USA, 2010), ASE '10, ACM, pp. 309–312. ACM ID: 1859058.
5. Manning, C., Raghavan, P., and Schtze, H. *Introduction to Information Retrieval.* Cambridge University Press New York, NY, USA, 2008.
6. Oracle. Java se documentation, February 2010.

On Distributed Querying of Linked Data*

Martin Svoboda, Jakub Stárka, and Irena Mlýnková

XML and Web Engineering Research Group
Faculty of Mathematics and Physics, Charles University in Prague
Malostranske namesti 25, 118 00 Prague 1, Czech Republic
{svoboda,starka,mlynkova}@ksi.mff.cuni.cz

Abstract. The concept of Linked Data has appeared recently in order to allow publishing data on the Web in a more suitable form enabling automated processing by programs and not only by human users. Linked Data are based primarily on RDF triples, which are also modeled as graph data. Despite the research effort in recent years, several questions in the area of Linked Data indexing and querying remain open, not only since the amount of Linked Data globally available significantly increases each year. Our ongoing research effort should result in a proposal of a new querying system dealing with several disadvantages of the existing approaches identified in our previous work. They are especially related to data scaling, dynamicity and distribution.

Keywords: Linked Data, RDF, indexing, querying, SPARQL.

1 Introduction

The concept of Linked Data [3] appeared in order to extend the Web of Documents towards the Web of Data. And the reason is simple – it is often not feasible to retrieve potentially structured information from traditional documents based on HTML [12] formats tailored for users and not programs.

Linked Data do not represent any particular standard; we only talk about a set of recommended principles and techniques, which lead to the publication of data in a way more suitable for their automated processing. First, each real-world entity should be described by a unique URL identifier. These identifiers can be dereferenced by HTTP to obtain information about the given entities. And, finally, these entity representations should be interlinked together to form a global open data cloud – the Web of Data.

A particular way to follow these principles is to use the RDF (Resource Description Framework) [7] standard, where data are modeled as triples conforming to the concept of *subject-predicate-object*. An alternative means to view these triples are graphs, where vertices correspond to *subjects* and *objects*, edges represent the triples themselves and are labeled by *predicates*. At the implementation level, we can publish RDF triples in a form of RDF/XML [2] syntax and

* This work was supported by the Charles University Grant Agency grant 4105/2011 and the Czech Science Foundation grant P202/10/0573.

along the data we can also publish RDFS [4] schemata or OWL [8] ontologies restraining the allowed content of such RDF data.

In recent years, a significant effort appeared not only in a theoretical research, but also in the amount of Linked Data globally available. However, we can still identify several open problems to which attention should be paid. The goal of our ongoing research effort is to propose a new querying system for Linked Data. In particular, we want to focus on indexing structures and techniques with respect to SPARQL [10], probably the most used querying language for RDF data.

The aim of this paper is to provide a description of the system we are attempting to propose. However, in order to understand our motivation, we also need to discuss the existing approaches from the area of Linked Data indexing and querying. Their thorough overview was presented in our previous work [15]. Although these approaches represent efficient systems (or at least promising interesting proposals), when we focus on large amounts of dynamic and distributed data concurrently, these approaches start showing their bottlenecks.

Preliminary ideas of our querying system were first introduced in [14]. Now, we will discuss main aspects and issues of the architecture in more detail. They are especially related to components for managing sources, distributed databases, storages for data triples and auxiliary indexing structures. Index structures in fact represent one of the crucial parts of our work, since the majority of existing methods does not assume dynamic data. When processing queries, we need to find suitable query evaluation plans, which involves the source selection and a set of optimization strategies.

Outline. In Section 2 we present a basic overview of the existing approaches. Section 3 provides the description of the architecture of system we are working on. Finally, Section 4 concludes.

2 Related Work

The existing approaches can probably be divided into three main categories: *local* querying systems, *distributed* querying systems and *global* searching engines. It is worth noting that even though we want to focus on distributed querying, its models and algorithms, wide range of relevant ideas can be found between approaches for local querying. For simplification, we will use abbreviations S, P, O and C for *subject*, *predicate*, *object* and *context* respectively.

We start our overview of existing approaches by local querying systems. Index structures proposed by Harth and Decker [5] enable querying of local data quads with context. These structures involve Lexicon (an inverted list for keywords and two-way translation maps for term identifiers based on B^+-trees) and Quad indices (B^+-trees for $SPOC$, POC, OCS, CSP, CP, OS orderings) allowing to query in all possible 16 access patterns. Despite data quads themselves, these indices also contain statistics about data.

The core of the stream processor RDF-X by Neumann and Weikum [9] is based on six B^+-tree indices for all SPO, SOP, OSP, OPS, PSO and POS access patterns. Additionally, they also use indices with statistics (S, P, O, SP,

PS, PO, OP, SO and OS projections) and selectivity histograms and statistics for pre-computed path or star patterns. Next, the idea of HexaStore approach by Weiss et al. [18] is based on similar SPO, SOP, OSP, OPS, PSO and POS index structures, however, these are implemented as ordered nested lists. All these lists contain only identifiers instead of strings, again.

BitMat is an approach proposed by Atre et al. [1]. Its index model is based on a matrix with three dimensions for S, P and O values (terms are translated to identifiers, which are used as matrix indices). Each cell contains a bit value equal to 1 if and only if the given triple is stored in the database, otherwise value 0. The index is organized as an ordinary file with all SO, OS, PO and PS slices stored using a bit run compression over individual slice rows.

Udrea et al. [17] introduced a model based on splitting data graphs into subgraph areas that are described by conditions limiting their content. The idea is derived from a metric defined on URIs and literals (e.g. a minimal number of edges in a data graph between a given pair of values). The index structure itself is a balanced binary tree, where internal nodes represent mentioned areas and leaf nodes store data triples conforming to these areas.

The last presented local approach is a parameterized index introduced by Tran and Ladwig [16]. Their model is based on bisimilarity relations, putting in a relation such two vertices of the data graph that share the same outgoing and ingoing edges (reflecting only predicates). Vertices from the same equivalence class have the same characteristics and, therefore, prompted queries can first be evaluated over these classes to prune required data.

Now, we move to distributed approaches. Quilitz and Leser [11] proposed a system for integrated querying over distributed and autonomous sources. The core of this approach is a language for description of distributed sources, in particular, data triples they contain, together with other source characteristics.

The purpose of a data summary index by Harth et al. [6] is to enable the source selection over distributed data sources. Data triples are modeled as points in a 3-dimensional space (S, P, and O coordinates are derived by hash functions). The index structure is a QTree based on standard R-Trees. Internal nodes act as minimal bounding boxes for nested nodes, leaf nodes contain statistics about data sources, not data triples themselves.

3 Framework

The system should provide transparent querying of distributed data – not in the context of the entire Web of Data, but only within a distributed database over which we have a full control. Linked Data are the subject of nontrivial changes in time and, thus, the aspect of the data volatility cannot be ignored in the framework architecture and index structures especially. Many existing approaches bring interesting ideas, but their indexing models only assume environments with static data. Therefore, the core part of our work is to propose an appropriate dynamic index structure.

3.1 Sources and Databases

The nature of Linked Data assumes that data are distributed within the entire global cloud of the Web of Data. Since completely centralized solutions seem not to precisely follow this idea, we want to find a suitable compromise between centralized and totally distributed approaches. For this purpose we can accept an idea that a distributed database is spread across a set of sources, as we can see in Figure 1 with a sample distributed infrastructure. Each source provides two main features – it is able to store data triples inside its local storages and provides interfaces for querying.

These sources can be viewed only as ordinary services, but we have the full control over sources we want to use in our database – either we own them completely (and decide what data they should store), or, at least, we can decide what independent sources we would like to use (and we accept data they provide). Anyway, submitting a query to a public interface of a particular source, it should transparently decompose the query into its elementary parts, decide which sources should be contacted to obtain relevant data, and, finally, to compose the entire query result. In other words, the user should define data to be used (by building its distributed database), but the query itself should be evaluated automatically without his or her explicit help.

For this purpose, we first need to have a technique for describing capabilities of individual sources. A promising concept was already introduced by Quilitz and Leser [11], as we already noted in the previous section. Anyway, we must be able to clearly describe data the given source contains. This can be achieved by a set of various conditions on triples and their S, P and O components. However, this is not an easy task, since descriptions must be as accurate as possible. But on the other hand, too complicated and big descriptions would be useless as well. Moreover, if we assume data dynamicity and query evaluation, we also need to publish statistics about given data, their versions or availability. And still we cannot end, because if we recall the second purpose of each source, we must also capture other issues of the query evaluation process. For example, if two different sources contain the same data, it would be worth to know which of them has better assumptions to execute the evaluation more efficiently.

Having defined the way how sources publish information about data they contain, we need to manage sources themselves. So, assume that we have the knowledge of sources we want to use, their locations or other technical details. Now, we must define, which sources (and, in particular, which data) constitute our database. This management seems to be easy, but cannot be omitted.

3.2 Storages and Indices

Data triples are stored in physical storages. Their role, however, might be a bit different comparing to traditional relational databases or others. The model of RDF triples is so simple that we can store data directly in indices, but as we will see, this still does not mean that physical storages should not be included in the architecture of querying systems. Although triples really are easy to grab,

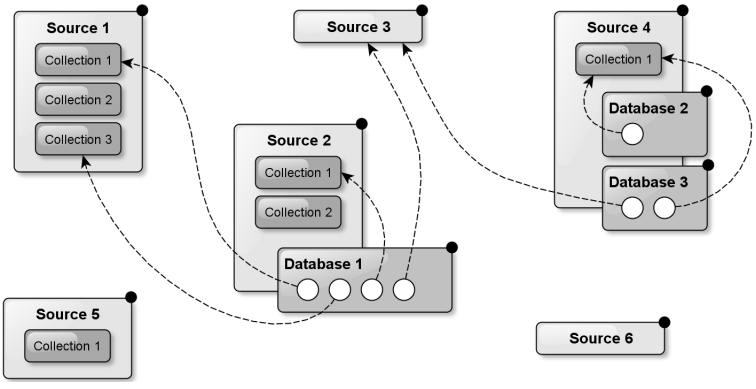


Fig. 1. Sample infrastructure with sources and distributed databases

it would be misleading to think that we do not need to handle different data differently. Relational databases allow users to create schemata and explicitly declare how their data should be stored in relational tables. However, we are not offered similar features in existing native approaches for RDF data.

Therefore, we assume that a storage is a component for storing RDF triples, but we do not have any further assumptions on their internal structure or characteristics. The only important is to comply with an agreed public interface. As a consequence, we can work with native storages, we can create wrappers around relational databases, or even to access remote storages via network. In other words, it would be interesting to access local storages within a particular source with the same (or at least very similar) interfaces as we would use between distributed sources during the query evaluation phase. In fact, we indeed need to achieve this behavior, since if we formulate a query on a given source against a particular database, apparently, some data may be available locally and other not – but from the point of the query evaluation process, both these data play the same role. A sample storages configuration can be seen in Figure 2.

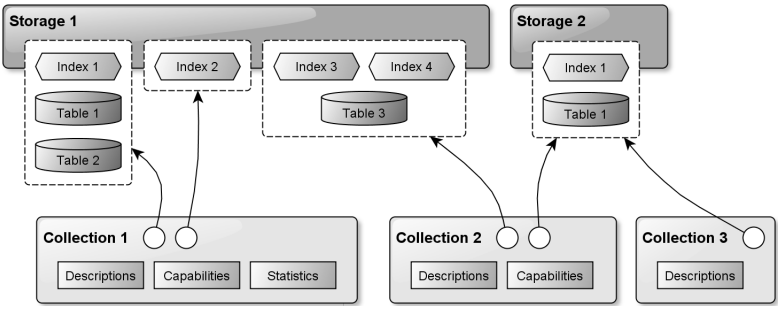


Fig. 2. Sample collections and storages composition

The shared idea by the majority of indexing methods is the way of storing string values of URIs and literals, because there is a high probability that strings

(or substrings) may have multiple occurrences in the database. Therefore, it is very effective to store these strings only once in a special storage, assign them unique integer identifiers, and use them in RDF triples instead of the original terms. As a consequence, frequently executed value equality tests during the query evaluation may then be executed much faster and the space required for storages decreases as well.

Having a particular domain of our problem, we should know at least something about data and even queries, and, thus, to design our database effectively. We do the same years and years in relational databases, so we should be able to select appropriate storages and indexing structures directly for our situation in RDF approaches, too. This functionality should be one of the core parts of our system. When storing data in a local storage of a source within a given database, users should be encouraged to choose from a palette of implemented approaches, best conforming to their situation.

The main disadvantage of the majority of interesting models for indexing RDF data is the static nature of indexing structures themselves. We do not enable working with extremely volatile data, but it is not a good idea to strictly assume only a static database. Therefore, one of our main goals is to extended some existing approach [1, 13] towards support for adding, modifying and removing data from storages and associated indices. Another problem could represent the necessity to heuristically configure indices. For example, in the index structure proposed by Tran and Ladwig [16], we need to define sets of predicates that are used for restricting ingoing and outgoing edges from vertices when constructing equivalence classes based on a relation on vertices. Unfortunately, this configuration may not always be feasible easily.

3.3 Queries

We have chosen SPARQL as a querying language. Having a query statement, we first need to parse it into an internal representation of a graph pattern. For simplicity, we can assume that this pattern is built from a set of triples, where we can use variables instead of only fixed terms. They may serve for joining individual patterns together, or may only state that we do not care about values of a corresponding triple particle. The latter purpose in fact corresponds to the idea of joining – if we first evaluate each pattern separately, then we need to join intermediate results together – joining only those pairs of triples that have equal values at positions of corresponding variables.

The problem is that our database is distributed between several sources. We have descriptions of data these sources provide, thus, we need to decide for each pattern (or even more complicated subqueries), where relevant data are located. This problem is referred as a source selection. Whereas data summary index by Harth et al. [6] works with detected summaries about data, we would like to rely on discussed descriptions. However, relevant data can be split between sources in different ways, or they can even partially or fully overlap. Therefore, this source selection is not always simple and we must take care what sources we want to access. We can see a sample query evaluation in Figure 3.

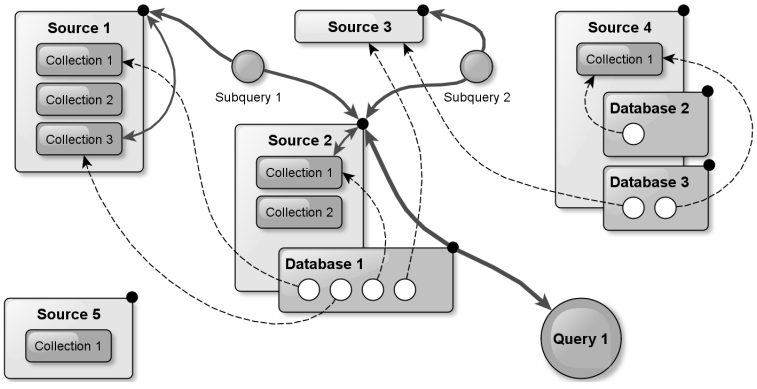


Fig. 3. Distributed evaluation process of a sample query

When we have decided which sources contain relevant data, we are still not finished with preparing the query evaluation plan. Data in sources are physically maintained in storages and these storages may have different capabilities to return required data. Thus, during the source selection, we also need to consider these capabilities. And moreover, different indices may be available, too.

If we want to find a suitable query evaluation plan, which is a must, since the complexity of different plans may be significantly different, we have to consider all the following aspects: different sources, storages, indices and different algorithms available for executing operations. The theoretical goal could be to find the optimal plan, but in practice we must settle only for approximations. Usually, we cannot inspect all possible plans, so we have to use only suitable heuristics. Another problem is that we are often forced to use incomplete and only approximate statistics.

The general idea of all optimizations is to avoid processing of irrelevant data wherever possible and to perform all computations effectively. If the existing approaches are not able to directly access only the required data, they at least attempt to prune data using other methods or ideas. For example, we can move data filtering selections as close as possible to their fetching, or we can perform data pruning before the phase of joining. Probably the most important position in query optimization techniques has the join ordering. It is quite interesting that we can use similar ideas to the nested loop algorithm from relational databases. However, there are other aspects that need to be considered, too.

4 Conclusion

In this paper we described the architecture of the querying system we are proposing. Issues related to this architecture can be divided into two groups: data and queries. First, we discussed observations and ideas related to the model of a distributed database spread across a set of selected sources, motivation and features of physical storages for RDF triples and indexing structures supporting the query evaluation process. Finally, this process must discuss methods for finding

optimal query evaluation plans, selection of relevant distributed sources and a set of optimization techniques, too.

Although the existing solutions already focus on the same area, these approaches do not target at all three main open challenges concurrently – data scaling, distribution and dynamicity.

References

1. Atre, M., Chaoji, V., Zaki, M.J., Hendler, J.A.: Matrix "Bit" loaded: A Scalable Lightweight Join Query Processor for RDF Data. In: Proceedings of the 19th Int. Conf. on World Wide Web. pp. 41–50. WWW '10, ACM, NY, USA (2010)
2. Beckett, D.: RDF/XML Syntax Specification (Revised) (2004), http://www.w3.org/TR/rdf-syntax-grammar/
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story so far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009)
4. Brickley, D., Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema (2004), http://www.w3.org/TR/rdf-schema/
5. Harth, A., Decker, S.: Optimized Index Structures for Querying RDF from the Web. In: Third Latin American Web Congress, 2005. LA-WEB 2005. IEEE (2005)
6. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.U., Umbrich, J.: Data Summaries for On-demand Queries over Linked Data. In: Proceedings of the 19th Int. Conf. on World Wide Web. pp. 411–420. WWW '10, ACM, NY, USA (2010)
7. Manola, F., Miller, E.: RDF Primer (2004), http://www.w3.org/TR/rdf-primer/
8. McGuinness, D.L., Harmelen, F.v.: OWL Web Ontology Language: Overview (2004), http://www.w3.org/TR/owl-features/
9. Neumann, T., Weikum, G.: RDF-3X: A RISC-style Engine for RDF. Proc. VLDB Endow. 1, 647–659 (August 2008)
10. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF (2008), http://www.w3.org/TR/rdf-sparql-query/
11. Quilitz, B., Leser, U.: Querying Distributed RDF Data Sources with SPARQL. In: The Semantic Web: Research and Applications. Lecture Notes in Computer Science, vol. 5021, pp. 524–538. Springer Berlin / Heidelberg (2008)
12. Raggett, D., Hors, A.L., Jacobs, I.: HTML 4.01 Specification (1999), http://www.w3.org/TR/html401/
13. Stuckenschmidt, H., Vdovjak, R., Houben, G.J., Broekstra, J.: Index Structures and Algorithms for Querying Distributed RDF Repositories. In: Proc. of the 13th Int. Conf. on World Wide Web. pp. 631–639. WWW '04, ACM, NY, USA (2004)
14. Svoboda, M., Mlynkova, I.: Efficient Querying of Distributed Linked Data. In: Proceedings of the 2011 Joint EDBT/ICDT Ph.D. Workshop. pp. 45–50. PhD '11, ACM, New York, NY, USA (2011)
15. Svoboda, M., Mlynkova, I.: Linked Data Indexing Methods: A Survey. In: On the Move to Meaningful Internet Systems: OTM 2011 Workshops. pp. 474–483. Springer (2011)
16. Tran, T., Ladwig, G.: Structure Index for RDF Data. In: Workshop on Semantic Data Management (SemData@VLDB) 2010 (2010)
17. Udrea, O., Pugliese, A., Subrahmanian, V.S.: GRIN: A Graph Based RDF Index. In: Proceedings of the 22nd National Conference on Artificial Intelligence – Volume 2. pp. 1465–1470. AAAI Press (2007)
18. Weiss, C., Karras, P., Bernstein, A.: Hexastore: Sextuple Indexing for Semantic Web Data Management. Proc. VLDB Endow. 1, 1008–1019 (August 2008)

Social Network Analysis: Selected Methods and Applications

Przemyslaw Kazienko

Instytut Informatyki (I-32), Wydział Informatyki i Zarzadzania (W-8), Politechnika Wrocławska (PWr)
Przemyslaw.Kazienko@pwr.wroc.pl

Abstract. A social network (SN) is a network containing nodes – social entities (people or groups of people) and links between these nodes. Social networks are examples of more general concept of complex networks and SNs are usually free-scale and have power distribution of node degree. Overall, several types of social networks can be enumerated: (i) simple SNs, (ii) multi-layered SNs (with many links between a pair of nodes), (iii) bipartite or multi-modal, heterogeneous SNs (with two or many different types of nodes), (iv) multidimensional SNs (reflecting the data warehousing multidimensional modelling concept), and some more specific like (v) temporal SNs, (vi) large scale SNs, and (vii) virtual SNs. For all these social networks suitable analytical methods may be applied commonly called social network analysis (SNA). They cover in particular: appropriate structural measures, efficient algorithms for their calculation, statistics and data mining methods, e.g. extraction of social communities (clustering). Some types of social networks have their own measures and methods developed. Several real application domains of SNA may be distinguished: classification of nodes for the purpose of marketing, evaluation of organizational structure versus communication structures in companies, recommender systems for hidden knowledge acquisition and for user support in web 2.0, analysis of social groups on web forums and prediction of their evolution. The above SNA methods and applications will be discussed in some details.

© J. Pokorný, V. Snášel, K. Richta (Eds.): Dateso 2012, pp. 151–151, ISBN 978-80-7378-171-2.
Charles University in Prague, MFF, Department of Software Engineering, 2012.

Author Index