Information Systems & Grid Technologies

Ninth International Conference ISGT'2015

Sofia, Bulgaria, May, 29 – 30., 2015.

ISGT'2015 Conference Committees

Chair

Prof Vladimir DIMITROV

Program Committee

- Míchéal Mac an AIRCHINNIGH, Trinity College, University of Dublin
- Pavel AZALOV, Pennsylvania State University
- Irena BOJANOVA, University of Maryland University College
- Marco DE MARCO, Catholic University of Milan
- Milena DOBREVA, University of Malta
- Vladimir GETOV, University of Westminster
- Seifedine KADRY, American University of the Middle East, Kuwait
- Kalinka KALOYANOVA, University of Sofia "St Cl Ochridsky"
- Angelika KOKKINAKI, University of Nicosia
- Violeta MANEVSKA, University of Bitola "St Cl Ochridsky"
- Maria NISHEVA, University of Sofia "St Cl Ochridsky"
- Dov TE'ENI, Tel-Aviv University
- Stanislaw WRYCZA, University of Gdansk
- Fani ZLATAROVA, Elizabethtown College

Organizing Committee

- Vasil GEORGIEV
- Maria KOLEVA

Vladimir Dimitrov, Vasil Georgiev (Editors)

Information Systems & Grid Technologies

Ninth International Conference ISGT'2015

Sofia, Bulgaria, May, 29 – 30., 2015.

Proceedings

organized by

Faculty on Mathematics and Informatics.
University of Sofia St. Kliment Ohridski

Bulgarian Chapter of the
Association for Information Systems (BulAIS)

St. Kliment Ohridski University Press

Preface

This conference was being held for the ninth time in the end of May, 2015 in the Rector's meeting hall of the University of Sofia "St. Kliment Ohridski, Bulgaria. It is supported by the Science Fund of the University of Sofia "St. Kliment Ohridski" and by the Bulgarian Chapter of the Association for Information Systems (BulAIS).

Total number of papers submitted for participation in ISGT'2015 was 21. They undergo the due selection by at least two members of the Program Committee. This book comprises 18 papers of 17 Bulgarian and 9 foreign authors. The conference papers are available also on the ISGT web page http://isgt.fmi.uni-sofia.bg/ (under «Former ISGTs» tab).

The editors

ISSN 1314-4855
St. Kliment Ohridski University Press

TABLE OF CONTENTS

Web Application as a Tool for Enhancing Clientelism in the Public Sector

Snezana Savoska[1], Branko Dimeski[2], Dragana Anceva[1]

[1]Faculty of information and communication technology, Bitolska bb,
University „St.Kliment Ohridski“ – Bitola,7000Bitola,
[2]Faculty of Law, Kicevo,University „St.Kliment Ohridski“ – Bitola, 7000 Bitola,
R.of Macedonia,

savoskasnezana@gmail.com, branko_dim@yahoo.com, dragana_anceva@hotmail.com

Abstract. In recent years, the whole activities of people and businesses are on the Internet and can be done from anywhere and timely by using web applications. For web applications development, many tools and open source platforms can be used, such as: LAMP helped by WordPress, Joomla, Drupal, XOOPS, and Alfresco etc. However, it should be mentioned that Microsoft VisualStudio.NET is also an excellent solution for web applications development. The paper will present an integration of three specific areas: public relations, clientelism as administrative modernization and undertaken activities for the needs of the local community and programming of Web application. Their symbiosis will produce socially useful tool that will be designed to improve the public sector by applying the relations clientelism, service by issuing preferential tickets to students and raising customer satisfaction. The design of web application with VisualStudio.NET will help in solving the above mentioned issues in the public sector.

Keywords: Web applications, Public services, Clientelism, ASP.NET, ADO.NET, IIS, Public Transport Enterprise.

1. Introduction

The “citizen-center oriented” concept influenced public administration institutions in improving the operations of the public sector. The relationships between institutions and citizens sometimes are not regulated by obligatory relations, but rather, they are regulated with contracts for reciprocal exchange of goods and services based on moral obligations, known as clientelism or “democratic patronage” which supports administrative modernization projects that mostly address the needs of local governments. In this paper the term is used to show the improvement of the process of issuing high school tickets (HST) by the Public Transport Enterprise (PTE) using the electronic connection between the institution and the public schools which have to input record for the students who should receive high school tickets.

There isn’t efficient way to improve the efficiency, effectiveness and control

of the processes than introduction of information technologies (IT) and systems in the operating of companies and institutions. Web applications that appear every day on the web spaces perform a myriad of tasks in order to increase the availability of services from companies via institutions to citizens.

With a well-designed Web application some improvement of performances in the public sector can be achieved as well as customer relationships enhancement and better processes in the overall society integration. Analysis of the needs and requirements of users for the service are made for the process of issuing tickets (HST) by the PTE. In addition, in order to improve the efficiency and effectiveness of employees in the PTE Skopje in issuing HST, a web application is developed. Web application is available via Internet and allows monitoring of the work through the company network or from any location that has Internet access.

Web application can allow PTE employees to increase their efficiency in part, with obtaining data for high school students. By using such data, the employees of PTE will be able to create and issue HST tickets. The use of the application will allow facilitating of employees' work and will significantly shorten the time needed for finishing the job and will reduce the documents required from students in the process of purchasing tickets. In doing that, schools staff should input data for HST, which is not obligatory commitment, but socially useful.

In creating of web application ASP.NET tool and other parts of MS VisualStudio.NET (C# and ADO.NET) are used. The paper describes how the Web application is created, the need for creating this application, and the analysis and design of the proposed solution based on information needs of the application users. Joint operation of different platforms is imperative in distributed applications. It was therefore necessary to ensure standardization in data exchange. So .NET platform is built on XML technology standards and SOAP (Simple Object Access Protocol). Also, work with this kind of application implies the existence of the .NET Framework that contains CLR (Common Language Runtime) and collections of classes in this environment. These features provide creating distributed applications whose units can operate with independent platforms, even written in different programming languages. All languages covered by the new package of Microsoft Visual Studio.NET have support for .NET classes [8].

2. The concept of "clientelism"

The concept of "clientelism" is defined as a relationship between individuals with unequal economic and social status ("boss" and his "customers"), which involves a reciprocal exchange of goods and services on a personal basis that is generally regarded as a moral obligation [11, 12]. Defined in this way,

clientelism means moral obligation "stronger" to help "weaker" on a moral basis. It is also known as "democratic patronage" or way of supporting administrative modernization projects, which primarily address the needs of local governments. However, the expansion of interventions from various states and local governments generate new opportunities for politicians to be able to control public resources and thereby mobilize electoral support. Social policies, urban renewal and economic development subsidies could be used to strengthen these "political machines". Some political scientists have gone so far that have started using the term "clientelistic" and ranked the state as a political system in which the dominant party takes the bureaucracy, collective goods and their distribution. Yet, so far, clientelistic' practices despite generally "moral obligation or democratic auspices which supports administrative modernization projects that primarily address the needs of local governments" can be viewed either as a relic of the past and tradition, as a sign of improper functioning of democracy, anomaly of political systems caused by the lack of civic culture or "capture" of institutions by politicians interested only to preserve their power [13]. Thus, the clientelism, despite the democratic modernization and support of the central authorities to local communities, is perceived as a political "pathology" that blocks democracy and wastes public goods.

Web application designed for PTE is a powerful tool for communicating with existing and potential customers. Through its proper functioning, employees and customers are constantly in contact by mutual flow of useful information.

3. Public sector problems in operating with clients and possible IT solutions

Part of the research is focused on the fact how web applications contribute in solving problems in the public sector. With empirical research and interviews with the users, it was noticed a lack of web applications to enhance public sector performance. The emphasis is placed on customer relations and how the implementation of Web application affects clientelism and improves the quality of public services through achieving greater efficiency and effectiveness in the public sector by using ICT. A concrete example on how to overcome the problems relates to the problem of lack of data for students in the City of Skopje who are users of public transport and to enable easier and more effective, more accurate and faster way in getting tickets for students by using web application.

By using Web application, faster information that is needed for issuing HST for students by the PTE is enabled. The idea of creating this application arose from the need of facilitating the process of obtaining information for high school students, easing employees in PTE work and obtaining data for students from their schools faster manner. Data on students should be input in the database

by the persons in charge of high schools, thereby reducing the time needed to obtain that information. In fact - getting "zero" delay data and avoiding additional procedures for issuing certificates to students by schools, avoiding misuse and other anomalies are just some of the benefits arising from the use of web application. Through the usage of application, employees from schools that have user-code and password can input and update data in order to create openwork database that can issue HST for students by the debtor in PTE.

4. Platform and software in the problem's context

Almost every application relies on a wide range of software as operating systems, database management software and sometimes public cloud software. Regarding the fact in which part works, this software apply application platform which play fundamental role in modern computing environments [8]. Applications and data which are used to provide all values that IT brings depend on application platform. Modern application platforms are not simply. Figure 1 shows layered software structure, which gives a general picture of modern application platform. The components of the software support which are used in the web application for HST creating are also explained.

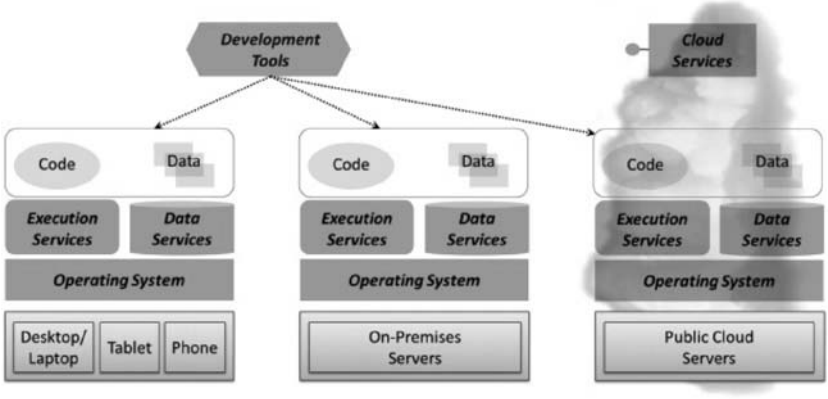


Figure 1 – Modern web application platform [8]

As shown in the Fig. 1, the modern application platform is broad and supports all types of applications. This includes individual customers, distributed applications and applications using cloud services. .NET Platform includes a large library of classes Framework Class Library (FCL) and provides inter-operability. Programs written for the .NET platform are executed in a software environment, the Common Language Runtime (CLR), and application virtual machine that provide services such as security and memory management. FCL and the CLR together are the part of the .NET platform. FCL is standard library

and a collection of classes, interfaces and value types that provide the user interface, data access, connection to the database, cryptography, web application development, numerical algorithms, and network communications.

CLR is a virtual machine on the Microsoft .NET platform and manages the execution of .NET programs. The process is known as just-in-time compilation and converts the translated code into machine instructions. CLR provides additional services including memory management, security type, dealing with exceptions, garbage collection and security and management of threads. All programs written for .NET, independent of the programming language are performed by the CLR [8].

The purpose of the Common Language Infrastructure (CLI) is to provide a neutral platform for application development and execution, including functions for managing exceptions, garbage collection, security and inter-operability. By implementing the main aspects of .NET within the CLI, this functionality will not be tied to one language but will be available in many languages supported by the platform. .NET framework includes many standard class libraries that are organized in a hierarchy of namespaces. Most embedded application programming interfaces are part of the System.* or Microsoft.* Namespaces. These class libraries implement many common functions such as reading and writing files, interaction database and XML document manipulation. .NET Class library is divided into two parts: FCL and BCL (Base Class Library).

FCL includes a small subset of the entire class library and is the core set of classes that serve as the basic application programming interfaces and CLR. BCL is a superset of FCL and refers to the entire class library that is sent with .NET. It includes an expanded set of libraries, including Windows Forms, ADO. NET, ASP.NET, language integrated query, Windows Presentation Foundation, Windows Communication Foundation, etc. BCL is much larger in scope than standard libraries for languages and is comparable in size to the standard Java libraries.

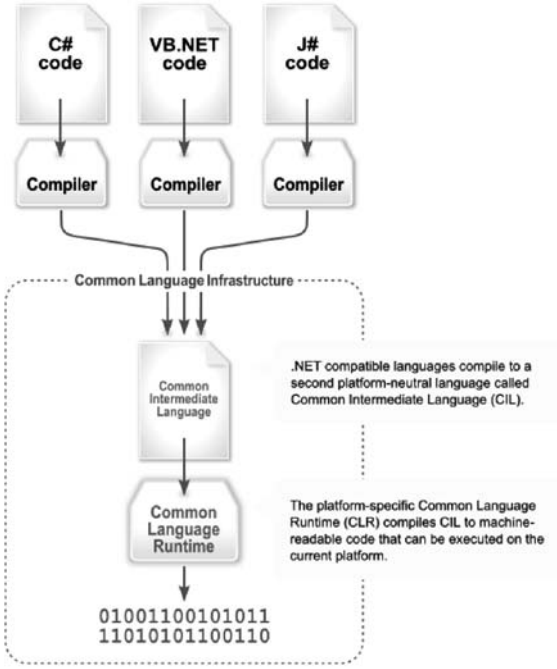


Figure 2. .NET platform and usage of CLI [8]

Mainly, the program support is divided into system support and customer support program. System support is a set of programs which is ready to accept and implement customized programs. Customer support program consists of programs that are used to solve a specific task prepared by the computer system. The program consists of a set of commands and data used to solve a specific task.

4.1. IIS (Internet Information Services)

Internet Information Service is a set of Internet based services created by Microsoft for the Windows platform. It is the second popular web server then Apache Http Server. The services that supports are FTP, FTPS, SMTP, NNTP, and HTTP/ HTTPS. IIS is a secure and scalable web server that provides easy handling platform for developing and hosting Web applications and services. IIS includes tools for creating reliable communications platform of dynamic network applications for various environments, small and large companies. It is a robust platform for the development, implementation and management of new Web sites and applications. In addition, IIS can run as an application server [10].

4.2. ASP.NET

ASP.NET is a part of the integrated .NET Framework, designed to provide services to create dynamic web applications and web services. Built on the CLR (Common Language Runtime) the .NET framework includes benefits such as: multi language support, inter-operability, security type, garbage collection and inheritance [1, 3]. ASP.NET works with Internet Information Server (IIS) to provide content in response to customer requests. In processing applications, ASP. NET provides access to all .NET classes, adjustable components and databases.

Web forms are the basis of the development of application in ASP.NET. They provide flexibility by allowing controls to be used as objects. These controls can manage events. Except for Web forms, ASP.NET is used when creating XML Web services that allow the construction of modular, distributed Web applications written in any language. These services are inter-operable across different platforms and devices. ASP.NET handles management conditions, by sending information to the controls in the web form to the server, ensuring execution of applications, regardless of the different versions of the .NET framework. We use XML support for data storage, manipulation and configuration. However, when it comes to security applications, ASP.NET uses the security code to access role-based security features of the .NET Framework and IIS methods for authentication of user authority. ASP.NET provides two different types of server controls - HTML server controls and Web server controls. Each type of control is different, but the focus is increasingly placed on the web server controls. HTML server controls map to specific HTML elements. Some developers want to separate some of the other controls and store them in their categories [4, 2]:

4.3. ADO.NET

ADO.NET is a technology family that enables .NET developers to interact with data with standard, structured and primarily disconnected ways. ADO. NET, expressed through the System.Data namespace, implements a small set of libraries consuming and manipulating large amounts of simple and clear data. ADO.NET manages both types of data i.e. internal data - created in memory and used within the application and external data - stored in the storage section, usually relational database or a text file. ADO.NET generalizes relevant data and submits them to the code table - columns and rows. When communicating with external databases, ADO.NET represents unbound (disconnected) data. ADO. NET is a set of classes that come with Microsoft.Net platform to facilitate access to data. The power of ADO.NET initially allows the application to access the various data types by using the same methodology. Second, ADO.NET provides two models to access the data: linked model where you can keep the connection to the database and to perform data access, and the other way is to provide all data

in ADO.NET objects that will enable to perform data access to the unconnected structures [5].

Platforms based on data without a specific provider use more generic ODBC and OLE DB providers, involved in ADO.NET. All communication with the external data source is via the object Connection. ADO.NET supports connection pooling to maximize efficiency between queries. DataAdapter object provides standard definitions of queries to communicate with the database. DataReader object provides fast, read-only access to the results of queries for a number of times when you just need to take quick data.

5. Case study - public sector improvements by using web applications

The case study that is used in this paper includes PTE needs analysis, solution design and development of web application connected to the database. Web application is made in the C# and ASP.NET programming language. The database in which data is stored is MS SQL Server. This program platform provides several impressive benefits [6] as an integrated error checking, web forms designer, an integrated web server, productivity enhancements programmer, granular debugging, and the ability to fully expand with macros to change the templates of the project, even adding their own enhancements in Visual studio.

Database is an essential part of the whole web applications where data are daily entered, read, updated and deleted and they are necessary for the operating of web application. From web programming aspect, their major advantage relates to the facilitation of how to work with them. Thus, MS SQL Server is used which is based on relational algebra that is easy to learn, it has a simple grammar and simple syntax [7] with standard SQL commands. When using the base, in this case MS SQL Server, it is about developing a dynamic web application.

When creating a database, it remains to organize data and to determine objectives: elimination of redundancy, fast search and database consistency. Therefore, key activities are: application modeling, defining the entities and relationships needed for the application, organizing data into tables, establishing a connection between tables, determining the requirements for indexing and data evaluation as well as preparation and preservation of all necessary requirements related with application. Figure 3 shows the layout of the database application for the issuance of tickets to high school. It is composed of several tables (entities): High school tickets, Logging, Municipality and School. Database “HST” contains data for students who need tickets and login users that log into the system.

5.1. Designing and creating application software i.e. web application

C#.NET is used for creating of web application. It is simple, multi-purpose

programming language with object - oriented syntax. In the application five forms were created. Created forms allow users to see, change or add data. It allows functions Insert, Delete, Update and read only. Suitable design forms allow users to gain easy access to information and enable user interaction by placing objects on screens.

Figure 3 below shows the structure of application's code that consists of several forms. Specific Solution Explorer window is shown that allows management of every part of application [9]. He shows all parts of the code of the Web application. Five forms are created: registration form, login form, and form by choosing whether to input a new student or consider the other option, the form for entering data and a new student selection in class or year. See below Fig. 3a and Fig. 3b.

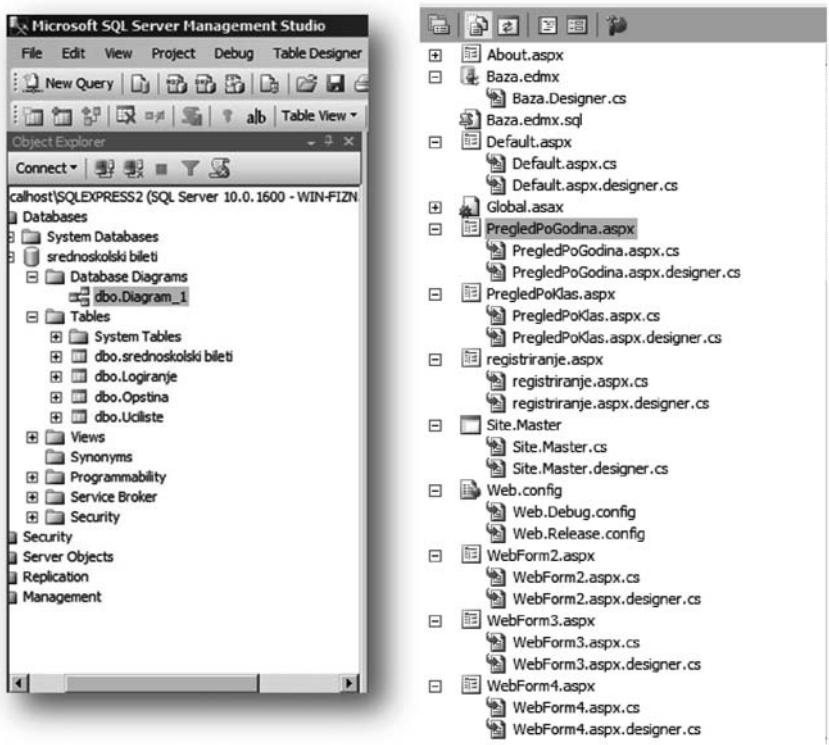


Figure 3a – Overview of Database structure created in MS SQL Server,
3b - The application code structure

Figure 4a below shows the web application entry screen in which are set input forms for entering the user's username and password. Two buttons are inserted: "Login" and "Register". By the button "Login" label that informs the user whether they typed the correct data when logging is set.

Figure 4a – Preview of the web application log screen

Only the employees of the high schools have access to the application once their accounts are verified by PTE and only they can register. Once the user will receive login information, it can access the application where the screen appears with the selection button for entering new student and preview button for already entered student. Additional search buttons were created on the screen (Se below Figure 4b).

Figure 4b – Preview of screen for input data for a new student

Preview of student intake or review of year or class is also created for the needs of one of the schools (See below Figure 5a and 5b).

ID	IME	PREZIME	ULICA	BROJ	Opstina	Naseleno Mesto	Uciliste	GODINA
1	ЕЛМЕДИНА	БАЈРАМИ	С.БРЕЗА				СУТС ЦВЕТАН ДИМОВ	1
2	АДМИР	СИНАНИ	С.ГОЛОШИНЦЕ				СУТС ЦВЕТАН ДИМОВ	1
3	БУЏРА	ЗЕНЕЛИ	1	63	КОНДОВО		СУТС ЦВЕТАН ДИМОВ	1
4	ЕМБРАХ	ХАЛИЛИ	1	91	КОНДОВО		СУТС ЦВЕТАН ДИМОВ	1
5	НАЗИМ	САИТИ	1	86	КОНДОВО		СУТС ЦВЕТАН ДИМОВ	1
6	БЕРАТ	ДУРАКУ	10	17	САРАЈ		СУТС ЦВЕТАН ДИМОВ	1
7	ДУРАН	ХАЗИРИ	171	13	ЧАИР		СУТС ЦВЕТАН ДИМОВ	1
8	АМИР	АДЕМИ	4	7	КОНДОВО		СУТС ЦВЕТАН ДИМОВ	1
9	ЗЕЉЕП	ЛИКА	167	25			СУТС ЦВЕТАН ДИМОВ	1
10	АДИС	ИСМАНИ	2	8	РАШЕ		СУТС ЦВЕТАН ДИМОВ	1

Figure 5a – Students Report per class

ID	IME	PREZIME	ULICA	BROJ	Opstina	NaseleneMesto	Uciliste	KLAS
1	ЕЛМЕДИНА	БАЈРАМИ	С.БРЕЗА				СУТС ЦВЕТАН ДИМОВ	10
2	АДМИР	СИНАНИ	С.ГОЛОШИНЦЕ				СУТС ЦВЕТАН ДИМОВ	10
3	БУЏРА	ЗЕНЕЛИ	1	63	КОНДОВО		СУТС ЦВЕТАН ДИМОВ	10
4	ЕМБРАХ	ХАЛИЛИ	1	91	КОНДОВО		СУТС ЦВЕТАН ДИМОВ	10
5	НАЗИМ	САИТИ	1	86	КОНДОВО		СУТС ЦВЕТАН ДИМОВ	10
6	БЕРАТ	ДУРАКУ	10	17	САРАЈ		СУТС ЦВЕТАН ДИМОВ	10
7	ДУРАН	ХАЗИРИ	171	13	ЧАИР		СУТС ЦВЕТАН ДИМОВ	10
8	АМИР	АДЕМИ	4	7	КОНДОВО		СУТС ЦВЕТАН ДИМОВ	10
9	ЗЕЉЕП	ЛИКА	167	25			СУТС ЦВЕТАН ДИМОВ	10
10	АДИС	ИСМАНИ	2	8	РАШЕ		СУТС ЦВЕТАН ДИМОВ	10

Figure 5b – Students reports per year

Empirical methods, including interview were used to analyze the users' information needs. In this survey done with the employees of the PTE, interviews were made in order to determine the needs before and after implementation as well as the benefits of web application. To this end, part of the employees in the PTE and managers were interviewed and asked questions about effect of time of getting information when they don't have application, and expected time results from the implementation of the application, the number of employees engaged in the process and their expectations in reducing the number of involved employees, as well as the effects on the community expectations. From the answers it can be concluded that they must provide precise data, the expectation from web application are that the number of engaged employees have to decrease (from 6 to 2), the whole process has to be faster and the needed time for obtaining data have to be shortened from several weeks to two or three days. The printing of HST will run through a computer which further reduces the time of issuance of the forms of identity cards. Through the answers of the questions in these interviews, it can be asserted that the contributions of this web application for PTE and students for high schools are really great. In this case, do not only facilitate the operations in PTE but it facilitated the work of the school staff and shortening the time of issuing of HST tickets for students. Instead of wasting paper and time for issuing students 'certificates, the data are entered into the application which saves time but also saves paper, and besides the financial benefit it also contributes in promoting of a green society.

6. Conclusions

It is a fact that Internet and IT has changed the traditional life-style. There is increase use of various forms of electronic business between companies, government institutions and customers. In the public sector the introduction of ICT is associated with the emergence of the concept of e-government where ICT are not just a tool but rather a basis for reforming the functioning of the public management and public services in order to improve the efficiency and effectiveness of public sector operations. The introduction of ICT is used to improve and facilitate the administrative procedures related to different relations of the public sector management and citizens as customers and interconnection of public sector institutions known as relation G2G (government-to-government). This conclusion is logical if we take into consideration all the benefits associated with this way of working in which each public employee, associate or service customer, wherever they are located, can complete their work quickly, efficiently and economically using a web application.

Establishing relationships with customers and users for every company and institution certainly represents the most important element in private and business

lives, but also in the "life" of PTE. When it comes to business relations between the local company, customers and other enterprises, and in terms of building long-term relationships based on mutual interest and moral obligation, we talk about customer oriented government / public administration [11]. Creating support for such kind of relationships through a web application by creating satisfaction and loyalty, we want to make a symbiosis between technology, public sector enterprises and clients. From the interviews made with customers is very important to determine if only companies understand the establishment of these relations or they are under-assessed from the clients.

With a web application developed according to customer needs for PTE can be achieved overall improvement of the public sector management performances, customer relationships and greater integration of all society processes. During the creation of the application, VisualStudio.NET platform is used and it enables integrated .NET Framework that is designed to provide services to create dynamic web applications and web services. Applications and data are used to provide all values that information technology brings depend on application platform. Because almost every public organization today tends on work by web applications, there is a clear relation between public business values and web application platforms. Web application as easy, fast and simple way, allows completing the obligation for which it was designed. By using it, we save resources, improve customer satisfaction and therefore, can be a much powerful and valuable tool in the hands of public sector managers.

References

1. Matthew MacDonald, Beginning ASP.NET 4.5 in C#, Apress, 2012
2. Sandeep Chanda, Damien Foggon . Beginning ASP.NET 4.5 Databases, 3rd Edition, Apress, 2013
3. Jason N. Gaylord, Christian Wenz, Pranav Rastogi, Todd Miranda, Scott Hanselman (2013). Professional ASP.NET 4.5 in C# and VB, Wiley, 2013
4. Morgan Skinner (2010). C#4, ASP.NET 4, and WPF, with Visual Studio 2010 Jump Start, Wiley, 2010
5. Tim Patrick, Microsoft ADO.NET 4 Step by Step, O'Reilly Media Inc.,2010
6. Sur A., Visual Studio 2012 and .NET 4.5 Expert Development Cookbook, Packt Publishing Ltd., 2013
7. Alex Kriegel, Discovering SQL, A Hands-On Guide for Beginners, Wiley, 2011
8. Thuan Thai, Hoang Q.Lam, .NET Framework Essentials, O'Reilly Media, 2003
9. Andrew Troelsen, Pro C# with .NET 3.0, Apress, 2007

10. Kenneth Schaefer, Jeff Cochran, Scott Forsyth, Rob Baugh, Mike Everest, Dennis Glendenning, Professional IIS 7.0, Wiley,2008
11. http://en.wikipedia.org/wiki/Clientelism, 02.05.2015
12. http://www.britannica.com/EBchecked/topic/1916881/clientelism, 05.06.2015
13. Robinson J., Verdier T., The political economy to clientelism, Scand. J.of Economics, 115(2), 260-291, 2013

Ontologies in Bioinformatics: Main Features and Applications

Maria Nisheva-Pavlova, Pavel Pavlov

Faculty of Mathematics and Informatics, Sofia University St. Kliment Ohridski
5 James Bourchier blvd., 1164 Sofia, Bulgaria
{marian, pavlovp}@fmi.uni-sofia.bg

Abstract. The paper discusses the main features of some of the most popular ontologies for bioinformatics: the variety of their concepts, types of relations and logical characteristics. Multiform real and potential applications of ontologies in bioinformatics research are analyzed. The presentation is focused on the usability of proper types of ontologies as models and formalisms for knowledge representation, knowledge sharing, information integration, and knowledge discovery in Bioinformatics.

Keywords: bioinformatics, ontology, knowledge representation, knowledge sharing, information integration, knowledge discovery.

1 Introduction

Bioinformatics is an interdisciplinary field that develops methods and software tools for automated analysis and interpretation of biological data. Its primary goal is to increase the understanding of biological processes and its main activities are focused on developing and applying computationally intensive techniques to achieve this goal.

Lately bioinformatics provides important tools for many areas of biology. For example, it assists significantly in sequencing and annotating genomes and their mutations. It helps in the text mining of biological literature and the development of biological and gene ontologies to describe and query biological data.

Bioinformatics uses mostly community knowledge stored as natural language texts. This knowledge needs to be represented in a computationally appropriate form containing strong definitions of its meaning, main features and restrictions. Ontologies offer adequate means for this purpose.

Ontologies have a rapidly extending range of bioinformatics applications. Bioinformatics can be considered as one of the most remarkable areas which demonstrate the real advantages of the use of ontologies, knowledge-based methodology and semantic technologies for achieving their specific research goals. From the other side, problems arising from real application areas like bioinformatics motivate new research activities in many directions of ontology-

V. Dimitrov, V. Georgiev (Editors): ISGT'2015. ISSN 1314-4855
Proceedings of the 9th International Conference on
Information Systems and Grid Technologies, Sofia, 29–30. May, 2015.

based technologies such as ontology engineering, ontology mapping, ontology alignment, etc.

2 Types of Knowledge and Formal Reasoning for Bioinformatics

Two main types of knowledge may be distinguished in bioinformatics research and practical applications [1]: common knowledge and local-domain (or novel) knowledge.

2.1 Common Knowledge

The term *common knowledge* refers to knowledge that is generally known and accessible to everyone in a particular community, and which can be formally described. *Explicit common knowledge* is the one that is already formally described in a proper format and is directly available to the corresponding applications.

An example of explicit common knowledge is the following rule describing the SNP (single nucleotide polymorphism) associations with diseases, where the polymorphism does not alter the codons directly, but the protein is either truncated or spliced differently [1]:

```
∀ Genetic_Disease ∃ Gene ∃ Protein ∃ SNP
   (SNP within Gene ∧ Gene expresses Protein ∧
    SNP modifies Protein ∧ SNP associated Genetic_Disease)
⇒
SNP root_cause_of Genetic_Disease
```

Common knowledge is useful for most forms of reasoning in bioinformatics, since it facilitates making connections between specific data in local problems and generalized rules or facts. Many authors point out that true inference is not possible without the proper encoding of complete common knowledge.

Due to the vastness of common knowledge around all biomedical domains (including all instances of genes, diseases, and genotypes), it is very difficult to explicitly formalize all of it and place it in a single knowledge base. However, if public data sources are considered as references of knowledge, then the amount of explicit common knowledge can be easily and greatly augmented. This process will require some mechanism for wrapping these sources with proper knowledge representation formalisms, for example associating entities with classes.

2.2 Local-domain or Novel Knowledge

There are large collections of local-domain knowledge consisting of works and models created by individual research groups. This is usually novel knowledge that has not been completely validated yet. It has mainly the form of hypotheses or beliefs, but nonetheless may be of interest for other researchers working

on the same or similar problems in order to compare or corroborate the used methodology and the proposed hypotheses. This knowledge is connected to and relies on the fundamentals of biology, which are in fact common knowledge. Because of that researchers are looking for a model which allows connecting local-domain knowledge easily with common knowledge.

2.3 Types of Formal Reasoning for Bioinformatics

Logical reasoning in classical knowledge-based systems is realized in three main forms: *deduction*, *induction*, and *abduction*. All of them work on rules (usually called *production rules*) which consist of *preconditions* (antecedents) and *conclusions* (consequents). Deduction is directed to solving for the consequent given the antecedent and the rule. Induction is about finding the rule that determines the consequent based on the known precondition. Abduction is directed to determining the preconditions based on the conclusions and the rules followed. It is not so frequently used in knowledge-based systems designed to perform reasoning in the area of Bioinformatics [1].

Formal methods and software tools for *Data Mining and Knowledge Discovery* (KDD) are widely used in Bioinformatics research. KDD is aimed at extracting patterns from data, but distinguishes itself from other subjects in that the patterns have to be validated, made intelligently interpretable, and applicable to the particular problem.

The set of key issues that KDD is successfully addressing in the area of Bioinformatics include [1]:

- prediction of gene/protein function and localization;
- prediction of molecular bioactivity for drug design;
- information extraction from biomedical articles;
- yeast gene regulation prediction;
- identification of pulmonary embolisms from three-dimensional computed tomography data
- computer aided detection of early stage breast cancer from X-ray images.

The progress in development of modern semantic technologies and the success in implementation of the Linked Open Data initiative (http://linkeddata.org/) face KDD with new challenges and opportunities.

2.4 Sources of Data in Bioinformatics Knowledge Bases

Bioinformatics knowledge bases include data originating from three types of sources [1]: data added by internal curators, data submitted by external collaborators, and data added automatically.

(1) Data added by internal curators

A central to the biological knowledge concept is data curation. Biological data

curation typically involves experts interpreting data and specialized literature and using this information to populate the corresponding database(s). It is a manual, highly time-consuming process that adds confidence, value and quality to the used resources.

The range of work performed by a curator can include the entire process of addition of new information [1]: stating whether some particular data are ready to be released to the public or not; composition of natural language text from literature sources to describe a specific aspect of a database entity; association of ontological terms (concepts) to new entries; structuring of relationships between data in the database.

(2) Data submitted by external users and collaborators

Knowledge bases often allow submission of data by users who are not part of the institution where the particular resource was developed and maintained. Many databanks provide tools to allow submission of data on both a small and a large scale. For small-scale submissions, Web-based forms tend to be provided where a user can interactively enter data that are subsequently validated. In the case of large-scale submissions, a standardized data format is typically defined in advance (usually in a markup language such as XML) and the user should prepare the "new" data in accordance with this format.

(3) Data added automatically

A significant part of data contained in Bioinformatics knowledge bases is a result of automatic associations or predictive calculations. The information obtained and added in this way is not as in-depth as that added by a human curator but nevertheless these processes' output is much greater.

In addition, automated pipelines that require minimal human intervention are implemented more and more in practice instead of relying on manual processes to load and transform data. This can be considered as a consequence of the need to handle very large amounts of data in an efficient way.

3 Ontologies and their Potential for Knowledge Representation and Reasoning in Bioinformatics

Bioinformatics is a large and complex domain, appropriate for testing knowledge representation techniques to the limit of their expressive power, precision and adaptability. Research in bioinformatics relies mostly on the use of community knowledge, rather than logical laws and axioms, for further understanding and knowledge generation. This knowledge has traditionally been kept as natural language texts and nowadays needs to be represented in computationally amenable forms. Ontologies offer a widely accepted mechanism for creating an understanding of the meaning of biological knowledge which can be shared by humans and computer systems.

An ontology may take various forms, but it always contains a *vocabulary of terms* (or *concepts*) and some *specification of their meaning*. This includes definitions of concepts, their relationships and some constraints on the possible interpretations of terms.

Gruber defines an ontology as "the specification of conceptualizations, used to help programs and humans share knowledge" [2]. The *conceptualization* is the expressing of knowledge about the world in terms of entities (things, the relationships they hold and the constraints between them). The *specification* is the representation of this conceptualization in a concrete form. One step in this specification is the encoding of the conceptualization in a knowledge representation language. The goal is to create an agreed vocabulary and semantic structure for exchanging information about the corresponding domain.

Ontologies are most often used in biology for conceptual annotation. They are also used to support the development of various types of intelligent search engines that are able to perform complex queries over heterogeneous, distributed information sources. In this case an ontology creates a shared understanding of the meaning of the domain-specific terms and the relations between the corresponding concepts. Ontologies provide a means for a schema definition suitable for the complexity and precision required for biological knowledge bases.

Ontologies designed mainly for bioinformatics applications can be divided into three types [3]:

- *domain-oriented*, which are based either on knowledge about domain-specific concepts (e.g. E. coli) or on domain generalizations (e.g. gene function or ribosomes);
- *task-oriented*, which are either task-specific (e.g. annotation analysis) or task generalizations (e.g. problem solving);
- *generic*, which capture common high level concepts, such as Physical, Abstract, Structure, Substance, etc. Generic ontologies are also known as *upper ontologies*, *core ontologies* or *reference ontologies.* They provide flexibility and reusability of the corresponding software applications.

Ontologies are used in a wide range of biology application scenarios [3, 4, 5]:

- as common vocabularies for describing, sharing, linking, classifying, querying and indexing database annotations. This is currently the most popular use of ontologies in bioinformatics. The Gene Ontology (GO, http://geneontology.org/) and the MGED Ontology (MO, http://mged.sourceforge.net/ontologies/MGEDontology.php) are mentioned as most representative examples in this sense;
- in understanding database annotation and technical literature. Usually ontologies are designed to support natural language processing that links domain knowledge and linguistic structures;
- as means for data integration and support of interoperability between

multiple resources. Various forms may be mentioned, for example: indexing across databases by shared vocabularies of their content, inter-database navigation and querying, using an ontology as a virtual schema for federation of databases, etc.;

- as knowledge sources for *intelligent search* (also known as *semantic search* or *ontology-based search*) over heterogeneous databases or other repositories of biological information. Search queries can be augmented and refined by following some relationships within the ontologies, in particular the taxonomic relationships.

The use of ontologies can also help to address a challenge that machine learning and data mining approaches face: the incorporation of different types of features (or attributes) for application of learning and especially classification algorithms [4]. Extraction and combination of information from text, images, videos, molecular data or structured data in knowledge bases aimed at improving classification can be facilitated through the use ontologies, by first extracting relevant attributes from each type of information and then representing the results using a single ontology that combines the information used for training the classifier.

4 Current Knowledge Sources and Knowledge-based Tools for Bioinformatics Research

The National Center for Biomedical Ontology (NCBO, http://www.bioontology.org/) and the European Bioinformatics Institute (EMBL-EBI, http://www.ebi.ac.uk/) are widely recognized as the most important primary knowledge sources for bioinformatics researchers. According to [6], the services hosted by both institutions include more than 60 ontologies of the Open Biomedical Ontologies Foundry (OBO Foundry, http://www.obofoundry.org/). The OBO Foundry is a collaborative experiment involving developers of science-based ontologies who are establishing a set of principles for ontology development with the goal of creating a suite of "orthogonal" interoperable reference ontologies in the biomedical domain.

4.1 Ontologies Used in Bioinformatics Research

Ontologies used in bioinformatics differ with respect to their scope, size and granularity. Some of them are devoted to anatomy and physiology of specific organisms, others provide descriptions of biomedical resources and experiments in analytical research labs. A significant number of ontologies are oriented to related areas like health care and medical applications.

The Gene Ontology (GO) [6, 7] is the most popular ontology in bioinformatics. Its origin dates back to 1999 when several model-organism

database projects (initially FlyBase, the Mouse Genome Informatics Database and the Saccharymyces Genome Database) noticed that a common vocabulary will improve the interoperability between databases and simplify data integration. The key to associating these model databases was the genetic structure of organisms. The GO integrates concepts that serve for classifying gene products according to what they do, where they act and how they perform these activities. It actually comprises three separate ontologies, one for molecular functions, one for cellular components, and one for biological processes. In addition to subsumption (*is-a*) and meronymy (*part-of*) it provides a third kind of relationships for describing interactions between biological processes, molecular functions and biological qualities.

The GO provides a standardized set of names for genes and proteins and the terms for characterizing – or annotating – their behaviors. Concept definitions have the form of textual descriptions (see for example Fig. 1).

Gene product semantics are organized into three categories which capture the primary "aspects" (or "views") of genes: (1) *biological process*, that identifies the largest process in which the gene product is active; (2) *molecular function*, that specifies the biochemical function the gene product contributes to through the pointed process; (3) *cellular component*, that fixes the location in the cell where that particular function is realized or expressed. Annotations for the same term in each "view" are cross-referenced on the basis of the unique identifier assigned to each term in the GO.

```
[Term]
id:         GO:0000015
name:       phosphopyruvate hydratase complex
namespace:  cellular_component
def:        "A multimeric enzyme complex,
            usually a dimer or an octamer,
            that catalyzes the conversion of
            2-phospho-D-glycerate to phosphoenolpyruvate
            and water." [GOC:jl, ISBN:0198506732]
subset:     gosubset_prok
synonym:    "enolase complex" EXACT []
is_a:       GO:0044445 ! cytosolic part
is_a:       GO:1902494 ! catalytic complex
```

Fig. 1. An example of concept definition in the Gene Ontology.

The MGED Ontology (MO) [8] provides terms for annotating all aspects of a microarray experiment from the design of the experiment and array layout, to the preparation of the biological sample and the protocols used to hybridize the RNA and analyze the data. The MO does not attempt to incorporate terms from existing ontologies, e.g. those that deal with anatomical parts or developmental stages terms, but provides a framework to reference terms in other ontologies and in this way facilitates the use of ontologies in microarray data annotation. The MO is primarily used to annotate microarray experiments, but it contains

concepts that are universal to other types of functional genomics experiments such as protocol and experiment design and thus can also be used for annotation of some of the data in these domains.

The MO is primarily used in three ways [8]: (1) embedded within an application to annotate or query microarray data, e.g. by biologists who have some knowledge of the MO structure; (2) directly for annotating microarray data, e.g. by an annotator; (3) for producing an application that uses the MO, e.g. by a software developer.

The TAMBIS ontology [9] covers a wide range of biological concepts. It is used as a unified schema to support queries over multiple biological data sources in an information integration system. The aim of the TAMBIS Ontology is thus to integrate biological and bioinformatics knowledge in a logical conceptual framework constrained in such a way that [9]: it classifies correctly only biologically sensible concepts; it can encompass different user views; it makes biological concepts and their relationships computationally accessible. TAMBIS can be considered as a knowledge base which is based on description logics.

The FungalWeb Ontology [10] is a large-scale integrated bio-ontology in the domain of fungal genomics based on the use of modern semantic technologies. The ontology provides simplified access to units that describe intersecting information from different biological databases and existing bio-ontologies. In particular, the FungalWeb ontology is being used as a core for a Semantic Web system. This system can be used by humans or bioinformatics software applications, including intelligent systems for ontology-based information retrieval to provide extended interpretations and annotations.

The FungalWeb Ontology reuses other existing domain specific bio-ontologies such as GO and TAMBIS. This is done by mapping, merging, and sharing common concepts and partially by importing instances. The ontology is designed with a high level of granularity and implemented in OWL-DL language in order to take advantage of the combination of frame-based knowledge representation within the OWL framework and the expressive power of description logics.

4.2 Usability of Semantic Technologies

Ontologies like FungalWeb make it possible to apply various semantic technologies in bioinformatics research. For example, [10] describes the use of the popular software tool RacerPro (https://www.w3.org/2001/sw/wiki/RacerPro) as a description logics reasoning system with support for the so-called T-Box (containing the axioms about class definitions) and A-Box (with the assertions about individuals) for reasoning on this ontology and checking the A-Box and T-Box consistency. Moreover, the FungalWeb Ontology currently supports a number of application scenarios including [10]:

- identification of enzymes acting on substrates;
- identification of enzyme provenance and common taxonomic lineage;
- identification of commercial enzyme products for enzyme benchmark testing;
- identification of enzymes with unique properties suited for industrial application.

These scenarios are realized by defining and fulfilling semantic queries to the FungalWeb knowledge base using the description logics based query language nRQL (new Racer Query Language, http://franz.com/agraph/racer/racer_features.lhtml). nRQL is implemented in RacerPro with its applicability to OWL Semantic Web repositories to retrieve A-box individuals under specific conditions. nRQL is more expressive than the traditional concept-based retrieval languages provided by other description logics reasoning systems.

A number of successful projects directed to automatic ontology generation, ontology engineering, and ontology matching (including ontology alignment and ontology merging) in the field of bioinformatics have also been realized [11].

5 Machine Learning Applications in Bioinformatics

Machine Learning (ML) is a scientific discipline concerned with the design and development of algorithms that allow computers to change their behavior on the base of available data (also known as *training data* or *training examples*). Major focuses of machine learning research are to automatically learn to recognize complex patterns and to make intelligent decisions based on data.

ML methods have been applied to a broad range of areas within genetics and genomics. ML is considered as most useful for the interpretation of large genomic datasets and has been used in the annotation of a wide variety of genomic sequence elements. ML applications have also been used to assign functional annotations to genes. Such annotations usually take the form of GO term assignments. A wide variety of bioinformatics data have been used as input to ML algorithms: genomic sequences, gene expression profiles across various experimental conditions or phenotypes, protein–protein interaction data, etc. [12]. A number of ML methods have been especially developed to help to understand the mechanisms underlying gene expression and to predict the expression of a gene on the basis of the DNA sequence.

The selection of features (or *attribute selection*) is recognized as one of the most significant challenges to the application of ML methods in bioinformatics research. In practice, it is important to distinguish among three distinct motivations for carrying out attribute/feature selection. First, in some cases, it is useful to identify a small set of features that yield the best possible classifier. For example [12], we may want to produce an inexpensive way to identify a disease

phenotype on the basis of the measured expression levels of a set of genes. Such a classifier, if it is precise enough, might form the basis of an inexpensive clinical assay. Second, it might be advisable to use a classifier to help to understand the underlying biology. In this case we expect the attribute selection procedure to identify only the genes with expression levels that are actually relevant to the task, hoping that the corresponding functional annotations or biological pathways might provide insights into the aetiology of disease. Third, it is usually desirable to have proper data for training the most accurate possible classifier. In this case the attribute selection is expected to enable the classifier to identify and eliminate noisy or redundant attributes.

As common difficulties in many applications of ML techniques to bioinformatics research [12] indicates (1) the large imbalance in the relative sizes of the groups being classified and (2) the fact of missing data values that can come from various sources, such as defective cells in gene expression microarrays, unmappable genome positions in functional genomic assays, etc.

6 Conclusion

Lately the application of ontology-based (and more generally, knowledge-based) approaches and ML methods has an important role in bioinformatics research. From the other side, the use of new technologies generates more and more large, complex genomic and proteomic datasets, therefore the proficient application of Artificial Intelligence models and algorithms is expected to become increasingly important to advancing genetics and genomics.

Acknowledgments. This work has been supported by the National Science Fund of Bulgaria within the "Methods for Data Analysis and Knowledge Discovery in Big Sequencing Datasets" Project, Contract DFNI-I02/7 of 12 December 2014.

References

1. Alterovitz, G., Ramoni, M. (Eds.): Knowledge-Based Bioinformatics: From Analysis to Interpretation. John Wiley & Sons (2010)
2. Gruber, T.: Towards Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal of Human-Computer Studies, Vol. 43 (1995), pp. 907-928 (1995)
3. Stevens, R., Goble, C., Bechhofer, S.: Ontology-based Knowledge Representation for Bioinformatics. Briefings in Bioinformatics, Vol. 1 (2000), pp. 398—414. Oxford University Press (2000)
4. Hoehndorf, R., Schofield, P., Gkoutos, G.: The Role of Ontologies in Biological and Biomedical Research: A Functional Perspective. Briefings in Bioinformatics, Vol. 16 (2015), pp. 1—12. Oxford University Press (2015)
5. Jakoniene, V., Lambrix, P.: Ontology-based Integration for Bioinformatics. In: Proceedings of the VLDB Workshop on Ontologies-based Techniques for Databases and Information Systems – ODBIS 2005, pp. 55—58 (2005)

6. Hartmann, S., Köhler, S., Wang, J.: Ontology Consolidation in Bioinformatics. In: Proceedings of the 7th Asia-Pacific Conference on Conceptual Modelling APCCM 2010, pp. 15—22 (2010)
7. Schuurman, N., Leszczynski, A.: Ontologies for Bioinformatics. Bioinformatics and Biology Insights, Vol. 2, pp. 187—200 (2008)
8. Whetzel, P., et al.: The MGED Ontology: A Resource for Semantics-based Description of Microarray Experiments. Bioinformatics, Vol. 22 (2006), pp. 866-873 (2006)
9. Baker, P., et al.: An Ontology for Bioinformatics Applications. Bioinformatics, Vol. 15 (1999), pp. 510–520 (1999)
10. Shaban-Nejad, A., et al.: The FungalWeb Ontology: Semantic Web Challenges in Bioinformatics and Genomics. LNCS, Vol. 3729, pp. 1063—1066. Springer, Heidelberg (2005)
11. Huang, J., et al.: Ontology-Based Knowledge Discovery and Sharing in Bioinformatics and Medical Informatics: A Brief Survey. In: Proceedings of the Seventh International Conference on Fuzzy Systems and Knowledge Discovery FSKD 2010, Vol. 5, pp. 2203—2208. IEEE (2010)
12. Libbrecht, M., Noble, W.: Machine Learning Applications in Genetics and Genomics. Nature Reviews Genetics, Vol. 16, No 6, pp. 321—332. Macmillan Publishers (2015)

Quantum Bits

Mícheál Mac an Airchinnigh | Михал Мак ан Аирхини

<mmaa@cs.tcd.ie>
School of Computer Science and Statistics
University of Dublin, Trinity College
Dublin 2, Ireland

Abstract. One takes the theme for granted: Intelligent and Information Systems and Grid technologies (IISG)... In this paper one also takes the "Narrative Scheme of Story Telling" for granted. Once again, there is a new story to be told. The story will be, yet again, another journey into the realm of Intelligent Information Systems [IIS]... deliberately and constructively different from the usual historical trajectory of earlier contributions to the theme. One notes the very focused idea of a "forum where new concepts can be introduced and explored." Technology evolves at a quicker rate today than yesterday! But, history teaches, that what comes round, goes round, in the great cycle of life, however one might imagine that! Today, there appears to be a dichotomy between the Arts and the Sciences. Today, there appears to be the burden of Technology, of Science, of all those things opposite to the Arts. There can never be a reconciliation between such opposites... Unless, of course, there is a Middle ground. IISG is a platform, a stage, upon which one can explore the potential of synthesis. In this paper, an attempt is made to demonstrate the potential compatibility between three (not two) "opposites"! There is the mathematical/scientific core of the "mysterious Qubit"! But, if one is already familiar with the astonishing World of Quaternions, then taking QuBits on board is child's play, so to speak!

Keywords. blockchain, disruption, eDisplay, entanglement, quantum, qubit,

1. Preface: Information comes in all kinds of forms.

Naturally, one distinguishes between information and knowledge. For example, waiting on a Railway Station Platform for one's train, one is inclined to look at the very crisp modern electronic displays (abbreviated eDisplays), with a certain regularity, anticipating the actual arrival of the train. Those who are waiting for buses, elsewhere, are also subject to similar eDisplays. From experience, one becomes knowledgeable as to the "reality" of the announced/displayed arrival time. Specifically, one expects the train to arrive on time; one expects the bus to be late.

This is normal. The bus is subject to the chaotic behavior of general traffic, whereas the train is usually on time.

We have become accustomed to search for information online via certain well-known search engines. In the general field of mathematics one is, perhaps, accustomed to Wolfram|Alpha? One expects to get the information looked for? Do you need some information on Chaos? Wolfram|Alpha will give you the choice: "Assuming "chaos" is a word | Use as a unit or a television series or a mythological figure or a class of mathematical terms or a plant or a movie or referring to a mathematical definition or a species specification instead." Experienced users know that work must be done to pick the right thing! In other words, one vaguely knows the answer and searches for confirmation. In other words, Wolfram|Alpha is not the Oracle! However, one does need the Oracle on occasion. {Please note that the Oracle link takes one directly to the Oracle Corporation and not to the real Oracle as intended.} The real Oracle is not yet part of the "Systemic Information Revelation"!

Figure 1 Apple Watch 2015

1.1 Fruitful Ambiguity

One seeks for information, vaguely remembered, perhaps even vaguely imagined. Our discovery of the Oracle above, is a classical example of what shall be called "fruitful ambiguity." Fruitful has the meaning of bearing fruit. The meaning is positive! But the fruit is definitely ambiguous. The classical example in our technological era is the Apple!

The Apple of the technological era provokes the idea of Time! Now, even more so, one intends the pun, with anticipation, of the iWatch? But is iWatch going to be the name! Instead, is Apple going to choose the Apple Watch? The Apple Watch will be a smart watch, being shortened to "smartwatch?" At the time of writing [2015-03-13] Wolfram|Alpha does not return an answer to the query on the iWatch.

Naturally, when one turns … as the "minute hand" turns… one might not be so sure about developments. In other words… names might change… [let us now allow for the passage of time… ?]

Time has passed! The Apple Watch is now exposed. Pictures are available. It falls under the category/umbrella of the https://en.wikipedia.org/wiki/Smartwatch [2015-04-06][start-class].

Let us pause here? Let us be a little pedantic? Let us examine the meaning(s) of "smart?" One often speaks of a person being smart. On Wikipedia, there is an article

on "SMART criteria" https://en.wikipedia.org/wiki/SMART_criteria. [start-class][2015-04-15]:

Specific: target a specific area for improvement.
Measurable: quantify or at least suggest an indicator of progress.
Assignable: specify *who* will do it.
Realistic: state *what* results can realistically be achieved, given available resources.
Time-related: specify *when* the result(s) can be achieved.

The author is presumably George T. Doran. A brief account is provided at http://www.projectsmart.co.uk/brief-history-of-smart-goals.php. The seminal ideas were published in (Doran 1981).

It turns out that there are many more other interpretations of the letters of "SMART" on the said Wikipedia page. One might summarize the smart possibilities as "encyclopedic," i.e., potentially ambiguous, to say the least.

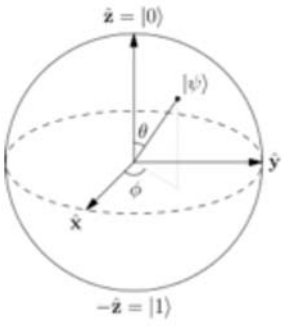


Figure 2 Bloch sphere

Unfortunately, at the time of writing, there is no Irish (Gaelic) version; nor is there a Bulgarian version. Fortunately, there is a revival in the development of the Irish (Gaelic) Wikipedia pages concerning Mathematics and Science. Ríomhaire candamach is the Irish for quantum computing; there is no corresponding Bulgarian page [2015-05-23]. To exhibit something of the nature of quantum computing one might consider the Bloch sphere, shown above in Fig.2. It looks just like the globe, a ball, the Earth? Geometrically, it is obviously a sphere. It is the standard representation of the qubit. One also immediately associates the quaternions with it. Can one not see the correspondence of i, j, k with x, y, z? [Aside: the use of Word does not readily permit the exhibition of precise mathematical notation.]

1.2 Picture me this?

"Conceptual graphs are a system of logic for representing natural language semantics. Unlike the predicate calculus, which was designed for studies in the foundations of mathematics, conceptual graphs were designed to simplify the mapping to and from natural languages." Sowa p.157.

Essentially, the graphs are pictorial. The logic is visible. The Wikipedia article [start-class] gives a succinct account: "A linear notation, called the **Conceptual Graph Interchange Format (CGIF)**, has been standardized in the ISO standard for common logic." There is no need to dwell on these matters here. Our preference has been for the ***Conceptual Reference Model,*** ECRM 140220 / CIDOC-CRM 5.1.2 (draft) http://www.cidoc-crm.org/ It does not take long to see that there is at least one way in which to catargorize quantum computing and the qubits. Specifically, there is a
<< comment [type: string]

Scope note:
"This class comprises 4 dimensional point sets (volumes) in physical spacetime regardless its true geometric form. They may derive their identity from being the extent of a material phenomenon or from being the interpretation of an expression defining an extent in spacetime. Intersections of instances of E92 Spacetime Volume, Place and Timespan are also regarded as instances of E92 Spacetime Volume. An instance of E92 Spacetime Volume is either contiguous or composed of a finite number of contiguous subsets. Its boundaries may be fuzzy due to the properties of the phenomena it derives from or due to the limited precision up to which defining expression can be identified with a real extent in spacetime. The duration of existence of an instance of a spacetimevolume is trivially its projection on time."
Naturally, one need to have some concrete examples of this so-called spacetimevolume:

Examples:

1 the spacetime Volume of the Event of (Julius) Caesar's murder [15 March 44 BC].
https://en.wikipedia.org/wiki/Julius_Caesar [B-class][2015-05-23].

2 the spacetime Volume, where and when, the carbon 14 dating of the Schoeninger Speer II took place in 1996. Specifically: concerns "[an about 400.000 years old Palaeolithic complete wooden spear found in Schoeningen, Niedersachsen, Germany in 1995]"
http://erlangen-crm.org/docs/ecrm/120111/classes/E16Measurement___-39440126.html

3 the spatiotemporal trajectory of the H.M.S. Victory from its building to its actual location
https://en.wikipedia.org/wiki/HMS_Victory [B-class][2015-05-23]

4 the spacetime volume defined by a polygon approximating the Danube river flood in Austria between 6th and 9th of August 2002.
https://en.wikipedia.org/wiki/2002_European_floods
A start-class article from Wikipedia, the free encyclopedia.
A former good article nominee.

1.3 Quantum computing

One might ask what practical use can we make of quantum computing? It does seem to be an esoteric subject with respect to the business of the real world. One might ask can it really be applied to our subject discipline of Information Systems and Grid technologies? To make a first shot at an analysis, let us consider a very standard business approach? There are many kinds of pictures intended to show various aspects of a business.

Consider the Gartner corporation (Gartner, Inc.)? Hype Cycle
https://en.wikipedia.org/wiki/Gartner [start-class][2015-04-15]

One of its most successful pictures is the well-known "Hype Cycle," a ***branded*** graphical representation to represent the *maturity*, *adoption* and *social application* of specific technologies. It is to be noted that there is not only a Wikipedia article on the Hype Cycle (unassessed), but also an accompanying diagram shown above in Fig. 3.

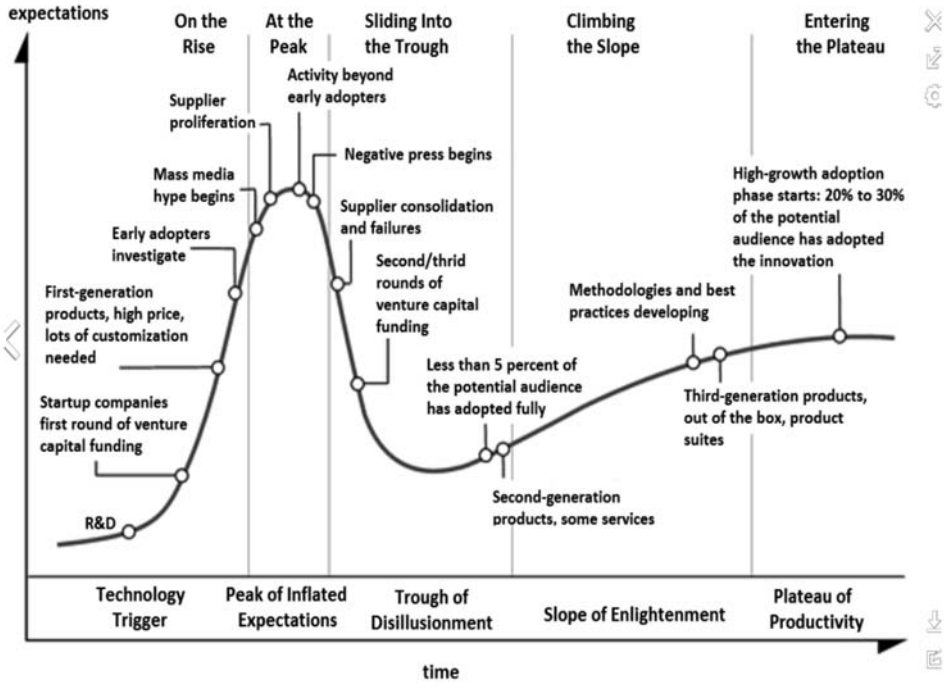


Figure 3: Wikipedia exposition of the Hype Cycle

Technically, anyone can decide on its degree of "hypedness." The picture tells a story.

The story may be read in the sequence:

> **T**echnology trigger… [R&D]? // [author: **R**ocket launch…]
> **P**eak of Inflated Expectations… [On a high]? // [author: **S**ummit…]
> **T**rough of Disillusionment… [In the Doldrums]? // [author: **D**ecline…]
> **S**lope of Enlightenment… [Now I get it]? // [author: **B**ounce back…]
> **P**lateau of Productivity… [Churn it out]? // [author: **S**hoot for the stars…]

So! Let us run with this story? It is the normal one. Being in a trough is depressing. One wonders if there might be a ***disruptive*** way to break out?

Trough of Disillusionment

Instead of following the usual upbeat climb (i.e. getting out of the doldrums, getting out of a hole), one might pause and consider a radical disruption?

II. Human Information Interfaces: People matter

Technologies of Control

*"A variety of technologies of control have emerged from the intertwined interests of commerce and governments. There are technologies of **identification**, of **surveillance**, and of **investigation**.*

Technologies of Identification
Show me your ID! (Apple uses fingerprints?)

Technologies of Surveillance
Cameras are everywhere! (CCTV stalks us?)

Technologies of Investigation
Seek and ye shall find!

Refer to the building of databases from the results of both surveillance and storage of routinely recorded information (Garfinkel, 2000).

III. Qubits: Quantum view of Life/Existence

Consider the Wikipedia article on quantum superposition!
https://en.wikipedia.org/wiki/Quantum_superposition.

> It is currently classified as start-class [2015-04-13]. Moreover, there is an exclamation banner**!**
>
> This article may be too technical for most readers to understand. (April 2011)
>
> This article possibly contains original research. (March 2010)
>
> This article needs attention from an expert in Physics. (October 2011)

Naturally, one would think that this issue would have been cleared up by now, in 2015?

If one now turns to the "Talk page":
https://en.wikipedia.org/wiki/Talk:Quantum_superposition, then one notices the almost quantum-like debate on the topic: "Some topics can't be explained or discussed in language intelligible to the average reader, nor should they be, if in attempting to water down some concept the meaning is changed or even partially lost."

More recently: "A big defect of this article is that fails to clarify how mixture and superposition differ." Chjoaygame (talk) 09:54, 4 December 2014 (UTC)
https://en.wikipedia.org/wiki/Quantum_computing.

How shall one try to tame quantum computing so that the "average reader" might feel comfortable to grasp the essentials?

"Welcome to Oxford Quantum! Oxford University is the UK's largest and most diverse centre for quantum research. We have 38 separate research teams, with a total of around 200 researchers. Oxford is therefore one of the world's largest centres for quantum science."

http://oxfordquantum.org (2015-04-06);
http://nqit.ox.ac.uk (2015-04-06);

How might one verify this Oxford claim, independently?

A rival to Oxford is, naturally/historically Cambridge.

http://www.cl.cam.ac.uk/teaching/0910/QuantComp/notes.pdf
http://www.cl.cam.ac.uk/teaching/1415/QuantComp/
http://www.cl.cam.ac.uk/~ad260/

Consider the Wikipedia article on quantum superposition https://en.wikipedia.org/wiki/Quantum_superposition. It is currently classified as start-class [2015-04-13]. Moreover, there is an exclamation banner**!**

1. This article may be too technical for most readers to understand. (April 2011)
2. This article possibly contains original research. (March 2010)
3. This article needs attention from an expert in Physics. (October 2011)

Naturally, one would think that this issue would have been cleared up by now, in 2015? If one now turns to the "Talk page":
https://en.wikipedia.org/wiki/Talk:Quantum_superposition, then one notices the almost quantum-like debate on the topic: "Some topics can't be explained or discussed in language intelligible to the average reader, nor should they be, if in attempting to water down some concept the meaning is changed or even partially lost."
More recently: "A big defect of this article is that fails to clarify how mixture and superposition differ." Chjoaygame (talk) 09:54, 4 December 2014 (UTC)

Conclusion

The title "Quantum Bits" was deliberately chosen to hint at intrinsic ambiguity. A "quantum bit" is conveniently abbreviated to "qubit." To further entangle the understanding of "Intelligent and Information Systems and Grid technologies," the qubit has been deliberately deployed strategically. Further entanglement may be introduced via
the https://en.wikipedia.org/wiki/Homophone Qubit[start-class] [2015-05-23]
and https://en.wikipedia.org/wiki/Cubit [start-class][2015-05-23].
The challenge… ultimately is to figure out which is which!

Disruption

"Bitcoins really are useful. But not in the way you think." — John Naughton.

The paper has focused on various ideas that might enliven, reinvigorate some aspects of the Intelligent and Information Systems and Grid technologies (IISG). Information is at the heart of the story. One remarkable modern disruption is the creation of Bitcoins. The thing that makes Bitcoin work is the blockchain. But for this conference, it is not the monetary aspect that matters! Instead, the focus is on the blockchain! Specifically, it has just been announced that "the state of Honduras, one of the poorest countried in Latin America, announced that it was going to use a blockchain to build a permanent and secure land title registry." Another analogous disruption is provided by "ascribe, a startup that enables artists to register a legal claim to the copyright on digital works they have created. The company timestamps those claims on to the bitcoin blockchain. When artists transfer their rights (eg., by selling the work) the blockchain confirms that they are transferring the copyright (in the form of a licence) to the new owner.

The author recognizes the automatic self-entanglement introduced and induced throughout the present paper. The goal was a self-analysis re IISG. By reflection, one also wonders how the future of IISG might evolve? The presentation shall speak for itself.

References

The usual organization of references: I Books, II Articles, III Electronic Books, IV Electronic Articles, V Other.

Books:

Garfinkel, Simson. Database Nation: The Death of Privacy in the 21st Century. O'Reilly Media, Inc. ISBN-13: 978-0596001056.

Moore, Walter John. Schrodinger, life and thought. Cambridge University Press. ISBN 0-521-43767-9. 1989

Nielsen, Michael A. & Chuang, Isaac L. Quantum Computation and Quantum Information. 10th Anniversary Edition. Cambridge University Press. ISBN 978-1-107-00217-3. First published 2000.

Sowa, John F. (Editor). Principles of Semantic Networks. Explorations in the Representation of Knowledge. Morgan Kaufmann Publishers, Inc. San Mateo, California. ISBN 1-55860-088-4. 1991.

Articles:

Haughton, John (The Networker). “Bitcoins really are useful. But not in the way you think.” The New Review, The Observer, 24.06.15, UK.

Code and Other Laws of Cyberspace
https://en.wikipedia.org/wiki/Code_and_Other_Laws_of_Cyberspace [start-class][2015-04-15]
Doran, G. T. (1981). "There's a S.M.A.R.T. Way to Write Management's Goals and Objectives," Management Review, Vol. 70, Issue 11, pp. 35-36.
Quantum bit [= Qubit]
https://en.wikipedia.org/wiki/Qubit [start-class][2015-04-20]

https://www.apple.com/watch/gallery/#apple-watch-sport-silver-aluminum-case-white-sport-band [2015-05-21]

Organizations

Gartner Group
https://en.wikipedia.org/wiki/Gartner [start-class] [2015-03-11]

Oracle
http://en.wikipedia.org/wiki/Oracle [C-class]

Mathematics & Computing

Bitcoin https://bitcoin.org/en/, https://bitcoin.org/bg/
Quantum computing https://en.wikipedia.org/wiki/Quantum_computing [B-class] [2015-02-18]
QuBit http://mathworld.wolfram.com/Qubit.html [2015-05-22]

Miscellaneous

https://en.wikipedia.org/wiki/Smartwatch [start-class][2015-04-06]

People

https://en.wikipedia.org/wiki/Felix_Bloch [start-class][2015-05-06]

Usage and Analysis of Game Development Tools for Android Mobile Operating System

Saule Sarsimbayeva[1*], Bereket Kamash[2]

1- Mathematics and Physics Department, K.Zhubanov Aktobe Regional State University, 7 Grishina Str., Aktobe 030000 Kazakhstan
2 - Mathematics and Physics Department, K.Zhubanov Aktobe Regional State University, 7 Grishina Str., Aktobe 030000 Kazakhstan
* - corresponding author (saulesarsi@gmail.com)

Abstract. The article is devoted to the problems of using and analyzing game development tools for Android mobile operating system.

Keywords: software, smartphone application, programming, Android, IDE

Nowadays a rapid increase of high capacity smartphones and growth of mobile applications have made application developers pay attention to the new mobile platforms - Android, iOS, Blackberry. Such mobile digital distribution services as PlayMarket and AppStore effected the rates of mobile applications market development. Simplified registration procedure in these services enabled the developers to get a profit from selling their mobile applications. Mobile games being a large segment of mobile applications market show their significant growth. Nowadays smartphones can compete with game consoles and personal computers for being the most popular game platform. It forced game development companies to add support of the most popular mobile platforms for their games. The motivation for the studyis determined by the demand for mobile games and loads of professional game development tools. This work is devoted to the analysis and usage of game development tools for virtual reality mobile games.

Game engine is a universal game development tool that combines a graphical engine, a physical engine, a script editor, a compiler and a level editor. These components are basic for every game engine. Game engines provide main technological reliability and performance of the project and they are usually platform irrespective. Modular architecture of the game engine provides an opportunity to select its components. For example, game developers can change built-in physical engine for the more specialized one.

Graphical engine [1] is a program application that responds for drawing game graphics. It's an important component of the game engine. Some of the game engines consist of graphical engine without other components. Major difference between game graphical engines and rendering engines is that the game graphical

engines provide real-time rendering and visualization. The capabilities of the graphical engines is not enough to create a modern video game, that' why game engines should include other components.

Physical engine [1] is a computer software that provides an accurate modeling of certain physical systems based on mathematical calculations. They are mainly used in video games and scientific spheres. In first case, there aresome strict requirements foran accurate modeling, but not for the calculating speed. In the second case, the calculating speed is highly précised, particularly in real time. Physical game engines provide such kindsof modeling as rigid body dynamics(including collision detection), soft body dynamics and fluid dynamics. Physical engine uses such abstractions as a body which has its form and number of parameters, a link being game physics limitations. When existing algorithms of the physical interaction aren't enough, the developers may create their own ones as the most of physical engines give that opportunity.

Audio engine [1] is a component consisting of any algorithms related to sound and acoustic phenomena. The most known audio engines are Environmental Audio Extensions, OpenAL, DercictSound3D.

In this work the virtual reality game was developed, existing game engines were analyzed among which the most suitable one was selected.

App Game Kit [2] is a development of "The Game Creators" company. The Engine is a special development environment which was designed for the programming of the games for various target platforms. The list of the supported platforms includes iOS, MacOS, Windows, SamsungBada, MeeGo. Game development enthusiasts, which are fond of game development, choose this engine. The AGK includes 2 interpreters: Basic and C++. Programmers write scripts using one of these programming languages, then the compile project in handy IDE. The licensing policy is flexible and allows small companies and giants of the game development industry to use the engine.

Project "Anarchy" [2] is a product of the Havokcompany. The main components of the engine are HavokPhysics physical engine, HavokVision graphical engine, HavokAI – game artificial intelligence module, WYSIWYG level editor. Such programming languages as LUA and C++ are available for the developers. Such platforms as Android, Tizen and Windows are supported by the engine. The engine will be available to everyone for free if anadvertising company is hold jointly with the Havokcompany. Commercial version of the software unlocks additional services. The Havokcompany is a leader of the virtual physics and artificial intelligence, that's why those engine components are considered to be reliable.

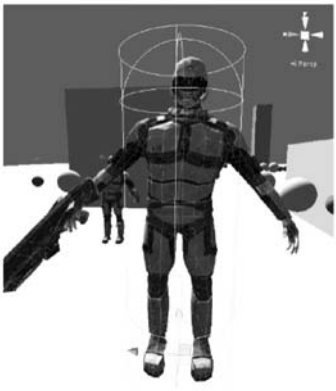
Cry Engine 3 [2] is a development of Crytekcompany, a descendant of Cry Engine 2. This game engine is the most popular choice of the game developers. One of the main reason of its popularity is a demonstration of its efficiency on the

example of Crysis 2 video game that had the most advanced graphics when it was first released. It worse to mark Ubershadershader system, used in Cry Engine3, that helps to create complicated visual effects. Graphical abilities of Cry Engine 3 allow to create such effects as reflections, refraction of light, effect of volumetric fever and flecks of sunlight. Maximal screen resolution is 7680x3200 pixels. Technology implementing this screen resolution is exclusive for Radeon 5000 – series video cards. The Crytekcompany has several engine licensing options.

Unreal Engine 4 [2] is a development of the Epic Game studio and it is one of the most advanced game engines. Technical demonstration was heldin June 2012. The main feature of this game engine is an access to source codes. They could be easily found in github-repository. Programming language is C++. Synchronization with Visual C++ is implemented. A hoy reboot system allows to edit a code when application is running as well as to estimate changes without stopping the game. An engine provides a wide range of tools for working with graphics: a visual shadereditor which allows to create complicated shaders without necessity to write them manually. There is a system of animation control called Matinee. An engine supports advanced abilities of DirectX 11, 12 visualization. CascadeVFX particle editor allows to create detailed effects of flame, snow, smoke and dust. It allows to calculate high-performance particle simulation and collision detection and illumination from each of million particles. License policy of the Epic Games studio allows everyone to use Unreal Engine 4fully andfor free.

Unity 3D [2,3] is 2D and 3D game development tool available for Windows and OSX. This cross-platform game engine allows to create games for all platforms including web-browsers. This advantage, on the other hand, is a disadvantage of the engine, because cross-platforming brings additional computational loads. Nvidia Physics is used as a physical engine. A list of build-in tools includes a landscape editor which allows creation of sinuous surfaces. There are three programming languages such as Boo, C# and JavaScript. Game project consists of one or multiple scenes which store their realms. Basic unit of the scene is an object.

A detailed review and analysis of multiple game engines and packages of 3D graphics were made in this research work. Unity3D as game engine and 3D Max as package of 3D graphics arethe most optimal choice. The choice was made taking into account all the advantages and disadvantages of reviewed options and specifics of the project. A process of the game development on the example of virtual reality game developmentwasstudied in details. Such aspects of game developing as implementation of main and minor aspects of gameplay, planning and filling of game level were described in details. A mobile virtual reality game was the finding of the current study. Game character is shown on picture 1.



Picture1. Game Character.

A review and analysis made in the work could help novice game developers get an idea of instrumental tools required for the game development as well as discover advantages and disadvantages of the most popular game development tools.

References

[1] Gregory J. Game Engine Architecture. – A K Peters, 2014. – 1052 c.

[2] Rabin S.,Game AI Pro: Collected Wisdom of Game AI professionals.–A K Peters, 2013.– 626 c.

[3] Gibson J. Introduction to Game Design, Prototyping, and Development: From Concept to Playable game with Unity and C#. – Addison-Wesley Professional, 2014. – 944 c.

The Development of "Multi-Field Search" Forms to look into the Library System of Scientific Organization

Kalina Ilieva[1] and Svetlana Vasileva[2],

[1] Dobrudzha Agricultural Institute - General Toshevo
9500 General Toshevo, Bulgaria
[2] Shumen University „Bishop Konstantin Preslavski", College-Dobrich, Dobrotitsa 12
9302 Dobrich, Bulgaria

{kalito_21, svetlanaeli}@abv.bg

Abstract. The paper considers "multi-field search" forms, which are an integral part of a project to implement an information-retrieval system designed for the needs of the research library. The system was developed two years ago by means of MS Access 2003. The project aims to facilitate the search of the specialized literature by researchers, users of the system, as well by the staff of the library of the Dobrudzha Agricultural Institute - General Toshevo municipality. Thanks to the developed "multi-field search" forms, the demand of the researchers for the desired information (by more than one criterion) it is optimize significant.

Keywords: search, "multi-field search" forms, information-retrieval system, users.

1 Introduction

The forms serve as the primary interface between users and the information system. Designing search forms in one information-retrieval system aims to optimize the search for the desired refund quote. In this case it is to search for literature in library science in the field of agricultural sciences. The developed system of scientific library is an advantage in terms of not only the demand for information, but also of the ongoing daily library processes - registration, logging, an inventory, loan, etc.

The library users occupy a very important part in the development of the system. With their help, the system is functional even in terms of the demand for information about a research team is constantly necessary and valuable.

2 The Switchboard of the Library Information System

Fig. 1 shows the system startup form, called Switchboard. Original startup form of contact with users of the Library Information System is presented in [1]. The form contains six basic forms, as when selecting each of them, they reference

to other forms satisfying consumer interests. Unlike the original version of the system here has added a new form to search for scientific publications. For the research team that is a convenience to search for posts directly. It can be searched by author, keyword or edition. Each researcher works daily with publications and this would save time, even just in terms of what can be informed.

Fig. 1. Switchboard of the library system.

3 The forms „Search" and „Advanced Search"

Fig. 2 illustrates a search form that includes five search fields. The form was created in Design view. The form is "multisearch", which allows user to search more than one criterion. To this end, multiple queries are used, which correspond to each criterion separately. The more criteria contain a form the more useful is the search for requested information and the better the results display system. For users of library science that is a functional form that meets the most elementary questions the system.

On Fig. 3 can be seen a search for a collective author who for scientific organization is common, as the library has many publications issued by universities, institutes, academies, etc. Scientific Library keeps many editions of Agricultural Academy. When looking for Collective author does not need to be displayed full name, user can display only the abbreviation "SAA".

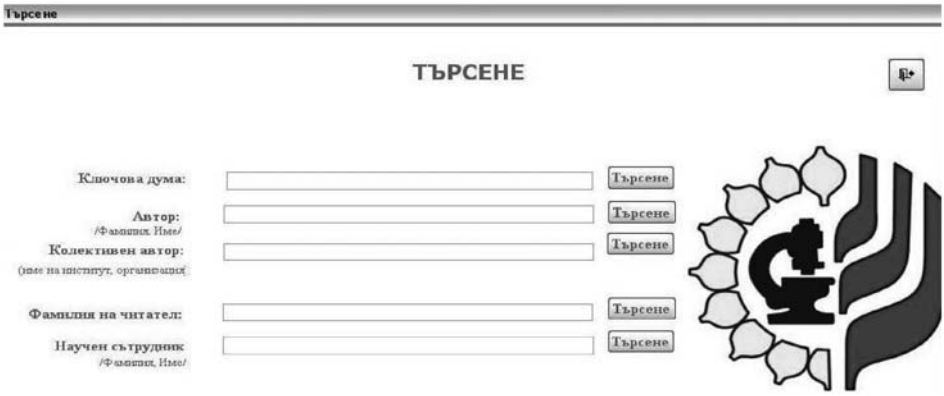


Fig. 2. Form for the consumer demand.

Fig. 3. Search Result for Collective author.

In search forms for each desired criterion a request is made, and participates in it an expression which is less complex but has its specifics [2]:

For example, the criterion for „Search by Author“ expression will type:

```
Like “*” & [Forms]! [SearchF_Users]! [Text56] & “*”,
```

which is understood as follows:

```
Like “*” & [Forms]! [Form Name]! [Search Criteria] & “*”,
where [Tekst56]
```

appears sought expression that can be a keyword, author’s name, reader, etc.

Such an example is shown in Fig. 4, where user can search by author, meaning by his family name. As can be seen from the result, we have two authors in with family Kolev. It is advisable to seek a family, as it appears in the tables at their introduction.

Fig. 4. Search Result for Collective author.

Another further developed for the needs of users is an advanced search form, shown in Fig. 5.

Each search criteria as "*Books*", "*Magazines*" and others is made through a separate request matching your search criteria. In this form are added two new search fields [1]: *Materials of conferences* and *Series*. Creating more search fields is convenient not only for scientists who constantly publish articles and looking constantly in collections, series, conference proceedings, but also for developing, facilitating the work of serving the research team.

Fig. 5. Advanced search form.

The seeking forms are a tool to optimize the performance of both library professionals and consumers scientists in the process of searching for the desired information.

Fig. 6 shows search using the form keyword “hel”. As already mentioned is not necessary to be displayed the whole word searched. Using the expression above, including the operator Like it only searches the database. Fig. 7 shows search scientific journal by the Advanced Search form.

Fig. 6 Search by series in the Advanced Search form.

Fig. 7 Search Result on magazines from the Advanced Search form.

4 Search form for science publications

This new form has been developed further in another way, and includes only a query, unlike the forms described above. Again using specific expression operator Like. [3], [4] User can search by author (only Research Fellow), publication (keyword) and by edition.

The request includes only those tables which will be used, and will be easy to manipulate the information in them. Most scientists are interested in the title of the publication in which edition is published and the year of issue.

This form is simplified but also easy to work with it. For example, if researcher searches by author, the user gets information on both the publication

and the magazine, which was published for the signature of publication, number of volumes, number of books, which are the page numbers, year of issue.

As it is shown in Fig. 8 the search for the author of the publication provides information and co-authors of the article designated. This is useful for user search. These data are important and valuable in shaping bibliographic lists of publications, which often face researchers in their daily lives.

The form is useful for scientists, containing the necessary data and for the library specialist who may at any time to provide timely and relevant information on any new scientific publication.

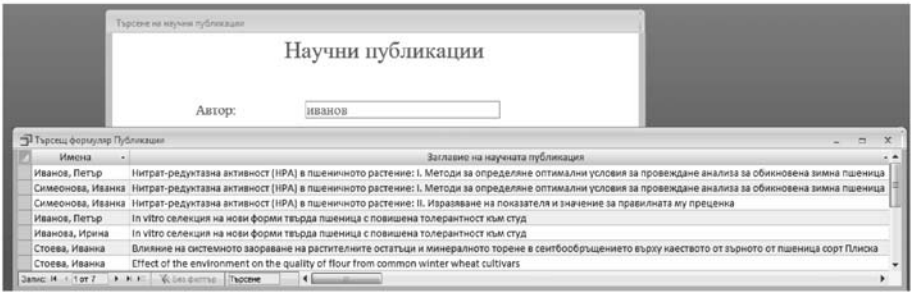


Fig. 8 Search Result by author of scientific publication.

5 Queries for creation of search fields in the forms

For the development of these forms are executed several important steps, and one of them are the queries which are built on the basis of the expression written above (Paragraph 3) with the operator Like [2], [3], [4], etc.

For the implementation of the query are chosen the desired tables from the database. The other point is to choose a box which will seek the desired information, such as title, the most common search criteria. Of course, we must first do a basic form to use it for searching particular information.

In our case, the form shown in Fig. 8 shows clearly the seeking field and the search results. It is evident that searching by author surname “Ivanov” as an author are displayed all authors with that surname plus coauthors of specific scientific publication. Initially, the user can display the first co-author of the publication and still obtain the desired information. This is an advantage of this form because it is sufficiently informed.

Returning to the query for the establishment search fields besides creating a basic form is important the expression displayed in the search box on the query and also the relationship with the query form to get to putting the results.

The specifics in this form is that there is no separate request for each field, as in the other forms described above, it is used only one single query in the case.

The expression, written in the criterion looks like this:

```
Like "*" & [Forms]![Search_publication]![Imena] & "*"
Like "*" & [Forms]![Search_publication]!
[Zaglavie_nauchna_publikacia] & "*" Like "*" &
[Forms]![Search_publication]![Zaglavie] & "*"
```

From the expression in the example shown, the form is entitled *[Search_ publication]* and the search field is *[Imena]*, which includes the surname and name of the author. The same applies to other fields.

This kind of queries are parametric requiring an input value, which is the criteria that will be sought, whether title, year of publication or release, etc.

It is very important, how it is built and whether the query allows for demand and which fields will be sought. It is important that the fields that will be used to have a lot of "information" to incorporate more information charge. Therefore, the system is an information-seeking and forms must meet this criteria.

Developed forms are tailored to consumer needs and interests. Developed based on inquiries addressed to library professionals in the science library. For example, often looking for information on selection and genetics of wheat, sunflower and other crops and therefore there is a request to filter titles include these keywords. The results are consistent with the criteria in the application, which is ancillary to systematize the large volume of information. Therefore the involvement of consumers in the development is very valuable for the system to be more functional, flexible and effective.

6 Conclusion

The development of the search forms on an information system is a complex process. Displayed above forms are simpler, which is advantageous when used by the user.

For an information system in such organizations the search is very important, especially for the research team, seeking always up to date scientific information.

The seeking forms are an effective means of communicating with the user system, but also a challenge for the developer, which operates continuously with her familiar in detail with the user's needs and constantly tested.

These forms are "calling card" of an information system.

Applications created with MS Access should not be underestimated because they are a stable means of developing information systems that do not require funds to maintain. Information system for scientific library is constantly in the process of improvement until the complete and efficient system. It needs more complex search forms and forms for data entry convenient for staff. Considering to make an application using SQL Server 2012 MS and PHP, but unfortunately it will take time and it is difficult to achieve.

Creating a database is a difficult and time consuming process, because always the addition of new tables, queries, forms. Perhaps most suitable programming language is Visual Basic, which is the most compatible with the application, but is yet to be developed. The use of additional programming language is an important step in the completion of the system and its improvement as a prerequisite for soundness and quality.

Completion of the system is an advantage of such an organization, storing thousands of valuable volumes of scientific literature in the field of agricultural sciences. Therefore, guarantees the effectiveness of the system is the continuous testing, which helps to avoid errors and adverse outcomes. Before testing, a very important place has the processes related to the construction of relational scheme, the creation of applications and best looking and qualitative forms corresponding to the user requirements.

Scientific organization is always demanding and similar system except that inform consumers also promotes scientific information in the field of agricultural science and to scientists and to anyone who wants to be informed in this scientific field.

References

1. Илиева, К., С. Василева.: Разработка поисковых форм для информационной системы обслуживающей научную библиотеку. In: Сборник научных трудов по материалам VII Международной научно-практической конференции „Теоретические и прикладные аспекты современной науки", Часть II, сс. 154—161. Агентство перспективных научных исследований, Белгород, 2015. / Ilieva, K., Vasileva S.: Development of seeking forms for information system servicing scientific library. In: Proceedings of VII International conference on theoretical and applied aspects of modern science, Part II, pp. 154--161. Perspective Investigations Agency, Belgorod, Russia, 2015.
2. Austin, R. Using the Like operator in queries. Access All In One .http://www.accessallinone.com/using-the-like-operator-in-queries. (2013)
3. Harkins, S. 10 Tips for using wildcard characters in Microsoft Access criteria expressions. TechRepublic. http://www.techrepublic.com/article/10-tips-for-using-wildcard-characters-in-microsoft-access-criteria-expressions/ (2007)
4. Rost, R. Build a Multi-Field Search form. Access Learning Zone. http://www.599cd.com/tips/access/multi-field-search-form-like/Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181--184. IEEE Press, New York (2001)

Aknowledgments This Paper is supported by Project N РД-08-277/11.03.2015 of Shumen University "Bishop Konstantin Preslavski"

Performance Study of SQL and Graph Solutions for Analytical Loads

Emanuela Mitreva[1], Hristo Kyurkchiev[1]

[1]Faculty of Mathematics and Informatics, Sofia University, 5 James Bourchier blvd., 1164, Sofia, Bulgaria

emitreva@gmail.com, hkyurkchiev@fmi.uni-sofia.bg

Abstract. NoSQL has become an extremely popular term in the database world. We take one of its concrete instances - graph databases and more precisely Neo4j on its promise of great analytical performance and compare it against a traditional relational DBMS (Oracle) and other NoSQL siblings (Vertica, MongoDB). Starting with the data models of each some preliminary hypothesis is build about each one's strengths and weaknesses. These are checked by using our previously developed performance benchmarking model. Stepping on the results we are able to make a list of all the advantages and disadvantages of each and come up with a recommendation on the usage and suitability.

Keywords: Database systems, Relational databases, Data warehouses, NoSQL, Column stores, C-Store, graph databases, Neo4j, performance evaluation, analytical queries

1 Introduction

Data mining has become increasingly popular area in recent years, thus the requirements for tools to handle big data and efficiently retrieving information or important patterns are more and more demanding and data mining could be used on different levels [5]. Therefore the research into database management systems (DBMS), which can provide fast performance for analytical queries, has great significance to the business world. In a series of articles [4], [6], [7], [8], [9] we are trying to find the best solution for such purposes by making performance tests on different types of data stores. In the overview article [10] we have presented the alternative data stores – key-value, column-based, document and graph stores. They all have their advantages and disadvantages considering how they store the data, how efficient they are for DML statements or for selects, etc. In the first article [6] we have considered a column store solution - Abadi's Vertica [8], the commercialization of C-Store [1], [2], [12] as a prominent example of its group. Vertica is a DBMS, which stores data in columns, rather than rows, providing both the standard SQL language for querying databases, and the performance needed for effective data mining. In previous research [6] we aimed at researching how

Vertica compares to Oracle - a commercial grade row store. We found out that by using Vertica instead of Oracle one achieves significant performance gains, which are, however, not in the same order of magnitude as previously suggested [15]. In the next article [7] we decided to choose a more modern solution – MongoDB, a document store NoSQL solution [11]. The setup of the data was done to mirror the Oracle and Vertica's structure and whether because this (document store does not support joins and the data from the different tables was stored in different collections, thus making alternative ways of joining the data a necessity) or because of the performance of the store itself, we achieved significantly low results, specially when compared to Vertica. Our next candidate, as outlined in previous research [7] were the graph stores – another NoSQL solution with a different data model and structure.

2 Architectural overview

2.1 Neo4j data model

Considering the problems and the low performance of MongoDB for our queries and data set, we have decided to try a data store, focusing more on the importance of the relationships between the data objects than on the structure of the data objects themselves. The assumption is that if the focus is on the relationships, queries with joins will be executed much faster. As the world's best and first graph database [14] Neo4j seemed the logical choice for our performance.

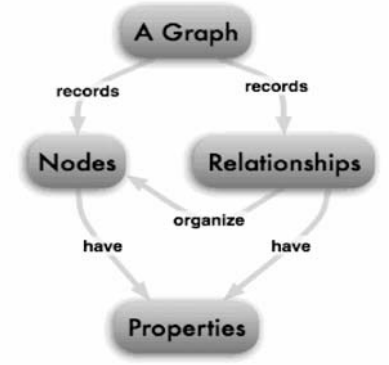


Fig. 1. Graph **database structure**

Logical and physical database structure. The Neo4j solution is a graph database, i.e. the data is stored in the form of graph like it is shown in Figure 1. A graph has nodes and edges – the records are the nodes and the edges are the relationships between the edges. There is no limitation to how many edges a node could have, thus making the graph store a good approach in the case of a many-to-many relationship. As in MongoDB, the schema of the records/nodes does not

need to be defined beforehand, but the nodes are defined upon insertion. This mutable schema allows the user to introduce new attributes (in Neo4j are called properties) at any time, unlike Neo4j the relational model's rigid schema makes it hard for any changes to be made [3].

Read optimization. Neo4j has high performance read and write scalability, without compromise. It delivers fast read and write performance, while still protecting the data integrity. It is the only enterprise-strength graph database that combines native graph storage, scalable architecture optimized for speed, and ACID compliance to ensure predictability of relationship-based queries [14]. As a result of its characteristics and strengths, Neo4j is really suitable in the cases when:

- Queries with many joins need to be executed;
- Some hierarchical data need to be retrieved;
- There are a lot of many-to-many joins or tree-like data structures;
- The data is already in a graph form (e.g. information about who is friends with whom in a social network) [13].

2.2 Compared to Vertica and Oracle

There are both similarities and differences between Neo4j, MongoDB, Oracle, and Vertica. A comparison of the main properties is presented below.

Table 1. Basic comparison between Neo4j, MongoDB, Vertica and Oracle

Characteristic	Neo4j	MongoDB	Vertica	Oracle
Data storage	Nodes and edges	XML or (B)JSON	Columns	Tables, rows
CAP	Consistency and Availability	Consistency and Partition tolerance	Consistency and Availability	Consistency and Availability
ACID rule	Yes	No	Yes	Yes
Transactions	Yes	No	Yes	Yes
JOINs	Yes, with predefined relationships	No	Yes	Yes
Indices	Yes	Supports single and compound indices on every level of the JSON	Does not support indices at all, uses projections for optimization	Supports single and compound indices on all columns
Replication	Yes	Yes	Yes	Yes
Sharding	Yes	Yes	Yes	Yes

As we can see from Table 1 we have been researching for the best solution among databases that store the data in different format – JSON, columns, tables and rows and now for the latest example – Neo4j - uses nodes and edges and is optimized to store data that is related in some way or even have hierarchy. The

data on which we are making the performance study have several tables, which have foreign keys to the other tables, so this will be viable data sample to be used for the graph store. This solution also adhere to the ACID properties and supports indices, which leads us to the hypothesis that Neo4j will be reliable and fast enough, but probably not as fast as Oracle and even Vertica when making DML queries (insert, update, delete) and would be similar to MongoDB in this regard. It, however, would be much more efficient when it comes to OLAP processing due to the optimization of the graph structure toward the relationships.

3 Experiment setup

In this section the main aspects of the experiment such as the hardware and software, the database schema, the benchmark, and the measuring tools are discussed. Only a part of the original database schema is presented, as it is proprietary information.

3.1 Setup description

In order for the comparison to be valid the same setup as in the original performance study [6] is used. For completeness, we have included a brief outline bellow.

Hardware setup. The DBMSs were run as virtual machines on VMWare ESXi, each equipped with 2 CPU cores (2.4GHz each), 6 GB of RAM and 16 GB of storage.

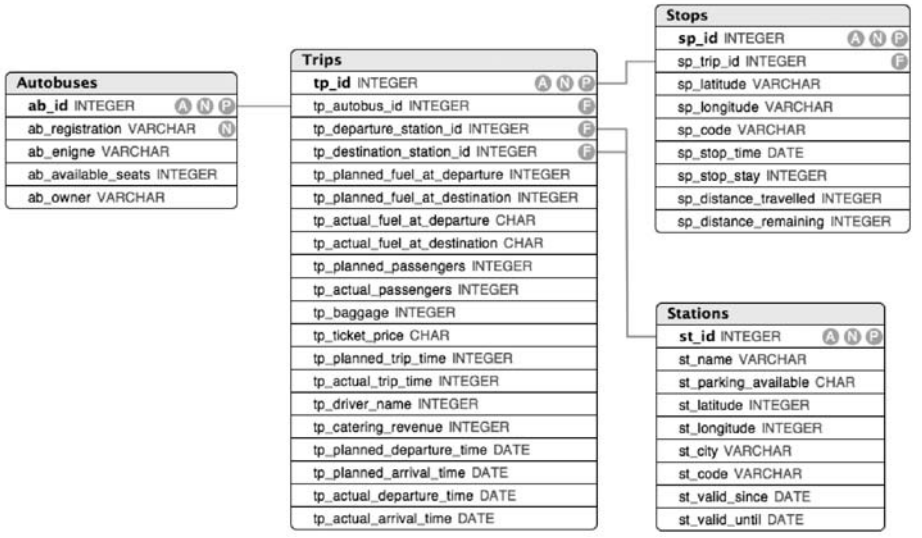


Fig. 2. Database

Database setup. No specific tweaks have been performed on each database to improve performance, except the ones, which are implicit or completely trivial:

- For Oracle, Oracle 11g SE One was used with implicit indices on all of the surrogate keys, as well as explicit indices on all of the foreign keys.
- HP Vertica Community Edition was used as the Vertica instance with the compulsory super projection on each table.
- The latest version of MongoDB (2.6.1) was installed on server with indices on the same columns as the ones in Oracle.
- The latest version of Neo4j (2.3) was installed on server with indices on the same columns as the ones in Oracle.

Database schema. The same schema as the one used in previous research [4] (Figure 2) is employed here as well – three dimension tables and one fact table. The data used is from a travelling agency and concerns autobus trips. It includes autobus data (~ 50 records), station data (~ 5000 records), trips data (~ 100000 records), and stops data (~ 1750000 records), which have been transferred to Neo4j without any changes.

3.2 Benchmark

In order for the results to be comparable, we will use the same set of queries, used in the previous articles [5], [6] and the same hardware. Of course, since the queries were originally written in SQL and Neo4j does not support SQL for querying data, they had to be rewritten in Cypher – one of the native query languages of Neo4j. Neo4j also provides REST APIs to manipulate the data or to retrieve it, but we have decided that using the Cypher language will be more close to the research done in the previous works. The Cypher language is not that different from the SQL language, when it comes to the simple queries and has the same basic clauses, but a slightly different syntax. A select query has the following basic syntax:

MATCH (alias:TABLE) WHERE conditions RETURN alias.col1, alias.col2;

A delete statement has the following form:

MATCH (alias:TABLE) WHERE alias.ID = 1 DELETE alias;

An update statement looks like this:

MATCH (alias:TABLE) WHERE alias.column = ‘value’ SET alias.column2 = ‘value’ RETURN alias;

Those statements, although a little different than the SQL syntax have similar logic and elements. However, what is the most interesting in the graph stores and the reason why we have chosen such a store are the relations. In this type of database, the relationships are links between nodes (records), which makes the execution of the queries more efficient. In the common case of a query with joins executed on a relational database, the query is scanning Cartesian product

of the tables and then it is filtering only the necessary records. In the case of the graph store, the nodes that are not connected to nodes that should be visited are not scanned; therefore the query does not need a full scan of the nodes to get the result of a query. Another thing that is important to be noted is that because of the afore mentioned feature of the graph store the execution time of the queries is not as dependent on the number of records as it is in the relational model.

When we need to make use of the relations between two nodes, we should use the following syntax:

MATCH (alias:TABLE)-[:RELATION]->(alias2:TABLE2) WHERE statements

RETURN alias.col1, alias2.col2;

The relation between the two nodes is named RELATION and unlike in the relational model, where the name of the foreign key is not important and it is used more for the data to be consistent, in Neo4j behind the name RELATION is the actual relation between the nodes and it is making the link (edge) between the two nodes.

4 Performance comparison

Aqua Fold's Aqua Data Studio was used to measure the response time and the figures were generated using Shield UI. All queries have been run multiple times and the results were averaged so that any differences are smoothened. It should be noted that for response times of less than 1 second the measured results varied significantly on Neo4j.

4.1 General queries performance analysis

Figure 3-5, which contain the results for the DML statements and selects, show that the patterns of the results for Oracle, Vertica and Neo4j are on par, except some minor difference in the case of the insert in stations and delete of an autobus in Vertica's case, which can be attributed to a measurement error. In comparison MongoDB is significantly behind in almost all DML statements. Unlike the case of Oracle and Vertica it is worth noting that both MongoDB and Neo4j offer different execution times if the statement is rerun and while with MongoDB we have not seen any pattern, Neo4j shows a clear declination with the number of subsequent reruns.

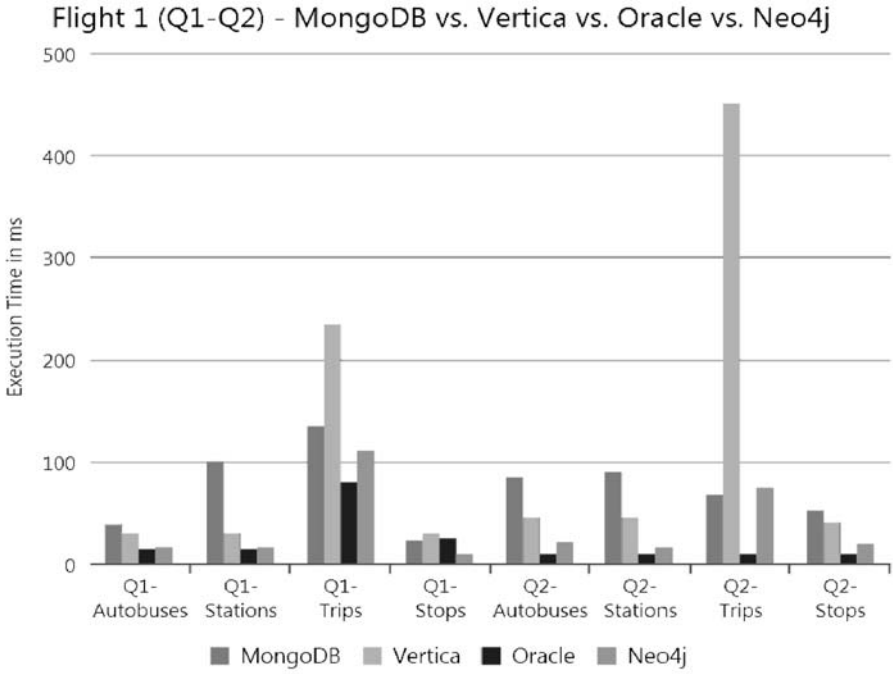


Fig. 3. Simple inserts and updates

The rest of the flight consists of simple select statements. We see a very similar pattern, which matches the one we saw in Vertica in the previous study [6]. The results show a significant degradation in performance. As expected high selectivity is not the best querying scenario for a graph database as Neo4j. It should also be noted that the last two queries were not performed because of the database being too memory hungry to allow its execution. Each try resulted in a JVM OutOfMemory exception, something, which did not happen with any of the other solutions. This leads us to believe that Neo4j has higher memory requirements, needs more performance tweaking in regards to server configuration than the rest of the studied DBMSs.

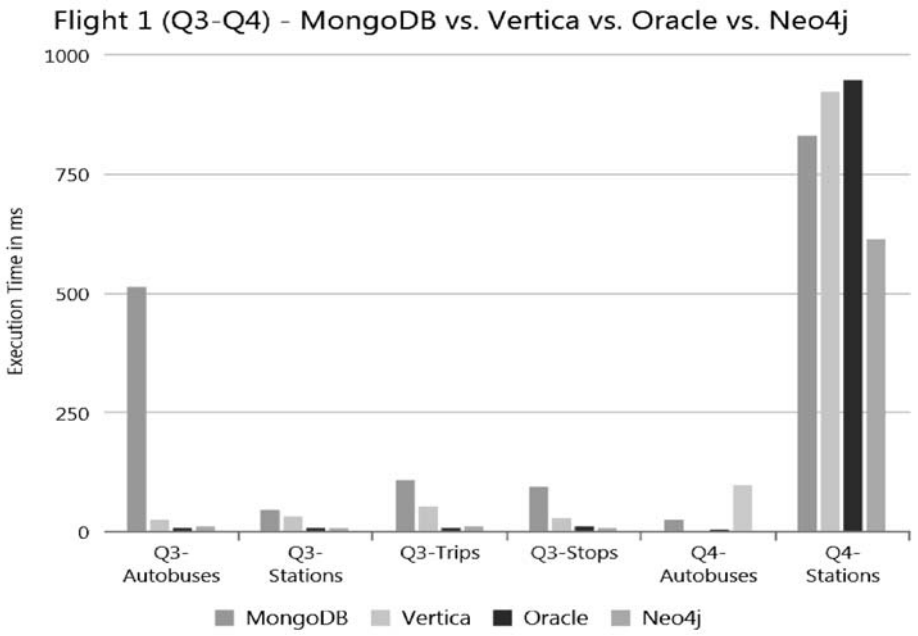


Fig. 4. Simple deletes and selects

4.2 Analytical queries performance analysis

We ran the analytical queries (flight two and three – Figure 5) several times and averaged the results. Unlike MongoDB, Neo4j performed on par with Vertica and was significantly faster than Oracle. On some queries it was even considerably faster than Vertica, which shows the best performance so far. This is no doubt connected with the higher expressiveness of the Cypher query language, which supports native joining operations unlike MongoDB's JavaScript dialect. In the second flight Neo4j matched the performance of Vertica query for query with even some slightly better results.

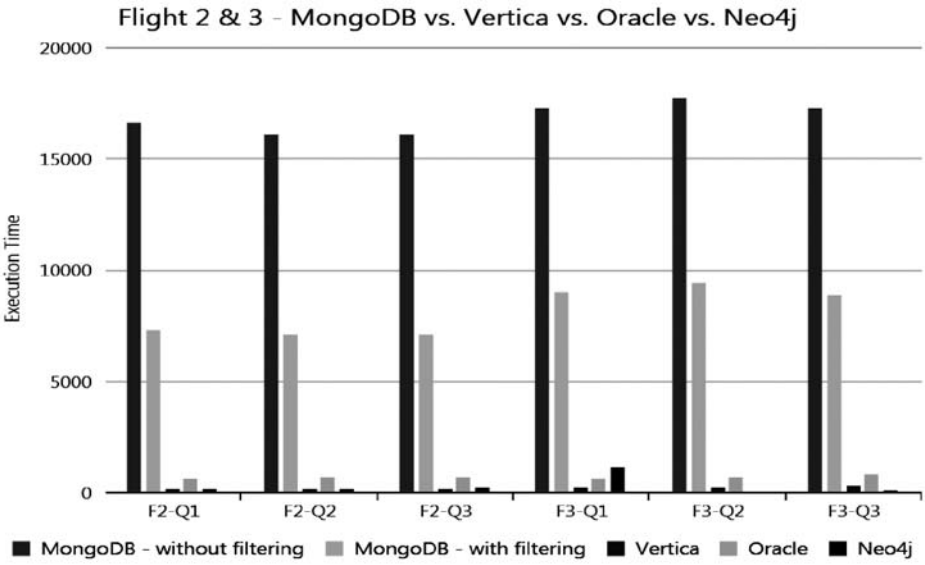


Fig. 5. Flight 2 and 3 results

What really are interesting are the results of the third flight. These are in the last two cases significantly better than both Oracle and Vertica and in the first one, much slower than both. The rationale behind these observations most likely lies in the selectivity, which we already saw presents an issue for Neo4j. Each of the three queries returns three groups with varying number of records. This combined with some clever caching can explain the differences in the execution times between the queries of the flight. What can be worrying is that in contrast to all other three databases – Oracle, Vertica, and MongoDB there is no consistency in the results of this test set.

The forth flight of queries was not performed as due to high consumption of memory the relationship between the Trips and the Stops could not be created. Each try as with the simple select failed with JVM OutOfMemory exception. Without the relationship the query leads to the same exception and we have reasons to believe that it would be unreasonably high, as we have run some tests without relationships for the other smaller tables, which were taking considerably more time than the final results shown above in the charts.

5 Conclusion

The overall impression of the graph store Neo4j is good – it is very easy to setup and load data; however there were some configuration issues with poor memory management during the data loading and querying tests. One of its

native languages – Cypher - is easy to learn, straightforward and is expressive enough to substitute the well-known and widely used SQL. When it comes to the performance results, as expected, it is pretty much on par with Oracle and Vertica, it is even faster than Vertica on some analytical queries, which is the best DBMS we have tested so far. Neo4j provides a great balance with fast simple inserts, updates and deletes and great speeds for analytical loads. What seems to be one of the few problems with Neo4j was the high selectivity, thus if it were not for the aforementioned issues with stability and memory consumption, it would have been the best store in our performance study. In conclusion, although it might be a very potent option for resource rich environment, we would still rather recommend sticking to Vertica as it is the more accomplished product. However, with the maturing of the technology, Neo4j deserves to be followed closely and considered when choosing an analytical database. As a natural continuation of our performance study of the different types of stores is the recently developed array databases, which we intend to study in future research.

Acknowledgment. This paper is supported by Sofia University "St. Kliment Ohriski" SRF under Contract 36/2015.

References

1. Abadi, D.J. et al.: Column-oriented database systems. Proc. VLDB. 1664–1665 (2009).
2. Abadi, D.J., Madden, S.R.: Column-Stores vs . Row-Stores : How Different Are They Really? SIGMOD. pp. 967–980 (2008).
3. Batra, S., Tyagi, C.: Comparative analysis of Relational and graph databases. IJSCE Int. J. Soft Comput. Eng. 2, 2, 509–512 (2012).
4. Hristov Hr., K. Kaloyanova A graph representation of query cache in OLAP environment, Proceedings of the 7-th ISGT International Conference, pp. 108-119, Sofia, (2013).
5. Kaloyanova K., Improving Data Integration for Dara Warehouse: A Data Mining Approach, Proceedings of the International Workshop "Computer Science And Edication", pp. 39-44, Borovetz-Sofia, (2005).
6. Kyurkchiev, H., Kaloyanova, K.: Performance Study of Analytical Queries of Oracle and Vertica. Proceedings of the 7th ISGT International Conference. pp. 127 – 139 , Sofia (2013).
7. Kyurkchiev, H., Mitreva, E.: Performance Study of SQL and NoSQL Solutions for Analytical Loads. 1–9.
8. Lamb, A. et al.: The Vertica Analytic Database : C-Store 7 Years Later. Proceedings of the VLDB Endowment. pp. 1790–1801 (2012).
9. Mitreva, E., Georgiev, V.: Using Document Store for 3D Virtual Collections. Proc. DiPP. (2015).
10. Mitreva, E., Kaloyanova, K.: NoSQL Solutions to Handle Big Data. Proc. Dr. Conf. MIE. 77–86 (2013).
11. Scholz, J.: Coping with Dynamic, Unstructured Data Sets – NoSQL: a buzzword or a savior? Johannes Scholz. May, 1–9 (2011).

12. Stonebraker, M. et al.: C-store: a column-oriented DBMS. Proceedings of the 31st VLDB Conference. pp. 553–564 (2005).
13. TreeHouse: Should you go Beyond Relational Databases?, http://blog.teamtreehouse.com/should-you-go-beyond-relational-databases.
14. Neo4j documenation, http://neo4j.com/.
15. The Vertica ® Analytic Database Technical Overview White Paper. (2010).

A Framework for RUP and PMI Artifacts

Kalinka Kaloyanova[1], Elitza Koleva[1]

[1]Faculty of Mathematics and Informatics, Sofia University, 5 James Bourchier blvd., 1164, Sofia, Bulgaria

kkaloyanova@fmi.uni-sofia.bg, elitza.koleva@gmail.com

Abstract. The paper explores the possibilities of an integration of best practices, guidelines and document templates of Rational Unified Process and PMI methodology and provides a framework for software project management using an optimal set of documents, based on both methodologies.

Keywords: Project Management, Rational Unified Process (RUP), Software Development

1. Introduction

Software development is a process that involves management of a set of activities and required resources to achieve a desired result. To reach that goal, the development team can choose from a variety of software development processes that are usually based on several basic models for software development - sequential, iterative and agile. The software development process is also a subject of many kinds of uncertainty that are more related to project management rather than the development process itself. Therefore the elements involved in the process of the development of software products can be seen in two dimensions - the software development process and the management process. To produce a quality product in time and budget there should be an appropriate combination of these two dimensions.

Project Management Institute (PMI) provides a methodology that supports a wide range of projects in different areas. The Project Management Body of Knowledge (PMBOK) Guide [10], developed by Project Management Institute ensures a set of best practices, tools and techniques that are successfully used for project management in a variety of industries. When project management concerns software area, the software process is important, as well.

In this paper we explore the possibilities of integrating two widely used methodologies – Rational Unified Process (RUP) for software development and the PMI methodology for project management. Although several comparisons of RUP, PMI and other methodologies have been done [2],[3],[7], as the methodologies have changed during the years, a new parallel is needed.

The purpose of the article is to reconcile the best practices, guidelines and document templates of RUP and latest PMBOK version and to focus on creating a framework of RUP artifacts and documents from PMI methodology that pretends to provide an optimal predefined set of documents for software

project management. The framework has a practical value to both – software development teams and students, who explore these methodologies, mainly the students in Information Systems and Project Management areas [5], [6].

2. PMI Framework

The Project Management Body of Knowledge is organized as a Guide that describes the different Project Management (PM) processes and their interactions. The processes are grouped in two dimensions: into five categories known as Process Groups and several categories known as Knowledge Areas. In the last 5th edition of PMBOK the number of Knowledge Areas is extended up to ten and the number of the processes is forty seven [9].

2.1 Process groups

Every project starts with processes from the Initiating Process Group that define a new project or a new phase of a project. The next Planning Process Group supports processes that aim to establish the scope of the project, refine the objectives, and define the course of action required to attain the objectives. The main goal of the processes from Executing Process Group is to complete the work defined in the project management plan and to ensure that the product satisfies the project specifications. The processes of the Monitoring and Controlling Process Group ensure that the project managers track, review and regulate the progress and performance, identify areas in which changes to the plan are required and initiate the corresponding changes. And finally, the Closing Process Group finalizes all activities across all Process Groups to formally close the project or a phase of the project [9].

PMBOK does not prescribe a precisely defined lifecycle for projects. Instead it specifies that the project lifecycle should be divided into phases based on the nature of the project and its area of application. During each of these phases the Process Groups are used, as appropriate, to ensure the successful completion of the project in a controlled manner. If it is necessary processes are repeated within each phase until phase completion criteria is satisfied [9].

It is possible a Process Group to be a one-time event, but most often they are overlapping activities that occur in different phases of the project and basically the output of one process becomes either a project deliverable or an input to another process.

2.2 Knowledge Areas

The processes are also grouped in ten Knowledge Areas by their nature. Each Knowledge Area represents a set of similar concepts and activities and forms a specific project management field [9]:

- Project Integration Management
- Project Scope Management
- Project Time Management
- Project Cost Management
- Project Quality Management
- Project Human Resources Management
- Project Communication Management
- Project Risk Management
- Project Procurement Management
- Project Stakeholder Management

The PMBOK also describes a set of documents that should be maintained with the project management processes to support the development of the project. Every project starts with signing a Project Charter that formally authorizes the existence of the project and also authorizes the project manager to apply organizational resources in the project development process.

The Project Plan presents more details about the project with Scope Statement, Work Breakdown Structure, Schedule, Cost Estimates, etc. Also Work Results, Change Requests, Lessons Learned and other documents are created and updated periodically to ensure that the project is developed in a controlled and predictable manner as well as that the product created corresponds to the stakeholders' requirements.

As the framework is wide and flexible it does not provide strict document templates. Although some sources [4] provide samples of templates, principally the form and the content of the documents are specific and depend on the application domain of the project.

Therefore the incorporation of an organizational project management with a specific software development process requires certain efforts for integration of artifacts, work products and documents from these two sources.

The next section will present an overview of a specific software development process and its corresponding artifacts.

3. Rational Unified Process

The Rational Unified Process is a software engineering process that provides a disciplined approach for software development [9].

RUP is an iterative, use-case driven and risk-driven process [1]. Each of these characteristics has a specific implementation here. In RUP the software development lifecycle is divided in phases and iterations that gradually produce executable versions of the application until the acceptance criteria is satisfied and the product is fully developed [11], [12].

Here functional requirements are expressed in the form of use cases that describe scenarios of using the designed system by users. RUP also puts a focus on risk reduction by tackling risk as soon as possible in the software development process, which includes prioritizing and building the important functionality before building the less important ones.

Generally, RUP implements software development in two main dimensions as it is shown on Fig.1:

- The horizontal dimension represents time and shows the lifecycle of the process as it unfolds.
- The vertical dimension represents core process disciplines, which logically group software engineering activities by their nature [3].

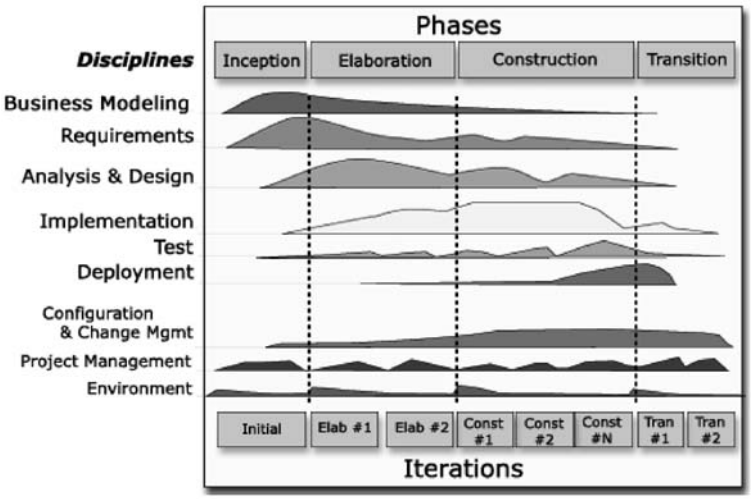


Fig. 1. Rational Unified Process [10]

3.1 RUP Phases

The lifecycle of any RUP project is divided into four phases – Inception, Elaboration, Construction and Transition.

The focus of Inception phase is to define the project scope. During the next Elaboration phase the project scope is refined, an executable architecture for the application is built, and risk assessment is performed. Based upon the baselined software architecture in the Construction phase most of the application's capabilities are developed. And finally, the goal of the Transition phase is to ensure that the application is available to the end-user community, which includes testing and preparation for release. Each phase has precisely defined criteria for completion and ends up with a major milestone. RUP phases can be further divided into iterations [8].

3.2 RUP Disciplines

RUP captures several software development best practices in a form suitable for a wide range of projects and organizations: adaption of the process, iteratively value demonstration, continuously focus on quality; balance between competing stakeholder priorities, collaboration across teams, etc. [8].

Based on the six best practices RUP organizes the activities and the supporting documentation involved in the development process in nine disciplines that basically describe who is doing what, how and when within the full development team. There are six main disciplines and three supporting disciplines that maintain the development process. The main disciplines are Business modeling, Requirements, Analysis and design, Implementation, Test and Deployment. The supporting disciplines are Project management, Environment and Configuration and Change management. Each discipline is expressed in terms of roles, activities, artifacts and workflows [10].

Within RUP, each discipline is expressed in terms of roles, activities, and artifacts. Every role defines who performs the task - the behavior and responsibilities of an individual, or a group of individuals, responsible for activities and artifacts. On the other hand, an activity is a task for which a role is responsible and may be asked to perform. It describes the steps required to create or update one or many artifacts or what the activity achieves [3].

3.3 RUP Artifacts

RUP specifies a set of artifacts that support the development process. Artifacts are either final or intermediate work products that are produced and used during a project to capture and convey project information. The artifacts serve as inputs/ outputs of the activities. An artifact can be a document, a model or a model element. RUP provides a wide range of document templates that can either be used directly by the organizations or can be adapted to the specific needs of the project [11], [12].

4. Rational Unified Process & PMI Compatibility

4.1 Concepts Compatibility

As a framework RUP is extremely effective for guiding the software development process, however it is a heavy and complicated process that requires the maintenance of a wide range of documents. Although it provides some management practices it is far from covering all aspects of project management. On the other hand the PMBOK Guide provides a set of best practices covering all aspects of project management, but the methodology of PMI is not concrete

and therefore it cannot be directly implemented. It rather provides a basic framework of management practices that can be successfully upgraded with a specific software development methodology. In this context, the PMBOK can provide the perspective of managing the scope and project life cycle, while RUP could provide a more technology-oriented perspective for managing the scope and product life cycle.

To create a framework of RUP artifact and documents from PMI methodology that provides an optimal set of documents for software project management we first have to:

- Explore the PMBOK practices that are covered by RUP
- Point out the PMBOK practices that can enrich the RUP
- Find out the RUP artifacts that are essential for the success of the project

The project management practices in RUP are mainly covered by the workflows in the PM discipline. The PM discipline provides a framework for managing software-intensive projects, in terms of practical guidelines for planning, staffing, executing, assessing risk and monitoring projects.

The PM discipline focuses mainly on the important aspects of an iterative development process such as risk management, planning an iterative project, through the whole lifecycle and for a particular iteration and monitoring progress of an iterative project.

The workflows, roles, activities, and artifacts in RUP that parallel the knowledge areas of the PMBOK reside in more than just the project management discipline.

4.2 PM practices from PMBOK that are not covered by RUP

Although RUP contains some concepts related to people management it does not cover all aspects of managing people such as hiring, training, coaching staff members, etc. Also the process does not provide a precise control on budget management such as planning, estimating, monitoring and controlling the cost of the product. RUP does not cover any of the issues regarding managing contacts with suppliers and customers and does not provide any techniques for managing communications in the project in general. Accordingly RUP does fully cover various fields of project management that could be critical for the success of the project such as Project Cost Management, Project Human Resource Management, Project Communications Management, Project Procurement Management and Project Stakeholder Management.

4.3 Integration of RUP artifacts and PMBOK documents

Based on above parallel of the two frameworks we suggest a possible

reconciliation of essential RUP artifacts and documents of the PMI methodology and present an optimal set of documents for software project management.

The documents are grouped in two categories – a list of documents required for the development of any software project (Table 1) and a supplementary list of documents that if applicable could be extremely useful and bring value to the software development lifecycle (Table 2).

Table 1. List of documents required for the development of any software project formed as integration of RUP artifacts and PMBOK documents at a project level.

PMI RUP		Initiating	Planning	Executing	Closing
	Monitoring and Control				
Inception	Iteration	1. **Business Case** 2. *Project Charter* 3. *Project Statement of Work* 4. Vison 5. Glossary 6. Use Case Specification 7. Supplementary Specification 8. Software Development Plan 9. Test Plan 10. Analysis Model 11. Deployment Model 12. Use Case Model 13. Development Case	1. *Project Management Plan* 2. **Quality Management Plan** 3. **Requirements Management Plan** 4. **Risk Management Plan** 5. *Communication Management Plan* 6. *Cost Management Plan* 7. *Procurement Management Plan* 8. *Schedule Management Plan* 9. *Scope Management Plan* 10. *Work Breakdown Structure* 11. *Stakeholder Management Plan* 12. *Milestone List* 13. *Resource Requirements* 14. *Change Management Plan* 15. *Master schedule* 16. **Risk Register**		
Elaboration			17. **Configuration Management Plan** 18. Data Model 19. Design Model 20. Domain Model 21. Test Model 22. Software Architecture 23. Software Requirements Specification 24. Integration Built Plan 25. Test Suite 26. Implementation Model 27. Deployment Plan	1. *Resource Calendar* 2. *Duration Estimate* 3. **Data migration specification/ Implementation and Migration Plan**	
Construction				4. Test Automation Architecture 5. Test Evaluation Summary 6. Test Case 7. User Support Material 8. *Employee Annual Review* 9. *Activity List*	
Transition				10. Status Assessment	1. *Lessons Learned* 2. *Post Project Review* 3. *Transition out plan*

Table 2. List of supplementary documents formed as integration of RUP artifacts and PMBOK documents at a project level.

PMI RUP		Initiating	Planning	Executing	Closing
	Monitoring and Control				
Inception	Iteration	1. *Stakeholder Register* 2. *Feasibility Study* 3. *Statement of work* 4. **Stakeholder Request** 5. Project specific Guidelines 6. Project Specific Templates 7. Problem Resolution Plan 8. Measurement Plan	1. **Project Acceptance / Acceptance Criteria** 2. *Human Resource Plan* 3. *Process Improvement Plan* 4. *Cost Baseline* 5. *Funding Requirements* 6. *Project Calendar* 7. *Meeting Agenda* 8. *Meeting Minutes*		
Elaboration			9. Bill of Materials 10. Navigation Map 11. Manual Styleguide 12. Configuration Audit 13. Storyboard		
Construction				1. Test Interface Specification	
Transition					1. *Formal Acceptance and closure*

Each document of the PMI methodology is in one of the following relations with one (or more) of the RUP artifacts:

- Fully overlap
- Partially overlap
- There is no analogue in the corresponding methodology

In bold are marked the documents that fully overlap, in italic are marked

the documents of PMI methodology that have no analogue in RUP, and the RUP artifacts that have no analogue in PMI methodology are underlined.

The decision what subset of documents from the supplementary list should be maintained for a specific project should be made by the development team based on the characteristics of the project such as size, type, and complexity.

As shown in Table 1 and Table 2, most documents are maintained in the Initiating and Planning Process Groups. This is understandable as they largely correspond to RUP phases Inception and Elaboration. However, in the other process groups there are important elements such as Test Evaluation Summary, Status Assessment, Lessons Learned, etc.

Due to the iterative nature of the development process documents should be updated periodically. The tables present the documents grouped by phase (in terms of RUP) and process group in which they are firstly created. The content of the tables does not pretend to be all-embracing.

Table 3 presents another possible reconciliation of essential documents used in RUP and PMI methodology - at an iteration level. The table also includes documents for monitoring and control since these documents are basically used in all phases/iterations during the project life cycle.

Table 3. Integration of essential RUP artifacts and PMBOK documents at an iteration level.

Initiation	Planning	Executing	Closing	Monitoring and Control
	1. Next Iteration Plan 2. Work Order			1. Iteration Assessment 2. Status Assessment 3. **Change Request** 4. **Change Log** 5. *Corrective Action* 6. *Performance Report* 7. *Root Cause Analysis* 8. **Issue Log**

Conclusion

Based on the overview of the main concepts of RUP and PMI framework, it can be concluded that both standards share similar practices and concepts concerning project management that are simply described by different terms. They do not contradict to each other. On the contrary, as highlighted in this paper RUP and PMI methodologies rather complement each other since PMBOK provides a basic framework of project management practices that can be successfully coordinated with a specific software development methodology as RUP.

The article proposes a framework containing essential RUP artifacts and

documents from the PMI methodology that provides an optimal set of documents to guide and support the software project management. In the proposed reconciliation PMBOK documents provide instruments for management of key project assets, including human resources, stakeholders' engagement, communications and suppliers in the project. The PMBOK practices also allow more precise control of the budget, schedule, scope and project planning in general. In addition, the PMI methodology provides an in-depth risk analysis, as well as some really good practices for retrospection, extraction of lessons learned and constant reassessment of project activities. These practices can significantly enrich RUP as a process that provides detailed, software-specific guidelines for all software development processes. Based on the proposed framework a small software product has been developed to support with proper templates the courses "Information Systems Analysis and Design" and "Project Management" at Sofia University undergraduate program "Information Systems".

Even though the proposed frame integrates two of the most commonly used methodologies of RUP and PMI it could also be adapted to other software development processes or separate stages of these processes.

Acknowledgment. This paper is supported by Sofia University "St. Kliment Ohridski" SRF under Contract 36/2015.

References

1. Bruegge B., Allen H. Dutoit: Object-Oriented Software Engineering Using UML, Patterns, and Java, Pearson Education, (2008)
2. Callegari, D.A, & Bastos, R. M, Project Management and Software Development Processes: Integrating RUP and PMBOK. White paper, (2007)
3. Charbonneau S., Software Project Management - A Mapping between RUP and the PMBOK (http://www.xelaration.com/documents/software_project_management_a_mapping_between_rup_and_the_pmbok.pdf), (2004)
4. Free Project Management Templates (http://www.projectmanagementdocs.com)
5. Kaloyanova K., Information Systems Analysis and Design Course with Projects Based on Real Customers Requirements, Proceedings of the 8th Mediterranean Conference on Information Systems, Verona, Italy;. Paper 31; http://aisel.aisnet.org/mcis2014/31, (2014)
6. Leybourne, S.A., Warburton, R., Kanabar, V., "Is Project Management the New Management 2.0?", Organisational Project Management, Vol. 1,No. 1, pp.16-36, (2014)
7. Napoli, J. P.; Kaloyanova, K. (2011) "An Integrated Approach for RUP, EA, SOA and BPM Implementation" In: CompSysTech '11-12th International Conference on Computer Systems and Technologies,Vienna, Austria, (2011)
8. Philippe Kruchten, The Rational Unified Process - An Introduction, Second Edition, Addison-Wesley, (2000)
9. Project Management Institute (PMI), A Guide to the Project Management Body of Knowledge (PMBOK), PMBOK Guide 5th edition, (2013)
10. RUP Fundamentals Presentation, https://era.nih.gov/docs/rup_fundamentals.htm
11. RUP for Large Projects - Rational Method Composer Version 7.5.1, IBM (2010)
12. RUP for Small Projects - Rational Method Composer Version 7.5.1, IBM (2010)

Possible improvements in the Big Data management with InnoDB Memcached integration

Svetoslav Savov*[1], Dimitar Vassilev[2,3]

[1] – Technical University, 8 St.Kliment Ohridski Blvd, Sofia 1756, Bulgaria
[2] – AgroBioInstitute, 8 Dragan Tsankov Blvd, Sofia 1164, Bulgaria
[3] – Faculty of Mathematics and Informatics, Sofia University "St. Kliment Ohridski", 5 James Bourchier Str, Sofia 1164, Bulgaria

* – corresponding author: svetlio_81@yahoo.com

Abstract: Big Data is a compound term that describes different sources of voluminous structured and unstructured data. The corresponding data sets are usually so enormous that it is a great challenge to manage (to gather, store, analyse, share, query, search and visualize) them in an acceptable period of time. On the other hand the users nowadays expect to get the information they are looking for in real time, without considerable delays. Sometimes the available hardware power is not enough, it is difficult to obtain the results in a proper time frame and innovative software solutions must be applied in order to meet the users' expectations.

One of the possible approaches is to store the data in MySQL databases and manage it through the InnoDB Memcached integration. The improvement is based on the direct access of the data through the fast Memcached protocol skipping the SQL query parser, the query optimizer and the overhead caused by the query execution. Combing it with the InnoDB stable and reliable functionality and the option to still run complex queries through SQL significantly benefits the database administrators and the applications' developers.

This study is focused on the appraoch how to setup the InnoDB integration with the Memcached daemon plugin. The solution is tested with sample data and compared with the standard query work flow through MySQL. The pros and cons are highlighted.

Keywords: Big Data, MySQL, InnoDB Memcached integration

1. Introduction

What is considered as Big Data? There are different definitions of the Big Data term. A popular definition says that it is extremely large data sets that may be analysed to reveal patterns, trends, and associations related to human behaviour and communication.

Big Data is a compound term that describes different sources of voluminous structured and unstructured data. The corresponding data sets are usually so enormous that it is a great challenge to manage (to gather, store, analyse, share,

query, search and visualize) them in an acceptable period of time. On the other hand the users nowadays expect to get the information they are looking for in real time, without considerable delays.[1]

One of the possible approaches is to store the data in MySQL databases [2] and manage it through the InnoDB integration with Memcached [3],[4]. The improvement is based on the direct access of the data through the fast memcached protocol skipping the SQL query parser, the query optimized and the overhead caused by the query execution. Combing it with the InnoDB stable and reliable functionality and the option to still run complex queries through SQL significantly benefits the database administrators and the applications developers.

This article will be focused on the way to setup the InnoDB integration with the Memcached daemon plugin. The solution will be tested with sample data and compared with the standard query work flow through MySQL. At the end pros and cons will be highlighted.

2. 3Vs model

Big Data can be represented with the 3Vs model [5].

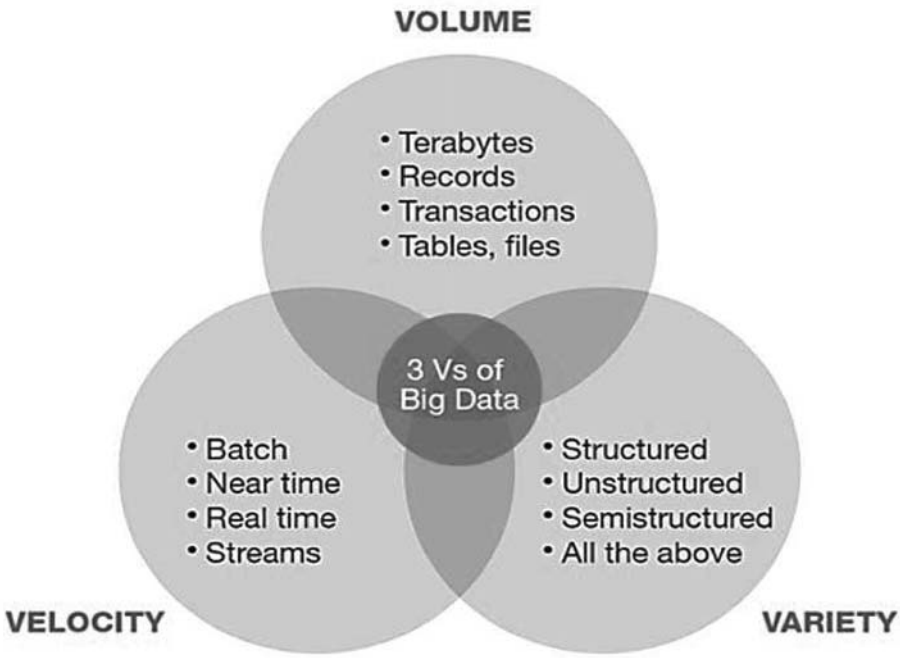


Fig.1. 3 Vs of Big Data

The characteristics are as follows:

Volume - Many factors should be taken into consideration when we analyse the recent increase in the volume of data transmitted all over the world. For example the data recorded during the years from different types of informational transactions, the unstructured data from social media platforms, the enormous amounts of records from smart sensors and machine-to-machine communication. Several years ago saving such volume of data has been a storage challenge. Nowadays, after the increase of the storage capacity and decrease of the related costs the focus is shifted towards the way to properly determine relevance within large data volumes and gain a certain value of it.

Velocity - During the last years the speed of the data streaming has been significantly increased. The huge volume of data should be handled within a timely manner. Some of the data streams should be managed in nearly real time. The main challenge that most applications‘ developers face is to deal quickly enough with the data velocity.

Variety - Today there are many different types of data formats: Structured data stored in traditional databases, logs from software activity (web logs, transactions logs, activity logs) and smart sensors, Internet of Things (IoT) appliances activity log, social networks data (media - videos, photos, online messaging services), Mobile Applications data, customers‘ private and billing data, content created from business applications, data from scientific researches, news feeds, search engines results‘ analytics, unstructured text documents, email messages, video, audio and financial transactions. It is still a struggle for most organizations to manage and merge all these different varieties of data.

3. MySQL InnoDB integration

There are different solutions for the storage and the management of the Big Data. A popular one is MySQL. You may wonder why?

MySQL is a leading database solution for web based applications. It is the most popular and widely used open-source RDBMS (relational database management system). It is already integrated in many solutions that manage and analyse Big Data. Its syntax is well known by most DBA, programmers and system administrators. The project is supported by a large community and there are many drivers for communication with other database systems. It comes with native support for most programming languages and is compatible with different operating systems. MySQL provides powerful replication mechanisms which can be of a great value for the Big Data management. Although there are questions about the future open-source nature of MySQL there is a very influential organization behind the project. Last but not least, MySQL has advanced security mechanisms and access privileges. It is considerably fast, stable, secure and reliable RDBMS.

There are several reasons to use the MySQL InnoDB Memcached integration.

Through it you can achieve a better performance – memcached works in the same process space as the MySQL server allowing direct access (read/write) to the data without going through the SQL layer and avoiding the overhead of standard MySQL queries. Structured as well as unstructured data can be handled. Multiple column values can be set as a single memcached item value. Automatic transfer between memory and disk is supported. The developers can still use regular MySQL queries for more advanced requests. The data is protected against memory crashes by regular commits and the storage on the disk in a MySQL database.

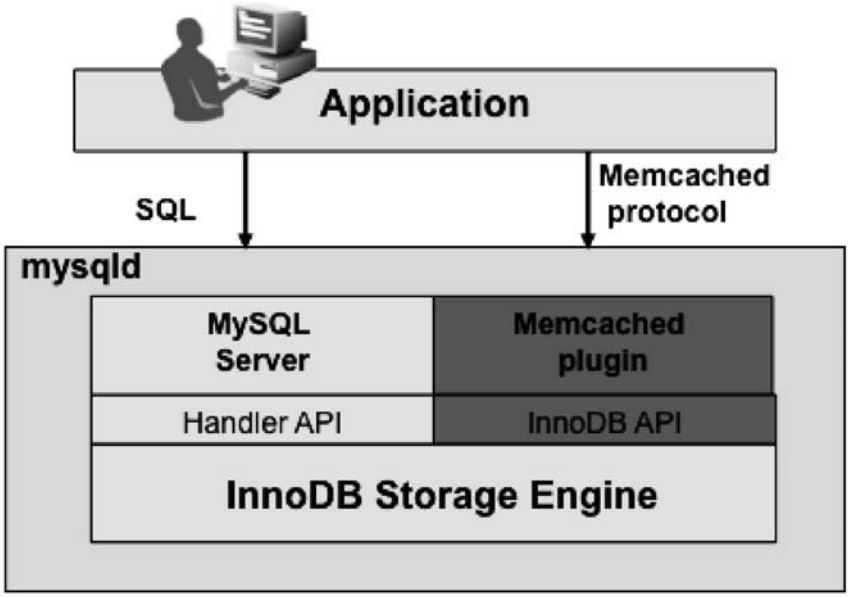


Fig.2. MySQL InnoDB integration

The integration of the Memcached InnoDB plugin is relatively simple. You should have MySQL 5.6 or later installed on your server. The Memcached daemon should be integrated in the MySQL server [6].

The commands used in this article are native for Centos. While the syntax and the files' locations slightly differ for other Linux distributions the approach is pretty much the same.

First, you should install the **libevent** library, which is required by Memcached:

-bash-4.1# yum install libevent

Then import the necessarily sql data and install the Memcached plugin so it can directly interact with the MySQL server:

-bash-4.1# mysql -uroot -p

Enter password:
mysql> SOURCE /usr/share/mysql/innodb_memcached_config.sql;
mysql> INSTALL PLUGIN daemon_memcached SONAME „libmemcached.so“;
The plugin activation can be verified through the following MySQL statement:
*mysql> select * from information_schema.plugins where plugin_name like ‘%memcached%’\G*
*************************** 1. row ***************************
PLUGIN_NAME: daemon_memcached
PLUGIN_VERSION: 1.0
PLUGIN_STATUS: ACTIVE
PLUGIN_TYPE: DAEMON
PLUGIN_TYPE_VERSION: 50624.0
PLUGIN_LIBRARY: libmemcached.so
PLUGIN_LIBRARY_VERSION: 1.4
PLUGIN_AUTHOR: Oracle Corporation
PLUGIN_DESCRIPTION: Memcached Daemon
PLUGIN_LICENSE: GPL
LOAD_OPTION: ON
1 row in set (0.00 sec)

Tables are set: *cache_policies, config_options and containers*. The last is used for the InnoDB Memcached mapping :

mysql> use innodb_memcache;
mysql> describe containers;

Field	***Type***	***Null***	***Key***	***Default***	***Extra***
name	*varchar(50)*	*NO*	*PRI*	*NULL*	
db_schema	*varchar(250)*	*NO*		*NULL*	
db_table	*varchar(250)*	*NO*		*NULL*	
key_columns	*varchar(250)*	*NO*		*NULL*	
value_columns	*varchar(250)*	*NO*		*NULL*	
flags	*varchar(250)*	*NO*		*NULL*	
cas_column	*varchar(250)*	*YES*		*NULL*	
expire_time_column	*varchar(250)*	*YES*		*NULL*	
unique_idx_name_on_key	*varchar(250)*	*NO*		*NULL*	

Table 1. Cache_policies and config_optioins
9 rows in set (0.00 sec)

mysql> select * from containers;

name	*db_ schema*	*db_table*	*value_ columns*	*key_ columns*	*flags*	*cas_ column*	*expire_time_ column*	*unique_idx_ name_on_key*
aaa	*test*	*demo_test*	*c1*	*c2*	*c3*	*c4*	*c5*	*PRIMARY*

Table 2. Containers

The description of the table's columns is as follows:

- **name**: This is the name that is like the primary key for the memcache data collection. If you have a value of default for name, then this will be the default entry that is used. Otherwise it uses the first entry in the container table. You can also specify this name value in the NoSQL statement.
- **db_schema**: The InnoDB database name that you will use to store the data.
- **db_table**: The InnoDB database table name that you will use to store the data.
- **key_columns**: The column that you will use for the key value lookup. Represents the alphanumerical value which will be the key for accessing the value of the item in the Memcached protocol.
- **value_columns**: Data will be pulled from and/or stored to these column/ columns of data. You use a separator value (such as a pipe „|" symbol) to separate the columns. The actual payload is kept in this column.
- **flags**: This column stores memcache flags, which is an integer that is used to mark rows for memcache operations. For example, it could be a flag whether or not to use compression.
- **cas_column**: Unique identifier of each item. Used for storing memcache compare-and-swap values.
- **expire_time_column**: Expiration time in seconds.
- **unique_idx_name_on_key**: This is the name of the unique index that you will use for the key column, and you should use the primary key for the table. You should not use an auto-incrementing index, as you can't insert into an auto-incrementing key.

The default port for the memcached communication is 11211. It is set to work on localhost. The telnet command can be used to establish a connection and verify the expected behavior. Use the standard **set** and **get** requests. The syntax is as follows:

set [key] [flag] [expiration] [length in bytes]
[value]

get [value]
-bash-4.1# telnet localhost 11211
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
set testkey 0 100 10
mytest1234
STORED
get testkey
VALUE testkey 0 10
mytest1234
END

To verify that the new key value pair is stored in the database as well run the following query:

*-bash-4.1# mysql -uroot -p -e "select * from test.demo_test where c1 like 'testkey';"*

Enter password:

```
+---------+-------------+------+------+---------------+
| c1      | c2          | c3   | c4   | c5            |
+---------+-------------+------+------+---------------+
|testkey|mytest1234|    0 |    2 |1442153272 |
+---------+-------------+------+------+---------------+
```

4. Benchmark the InnoDB Memcached integration

In order to test the performance improvement you can run scripts like the following ones:

Memcached

-bash-4.1# time php test_memcached.php
real   0m1.717s

-bash-4.1# cat test_memcached.php

```
<?php
$memcache_obj = new Memcache;
$memcache_obj->connect(‚localhost‘, 11211) or die („Could not connect“);
for ($x = 0; $x <= 10000; $x++) {
    $key=“mykey“ . $x;
    $random = substr( md5(rand()), 0, 7);
    $memcache_obj->set($key, $random) or die („Failed to save data at the
```

```
server“);
    }
    ?>
```

MySQL

```
-bash-4.1# time php test_mysql.php

real    0m2.109s

-bash-4.1# cat test_mysql.php
<?php
$servername = „localhost“;
$username = „user“;
$password = „pass“;
$dbname = „test“;
// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
   die(„Connection failed: „ . mysqli_connect_error());
}
for ($x = 0; $x <= 10000; $x++) {
    $key=“mykey“ . $x;
    $random = substr( md5(rand()), 0, 7);
    $sql = „UPDATE demo_test SET c2=‘$random‘ WHERE c1=‘$key‘“;
    mysqli_query($conn, $sql);
}
mysqli_close($conn);
?>
```

Both scripts insert 10000 key value pairs in a MySQL database. The first script completes the task through Memcache while the second uses the standard approach with MySQL.

5. Conclusion

The Memcached InnoDB integration has certain benefits. Some of the Pros and Cons are:

- ***Latency*** – the direct access to the InnoDB engine with simple memcached commands significantly speeds up the execution of the corresponding

query. The query through the SQL engine takes more time since it should be parsed, the execution should be first optimized and then completed.

- ***Throughput*** - the faster execution of the query through the memcached plugin (no parsing, no optimization - accessing a single row by an index) allows to complete more queries for a defined period of time.
- ***Scalability*** – with the standard MySQL approach the scalability can be achieved either by switching to MySQL Cluster that has a built-in auto-sharding functionality or by implementing the sharding on the application level. Memcached is designed to work as distributed cache and offers a sharding feature through the libraries (libmemcached for example) that support the connection to the database.
- ***MemCached*** does not use an authentication mechanism by default so it is not recommended to be used for sensitive data. Alternatively, additional Simple Authentication and Security Layer authentication should be implemented.

Acknowledgments. This work has been supported by the National Science Fund of Bulgaria within the "Methods for Data Analysis and Knowledge Discovery in Big Sequencing Datasets" Project, Contract DFNI-I02/7 of 12 December 2014.

References

[1] McAfee, A., Brynjolfsson, E., Davenport, T.H., Patil, D.J. and Barton, D., 2012. Big data. The management revolution. Harvard Bus Rev, 90(10), pp.61-67.

[2] MySQL AB, 1995. MySQL: the world's most popular open source database.

[3] Frühwirt, P., Kieseberg, P., Schrittwieser, S., Huber, M. and Weippl, E., 2012. InnoDB database forensics: reconstructing data manipulation queries from redo logs. In Availability, Reliability and Security (ARES), 2012 Seventh International Conference on (pp. 625-633). IEEE.

[4] Oracle, 1997-2015. MySQL 5.7 Reference Manual [Online]
(https://dev.mysql.com/doc/refman/5.7/en/innodb-memcached.html)

[5] Zikopoulos, P. and Eaton, C., 2011. Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media.

[6] Sun, C., 2012. New Enhancements for InnoDB Memcached. Transactions on InnoDB. Oracle blog [Online]
(https://blogs.oracle.com/mysqlinnodb/entry/new_enhancements_for_innodb_memcached)

PhyloEdit: a web-based GUI for visualization and analysis of evolutionary trees

Zhenya Duylgerova[1], Irena Avdjieva[2], Deyan Peychev[2], Dimitar Vassilev *[1,2]

[1] – Faculty of Mathematics and Informatics, Sofia University "St. Kliment Ohridski", 5 James Bourchier Str, Sofia 1164, Bulgaria
[2] – Bioinformatics group, AgroBioInstitute, 8 Dragan Tsankov Blvd, Sofia 1164, Bulgaria
* – Corresponding author: jim6329@gmail.com

Abstract
Evolution is the cornerstone of contemporary biology and unites all its fields under one theoretical concept. Evolutionary biology studies the evolutionary history and relationships among living organisms. These relationships are based on similarities and differences between biological entities and are graphically represented as phylogenetic trees. Today, reconstruction, refining, and analysis of phylogenetic trees relies extensively on sequence alignments and benefits greatly from computational methods and algorithms. Graphical representation of phylogenetic trees is an inseparable part of evolutionary analysis and there are many software applications designed to visualize trees. The majority of them, however, are limited in either functionality or accessibility. This is why the current study proposes a new visualization tool that is web-based, open-source, and, most importantly, provides additional information about the genes in the phylogenetic trees.

1. Introduction

Evolutionary biology studies the descent and the origin of species (and their genes). Current researches in this field cover diverse topics and incorporate ideas from molecular genetics, systematics, phylogenetics, bioinformatics, and other, to study the evolutionary history and relationships among living organisms. These relationships are discovered through phylogenetic inference methods that evaluate observed heritable traits, such as DNA sequences or morphology under a model of evolution of these traits [1].

The graphical result of evolutionary analyses is a phylogeny (also known as a phylogenetic tree). A phylogenetic tree shows inferred evolutionary relationships among various biological entities, based upon similarities and differences in their physical or genetic characteristics. It represents an undirected, acyclic, bifurcating graph with a distinct, albeit sometimes virtual, starting point. Phylogenetic trees (**Figure 1**) describe the "probable" evolutionary relationships between entities that correspond to the leaves (external nodes) of the tree. Entities can be species,

genes or any operational taxonomic units (OTUs), even unknown or multiple species. Internal tree nodes represent hypothetical common ancestors of the descending leaves [1], [2], [3].

Evolutionary biology, like many other science fields, strongly relies on computational methods and algorithms. There are various bioinformatics toolsets used in evolutionary analyses that include *in silico* methods developed to reconstruct, analyze, and visualize evolutionary events. Computationally, phylogenetic trees are presented in different file formats (Newick [4], PhyloXML [5], Nexus [6], etc.) and stored in phylogenetic (PhylomeDB [7], TreeBase [8], etc.) or general biological repositories (Ensembl [9], NCBI [10], etc.). Today, there are numerous visualization tools designed to read the phylogenetic tree structure from files and graphically represent it to users. These tools, however, seldom have any other functionality, such as, for example, visualizing additional information for the OTUs or storing/sharing trees. Also, most of these tools are desktop applications which make them dependable on users' operational system and hardware resources, or proprietary software [11].

Regarding the abovementioned limitations of current software, the goal of this research is to develop a platform-undependable, web-based application which allows visualization, interpretation, supplementary data addition, storage and sharing of phylogenetic trees.

2. Materials and methods

2.1. Phylogenetic trees

The application was designed to be used with data in Newick (*Newick notation* or *New Hampshire*) tree format, which is one of the most common phylogenetic formats and is incorporated in most tree reconstruction software packages. Trees in Newick format are represented as one-line strings using nested parentheses and commas. The tree starts with a parenthesis and ends with a semicolon. Interior nodes are represented by a pair of matched parentheses. Between them are representations of the nodes that are immediately descended from that node, separated by commas. Leaves can be left empty or represented by their names, which can be any string of printable characters except blanks, colons, semicolons, parentheses, and square brackets. Branch lengths can be incorporated into a tree by putting a real number, with or without decimal point, after a node and preceded by a colon. This represents the length of the branch immediately below that node. Typically, a tree's representation is rooted on an internal node and it is rare (but legal) to root a tree on a leaf node. When an unrooted tree is represented in Newick notation, an arbitrary node is chosen as its root [4].

The test data input for this project was retrieved from the 19th release of the Ensembl Plants database [9]. It consists of 18330 gene phylogenetic trees in a single flat file dump in Ensembl Multi Format (emf). In addition to the Newick string representing the tree itself, this format also includes additional information about each gene (**Figure 2**).

2.2. User interface

Both anonymous and registered users can use the application. Registered users have to provide a valid e-mail, username and a password to create an account. They are assigned a unique identification number (ID) during registration and are be able to view, import, edit, browse and exchange phylogenetic trees and supplement data, while anonymous users' abilities are limited to browsing and viewing.

Phylogenetic trees also receive unique IDs after being uploaded. The trees and the metadata associated with them are editable only by the users who have uploaded them and changes can be made by other users only after the initial uploader's authorization. Phylogenetic trees are described as Newick strings and one or more parameters in the form of metadata can be attributed to each leaf.

2.3. Technological background

Regarding the functionality and specifications of the tool, a PHP-based software application was developed implying the following practices:

- Object-oriented programming (OOP) – a programming paradigm based on the concept of "objects" (data) in the form of fields (attributes); and code in the form of procedures (methods). An object's procedures can access and often modify the data fields of the object with which they are associated [12].
- Model-View-Controller (MVC) – a software architectural pattern for implementing user interfaces that divides a given software application into three interconnected parts to separate internal representations of information from the ways that information is presented to or accepted from the user [13]. Traditionally used for desktop graphical user interface.
- Data access layer (DAL) and a simplified Object-relational mapping model that allows representing the input data as PHP objects and classes [14].
- Smarty – a web template system that allows PHP programmers to define custom functions that can be accessed using Smarty tags, later processed and substituted with other code [15].

3. Results

After determining the structure of the application and creation the test database, the approach to developing the software is straightforward. First, queries to the application trigger a configurational script. It loads all the necessary libraries and models, and creates instances of the classes *AdoDB* [16] and *Smarty*. The configuration file is located in */inc/config.inc. php*. All processes follow the same mechanism: the user makes a *HTTP* query to the server through the graphic user interface (GUI). Based on the query parameters, the corresponding controller invokes the suitable functions of the model, supplies the view with the processed data and retrieves a resulting *HTML*. To achieve this process the following tasks were fulfilled:

1) All necessary ORM (Object-relational mapping) models, classes and controllers were developed;

2) All views and their corresponding interfaces were designed;

3) All external libraries needed for the system functionalities (*Smarty*, *AdoDB*, *PhyloCanvas*, *gettext*) were integrated;

4) The *PhyloCanvas* [17] library was modified according to the project special requirements

3.1. Phylogenetic tree visualization:

The visualization of a current tree is done by the *PhyloCanvas* library in the *Tree/view.tpl* view. To achieve this, a *HTML* object, labelled with the ID „phylocanvas“, is created. The parameter for this HTML object is the ID of the PHP object from the Tree model.

The script sends an *AJAX* query to the *Tree* controller and thus retrieves the *Newick* string for the current tree. An example of the resulting visualization is shown of **figure 3**

For larger trees the *PhyloCanvas* functionality allows zooming in and out, collapsing and expanding subtrees at a given internal node and exporting trees/ subtrees as images or leaf labels as text.

3.2. Editing leaf-associated metadata:

The library uses a HTML 5 Canvas, where each leaf is a separate object. On mouse click over a leaf node, an AJAX query is sent to the controller which retrieves all supplementary data associated with the current leaf – in general, this includes information about the gene (retrieved from the local database or form Ensembl) and available functional annotations or predictions.

The GUI shown in response to the query can also allow edition of the data, provided that the current user is authorized to do so.

3.3. Multilinguialism and *Gettext* integration

To aid the user experience, the system interface is designed to be easily translated into multiple languages. A suitable solution to this issue is the integration of the application with *Gettext* library [18]. It is an internationalization and localization system commonly used for writing multilingual programs on Unix-like computer operating systems.

4. Discussion:

The resulting product, *PhyloEdit* (http://phyloedit.org/index.php), is a web-based, platform-independent, open-source GUI, able to view, store, share, and, to an extent, edit phylogenetic trees and supplemental data associated with them. The open-source software is one of the main advantages of this application and it will, ideally, wrap around itself a community of researchers and developers to further support and add new functionalities to it. This perspective would make *PhyloEdit* a popular and useful tool for scholars and scientists in the field of evolutionary analysis.

PhyloEdit relies on standard hosting technologies which make it flexible and adjustable to the volume of data and users working with it. Initially designed to use a shared hosting services, it can be conveniently transferred to a more resourceful service, if needed.

Compared to similar web-applications, *PhyloEdit* provides an extended functionality including, but not limited to, sharing information between users on different levels, integration with the Ensembl database and, simultaneously, storing information independently from it, integration with available functional annotations and ability to predict function of unannotated genes, and more.

Last, but not least, *PhyloEdit* is accessible to anyone. The project code can be found at the public repository *GitHub* [19] on the following address:

https://github.com/arimano/plyloedit

5. Conclusion

The current project successfully developed a model GUI for visualization, storage and analysis of evolutionary data. The application architecture is adjustable to additional functionalities and services. The main perspective of the conceptual model and its web-application is further development of its ability to predict functional annotations based on semantic models, as well as addition of other available phylogenetic data formats.

Acknowledgments. This work has been supported by the National Science Fund of Bulgaria within the “Methods for Data Analysis and Knowledge Discovery in Big Sequencing Datasets” Project, Contract DFNI-I02/7 of 12 December 2014.

References

[1] Barton N H, D E G Briggs, J A Eisen, D B Goldstein, N H Patel (2007). Evolution. Cold Spring Harbor LP

[2] Edwards A W F and L L Cavalli-Sforza (1964). Reconstruction of evolutionary trees. eds. Vernon Hilton Heywood and J McNeill. Phenetic and Phylogenetic Classification 6(6): 67-76.

[3] Hodge T and M Cope (2000). A myosin family tree. J Cell Sci 113 (19): 3353–4.

[4] Archie J, W H Day, J Felsenstein, W Maddison, C Meacham, E J Rohlf, D Swofford (1986). The Newick tree format. More information: http://evolution. genetics. washington. edu/phylip/newicktree. html.

[5] Han M V and C M Zmasek (2009). PhyloXML: XML for evolutionary biology and comparative genomics. BMC Bioinformatics (United Kingdom: BioMed Central) 10: 356

[6] Maddison DR, D L Swofford, W P Maddison (1997). NEXUS: An extensible file format for systematic information. Systematic Biology 46 (4): 590–621

[7] Huerta-Cepas J, A Bueno, J Dopazo, T Gabaldón. (2008). PhylomeDB: a database for genome-wide collections of gene phylogenies.Nucleic acids research, 36(suppl 1), D491-D496.

[8] Piel W H, M J Donoghue, M J Sanderson, L U T Netherlands (2000). TreeBASE: a database of phylogenetic information. In Proceedings of the 2nd International Workshop of Species 2000.

[9] Kersey P J, J Allen, M Christensen, P Davis, L J Falin, C Grabmueller, D Seth, T Hughes, J Humphrey, A Kerhornou, J Khobova, N Langridge, M McDowall, U Maheswari, G Maslen, M Nuhn, C K Ong, M Paulini, H Pedro, I Toneva, M A Tuli, B Walts, G Williams, D Wilson, K Youens-Clark, M K Monaco, J Stein, X Wei, D Ware, D M Bolser, K L Howe, E Kulesha, D Lawson, D M Staines (2014). Ensembl Genomes 2013: scaling up access to genome-wide data. Nucleic acids research, 42 (D1): D546-D552.

[10] Coordinators, N. R. (2015). Database resources of the national center for biotechnology information. Nucleic acids research, 43(Database issue), D6.

[11] Felsenstein, J, J Archie, W H E Day, W Maddison, C Meacham, F J Rohlf, D Swofford (1986). Society for the Study of Evolution meeting, Durham, New Hampshire, US http://evolution.genetics.washington.edu/phylip/newicktree.html

[12] Kindler E and I Krivy (2011). Object-Oriented Simulation of systems with sophisticated control. International Journal of General Systems pp 313–343.

[13] Reenskaug T and J Coplien (2009). The DCI Architecture: A New Vision of Object-Oriented Programming.

[14] Mitchell S (2010). Teach Yourself ASP.NET 4 in 24 Hours 1st edition. Pearson Education, Inc.

[15] Ohrt M, U Tews (2001-2011).Smarty - the compiling PHP template engine. New Digital Group, Inc (http://www.smarty.net)

[16] Regad D, M Newnham (2014). ADOdb - Database Abstraction Layer for PHP (http://adodb.sourceforge.net/)

[17] Goater R (2015). PhyloCanvas tree-drawing library. Centre for Genomic Pathogen Surveillance, WTSI (http://phylocanvas.org)

[18] Ueno D (2015). GNU gettext. Free Software Foundation, Inc (https://www.gnu.org/software/gettext/)

[19] Dabbish L, C Stuart, J Tsay, J Herbsleb (2012). Social coding in GitHub: transparency and collaboration in an open software repository. In Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work (pp. 1277-1286). ACM.

Figures and legends of figures

Figure 1: The tree structure incudes nodes (represented in the figure as circles) and branches (lines connecting nodes). External nodes (leaves) represent the actual OTUs – in this case, virus genomes. Internal nodes indicate putative ancestors for the sampled viruses. Trees A and B are rooted which suggests the location of the ultimate common ancestor of all sampled viruses. Knowing this gives the tree an order of branching events. Estimated branch lengths (numbers above branches in tree A) indicate nucleotide substitutions per site and correspond to the evolutionary distance between events. A scale bar for branch lengths is seen at the bottom of each tree. There are three types of phylogenetic tree formats, all describing the same tree: A) classical (rectangular); B) polar (circular) - big visual impact but generally have reduced readability; and C) radial – used when the rooting of the tree is not known.

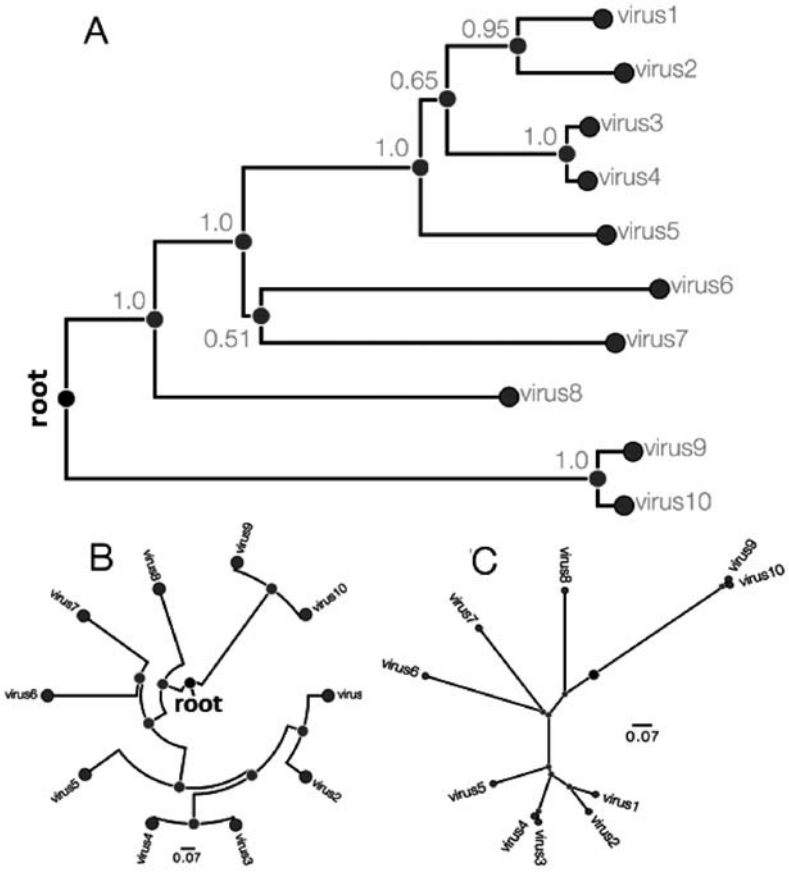


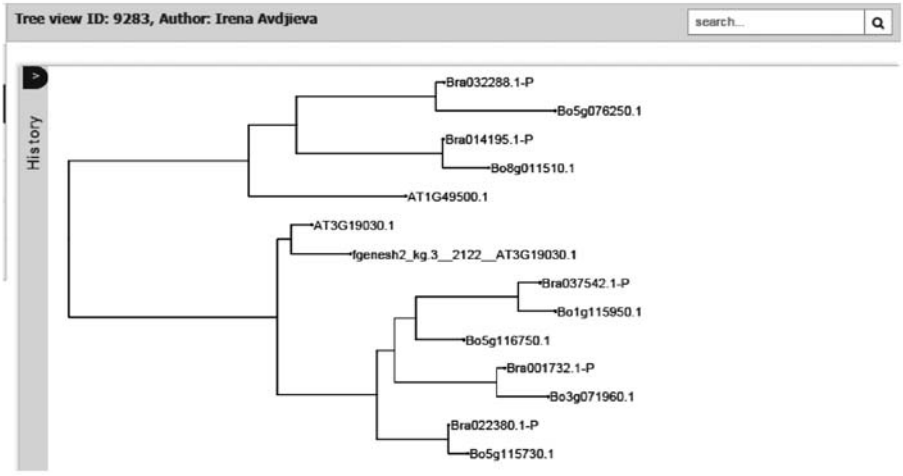
Figure 2: A single tree in Ensembl Plants EMF file.

```
1 SEQ setaria_italica Si004553m scaffold_5 43010075 43013802 1 Si004553m.g Si004553m.g
  SEQ setaria_italica Si003148m scaffold_5 25168442 25171496 -1 Si003148m.g Si003148m.g
  SEQ sorghum_bicolor Sb02g039633.1 2 73687862 73688508 1 Sb02g039633 SB02G039633
  SEQ zea_mays GRMZM2G424150_P01 5 187385096 187385485 -1 GRMZM2G424150 NULL
2 DATA
3 ((Sb02g039633.1:0.4013,GRMZM2G424150_P01:0.6260):0,(Si003148m:0.0911,Si004553m:0.1566):0):0;
4 //
```

KEY:

1. Supplementary data (a line for each gene)
[SEQ][organism][translation ID][chromosome][start(bp)][stop(bp)][strand][gene ID/NULL]
2. Separator between supplementary data and phylogenetic tree
3. Phylogenetic tree in NEWICK format (one line)
((gene1:branch_length,gene2:branch_length):0,(gene3:branch_length,gene4:branch_length):0):0;
4. Separator between individual phylogenetic trees

Figure 3: Visualization of a phylogenetic tree in PhyloEdit.



CWE-119 in Z-Notation

Vladimir Dimitrov

Faculty of Mathematics and Informatics, University of Sofia,
5 James Bourchier Blvd., 1164 Sofia, Bulgaria
cht@fmi.uni-sofia.bg

Abstract. Software weaknesses are described in formatted text. There is no widely accepted formal notation for that purpose. This paper shows how Z-notation can be used for formal specification of CWE-119.

Key words: *Z-notation, CWE, formal specification.*

1 Introduction

So-called weaknesses, introduced during the program development phase, are leading problem in Computer security. A weakness becomes vulnerability when a hacker attacks the software using successfully this weakness.

Some definitions of used above terms, taken from [1], are listed below:

- "*A1. What is CWE? What is a "software weakness"?*

Targeted at both the development community and the community of security practitioners, Common Weakness Enumeration (CWE™) is a formal list or dictionary of common software weaknesses that can occur in software's architecture, design, code or implementation that can lead to exploitable security vulnerabilities. CWE was created to serve as a common language for describing software security weaknesses; serve as a standard measuring stick for software security tools targeting these weaknesses; and to provide a common baseline standard for weakness identification, mitigation, and prevention efforts.

Software weaknesses are flaws, faults, bugs, vulnerabilities, and other errors in software implementation, code, design, or architecture that if left unaddressed could result in systems and networks being vulnerable to attack. Example software weaknesses include: buffer overflows, format strings, etc.; structure and validity problems; common special element manipulations; channel and path errors; handler errors; user interface errors; pathname traversal and equivalence errors; authentication errors; resource management errors; insufficient verification of data; code evaluation and injection; and randomness and predictability."

- "*A2. What is the difference between a software vulnerability and software weakness?*

Software weaknesses are errors that can lead to software vulnerabilities. A software vulnerability, such as those enumerated on the Common Vulnerabilities

and Exposures (CVE®) List, is a mistake in software that can be directly used by a hacker to gain access to a system or network."

- "*A7. What is the relationship between CWE and CAPEC?*

While CWE is a list of software weakness types, Common Attack Pattern Enumeration and Classification (CAPEC™) is a list of the most common methods attackers use to exploit vulnerabilities resulting from CWEs. Used together, CWE and CAPEC provide understanding and guidance to software development personnel of all levels as to where and how their software is likely to be attacked, thereby equipping them with the information they need to help them build more secure software."

- "*A8. What is the relationship between CWE and CVE?*

MITRE began working on the issue of categorizing software weaknesses as early 1999 when it launched the Common Vulnerabilities and Exposures (CVE®) List. As part of building CVE, MITRE's CVE Team developed a preliminary classification and categorization of vulnerabilities, attacks, faults, and other concepts beginning in 2005 to help define common software weaknesses. However, while sufficient for CVE those groupings were too rough to be used to identify and categorize the functionality offered within the offerings of the code security assessment industry. The CWE List was created to better address those additional needs."

- "*A1. What is CVE?*

CVE is a list of information security vulnerabilities and exposures that aims to provide common names for publicly known cyber security issues. The goal of CVE is to make it easier to share data across separate vulnerability capabilities (tools, repositories, and services) with this "common enumeration."""

- "*A8. What is a "vulnerability"?*

An information security vulnerability is a mistake in software that can be directly used by a hacker to gain access to a system or network. See the Terminology page for a complete explanation of how this term is used on the CVE Web site."

- "*A9. What is an "exposure"?*

An information security exposure is a mistake in software that allows access to information or capabilities that can be used by a hacker as a stepping-stone into a system or network. See the Terminology page for a complete explanation of how this term is used on the CVE Web site."

- "*What is an "attack pattern"?*

An attack pattern is an abstraction mechanism for helping describe how an attack against vulnerable systems or networks is executed. Each pattern defines a challenge that an attacker may face, provides a description of the common technique(s) used to meet the challenge, and presents recommended methods for mitigating an actual attack. Attack patterns help categorize attacks in a meaningful way in an effort to provide a coherent way of teaching designers

and developers how their systems may be attacked and how they can effectively defend them. The CAPEC List provides a formal list of known attack patterns."

In the databases CWE, CVE and CAPEC: the weaknesses, vulnerabilities and attacks are structured, but not formalized. This leads to misinterpretation of the text.

This paper is an initial attempt to CWEs formalization. Z-notation [1, 2] is well-known standard notation, suitable for that purpose. The paper focuses on CWE-119.

2 CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer

The description of CWE-119, taken from the database, is:

"**Description**

Description Summary

The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.

Extended Description

Certain languages allow direct addressing of memory locations and do not automatically ensure that these locations are valid for the memory buffer that is being referenced. This can cause read or write operations to be performed on memory locations that may be associated with other variables, data structures, or internal program data.

As a result, an attacker may be able to execute arbitrary code, alter the intended control flow, read sensitive information, or cause the system to crash.

Alternate Terms

Memory Corruption:

The generic term "memory corruption" is often used to describe the consequences of writing to memory outside the bounds of a buffer, when the root cause is something other than a sequential copies of excessive data from a fixed starting location (i.e., classic buffer overflows or CWE-120). This may include issues such as incorrect pointer arithmetic, accessing invalid pointers due to incomplete initialization or memory release, etc."

For more details visit http://cwe.mitre.org/data/definitions/119.html.

3 Formal Description of CWE-119

The basic subject of CWE-119 is the memory. It is a sequence of bytes. The basic type is the byte.

[*Byte*]

Memory address space starts from 0 and ends to some max address.

| $maxAddress: \mathbb{N}$

Memory is finite sequence of bytes – possibly empty sequence. The finite sequences elements in Z-notation are numbered starting from 1, but computer memory bytes are indexed starting from 0.

Memory

$$contents: \text{seq } Byte$$

$$\#contents = maxAddress + 1$$

There are two operations with the memory: read and write that a program can perform. First one is read.

read

$$\Xi Memory$$
$$i?: \mathbb{N}$$
$$r!: Byte$$

$$i? \in \text{dom } contents$$
$$r! = contents(i?+1)$$

Second one is write.

write

$$\Delta Memory$$
$$i?: \mathbb{N}$$
$$w?: Byte$$

$$i? \in \text{dom } contents$$
$$contents' = (1..i?) \lhd contents \frown \langle w? \rangle \frown ((i?+2)..\#contents) \lhd contents$$

$i? \in \text{dom } contents$
$contents' = (1..i?) \lhd contents \frown \langle w? \rangle \frown ((i?+2)..\#contents) \lhd contents$

A program performs in memory on a buffer. The buffer is specified with its address space.

Program

$$Memory$$
$$m, n: \mathbb{N}$$
$$BufferSpace: \mathbb{P}\, \mathbb{N}$$

$$m \leqslant n \leqslant maxAddress$$
$$BufferSpace = m..n$$

Every program run performs on a given buffer. The property bad run is CWE-119, i.e. read or write outside the buffer.

$$
\begin{array}{|l}
\textit{badRun} \\
\hline
\Delta Program \\
m?, n?: \mathbb{N} \\
\hline
m = m? \land n = n? \\
\exists i: \mathbb{N} \bullet i \notin BufferSpace \land ((\exists r: Byte \bullet read[i/i?, r/r!]) \lor (\exists w: Byte \bullet write[i/i?, w/w?])) \\
\hline
\end{array}
$$

4 Conclusion

CWE-119, following this description, is an operation that accesses the memory outside the buffer space. This operation can be part of any other program operation.

CWE-119 specification is enough abstract and formal in comparison with its textual representation. On the other hand, the CWE-119 description is at very low level – the memory. Therefore, the level of the specification is not very high, but CWE-119 is formalized.

The more general question is "How suitable is the Z-notation for formal specification of CWEs?" More general conclusion to do from one example is not correct – more research is needed.

How the specification can be used? Z-notation supporting tools can extract automatically from the specification pre- and post-conditions. Formal verification tools can use these conditions to check the software for CWE-119. However, the devil is in the details – the problem is how to link the abstract buffer space with the real one of the programs. There are some ideas in that direction, but more research is needed.

This research is a result of collaboration with NIST.

5 References

1. The MITRE Corporation, www.mitre.org.
2. "ISO/IEC 13568:2002". Information Technology — Z Formal Specification Notation — Syntax, Type System and Semantics. ISO. 2002-07-01. 196 pp.
3. "ISO/IEC 13568:2002/Cor.1:2007". Information Technology — Z Formal Specification Notation — Syntax, Type System and Semantics — Technical corrigendum 1 (PDF). ISO. 2007-07-15. 12 pp.

The Need to Formalize the Software Bugs

Vladimir Dimitrov

Faculty of Mathematics and Informatics, University of Sofia,
5 James Bourchier Blvd., 1164 Sofia, Bulgaria
cht@fmi.uni-sofia.bg

Abstract. This paper motivates the need for software bug formalization. It is based on MITRE classification.

Key words: *software bugs, formal specification.*

1 Introduction

Knowing what makes a software systems vulnerable to attacks is critical, as software vulnerabilities hurt security, reliability, and availability of the system as a whole. In addition, understanding how an adversary operates is essential to effective cyber security.

The term "software bug" [1] applies to the following concepts:

- **Weakness**: A type of mistake in software that, in proper conditions, could contribute to the introduction of vulnerabilities within that software. This term applies to mistakes regardless of whether they occur in implementation, design, or other phases of the SDLC." [2]
- **Vulnerability**: "An occurrence of a weakness (or multiple weaknesses) within software, in which the weakness can be used by a party to cause the software to modify or access unintended data, interrupt proper execution, or perform incorrect actions that were not specifically granted to the party who uses the weakness." [3]
- **Attack**: A well-defined set of actions that, if successful, would result in either damage to an asset, or undesirable operation. [4] (An attacker has to find and exploit a weakness, exposed by a vulnerability, and realize the vulnerability.

These are important concepts that are related but different. A weakness is a static presence existing in software systems -- it might stay in software and never cause any problems until it is exploited by an attacker, and when the attacker finds out the weakness(es) and exploit it (them), the vulnerability of this software is exposed [4].

2 Common Vulnerabilities and Exposures (CVE)

CVE is a dictionary of security vulnerabilities. It was established in 1999 in response to lack of standardization of names of vulnerabilities: different repositories could refer to the same vulnerability by a different name, resulting in difficulty in comparing software security tools. CVE provides standard identifiers for security vulnerabilities, and help in finding information about a vulnerability, including ways of, and available products for, eliminating the vulnerability. It can also help in determining whether particular tools are adequate for detecting attacks that are based on particular vulnerabilities[1].

After discovering a potential security vulnerability, a CVE Numbering Authority (CNA) can assign to it a CVE identifier [2]. Then the CVE Editor posts the information on the CVE List. The Primary CNA is MITRE Corporation. Other CNAs are software vendors, (for example, Apple Inc. and Adobe Systems Incorporated), third-party coordinators, (for example, CERT/CC), or researchers (for example, Core Security Technologies). The CVE Editor is MITRE Corporation.

3 Common Weaknesses Enumeration (CWE)

Common Weakness Enumeration (CWE) is a collection of descriptions of software weakness types stored as .xml, .xsd and .pdf documents. There are four major types of CWE-IDs:

1) Category
2) Compound Element
3) View
4) Weakness.

The weaknesses covered by CWE have weakness IDs. Category and Compound Element are aggregations of weaknesses. Category aggregates types of weaknesses, and Compound Element aggregates a group of several events that together can result in a successful attack. View IDs are “assigned to predefined perspectives with which one might look at the weaknesses in CWE.” [2]

CWE was established for those who create software, analyze software for security flaws, and provide tools and services for finding and defending against security flaws in software [1]. The CWE Compatibility and Effectiveness Program is based on six requirements:

1) “CWE Searchable,”
2) “CWE Output,”
3) “Mapping Accuracy,”
4) “CWE Documentation,”
5) “CWE Coverage,”
6) “CWE Test Results.”

4 Common Attack Pattern Enumeration and Classification (CAPEC)

Common Attack Pattern Enumeration and Classification (CAPEC) is a comprehensive dictionary and classification taxonomy of known attacks that can be used by analysts, developers, testers, and educators to advance community understanding and enhance defenses [3].

CAPEC is actually a catalog of attack patterns along with a comprehensive schema and classification taxonomy created to assist in the building of secure software. Attack patterns examples: HTTP response splitting, SQL injection, XSS in HTTP query strings, Session fixation, Phishing, Filter failure through buffer overflow, Removing or short-circuiting guard logic, Lifting data embedded in client distributions, Subvert code-signing facilities, Reflection attack in an authentication protocol, Cause web server misclassification, Rainbow table password cracking, Forced deadlock, Cache poisoning, Restful privilege escalation [5].

5 Problems with CWE, CVE, and CAPEC: Non-Precise Descriptions

CWE, CVE, and CAPEC are considerable efforts, providing foundational knowledge about software weaknesses, vulnerabilities, and attacks. However they are not sufficient, accurate and precise enough to serve as common measuring sticks and provide common base for software developers and security practitioners. There is luck of precise descriptions of attacks that lead to realization of vulnerabilities, exposed by software weaknesses.

CWE definitions have ambiguities, entangle phases, and do not match well with the classes reported by test tools. For example, the description summary of CWE-119 includes text such as “intended boundary”, which is too vague. It does not indicate that it is the boundary given by the formal semantics.

CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer - “The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.”

While, to mitigate the vagueness of the definition as much as possible, our tentative definition of CWE-119 is: The software can access through a buffer a memory location not allocated to that buffer [6].

CVE definitions need to describe unambiguously and fully describe the involved weakness (es) exposure. For example, CVE-160 description and Related Weaknesses parts do not provide information on possible chains of weaknesses that lead to the vulnerability realization.

CVE-160: Heartbleed - "The (1) TLS and (2) DTLS implementations in OpenSSL 1.0.1 before 1.0.1g do not properly handle Heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read, as demonstrated by reading private keys, related to d1_both.c and t1_lib.c, aka the Heartbleed bug."

CAPEC definitions also need formalization. For example, the CAPEC-47 description does not show the dynamics of the activities leading to the realization of the attack.

CAPEC-47: Buffer Overflow via Parameter Expansion - "In this attack, the target software is given input that the attacker knows will be modified and expanded in size during processing. This attack relies on the target software failing to anticipate that the expanded data may exceed some internal limit, thereby creating a buffer overflow."

This research would like to explore formalization of CWEs, CVEs, and CAPECs.

6 Formalization

What is formal specification? In [7] it is defined as:

"Formal specifications use mathematical notation to describe in a precise way the properties which an information system must have, without unduly constraining the way in which these properties are achieved. They describe what the system must do without saying how it is to be done. This abstraction makes formal specifications useful in the process of developing a computer system, because they allow questions about what the system does to be answered confidently, without the need to disentangle the information from a mass of detailed program code, or to speculate about the meaning of phrases in an imprecisely-worded prose description.

A formal specification can serve as a single, reliable reference point for those who investigate the customer's needs, those who implement programs to satisfy those needs, those who test the results, and those who write instruction manuals for the system. Because it is independent of the program code, a formal specification of a system can be completed early in its development. Although it might need to be changed as the design team gains in understanding and the perceived needs of the customer evolve, it can be a valuable means of promoting a common understanding among all those concerned with the system."

Candidate tools for formal specification of software bugs are Z-Notation [7], CSP [8] and UML [9]. In the next following a motivation for this choice is given.

Z-notation is a standard notation standardized by ISO. It is high level, based on mathematics notation. Z-notation is applicable on any level of abstraction. It is useful for software requirement specification and code verification. The

problem with it is the limited set of tools supporting the notation – mainly at academia. Z-notation can be used for CWE and CVE specification (weaknesses and vulnerabilities).

Communicating Sequential Processes (CSP) is well-designed algebra. It is useful for distributed application specification and verification. It is behavioral in nature and can be used for CAPEC specification (attacks). Its problem is like that of Z-notation.

An alternative of Z-notation and CSP is UML. This notation is well supported in industry. Activity diagrams can be used for CAPEC specification, class diagrams and object diagrams for CWE and CVE can be used. Even more, special kind of CWE, CVE and CAPEC can be developed; code for software bug tests can be automatically generated. The problem is that all these possibilities must be carefully investigated.

7 Conclusion

Some preliminary results in above mentioned directions have been done and soon will be published, but more research must be done.

8 References

1. MITRE. “CVE Common Vulnerabilities and Exposure.” http://cve.mitre.org
2. MITRE. “CWE Common Weakness Enumeration.” http://cwe.mitre.org
3. MITRE. “Common Attack Pattern Enumeration and Classification”, https://capec.mitre.org
4. Wu, Y., “Using Semantic Templates to Study Vulnerabilities Recorded in Large Software Repositories”, Dissertation, October 2011, Omaha, Nebraska.
5. Wu, Y., R. A. Gandhi, and H. Siy. “Using semantic templates to study vulnerabilities recorded in large software repositories.” 2010 ICSE Workshop on Software Engineering for Secure Systems, SESS ‘10, pages 22-28, New York, NY, USA, 2010. ACM.
6. R. Gandhi, H. Siy, Y. Wu. “Studying Software Vulnerabilities.” CrossTalk, The Journal of Defense Software Engineering, September/October 2010.
7. Spivey, J. M. “The Z Notation: A reference manual”, International Series in Computer Science (2nd ed.). Prentice Hall, 1992.
8. Hoare, C. A. R. (2004). Communicating Sequential Processes. Prentice Hall International, 1985.
9. Unified Modeling Language (UML), http://www.uml.org.

Semantic Templates and Software Fault Patterns – an Overview

Vladimir Dimitrov

Faculty of Mathematics and Informatics, University of Sofia,
5 James Bourchier Blvd., 1164 Sofia, Bulgaria
cht@fmi.uni-sofia.bg

Abstract. Semantic templates and Software fault patterns are overviewed as tools for software bugs specification tools. Further research on the topic is discussed.

Key words: software bugs, semantic templates, software fault patterns.

1 Introduction

The term "software bug" [1] applies to the following concepts:

- **Weakness** is a type of mistake in software that, in proper conditions, could contribute to the introduction of vulnerabilities within that software. This term applies to mistakes regardless of whether they occur in implementation, design, or other phases of the SDLC." [2]
- **Vulnerability** is "An occurrence of a weakness (or multiple weaknesses) within software, in which the weakness can be used by a party to cause the software to modify or access unintended data, interrupt proper execution, or perform incorrect actions that were not specifically granted to the party who uses the weakness." [3]
- **Attack**: is a well-defined set of actions that, if successful, would result in either damage to an asset, or undesirable operation [4] (An attacker has to find and exploit a weakness, exposed by a vulnerability, and realize the vulnerability.

These are important concepts that are related but different. A weakness is a static presence existing in software systems -- it might stay in software and never cause any problems until it is exploited by an attacker, and when the attacker finds out the weakness (es) and exploit it (them), the vulnerability of this software is exposed [4].

CVE is a dictionary of security vulnerabilities. CVE provides standard identifiers for security vulnerabilities, and help in finding information about a vulnerability, including ways of, and available products for, eliminating the vulnerability. It can also help in determining whether particular tools are adequate for detecting attacks that are based on particular vulnerabilities [1].

Common Weakness Enumeration (CWE) is a collection of descriptions of software weakness types stored as .xml, .xsd and .pdf documents. CWE was established for those who create software, analyze software for security flaws, and provide tools and services for finding and defending against security flaws in software [1].

Common Attack Pattern Enumeration and Classification (CAPEC) is a comprehensive dictionary and classification taxonomy of known attacks that can be used by analysts, developers, testers, and educators to advance community understanding and enhance defenses [3].

2 Semantic Templates

A Semantic Template is a human and machine understandable representation that contains the following four elements [5]:

1) Software faults that lead to a weakness;
2) Resources that a weakness affects;
3) Weakness characteristics;
4) Consequences/failures resulting from the weakness.

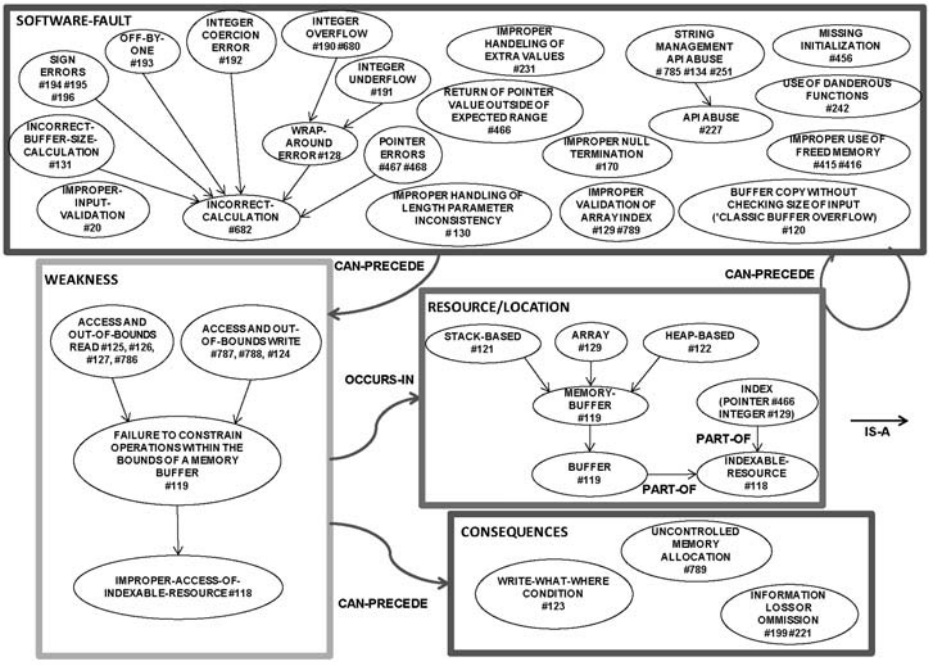


Fig. 1. Buffer Overflow Semantic Template.

The required information pieces are either expressed together within a single CWE entry or spread across multiple entries. Such complexity makes it difficult to trace the information expressed in the CWE to the information about a discovered vulnerability from multiple sources. Therefore, to facilitate CWE use in the study of vulnerabilities, easy-to-understand templates for each conceptually distinct weakness type has been developed. These templates can then be readily applied to study project-specific vulnerability information from project repositories. For example, Fig. 1 shows the Semantic Template for Buffer Overflow, which is an aggregation of information collected from 42 CWEs. In this Buffer Overflow Semantic Template, the four groups of relevant information were carefully collected and synthesized with "is-a" relationship inside of each group and "can-precede", "occurs-in" between the groups so that the lifecycle of a weakness from the starting point (software fault) to the end (consequences) is clearly presented.

The Semantic Templates also can provide intuitive visualization capabilities for the collected vulnerability information such as the CVE vulnerability descriptions, change history in the open source code repository, source code versions (before and after the fix), and related CAPECs [6]. Semantic Templates were shown to be helpful to programmers in constructing mental models of software vulnerabilities by an experiment described in [7]. In this experiment, 30 Computer Science students from a senior-level undergraduate Software Engineering course were selected to study six sets of vulnerability-related material with or without Semantic Templates in a pre-post randomized two-group design. The experimental results revealed that the group with the aid of Semantic Templates could analyze vulnerabilities with shorter time and higher recall on CWE identification accuracy.

3 Software Fault Patterns

Software Fault Patterns (SFPs) was developed by KDM Analytics Inc. By identifying and developing white box definitions for SFPs as a formalization process, they could be integrated into a standards-based tool analysis approach, benefiting both real-time embedded and enterprise software assurance systems. Those identified SFPs will be common to more than one CWE and can be used further to define CWEs [8].

The SFP is targeted at preventing cyber-attacks by collecting and managing knowledge about exploitable weaknesses and building more comprehensive prevention, detection and mitigation solutions. With the knowledge extracted from CWE taxonomy, three transformations were executed to extract common patterns and white-box knowledge, redefine existing weaknesses as specializations of the common patterns, and then invariant core and variation points are identified to redefine each SFP to represent further weakness specializations [8].

KDM Analytics defines an SFP as a common pattern with one or more associated pattern rules (conditions), representing a family of faulty computations. The SFP structure is organized by the primary SFP definition, which refers to the entire secondary cluster, and is arranged into invariant core and variation points [8]. SFPs can map to multiple CWEs in such a way that each CWE in the family can be defined as a specialization of the SFP with its specific variations on the identified parameters. To date, 21 primary clusters, which include totally 62 secondary clusters, and 36 unique SFPs have been identified. 632 CWEs have been categorized while only 310 of them are identified as discernible CWEs. Identified SFP definitions could lead to the development of more accurate testing tools and improve developer education and training. They also provide benefits for a possible future formalization, since for each CWE, only the variation extension to a formalized SFP is required.

As the proof of recognition of the SFP research work, CWE-888: Software Fault Pattern (SFP) Clusters was incorporated by MITRE as a view into the CWE dictionary.

Both Semantic Templates and SFPs are designed to help understand and automate the vulnerability study. While Semantic Templates emphasize mental model construction from the human perspective, with the explanation of the four main elements of a vulnerability's lifecycle, while SFP's approach focuses on the "foot-holds", which are places in the code that present the necessary conditions for vulnerabilities, with the emphasis on the computation side to aid the test cases generator's work.

4 The Use of Semantic Templates and Software Fault Patterns

One idea is to utilize Semantic Templates and Software Fault Patterns for software bug specification

For example, Fig. 2 demonstrates how the Buffer Overflow Semantic Template helps separate the distinct phases of software faults, weakness, resource/location, and consequence between which "can precede" and "occurs in" relationship exist. In addition, the figure contains how the Buffer Overflow SFP can be used to factor out parameters, such as resource and location.

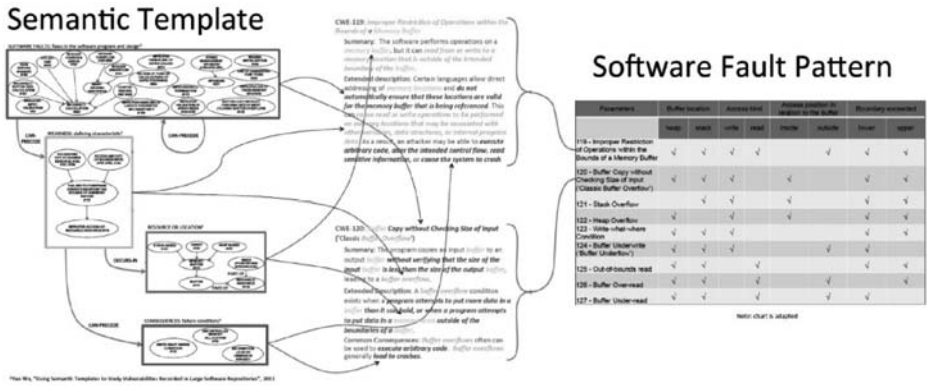


Fig. 2. Buffer Overflow Semantic Template and Software Fault Pattern.

Using this schema, accomplish the following tasks are accomplished:

- Task 1: Formally describe the vulnerabilities as chains of exploited weaknesses, corresponding to Yan Wu's Buffer Overflow and Injection Semantic Templates [4].
- Task 2: Formally describe the involved CWEs utilizing the corresponding SFPs.

However, the attempt to use this formalization approach failed midway, as we realized we needed to specify not only the static aspect of the weaknesses and the possible chains, but also the dynamic aspect of the attack.

For example, the Heartbleed vulnerability had been investigated [7], which was discovered on April 7, 2014 in OpenSSL, the most widely deployed cryptographic function on the web. This devastating vulnerability is currently defined under CVE-2014-0160 and possible chains of CWEs are described:

- both in CWE-130 and CWE-126 as: "Heartbleed bug receives an inconsistent length parameter (CWE-130) enabling an out-of-bounds read (CWE-126), returning memory that could include private cryptographic keys and other sensitive data."
- in "How to Prevent the next Heartbleed" (http://www.dwheeler.com/essays/heartbleed.html) as: "The key weakness can be classified as a buffer over-read (CWE-126) in the heap, which could happen because of improper input validation (CWE-20) of a heartbeat request message."

Note: CWE-126 is a special case of an "out-of-bounds read" (CWE-125), which itself is a special case of "improper restriction of operations within the bounds of a memory buffer" aka "improper restriction" (CWE-119).

```
Chain := +{ Software-Fault Weakness Consequence }

Note: The attacker can repeat this attack multiple times
```

and get different sets of bytes from memory as well – so it is a very important attack in that regard.

Software-Fault : = CWE-20 | CWE-130

Weakness := CWE-126

CWE-125 := (CWE-126 | …)

CWE-119 := (CWE-125 | …)

Consequence := (CWE-xxx | CWE-yyy | …)

Note: What CWE-xxx, CWE-yyy, etc. would cover leakage of sensitive data (user names passwords, encryption keys, instant messages emails, business documents, etc.?)

Resource := CWE-12x should be Heap based over-read (CWE-122 is over-write)

CWE-119 := (CWE-12x | …)

SFP8 Faulty Buffer Access -> heap-based buffer: as for CWE-126, with buffer location on the Heap. [8]

5 Conclusion

Our very first goal is to formalize one or two software vulnerabilities to expose the opportunities and challenges of formalization:

1. Develop a method for formal description of software attacks, exploited vulnerabilities, and involved CWEs.
2. Describe formally attacks exploiting some intriguing vulnerabilities: for example, Heartbleed or ShellShock.
3. Refine the involved CWEs, CVEs, and CAPECs definitions, so that they precisely and unambiguously described.

Note: This work closely related to the NIST Software Assurance Metrics and Tool Evaluation (SAMATE) project, which is dedicated to improving software assurance by developing methods to enable software tool evaluations, measuring the effectiveness of tools and techniques, and identifying gaps in tools and methods. The scope of the SAMATE project is broad: ranging from operating systems to firewalls, SCADA to web applications, source code security analyzers to correct-by-construction methods. Among the project objectives, the body

of knowledge in software weaknesses is the foundation of most research and application activities.

From Yan, Yacoov, Irena's Crosstalk paper [6]:

"To provoke further thinking and discussions throughout the Software Assurance community and beyond, we pose the following questions:

- What other formal methods can be used to help formalize CWEs with required accuracy and precision and at the same time allow for further extensions?
- To what granularity should CWEs be formalized? Finer granularity means more flexibility (especially when new weaknesses are identified, the extracted commonalities can reduce the re-invent work) but more effort to create them; Coarser granularity indicates the easy-to-use weakness items while we need to re-invent the wheel every time.
- How can the formalized CWEs be used and in which domains? For education and training? To prevent vulnerabilities? To integrate into software IDEs, test tools, and tools that generate test tools?
- How can an automatic system be constructed to record newly identified vulnerabilities and classify them by CWEs? With better formalization and finer granularity of CWE definitions (which also means limited dictionary for weaknesses, better taxonomy of vulnerabilities), text mining could be the potential technique to mapping CVEs to CWEs at least semi-automatically. "

In response to the above query, we decided to focus on the main fact that a vulnerability is a realization of a weakness of the software. This gives us the following two aspects:

1. A realization of a vulnerability happens through and attack or attacks, t.e. there is the dynamic aspect.
2. The exploited weakness itself is a property of the software, t.e. this is the static aspect.

The description of the dynamics of the attack can be done with CSP [8], while the property of the software (static) can be described with Z-notation [7].

Following the Semantic Templates idea, we explore in detail Yan Wu's dissertation, "Using Semantic Templates to Study Vulnerabilities Recorded in Large Software Repositories" [4]. The idea there is to build a database with knowledge about the vulnerabilities using the available repositories. In other words, first the repositories are annotated according the software template. Then, in each part of the template, semantic nets are organized with 3 kinds of arrows. She has reached only the idea that one vulnerability can be represented by several weaknesses in each component of the template. However, the problems are much more. and this rode is very difficult as for example, language problems will start interfering -- with two words, "artificial intellect" for example, the leading idea

is that Semantic Templates can be extracted automatically or semi-automatically from their natural language descriptions, but how clear and descriptive are these descriptions is an open question to do that. There are available instruments, but we doubt this is the proper direction to follow.

Let us look at the first task: Describe vulnerabilities as chains of weaknesses. What Yan does is connecting a given vulnerability with a concrete (root) weakness and from there develops the template. This means she misses the attack. While indeed, the attack is performed following a scheme, it is dynamic and rather the vulnerability is a successfully conducted attack, and not a property of the software. The latter is a weakness of the software.

We think that the vulnerabilities have to be described as attacks and not as chains of weaknesses.

If it is needed to describe the relationships between the weaknesses, as it is in Yan's dissertation, it is better to use the UML notation [9]. Diagrams are preferred nowadays. Even, UML can be specialized to define a diagram that reflects the relationships between the weaknesses. The dynamics of the vulnerabilities can also be presented with diagrams.

Anyway, we are talking about software and using the UML terminology, we need the three models: classes, states, and interactions. The static aspect of the relationships between the weaknesses can be presented with a class diagram or a specialized such. The dynamics in time of a separate weakness can be presented with a state diagram. The attacks and the connections between the weaknesses can be described with activity diagrams and other kinds of diagrams from the interactions model. Note that there are no good tools for re-engineering but there is work done in this direction.

6 References

1. MITRE. "CVE Common Vulnerabilities and Exposure." http://cve.mitre.org
2. MITRE. "CWE Common Weakness Enumeration." http://cwe.mitre.org
3. MITRE. "Common Attack Pattern Enumeration and Classification", https://capec.mitre.org
4. Wu, Y., "Using Semantic Templates to Study Vulnerabilities Recorded in Large Software Repositories", Dissertation, October 2011, Omaha, Nebraska.
5. Wu, Y., R. A. Gandhi, and H. Siy. "Using semantic templates to study vulnerabilities recorded in large software repositories." 2010 ICSE Workshop on Software Engineering for Secure Systems, SESS '10, pages 22-28, New York, NY, USA, 2010. ACM.
6. R. Gandhi, H. Siy, Y. Wu. "Studying Software Vulnerabilities." CrossTalk, The Journal of Defense Software Engineering, September/October 2010.
7. Spivey, J. M. "The Z Notation: A reference manual", International Series in Computer Science (2nd ed.). Prentice Hall, 1992.
8. Hoare, C. A. R. (2004). Communicating Sequential Processes. Prentice Hall International, 1985.
9. Unified Modeling Language (UML), http://www.uml.org.

Personalisation of learning environment for delivery of electronic services and electronic content

Daniela Orozova[1], Magdalina Todorova[2]

[1] Burgas Free University, Faculty of Computer Science and Engineering,
62 San Stefano Str., Burgas 8001, Bulgaria
orozova@bfu.bg
[2] "St. Kliment Ohridski" University of Sofia, Faculty of Mathematics and Informatics,
Sofia 1164, Bulgaria
todorova_magda@hotmail.com

Abstract. This article presents the key results of our work in the area of personalization of the electronic educational environment. The goal of effective design of mobile electronic content is achieved by developing SCORM packages of the course learning content. The delivery of electronic services and electronic content is based on autonomous software agents.

Keywords: Learning environment, Sharable Content Object Reference Model, Personal assistants, Data mining tools.

1 Virtual learning environment architecture

Integrating tools in the learning process used by the learners as a personal virtual space on an everyday basis is a way to engage their attention. The virtual learning environment is a context-dependent environment, through which an effective, supported by current information and communication technologies educational process is realized. The aim of the learning environment is to integrate the real learning process with the designed virtual world in an "intelligent" manner. The learning environments have to provide the following options: active and interactive participation; team work; searching for and sharing information; discussion and presentation; generating new knowledge; supporting learners' and tutors' activities; connection with experts, and last but not least personalization of the learning.

The learning environment infrastructure includes:

- ***Building elements*** of the environment: identities populating the environment. These can be: learners, tutors, administrators, personal assistants, digital libraries, electronic services, etc.
- ***Connections*** – the connections existing between the structural elements, which ensure their operating as a whole within the learning environment.

The context-dependence characterizes the behavior of the different identities

populating the environment. The concept of context-dependence is characterized by two main attributes [1]:

- Personalization – in the sense of planned adapting of the learning content with regards to the particular learners. Personalization can be viewed from different points of view, as a basis for any of them is a corresponding classification system. To this end, an individual user profile is maintained for each learner.
- Adaptability – shifting the learning process according to changes in the environment. In order to ensure such a quality, user modelling is performed. Regular measuring and evaluation of different indicators, showing the learning progress and acquisition development, is ensured. The model receives information from the environment on whether the learner has completed a given topic.

2 Presenting learning content

In order to effectively develop learning content, which to ensure its mobility integration and reusability of the components in different learning situations, the learning content has to be based on a predefined standard. To this end, research was conducted on the main concepts and grounding behind Sharable Content Object Reference Model (SCORM) – a collection of standards and specifications for web-based electronic educational technology. What was done was: choice of appropriate environments for creating and testing SCORM packages; systematization and schematization of the content of each course learning material.

2.1 SCORM Content Model

The SCORM Content Model describes the components used in designing learning content of learning objects. It also defines how the smaller learning objects are combined into bigger ones. It consists of assets, Sharable Content Object (SCO), activities and their combinations.

- **Asset**

Assets are the basic structure unit of a learning object. They give an electronic representation of data: text, images, sound, etc., which can be delivered by a web-client to the learners. Several assets can be combined to make a new asset. The assets can be described by metadata in order to be searched for and found in repositories, as well as to be reused and easily serviced.

- **Sharable Content Object**

SCO is a set of assets which comprise a united learning resource using

SCORM Run-Time Environment to communicate with the Learning Management System (LMS). The SCO is the lowest level of a learning resource, which is monitored by an LMS through SCORM Data Model. In order to increase the possibility to be reused, a given SCO has to be context independent. Thus it can be used in different learning items, and it can serve different purposes. In addition, a learning activity can combine more than one SCO, forming an item of a higher level.

The SCOs are relatively small units, which can be used in a multiple occasions in the learning context. SCORM does not impose a definite volume of these elements.

- **Activities**

A learning activity is a unit of instructions. A given activity can provide a learning object (SCO or asset), as well as can consist of several sub-activities. Activities that have no sub-activities are associated with learning resource (SCO or Asset), which presents relevant material of the learner. Activities which have sub-activities are clusters or parent activities.

- **Content Organization**

Content organization shows the relations among the activities while forming structural units. Sequences of units (Asset or SCO) can be applied on actions and groups of activities. Activities and relations between them are described in these sequences. The learning management system is responsible for the interpretation of the information, which is described in the content organization, and for applying it on the learning objects during the performance. In SCORM, the information about activity sequences is external for the learning objects. The LMS is responsible for providing the learning objects in the predefined order.

- **Content Aggregation**

Content aggregation can be used as an action or process towards developing functionally interrelated object set, which to be applied in an educational activity.

2.2 SCORM Content Packaging

The aim of the packaging is to offer standardized means of learning content exchange among different system or tools. It also provides space for describing the structure and expected behavior of the content. The IMS Content Packaging specification ensures a common input-output format which can be supported by any system. Each packet consists of two main components:

– an XML document which describes the structure content and the related resources in the package, which are called manifest files (imsmanifest.xml). The

manifest can be described as a part of a course, internal course, set of courses or just some content which has to transferred from one system to another. The package always consists of a main manifest file which, in turn, can contain one or more manifest files. The main one describes the package, all the rest describe the content at the level at which they are positioned.

– content which describes the physical files of the package. The package has to be stand-alone, i.e. when it is non-archived, it must contain all the information needed for education. The package represents a learning unit. This unit can be a course or a part of a course which is delivered independently. When a package is delivered at a specific place, it has to allow for being divided into smaller items and re-joining these again. The organization of the structure may be viewed as a structure map of the learning objects, called "activity tree". It describes the activities and guides the learner via an activity hierarchy using learning objects.

Thus an object can be comprised of a number of components. If an object is designed to communicate with the LMS, it is an SCO. Otherwise it is simply an asset. The collection of object components creates resources, which can be addressed by the structure. This collection of objects and the structure define the content organization.

uch a packet for e-learning, based on the SCORM standard, is developed for each course. The content has to be divided into separate topics. In turn, the topics must be split into sub-topics, which in their turn to be split into even smaller items. According to the SCORM standard, these assets are associated with learning objects and can be reused and involved in the assessment of the learning outcomes through testing. Based on these units, an activity tree is designed. Part of this description in the case of the course Data Bases is shown in fig. 1.

The tests in these packages are structured in such a way so each question to be associated with a SCO of the respective topic. The idea is to evaluate the degree to which the learning material is acquired, and to navigate accordingly so in case of a wrong answer to lead the learner to the respective content. The content of all topics of the courses is structured and the schemes used correspond to the SCORM 2004 standard, which allows supporting navigation and sequence of activities. The tests are based on the standard Common Cartridge [3].

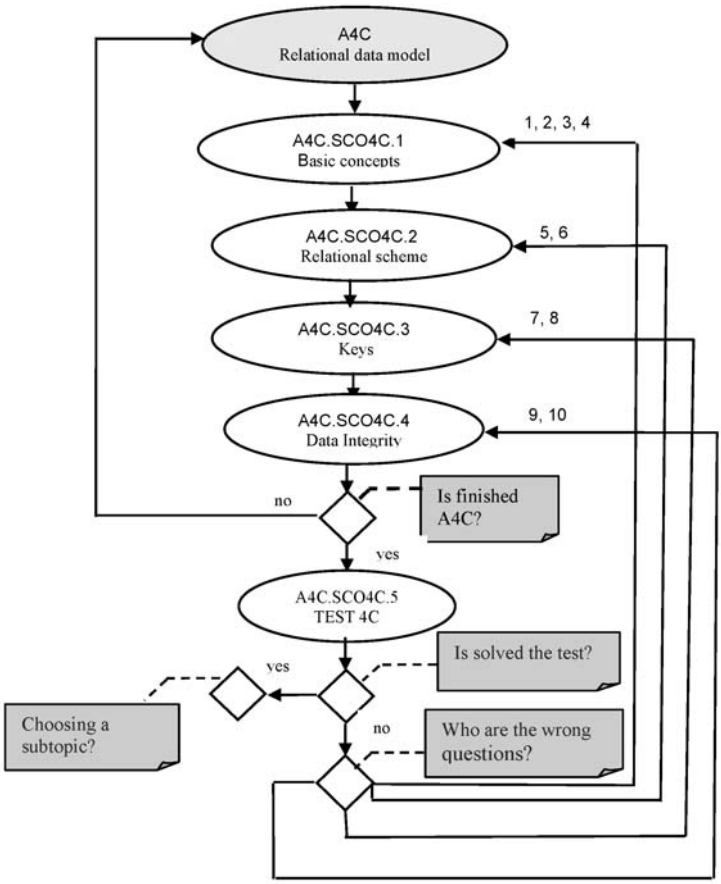


Fig. 1. A sample scheme of the SCORM packet of the course “Data Bases”.

3 Personal assistant

The “personal assistant” is an element of the context-dependent educational environment which delivers e-services and e-content. It is realized as an autonomous software agent based on Java virtual machine, in which the open develelopment environment Java Agent DEvelopment Framework (JADE) is integrated to develop intelligent agents [4]. The library BDI4JADE [5] is used, which implements Belief-Desire-Intention (BDI) architecture in JADE environment. BDI is a classic architecture for intelligent agents. It uses a human activity model for representing limited rationality based on: belief (environment model), desires (agent tasks which have not been transformed into particular intentions yet) and agent intentions which equal its existing engagements, including toward itself [2].

The process of functioning of the "personal assistant" as a BDI agent includes the following components [6]:

- *Belief* is the information available to the agent regarding the current state of its environment. This information comes from the administrative database of the respective higher educational institution, the exam session schedule, exam results, and other elements which belong to cause-effect relations.
- *Belief assessing function* accepts the information from the environment sensors of the agent as input; in this case this is the information about the formative assessment of the learner, forthcoming exams, and changes in the regulatory database, and forms the new set of beliefs on that basis, as shown on fig 2.

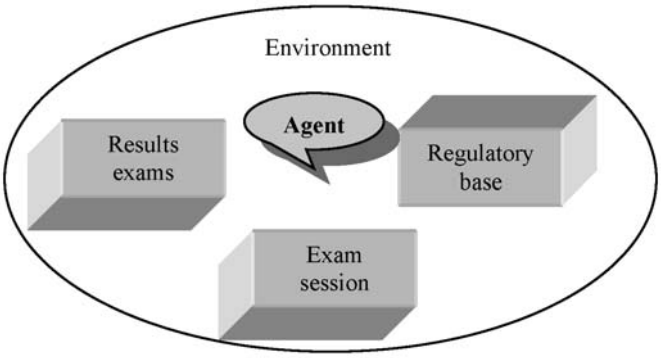


Fig. 2 "Personal assistant" environment

- *Options generating function:* based on the received information, the agent generates different possible actions, e.g. in case of forthcoming exam, informs the student via message, in case of fail mark, checks the next exam session schedule, etc.
- *Current desires* are the possible actions of the agent. Agent intentions are related in advance to the programmed behavioral models which correspond to the cause-effect relations in the real life. They are not yet concrete intentions.
- *Filtering function* is the one which represents the agent process of "consideration" and which identifies its intentions based on its current beliefs and desires. These are predefined programme rules which control the agent intentions. For example, in case of fail mark, the student to be informed about dates and hours for consultations, the exam session schedule to be checked for retaking the exam, or not to take any actions. What is planned in addition is implementing a means for self-paced learning. Thus, when particular actions are systematically ignored, they are not be activated.

- *Current intentions* are the state which the agent has promised to achieve. After filtering, the wishes are transformed into intentions, which are equivalent to the agent tasks.
- The *function for choice of action* defines what action the agent should take based on its current intentions. After revising the current intention, an action plan is chosen.

4 Integrating Data Mining tools in virtual learning environments

In the last years, we witness a wide use of Data Mining techniques for analysis and prediction in diverse real-life situations. Our efforts are dedicated to integrating Data mining tools and e-learning systems, towards personalization and adaptation of the latter to specific needs.

The learning environment is viewed in layers and respective functions, as shown in fig. 3.

Based on the data collected by working with different users in a virtual learning environment, applying Data Mining tools can help making decisions such as:

- developing optimized learning environments with options for personalized acquisition of key knowledge, skills and competencies;
- looking for tendencies regarding developing and supporting the e-learning processes;

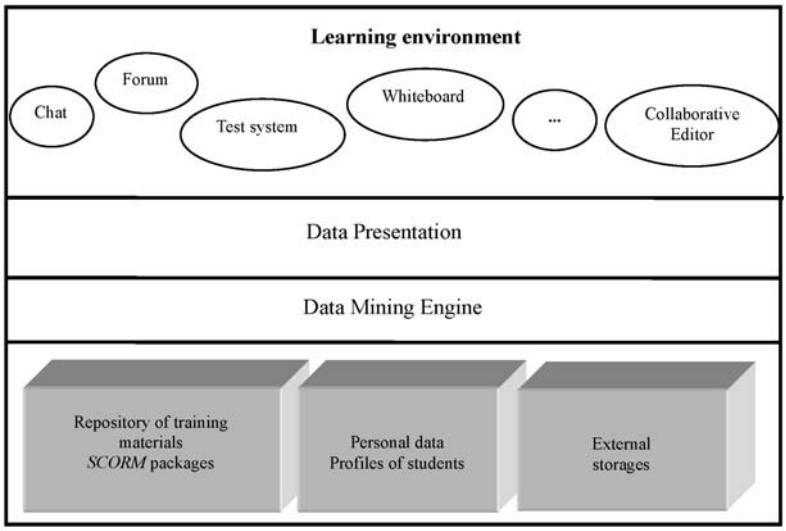


Fig. 3. New learning environment.

- optimizing the techniques for choosing test elements and relevant

influence on learner's activity;

- identifying types of learners and offering appropriate continuation of education;
- taking into consideration learners who may not cope with the education;
- analyzing the degree of knowledge acquisition and loss for different periods of time and different types of tasks, as well as comparison of the corresponding indicators in the years.

Integrating the e-learning systems and Data mining tools is needed for the process of personalization of distance learning courses. Based on the results, additional measures for course analysis and adaptation can be introduced. This, in turn, is a way towards improving the quality of education.

On the other hand, through analyzing the user profile data, means to evaluating a number of personal qualities which influence learning can be proposed. Such qualities are: curiosity – the desire to improve and change things which are generally accepted as norm; connection – the ability to find relations between data which have nothing in common at first sight; persistence – is the ability to continue looking for better solutions even if satisfactory ones have been found; complexity – the ability to work with a lot of information; etc.

Adapting and personalizing learning lay the foundation for solutions using learning environments. Learners' interest, their active role in the process of knowledge acquisition and skill development can be influenced to a great extent by the quality of the learning environment used.

References

1. Stoyanov, S.: Context-Aware and Adaptable eLearning Systems, Internal Report, Software Technology Research Laboratory, De Montfort University, Leicester, UK (2012)
2. Stoyanov, S., Valkanov, V., Popchev, I., Stoyanova-Doycheva, A., Doychev, E.: A Model of Context-Aware Agent Architecture, Comptes rendus de l'Académie bulgare des Sciences, Vol. 67, No 4, pp. 487-496 (2014)
3. Common Cartridge, http://www.imsglobal.org/commoncartridge.html
4. JADE, http://jade.tilab.com/
5. BDI4JADE, http://www.inf.ufrgs.br/prosoft/bdi4jade/
6. Rao, A. S., Georgeff, M. P.: 1995, BDI-agents: from theory to practice, in 'Proceedings of the First Intl. Conference on Multiagent Systems', San Francisco (1995)

Traffic Prioritization System Based on Embedded Components

Ioannis Patias, Vasil Georgiev

Faculty of Mathematics and Informatics
University of Sofia St.Kliment Ohridski"
Corresponding author: ioannis.patias@gmail.com

Abstract
Millions of people benefiting from Mass Urban Public Transport, traveling by bus every day. Widely speaking in our day's time is always important to be able to get the bus in exact timing. Especially for the people in cities there is another challenge, resulting form the frequent use of long routes travelling in city area, crossing the city one side through the other. In this case even if the frequency of the buses is good, the overall time still remains long. The proposed solution is a Bus Prioritization System Based on Arduino Microcontroller. The solution consists of a detection sub-system, with a bus component, using transmitter, and a receiver placed on the traffic light, and a traffic light sub-system. We minimize the time waist on waiting on traffic lights with this prioritization system, with minimum cost, and give an affordable solution. Such a system is a useful instrument for any public transport system.

Keywords: Information systems, embedded systems, transportation control

I. Introduction

The term Intelligent Transport Systems (ITS) refers to a wide range of applications. The most basic ones include simple traffic signal control and management systems, automatic number plate recognition with speed cameras, security CCTV systems. The more advanced applications can integrate real-time traffic and vehicle data and can regulate the traffic in real-time with using such historical data.

Although ITS may refer to all modes of transport, EU Directive 2010/40/EU (7 July 2010) defines ITS as systems in which information and communication technologies are applied in the field of road transport, including infrastructure, vehicles and users, and in traffic management and mobility management, as well as for interfaces with other modes of transport[1]. ITSs are important in increasing safety and also manage Europe's growing emission and congestion problems. They make transport safer, more efficient and more sustainable.

On the other side in other countries, like the United States, the increased interest in the area of ITSs is rather motivated by an increasing focus on homeland

security. Many of the proposed ITS systems also involve surveillance of the roadways, which is a priority of homeland security[23].

When talking about ITS, there is e a wide range of technologies applied[4]. Those technologies include:

- data processing, management and archiving technologies
- detection technologies
- communication technologies
- information dissemination technologies
- location referencing and positioning technologies
- traffic control and vehicle control technologies
- electronic payment technologies
- surveillance and enforcement technologies

Bus Prioritization System (BPS) or Transit Signal Priority (TSP) is an ITS aiming to reduce the time waist on traffic lights for Mass Urban Public Transport vehicles. Although are most often related with buses, they also are applied in trams, rails, etc. In terms of technologies BRS involve traffic control, and detection technologies.

There are two categories of BPS. The so-called active BPS is a system based on detecting Mass Urban Public Transport vehicles as they approach the traffic light and adjusting the traffic light's timing dynamically, and thus create "green wave", meaning uninterrupted traffic along the bus line route. It is important to mention here that implemented this way the system can also be used for the emergency vehicles, so from now on when we are talking about buses we always mean also emergency vehicles. Passive BPS called those systems, which are built with specialized hardware and try to optimize the traffic lights timing by using historical data, and this effect applies to all vehicles along a route.

The proposed here BPS is based on Arduino microcontroller, and is a typical active BPS. It consists of a detection system with a transmitter placed on-board on the bus and receivers placed on the traffic lights. Once the bus transmitter signal received the traffic light sub-system performs the appropriate traffic light timing. The adjustments realized by the Arduino based microcontroller are:

- Extended Green Interval

That means we have extension of the green light interval, when we have a signal from an approaching bus. Once the detection system with the transmitter placed on-board on the bus and the receiver placed on the traffic lights generate a signal the green light is extended with time allowing the bus to reduction its delay for the traffic light.

- Earlier Green Light

We shorten the red light duration, whenever a bus arrives at a red light. Those changes are not applied immediately, to avoid conflicting situations.

The system has two different operation levels, for the respective use. One is for Mass Urban Public Transport management, providing the interface to re-program the traffic lights timing, and another is the normal use.

II. Problem Definition

To purpose is to develop a Bus Prioritization System (BPS) Based on Arduino Microcontroller. The main objectives of this system are to create conditions for greater mobility for the citizens and visitors of the city using an isolated vehicle actuated system. By reducing the time of the buses waiting in traffic lights to increase the capacity of urban transport systems. And finally, to increase the number of passengers on public transport and increase the market share of public transport, with an affordable cost.

III. Literature Review

The idea of a post-desktop model of human-computer interaction, where we have integration of information processing into everyday objects and activities is called Ubiquitous Computing5. In many cases the end-user uses more than one distributed systems and devices even simultaneously, without even being aware of their existence. The implementation of this concept is not that easy. But the overall dividend is great. Our life would be quite easier if all objects in the world surrounding us get equipped with identifying devices.

The most widely used identifying devices are the ones using Radio-Frequency Identification (RFID). RFID tags, or electronic labels are used with objects to be monitored or tracked. The technology can be applied to any object, animal, or people. We can identify and track the objects by using radio waves or sensing signals. There are tags, which can be tracked with range of tens or hundreds of meters. The syntax of RFID tags contains two major parts at least. The first is storing and processing information integrated circuit, which is also modulating and demodulating a radio-frequency (RF) signal. The second part consists of an antenna, used for receiving and transmitting the radio signals.

There are active, semi-active, and passive RFID tags. Tags can store up data and consist of microchip, and antenna, and also battery for the cases of active and semi-passive tags. All the components can be enclosed in plastic, or silicon. In general RFID tags help us in our everyday activities, since they are not expensive, and at the same time they can apply in almost any object.

The use of RFID enriches the options of systems used for giving priority to buses. The concrete needs determine the most appropriate measures to be used. A feasibility study should be implemented prior to the concrete measures to be used can be defined.

There are various ways of giving priority to buses, which could be broadly categorized as6:

- physical measures,
- traffic signal priorities, and
- integrated measures.

Physical measures can include with and contra-flow lanes, bus only lanes or even streets. Traffic signal priorities method's typical example is the BPS. Integrated measures are those, which combine traffic signal measures with physical measures. The latest is applicable in cases where none of the first two systems alone is effective.

Focusing on the traffic signal priorities method, there are different systems implementations. Those differentiations usually called traffic signal control systems and strategies[7], and are categorized in:

- Isolated systems

In isolated systems the controlled by signal traffic light is located and operates independently, this is why the term isolated traffic light is used. Traffic light's signals can also be linked to a Control Centre, but only for fault monitoring purposes, not for management. Isolated system can further be divided into fixed time or vehicle actuated (VA).

- Co-ordinated systems

Co-ordinated systems, are so called because they co-ordinate the operations at a traffic light, with the operations at one or more neighboring traffic lights. All traffic lights have to be connected to a centralized system implementing a Control Center system. Co-ordinated under Control Center systems can be further divided into traffic responsive or fixed time.

VA systems rely on detectors placed on traffic lights. When a bus approaches to the traffic light, and once it is detected the traffic light performs the appropriate timing. A bus approaching a traffic light with red light sends to the controller a demand for a green light. The demand is then served by the controller, which can apply different timing cycles. After serving any signals and with no more incoming ones, the controller will continue the preprogrammed mode/s.

The VA system can give priority both to buses, and any other special purpose and/or emergency vehicle. Also the VA systems can serve different priority levels requests. This means that special purpose vehicles can transmit a higher priority level "priority request", and thus be served with privilege.

IV. The BPS System

BPS architecture

Using the terminology, introduced in the previous section, our proposal, the Arduino microcontroller BPS, is a typical isolated VA system.

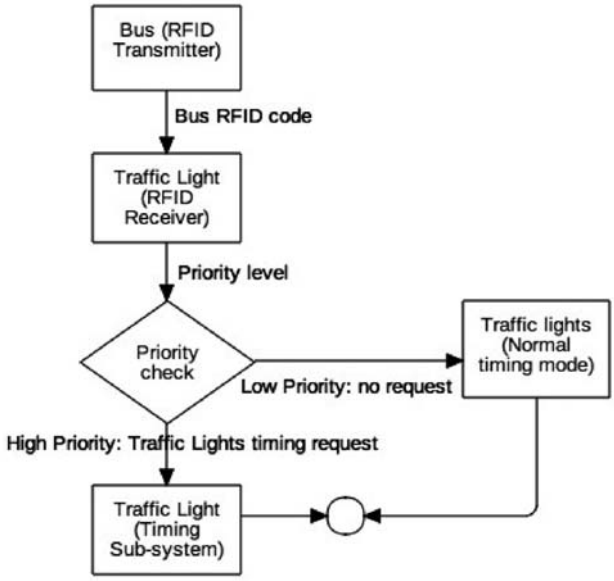


Figure 1: BPS Flow Chart Diagram

As we can see in the figure above, once a bus approaches to the traffic light, it sends a "priority request". In order to increase system's flexibility we can define more than one priority levels. For instance, one is for the buses, and the other for special purpose vehicles.

Our system implements all the above-mentioned, divided in the following modules:

1. detection sub-system, with:
   a. bus component, using transmitter, and
   b. receiver placed on the traffic light, and
2. traffic lights timing sub-system.

BPS operation

When a bus approaches a junction, the traffic lights receiver captures an RFID signal sent by the bus transmitter. The receivers should be placed so to focus on signal captured at about 30m distance. This way we can simulate Expected Time of Arrival (ETA) functionality. Meaning we cover variations of effective Extended Green Interval or Earlier Green Light traffic light timing adjustments.

The priority requests are converted as follows:

Case 1 (Extended Green Interval):

Every traffic lights timing sub-system when receives a priority request, and currently is on green light mode, extends the green light interval (additionally to

the normal green interval).

Case 2 (Earlier Green Light):

When the traffic lights timing sub-system receives a priority request, and currently is on red light mode, the red light duration is shortened. This shortened duration cannot be applied immediately, in order to avoid conflicting situations.

There is one more intermediate case, when a priority request arrives on a yellow sign. We have to check what the next signal is and respectively to apply Case1, when the next light is to be green, or Case2 when we expect red light. The described sequence of signals and operations in shown in the next figure.

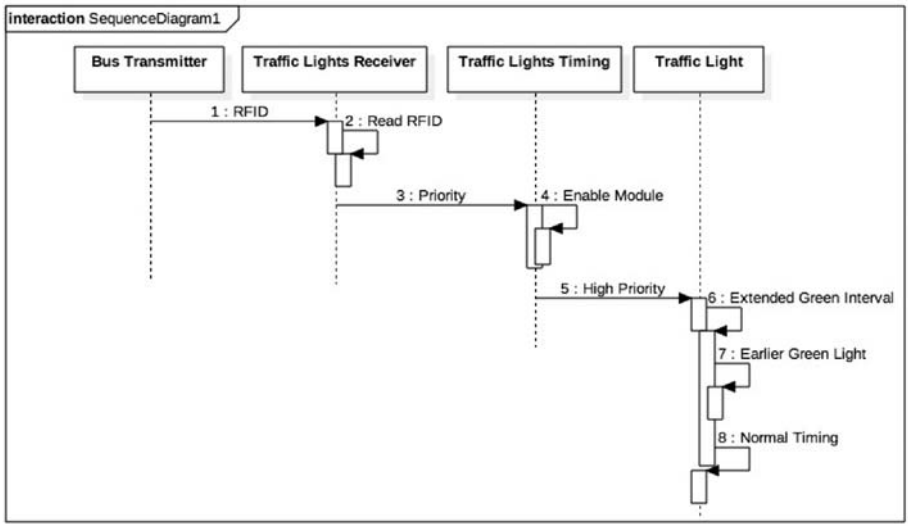
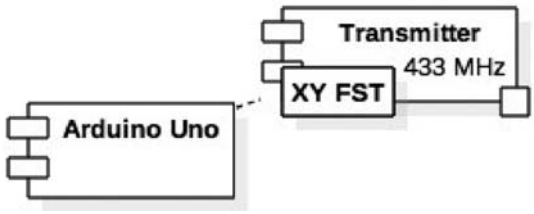


Figure 2: BPS System-level Sequence Diagram

V. Hardware Component Description

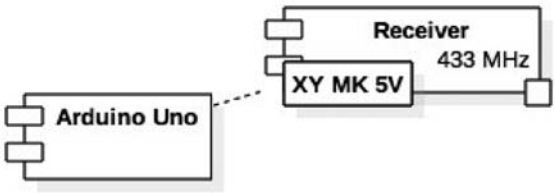
Bus component (transmitter):

(designed to have range of around 60 meters, when supported by antenna and has direct view)



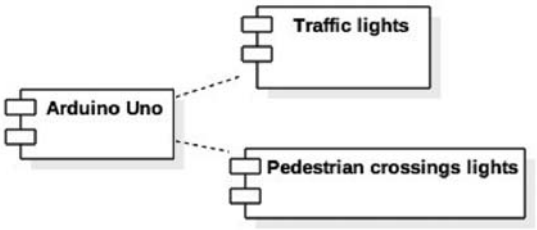
Traffic Light Receiver:

(to be placed so that can provide direct view to the bus component)



Traffic light timing sub-system:

(simplified demo version with light emitting diodes (LEDs), showing both traffic lights and pedestrian crossings lights)



Simplified algorithm

```
LOOP () {
Read the RFID reader
If
The PRIORITY goes high
Then decode the ID
    If
    The ID is for HIGH_PRIORITY_VEHICLE
    Then Enable the HIGH_PRIORITY module
            Else if
            The ID is for LOW_PRIORITY_VEHICLE
            Then Enable the LOW_PRIORITY module
Else
Enable the NORMAL_TIMING module
}
```

VI. Conclusions

The BPS presented can really help passengers. Although implemented with an affordable for any municipality budget, still is a reliable solution that can answer to the main objectives of the citizens and visitors of any city. It can reduce the time of the buses waiting in traffic lights, and so to increase the capacity of urban transport systems. We can implement the system at any traffic light, since it requires no infrastructural changes or upgrades. The system so described can

increase the number of passengers on public transport and thus the market share of public transport, with an affordable cost.

References

1 DIRECTIVE 2010/40/EU OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2010:207:0001:0013:EN:PDF

2 Torin Monahan, "WAR ROOMS" OF THE STREET: SURVEILLANCE PRACTICES IN TRANSPORTATION CONTROL CENTERS, http://publicsurveillance.com/papers/war_rooms.pdf, The Communication Review, 10: 367–389, 2007

3 http://www.its.dot.gov/landing/strategicplan2015.htm

4 ROAD NETWORK OPERATIONS & INTELLIGENT TRANSPORT SYSTEMS http://rno-its.piarc.org/en

5 Kai Hwang, Geoffrey C. Fox, and Jack J. Dongarra, "Distributed and Cloud Computing From Parallel Processing to the Internet of Things", 2012 Elsevier.

6 https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/329973/ltn-1-97__Keeping-buses-moving.pdf

7 http://content.tfl.gov.uk/interaction-of-buses-and-signals-at-road-crossings.pdf

Transformation and modernization of PRINTS database

Anatoliy Dimitrov[1*], Teresa Attwood[2], Ognyan Kulev[1], Dimitar Vassilev[2,3]

1 - Faculty of Mathematics and Informatics, Sofia University "St. Kliment Ohridski", 5 James Bourchier Str, Sofia 1164, Bulgaria
2 - Faculty of Life Sciences, The University of Manchester, Carys Bannister Building, Dover Street, Manchester M13 9PL, UK
3 - Bioinformatics group, AgroBioInstitute, 8 Dragan Tsankov Blvd, Sofia 1164, Bulgaria

* - Corresponding author: tollodim@gmail.com

Abstract: The PRINTS database is an important resource in bioinformatics and one of the most popular protein databases in the world. At the beginning of the project, PRINTS data was contained in a large text file with more than one million rows. The aim of the current project was to modernize PRINTS by parsing the information from the text file and storing it in a state-of-the-art relational database. In addition to that, tools would have been provided for work with the new database such as web interface and Linux command line scripts. The project accomplished successfully its goal as the current paper show in further details.

Keywords: PRINTS, Bioinformatics, Text Parsing, Relational Databases

1 Introduction

Protein Sequences, Motifs and Families

The large number of protein sequences has created valuable information which needs to be rationalised in order to expedite protein sequence and structure analysis. The protein motif takes an important part in this rationalisation. It is formed by the three-dimensional arrangement of amino acids, which may not be adjacent.

Homologous protein sequences have similar 3D structure, and carry out related molecular functions—this is the most fundamental premise of protein sequence analysis. This understanding has facilitated the grouping of proteins descending from clear common ancestry into homologous sequence groups known as protein families [1].

Introduction to PRINTS

PRINTS is a collection of patterns of protein motifs called 'fingerprints'. A fingerprint is defined as a group of motifs excised from conserved regions of a sequence alignment, whose diagnostic power or potency is refined by iterative

database scanning [2]. Furthermore, fingerprinting offers a powerful approach to the analysis of protein sequences: it inherently offers improved diagnostic reliability over single-motif methods by virtue of the mutual context provided-by motif neighbours [2].

Purpose of the project

Much of the knowledge in biological databases is stored in plain text (easily consumed by humans, but essentially opaque to computers), and many records are incomplete or contain 'grey', putative information [3]. Because of that, PRINTS was a typical resource storing information in plain text before its modernisation.

This paper describes the work on modernising PRINTS. More specifically, the following tasks were executed:

- Migration from a text file to a relational database as main data source.
- Development of tools for working with the data, including Web-based and command-line tools.

The purpose of this project was to align PRINTS with the current technologies for storing and representing information. This involved a process of transformation and modernisation, at the basis of which lies the creation of a relational database. Along with that, tools for accessing and managing the information in a fast and efficient way were provided.

PRINTS, just as any other existing IT resource, needs to follow current technology trends in order to grow and remain useful to the users. However, it takes some time to catch up technologically and, meanwhile, the modernisation gap increases. Thus, in the case of PRINTS, the main data source for the public database was, and still is, a plain text file using a custom tag system for identifying specific data fields within each database record. There are numerous disadvantages of working with such file. First, it is hard to manage the information, especially due to the large file size (more than one million text lines). Writing and reading information was both difficult and prone to errors. Second, working with a single text file as data source was a bottleneck for performance, because the speed to write in/read from it was much lower compared to a modern database, where caching, indices and other performance features exist.

The modernisation approach described in this article is rather general and universally applicable, even though all specifics of PRINTS have been taken into consideration. Thus, this same approach could be applied to other similar information resources that need to be refreshed and powered by a relational database. Undoubtedly, many such resources still exist, and the current work may allow them to follow a fast and straightforward process towards modernisation.

The modernisation of PRINTS involved the following tasks:

- Understanding PRINTS and its data;
- Design of the new PRINTS database;
- Parsing the old PRINTS data file;
- Creation of a CRUD (Create Retrieve Update Delete) application for the new PRINTS database;
- Provision of additional tools for working with the data.

2 Materials and methods

Understanding the specifics of the PRINTS project and the meaning of its data

Before we could begin any work on the data transformation, it was important to acquire a good understanding of the PRINTS data. Based on this understanding, we could begin to normalise the data, in order to create the new database tables.

While parsing PRINTS we extracted information for more than 2,100 fingerprints. A fingerprint is a group of conserved motifs used to characterise a protein family; its diagnostic power is refined by iterative scanning of a SWISS-PROT/TrEMBL composite [4]. Usually the motifs do not overlap, but are separated along a sequence, though they may be contiguous in 3D-space. Fingerprints encode protein folds and functionalities more flexibly and powerfully than single motifs, full diagnostic potency deriving from the mutual context provided by motif neighbours.

The PRINTS data were stored in more than one million text lines, which had to be automatically processed and parsed. Each fingerprint and its associated information followed one after another: the first line of each record containing the General Code for the entry, preceded by the specific tag, 'gc;'. Each subsequent line included information related to the same fingerprint until a new 'gc;' was reached.

Database format

Here's an example of the fingerprint GLABLOOD from the old text file:

```
gc; GLABLOOD
gx; PR00001
gn; COMPOUND(3)
ga; 12-JUL-1991; UPDATE 27-JUN-1999
gt; Coagulation factor GLA domain signature
gp; PRINTS; PR00002 GLABONE
gp; INTERPRO; IPR002383
gp; PROSITE; PS00011 GLU_CARBOXYLATION
gp; PFAM; PF00594 gla
gp; PDB; 1APO; 2PF2
```

gp; SCOP; 1APO; 2PF2
gp; CATH; 1APO; 2PF2
bb;
gr; 1. SORIANO-GARCIA, M., CHANG, H.P., TULINSKY, A., RAVICHANDRAN, K.G.
gr; AND SKRZYPCZAK-JANKUN, E.
gr; Structure of calcium prothrombin fragment 1 including the conformation
gr; of the Gla domain.
gr; BIOCHEMISTRY 28 6805-6810 (1989).
gr;
gr; 2. CHURCH, W.R., MESSIER, T., HOWARD, P.R., AMIRAL, J., MEYER, D.
gr; AND MANN, K.G.
gr; A conserved epitope on several human vitamin K-dependent proteins.
gr; J.BIOL.CHEM. 263 6259-6267 (1988).

Each line from the text starts with a custom, unique identifier which holds specific information about the fingerprint. In the above example the identifiers have the following meaning:

- gc – name of the fingerprint;
- gx – accession number;
- gn – number of motifs. The COMPOUND keyword is always present and the number of motifs is in the parenthesis;
- ga – creation and update date;
- gt – title;
- gp – cross references within PRINTS and to other protein databases;
- gr – literature references.

These are some identifiers in the beginning of each fingerprint text. To avoid shifting the focus of the article, we will mention the remaining ones only when needed. Once we got acquainted with all identifiers, the text format was relatively clear to understand. Unfortunately, there were no available libraries for working with this format for any popular programming language which meant that a custom parser had to be created for the current project.

The initial time spent on studying and understanding the PRINTS data was well invested, which was proven by the minimal number of redesigns of the database.

Problems and challenges

After studying carefully the old PRINTS data structure, it was decided that the most efficient way to parse the data was on a line-by-line basis. The principal reasons for this decision were two-fold:

- In its original format, PRINTS data were divided on the same principle, *i.e*. line by line, with unique indents/tags at the beginning of each line.
- Lines could be split into parts and matched in accordance with the known indents. In contrast, if multiple lines were processed simultaneously, regular expressions were necessary, which would slow down the process and also make it more error-prone.

However, the line-by-line parsing method introduced a few problems in the context of processing PRINTS:

1. In one case, the exact meaning of the current indent depends on the previous one. This was the case with 'special lines', which were introduced after the original PRINTS data-file format was specified, specifically in order to be able to record accession numbers (rather than simply identifiers) of proteins matched by each fingerprint. In this case, the tag 'KA' simply denotes the matched protein accession number, regardless of whether it is a true-positive (tp) match or a partial true-positive (st) match.
2. One line could contain more than one logical grouping of information, *i.e*. table row in the future relational database. For example, the line for database cross-references 'gp; PRINTS; PR00209 GLIADIN; PR00208 GLIADGLUTEN; PR00210 GLUTENIN' contains 3 separate cross-references:
   a. PR00209 GLIADIN;
   b. PR00208 GLIADGLUTEN;
   c. PR00210 GLUTENIN.
3. Each line could contain additional denotations for logically separating information. For example, in the line 'gp; PRINTS; PR00002 GLABONE', *PRINTS* means a cross-reference from the PRINTS database. *INTERPRO* would denote the InterPro database; and so on.
4. One line could contain different groupings of information, *i.e.* different table rows in the future relational database. For example, line 'ga; 16-NOV-1995; UPDATE 05-MAR-2010' denotes first the creation date, and then the update date.
5. One atomic piece of information could be spread over multiple lines. This was probably the biggest problem with the current line-by-line parsing method. As per the PRINTS excerpt above, the 'gr;' tag defines literature references. In the first literature entry, the title is spread over two lines on the third and fourth sequential lines. In the second literature reference, however, the title was found only on the third line.

These problems were the biggest challenges for the parsing and for the whole project, because the success of the project depended on their correct resolution. The remaining tasks, *i.e.* the design of the database and the creation of the tools, were relatively trivial and did not impose significant challenges, and hence are only briefly mentioned in the current paper.

3 Results and discussion

Parsing the data and loading the information into the database

In order to parse the data from the text file, we decided to write a custom program in Python. The reason to choose Python as the main programming language for the parser was its strong support for text processing and wide acceptance in the scientific community.

Preparation of the text for parsing

To tackle the first parsing problem, we also used the Perl programming language, partly for educational purposes and partly for expediting the task. The problem, as described above, concerned the protein sequence accession-number field, denoted with the KA tag, which could refer either to true positives (tp) or true partial-positives (st), depending on the preceding text line. Here is an example:

tp; O04691_METGY Q40933_PSEMZ Q38697_ASAEU B9H8M2_ POPTR
KA; O04691 D1 Q40933 M1 Q38697 M1 B9H8M2 M1
tp; Q39775_GNEGN
KA; Q39775 M1
bb;
sn; Codes involving 5 elements
st; LEGB_GOSHI 11S3_HELAN Q647H1_ARAHY Q9SQH7_ARAHY
KA; P09800 M1 P19084 M1 Q647H1 M1 Q9SQH7 M1

To eliminate this first major problem, we decided to replace the KA tag with a different tag (K1), where the accession number referred to true-partial positives. Perl is perfectly suited for the task because of its simplicity and powerful regular expression support. The Perl code opens the prints42_0.kdat file and performs a replacement based on the regular expression:

"s#(„^st.*\n)KA\;#$1K1;#g**;",** which means**:**

– Search and replace globally all matches.

– Match a new line that starts with "st" and is followed by a new line that starts with "KA".

– Reference the line that starts with "st" and preserve it as is in the replacement.

– Replace KA with K1 on the second line.

This fixed the first problem and made the text better suited for line-by-line processing, with unique tags for all related parts of information.

For convenience, we also made another simple replacement. In front of every

fingerprint (*i.e.*, for each gc; tag), with the help of a simple text editor, we inserted a custom string:

"---custom_delimiter_for_fingerprint---"

This was necessary to preserve the structure of the fingerprint intact when splitting the text into separate fingerprints.

Python, our main programming language of choice, required a string for its split function. That specific string had to be later removed from the split text. Thus, if we took some existing text as the delimiter, such as gc, this would be removed in the split text and hence leave the first line of each fingerprint without a tag, which would interfere with the logic for line-by-line processing.

With the preliminary tasks done, we could begin the essential parsing of PRINTS.

Parsing PRINTS with Python

The Python parser begins by reading the whole text file and then splitting it into parts. Each part represents one fingerprint. As mentioned, Python's built-in split method will be mainly used in the current project. In its first occurrence, the file's content is split into parts by the custom delimiter created previously, "---custom_delimiter_for_fingerprint---".

Another useful method for separating text in Python is partition. This divides text into parts using a given separation, and returns a 3-tuple containing the part before the separator, the separator itself, and the part after the separator.

In PRINTS, we used partition to create a tuple for each line with the tag and its content. The separator was ";", which could be removed as unnecessary. To accomplish the desired text partitioning, our code iterated through each PRINTS data line (represented by l) like this:

```
tag, sep, contents = l.partition(';')
```

After that, the values for tag and contents were appended to a new multi-dimensional list, called lines. This specific list would be used for the essential data matching, and iteration would be performed through it (for l in lines…).

The most simple value assignment was for tags such as that for fingerprint identifiers, denoted by gc. If the first part of the line was the "gc" tag, the second part of the line contained only the value of the fingerprint identifier, which could be assigned directly.

However, most tags required more complex processing. If the first part of the line was "gn", then we performed a regular expression search and looked for whole numbers after the string COMPOUND. When matched, only the whole numbers would be captured and assigned to the variable "no_motifs" used for number of motifs.

For other information, such as literature references, denoted by the "gr" tag, we used a different approach. The literature reference field could contain more than one reference, and any part of this reference (author, title or source) could be spread over more than one line. Hence, a new list, called reference, was created for each "gr" line and processed later, after the iteration of the fingerprint lines was over.

In the original PRINTS file, a new (blank) line occurs between each literature reference. Thus, we separated each reference with the re.split method (similar to the simple split, but using regular expressions for the delimiter). In order to do that, all the reference lines were first merged, using the join method. After that a few specifics found in every literature reference were taken into consideration:

- The first line always contained the author(s) and could be immediately assigned, hence removed from the temporary list and from further processing.
- The last line was always the journal/book name, and could be treated similarly.
- The middle line(s) was always the title. It could be immediately assigned only if there were three lines in total in the temporary list. Any other cases required additional processing.

If there were more than three lines, we used an important peculiarity of the literature reference: authors are always written with capital letters. Thus, if the line contained any lower-case letters, it was considered part of the title. Otherwise, that line was part of the author list.

Continuing further with the parser code and going back to processing the tags, in some cases, we used Python dictionaries for storing the information: for example, the true partial-positive values, denoted by "st". Once the number of elements had been captured, it was attached to the true partial-positive values. For this purpose, a dictionary tpp_number_of_elements was used. Keys in this dictionary were the true partial-positive values, with the values being taken from the last known number of elements.

The full code of the parser is available in the GitHub repository:
https://github.com/terry81/prints/blob/master/final.py

Creating a relational database

There was never a doubt that the new version of PRINTS should be stored in a database. Alternatives, such as using mark-up languages (*e.g.*, XML) and storing the information in plain text file, were ruled out as slower, less efficient, less consistent and more prone to anomalies.

While the easiest option would have been to use a NoSQL database (each fingerprint could be represented as a document), a relational management system was chosen for the following reasons:

- There were clear relations between the parts of each fingerprint. These relations were important for the sanity checks and integrity of the information.
- Each fingerprint had the same structure and could populate the same tables.
- The current and expected volume of PRINTS information was within the optimal size for a traditional relational database.

Regarding the choice of Relational Database Management System (RDBMS), PostgreSQL was preferred over MySQL die to:

- Fast performance and powerful features were needed.
- Open Source license to be in accordance with the best practices for scientific projects.
- The price should be low, and preferably free.
- The chosen RDBMS should be familiar to the scientific community.

Design of the PRINTS database

When designing the PRINTS database, we took into consideration the basic fact that all data were about and related to protein fingerprints. Thus, the basic database table was termed 'fingerprint'. All other tables were related to fingerprint with primary – foreign key relations.

Figure 1 illustrates all PRINTS tables, with their columns and relations. To create the diagram, we used the software DBVisualizer [5].

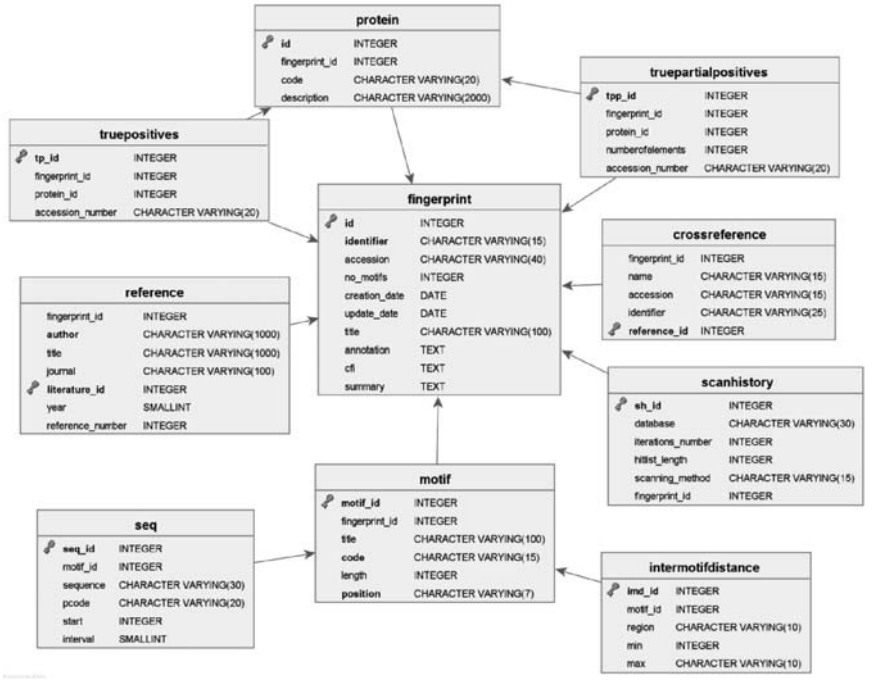


Figure 1. Tables in the new relational database

Standard practices were followed in the database design, namely:

1. Unique, automatically incremented primary key for each table.
2. Tables were connected with primary – foreign key relations.
3. Strict data types were enforced wherever possible.

Thus, a large number of possible mistakes, especially when inserting parsed information, were avoided and the database schema was optimal with regard to performance.

Inserting the parsed data in the new database

To insert the parsed data into the new database, we used the same Python script mentioned previously (https://github.com/terry81/prints/blob/master/final.py).

For interacting with the PostgreSQL database, we chose the popular Psycopg2 Python module (https://pypi.python.org/pypi/psycopg2), which provided excellent methods for inserting and retrieving information.

For inserting information, we Psycopg2's methods: once the fingerprint had been inserted, its identifier (ID) was recorded under fingerprint_id. From this point on, fingerprint_id, the primary key of the fingerprint table was used for inserting values in the rest of the tables. The rest of the PRINTS data were similarly inserted without presenting any fundamental challenges.

Providing a CRUD functionality

The ability to manage the PRINTS data efficiently was top priority of the new re-design, because the principal problem with the old data format was that the text file was hard to manage. For this purpose, we decided to implement a CRUD (Create Retrieve Update Delete) Web application. This application had to be fast, easy to maintain, compliant with the best practices, and free.

These requirements could be best met with a PHP Web application based on the Yii framework [6]. Yii is among the top PHP frameworks, with a long history, a solid support track-record, and open source licence (Berkeley Software Distribution license). Yii has a modern design, which follows the Model, View, Controller (MVC) principles [7], as illustrated in Figure 2. This allows fast initial setup, easy maintenance and future extension. In the context of PRINTS, the MVC concept meant that each database table would have a separate controller, model and view – ten in total.

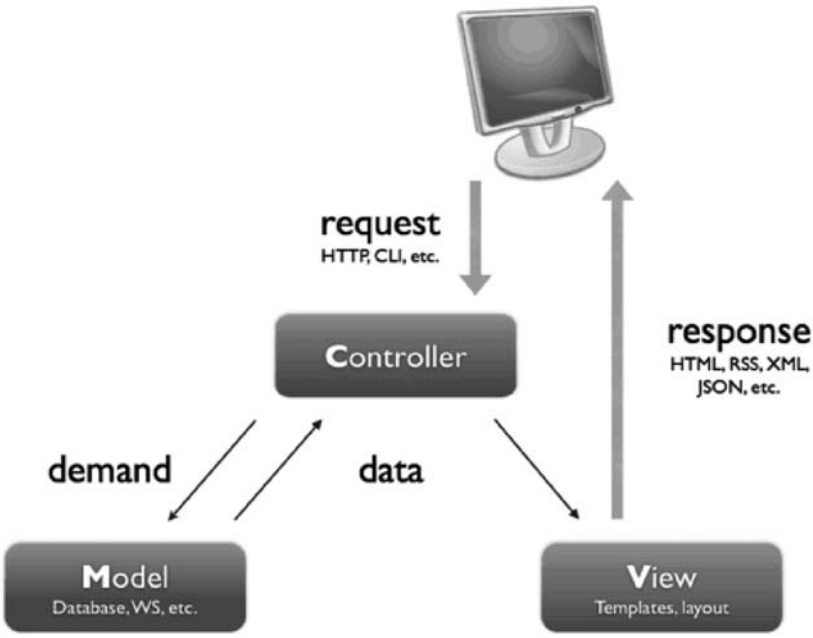


Figure 2. Relations between the components of MVC

Integrating PRINTS with the Yii web framework

To build the Web application for PRINTS with Yii we followed these steps:

1. Generated a new standard Web application with Yii and connected it to the new database.
2. With the help of the Giix extension (http://www.yiiframework.com/extension/giix/), we generated almost automatically all the views, controllers and models for the application.
3. We started customising the code in accordance with the requirements of the new functionality and appearance of the Web application.

Thanks to the use of Yii and its automation features, the development time for trivial tasks was reduced and efforts could be concentrated on the essential part of the development.

The most basic requirement of the new Web interface was to make it compatible with or similar to the existing one for the old PRINTS database. This would allow the use of previous proven functionality, look and feel, making the transition easier for existing PRINTS users. Generally, this meant two things:

- Preserve the order of the fingerprints elements on the.
- Preserve the formatting of the old data, including the spaces between every fingerprint detail.

– Preserving the alignment of text within rows of data tables (*e.g.*, such as those found in the Composite Feature Index) was critical for human readability (*e.g.,* to be able to compare values). For this purpose, we used mono-spaced fonts in addition to some programming.

Making the new interface resemble the previous one was a challenge because of the database redesign, the differences coming from the MVC concept and Yii. The fingerprint view (*i.e.*, Web page) in the new MVC design had to be the focal point for all the PRINTS information. This meant having to integrate most of the functionality of all ten views into a single view – that of the fingerprint. The full code of the fingerprint view can be found in the git repository:

https://github.com/terry81/prints/blob/master/public_html/protected/views/fingerprint/view.php

To preserve the fixed-width text, we used regular expressions in a way that spaces were used for padding and preserving the column alignment. The following example (figure 3) illustrates how particular data tables had to appear within the Web page:

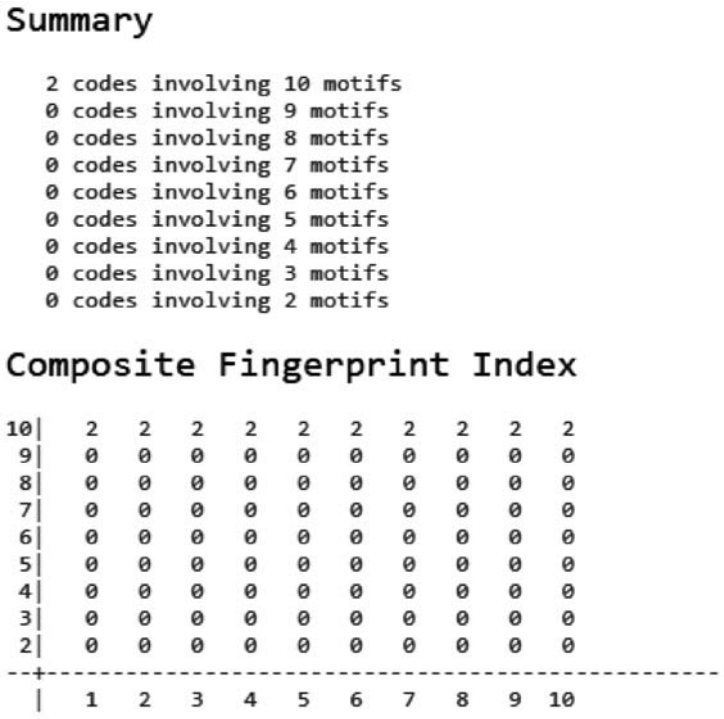


Figure 3. Screenshot from the new web page for PRINTS

To accomplish the above alignment in the Summary and Composite Fingerprint Index, we had to make some replacements, based on regular

expressions. The code inserts a number of blank spaces with respect to the number of original characters (digits in this case). The fewer the digits, the more padding was needed, and the more blank spaces were inserted. Thus, for a single digit in the Summary information, three blank spaces (in HTML) were placed; for two digits – two blank spaces; and for three digits – one blank space. Similarly, for the Composite Fingerprint Index, padding was added in all the columns to ensure that information was properly aligned.

The rest of the PHP code in the GitHub repository shows other techniques used for the visualisation and the data.

Providing additional tools for working with the PRINTS database

Alongside the intuitive and easy-to-use Web application, we decided to provide additional tools for working with PRINTS: in particular, we needed more powerful and faster tools for managing the PRINTS data. That's how we decided to use the Python code for the initial parsing and extend it further.

The most important additional tool was one to parse new fingerprints created in the old text format. To this end, we created the add_fp.py script (https://github.com/terry81/prints/blob/master/add_fp.py), which took one argument passed on the command line – the text file of the new fingerprint. The code was exactly the same as the original parser, only the original parser had the file with all the fingerprints (prints42_0_adapted.kdat) hard-coded. For convenience, we also created another script to delete fingerprints based on their identifiers – delete_fp.py (https://github.com/terry81/prints/blob/master/delete_fp.py). This script also accepted one argument – the fingerprint identifier – and went through the table relations, bottom to top, deleting all dependencies such that a fingerprint would be completely removed.

Of course, there also remained the option to use the PostgreSQL terminal. This provided the most powerful features based on the standard Structured Query Language (SQL). However, this requires SQL skills, and would be used only for the most advanced data-management needs.

4 Conclusion

To complete the project, we planned to thoroughly check and test all the provided tools. Along with such testing, the database integrity would be double checked to ensure that there were no discrepancies with the old database.

It was a huge step to move from a simple text file to a relational database with interactive tools connected to it. However, with the new features, power and complexity came many possibilities for problems. For this reason, it was critical to understand that once the essential phase of the modernisation process

was completed, substantial ongoing support should be provided. Only this would validate the whole point of modernising PRINTS.

To facilitate the future support and development of PRINTS, we avoided using any exotic technologies, but instead relied on proven solutions and programming techniques. Along with this philosophy, we described and documented the full modernisation process was in detail.

References

[1] Linial, M. & Loewenstein, Y., 2008. Connect the dots: exposing hidden protein family connections. ECCB, Volume 24, p. 193–i199.

[2] Attwood T.K., M. A. a. D.-S., 1994. PRINTS a database of protein motif fingerprints. Nucleic Acids Research, 22(17), pp. 3590 -3596.

[3] Stockinger H, T. A. S. N. C. R. C. P. C.-M., 2008. Experience using web services for biological sequence analysis. Briefings in Bioinformatics, 9(6), pp. 493-505.

[4] Coletta, A. & Farhan, R., n.d. PRINTS. [Online] (http://www.bioinf.manchester.ac.uk/dbbrowser/PRINTS/index.php)

[5] DbVis Software AB, 2010. DbVisualizer: The Universal Database Tool, Stockholm, Sweden [Online] (https://www.dbvis.com/)

[6] Winesett, J., 2012. Web Application Development with Yii and PHP. Packt Publishing Ltd. (http://www.yiiframework.com)

[7] Krasner, G.E. and Pope, S.T., 1988. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. Journal of object oriented programming, 1(3), pp.26-49.

Aknowledgments This project has been supported by the National Science Fund of Bulgaria within the “Methods for Data Analysis and Knowledge Discovery in Big Sequencing Dataset” project under contract DFNI02/7 of 12.12.2014.

Software protection integrating registration number and anti-debugging protections

Magdalina Todorova[1], Daniela Orozova[2]

[1] "St. Kliment Ohridski" University of Sofia, Faculty of Mathematics and Informatics,
Sofia 1164, Bulgaria
todorova_magda@hotmail.com

[2] Burgas Free University, Faculty of Computer Science and Engineering,
62 San Stefano Str., Burgas 8001, Bulgaria
orozova@bfu.bg

Abstract. The article is dedicated to a description of a method of software protection integrating registration number and anti-debugging protections. The means of generalized nets are used to achieve that. The registration number is defined by a randomly generated generalized net with a tree-like structure whose nodes are generalized nets also. The anti-debugging protection is realized via embedding software, which traverses the generalized net providing the registration number, into the protected software.

Keywords: Software Protection, Anti-debugging, Registration Number Protection.

1 Introduction

The popular means used by crackers to break code can be divided into three groups as follows: debuggers (SoftICE, TRW2000, Syser, OllyDbg, Rasta Ring 0 Debugger, HyperDBG, LinICE, BugChecker, etc.), disassemblers (WinDasm, Sourcer, IDA Pro, PEDasm, CRACKER, etc.) and decompilers (Mocha Decompiler, Java Decompiler, JAD Decompiler, JEB Decompiler, Android App Decompiler, etc.). Respectively, the most common ways and means of protection of the programming code from breaking are anti-debugging, anti-disassembly and anti-decompiling [1, 2, 3].

The method proposed here can be classified as an anti-debugging one. However, instead of blocking the tools for debugging the fragment which compares the registration number (key, code), it gets more complicated. The complication is realized by using generalized nets (GNs) [4]. The choice of a GN is determined by the fact that these nets are a useful tool for modelling and simulating parallel processes. By means of GSs hierarchical data structures can easily be modelled and processed. They are not popular with the software

specialists, but are well-studied from a mathematical point of view [4]. Software tools for efficient interpretation of generalized net models are developed [5, 6, 7].

2 Description of the protection method

The described method embeds software which realizes parallel traversing of a generalized net, into the code of the protected software. The GN used for this purpose has a tree-like structure. The nodes of the tree are oriented multigraphs presented by GNs (fig. 2). This GN defines the key of a particular instance of the protected software.

The programming code to be embedded is designed by iteration of the developed tree layer by layer from the leaves towards the root. The GNs at a given layer are executed in parallel. The execution of each GN which is positioned in a node is also in parallel.

In order to build the tree in fig. 2, two sets of generalized networks are developed: G_1 and G_2, of the same type as shown in fig. 1.

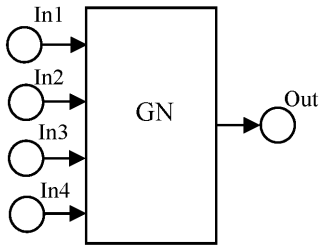


Fig. 1. Simple GN with 4 Input and 1 Output Places

The nets have four input places, denoted as In_1, In_2, In_3 and In_4 in the figure, and one output place Out. A net of this type starts functioning when each of its 4 input places receives a token. The characteristic of the token in place In_i is denoted by h_i and is related to the value 0 or 1 ($i = 1, ..., 4$) during the different executions of the net. Thus, the sequence of token characteristic values $\{h_1, h_2, h_3, h_4\}$ in the input places of the net can be interpreted as a 4-bit registration number. For the purposes of simplification, instead of "value of the characteristic of the token", we will use also "characteristic of the token". As a result of the GN execution, a token is received in the output place Out.

The set G_1 consists of such GNs which complete their execution at code 1 in case of exactly one 4-element variation of elements 0 and 1 with repetition which is given as a sequence of input values of the token characteristics in input places In_1, In_2, In_3 and In_4, respectively. In this case, a single token with characteristic value of 1 gets to their output place Out. This unique variation defines the correct

registration number of the respective net. In the cases of all the other 15 4-element variations of elements 0 and 1 with repetition, given as sequences of input values of the token characteristics in input places In_1, In_2, In_3 and In_4, these nets complete their execution so that a single token with characteristic value of 0 gets to their output place Out (which we also note as: their execution ends at code 0).

An example of such GN is given in fig. 4. This GN completes execution at code 1 only in the case of variation {1, 1, 0, 1} of the elements 0 and 1. The variation is given as a sequence of input values of the token characteristics in the respective places In_1, In_2, In_3 and In_4.

The set G_2 consists of GNs which, in the case of exactly one 4-element variation of elements 0 and 1 with repetition given as a sequence of input values of the token characteristics in input places In_1, In_2, In_3 and In_4, respectively, end their execution so that a single token with characteristic value of 0 gets to their output place Out (their execution ends at code 0). This unique variation defines the correct registration number of the respective GN. In the cases of all the other 15 4-element variations of elements 0 and 1 with repetition, given as sequences of input values of the token characteristics in input places In_1, In_2, In_3 and In_4, these nets complete their execution at code 1.

An example of such net is given in fig. 5. It completes its execution at code 0 only in case of variation {0, 1, 1, 1} of the elements 0 and 1, given as a sequence of input values of token characteristics in input places In_1, In_2, In_3 and In_4.

As the execution of any GN belonging to G_1 ends by code 1 (true), in case of a single 4-element variation of the elements 0 and 1 with repetition, it makes sense to choose as a protection tool a random net of this set. Let us denote the selected network with GN_1. However, it is obvious that this choice does not solve the task as finding the registration number will be done by manual checking of 2^4 4-element variations of the elements 0 and 1 with repetition. According to [8], an acceptable length of the registry number is 128. In order to increase the registry number length, we will upgrade the GN. To make this, a random GN, belonging to G_1 or to G_2, is chosen for each input place of the GN_1. If the value of the correct registration number in a place of GN_1 is 1, is chosen a random GN, belonging to G_1. If the value of the correct registration number in a place of GN_1 is 0, is chosen a random GN, belonging to G_2. The output of the chosen GN merges with the respective input place of the GN_1. After the first upgrade, we have a GN with 16 input places which represents the tree (fig. 2).

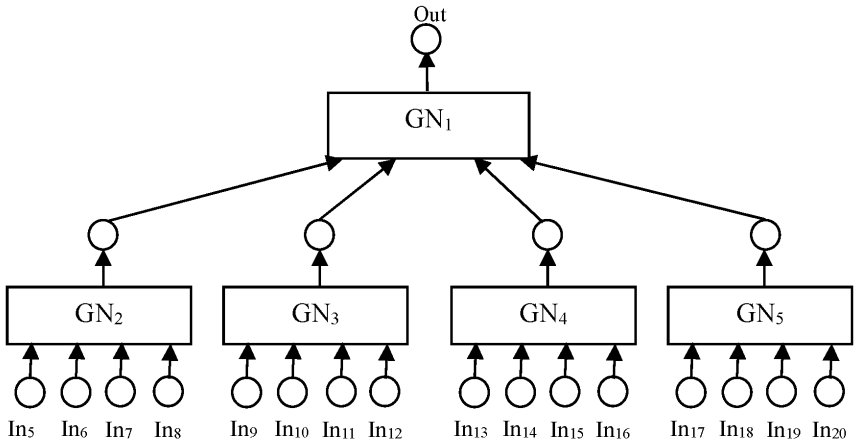


Fig. 2. Upgrading the GN_1 in its 4 input places

The sequence of correct registry numbers of the nets GN_2, GN_3, GN_4 and GN_5 defines the correct registration number of the tree-like GN.

Example. Let the GN in fig. 4 is chosen to protect the code. This GN completes execution by code 1 only in case of registration number {1, 1, 0, 1} given as a sequence of characteristics of the tokens in the input places In_1, In_2, In_3 and In_4. The GN is denoted as GN_e. In order to realize a GN with 7 input places, we upgrade GN_e in place In_1 by GN_e, as shown in fig. 3. Note that the registry number in place In_1 of GN_e has value of 1. In the case of this upgrade, the resulting GN will end its execution by a token with characteristic (code) 1 in place Out only in the case of the registration number – the variation {1, 1, 0, 1, 1, 0, 1}, given as a sequence of values of the characteristics of the tokens in places {In_5, In_6, In_7, In_8, In_2, In_3, In_4}. In the case of the other variations, it will end at a token with characteristic (code) 0. Similarly, the GN from fig. 3 can be upgraded in places In_2 and In_4 by random GNs belonging to set G_1 as they refer to code 1. The GNs belonging to G_1 are not suitable for upgrading place In_3 as they complete their execution at code 0 at Out for 15 4-element variations of the elements 0 and 1 with repetition. In this case, a suitable GN is one belonging to G_2.

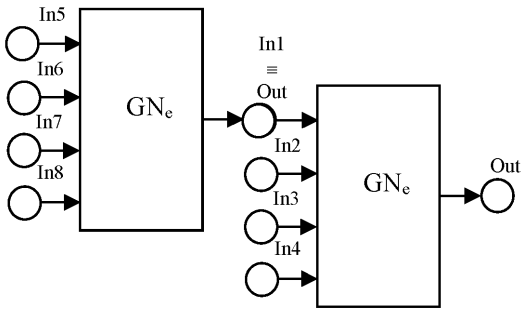


Fig. 3. Upgrading GN_e by GN_e in place In_1

The upgrading process can continue in the same manner. At the next level of upgrade, a GN with 64 input places will be produced; at the next: with 256, etc. The choice of a length of the registration number depends on the particular case of applying the protection method.

Choosing the appropriate level of security depends on the answers of the following questions: What is the price of the software to be protected? How long it should be protected for? What are the predicted resources available to the crackers, who would try to break the protection? [8].

An important point in applying the GN developed for software protection is its correctness. Part 4 of the article presents verification of the developed GN which is to be integrated into the protected software.

3 Examples of GNs belonging to sets G_1 and G_2, which are used by the method of software protection

This part is dedicated to two examples of GNs belonging to G_1 and G_2, respectively, which were used for describing the method of software protection.

Example of a GN belonging to G_1. The GN presented in fig. 4 has 4 transitions denoted by T_1, T_2, T_3 and T_4. It completes its execution after completing transition T_4. The only possibility to complete transition T_4 so that Out to contain a token with characteristic 1 is in case of one token with characteristic 1 in each place In_1, In_2 and In_4, and a token with characteristic 0 in place In_3. In case of all other values of the characteristics of the tokens in the input places of the GN, the execution of the latter will end at a state of token with characteristic of 0 in place Out.

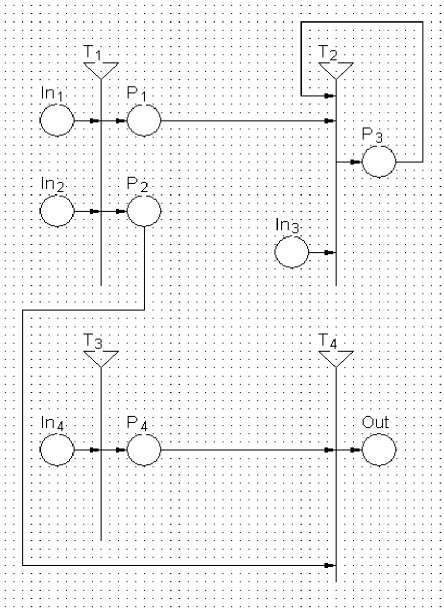


Fig. 4. An example of a GN which completes its execution at a token with code 1 in Out only in case of input tokens with characteristics {1, 1, 0, 1} in the respective input places {In_1, In_2, In_3 and In_4}

Definitions of the transitions of the GN are given below. The capacity of each GN arcs is 1. Let us note again that the tokens in places In_1, In_2, In_3 and In_4 are denoted by h_1, h_2, h_3 and h_4, respectively.

Transition T_1 is executed if there is one token in each of its input places In_1 and In_2. It is defined as follows:

$$T_1 = <\{In_1, In_2\}, \{P_1, P_2\}, t_{11}, t_{12}, r_1, M_1, \wedge(In_1, In_2)>, \text{ and}$$

r_1=	P_1	P_2
In_1	W1	¬W1
In_2	¬W1	W1

where
W1: h_1 is 1 $\wedge$ h_2 is 1

As a result of the transition execution, if W1 holds, the token from place In_1 transfers into place P_1, the tokens from place In_2 transfers into place P_2 and both tokens keep their characteristics. Otherwise (¬W1 holds), the token from place

In_1 transfers into place P_2, the tokens from place In_2 transfers into place P_1 and both tokens receive new characteristics with values of 0.

Transition T_2 is defined as follows:

$T_2 = <\{P_1, In_3, P_3\}, \{P_3\}, t_{21}, t_{22}, r_2, M_2, \wedge(P_1, In_3)>$, and

$r_2 =$	P_3
P_1	W2
In_3	¬W2
P_3	False

where

W2: The characteristic of the token in P_1 is 1 $\wedge$ h_3 is 1

T_2 is executed if there is one token in each of its input places P1 and In_3. As a result of the transition execution, if W2 holds, the token from place P_1 transfers to place P_3 and keeps its characteristic. Otherwise (¬W2 holds), the token from place In_3 transfers to place P_3 and receives a characteristic with value of 0.

Transition T_3 is executed if there is a token in its input place In_4. As a result of its execution, the token from place In_4 transfers into place P_4 and keeps its characteristics. It is defined as follows:

$T_3 = <\{In_4\}, \{P_4\}, t_{31}, t_{32}, r_3, M_3, \wedge(In_4)>$, and

$r_3 =$	P_4
In_4	True

Transition T_4 is executed if there is a token in each input place P_2 and P_4. Its priority is lower that this of transition T_2. It is defined as follows:

$T_4 = <\{P_2, P_4\}, \{Out\}, t_{41}, t_{42}, r_4, M_4, \wedge(P_2, P_4)>$, and

$r_4 =$	Out
P_2	¬W3
P_4	W3

where

W3: The characteristic of the token in P_2 is 1 $\wedge$
The characteristic of the token in P_3 is 0 $\wedge$
h_4 is 1

As a result of the transition execution, if W3 holds, the token from place P_4 transfers to place Out and receives characteristic of 1. Otherwise, the token from place P_2 transfers to place Out and receives characteristic of 0.

Example for a GN belonging to G_2: the GN presented at fig. 5 has 5 transitions (T_1, T_2, T_3, T_4 and T_5).

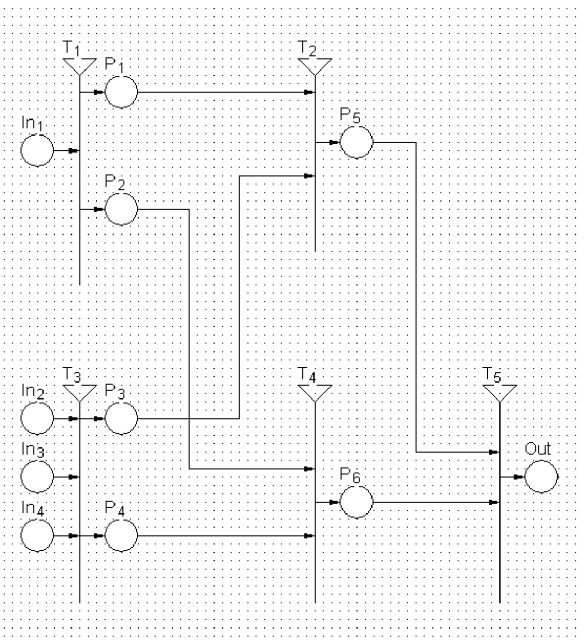


Fig. 5. Example for a GN, which completes its execution by code 0 only in case of values of the characteristics of the tokens {0, 1, 1, 1} in places {In_1, In_2, In_3, In_4}

This GN completes its execution after completing transition T_5. The only possibility for execution of transition T_5 so that Out contains a token with characteristic of 0 is if there is one token with characteristic of 1 in each place In_2, In_3 and In_4, and there is a token with characteristic of 0 in place In_1. In case of any of the other 15 possible values of the characteristics of the tokens in input places of the GN, the execution of the latter will end at a state of token with characteristic of 1 in place Out.

The definitions of the transitions of a GN follow. The capacity of each transition arc of a GN is 1. The characteristics of the tokens in places In_1, In_2, In_3 and In_4 are, again, respectively h_1, h_2, h_3 and h_4.

Transition T_1 is executed when there is a token in its input place In_1. As a result of its execution, the token in In_1 splits into two tokens which transfer in places P_1 and P_2. The characteristics of the new tokens are the same as this of the token in In_1. T_1 is defined as follows:

$T_1 = <\{In_1\}, \{P_1, P_2\}, t_{11}, t_{12}, r_1, *, \wedge(In_1)>$, and

r_1=	P_1	P_2
In_1	True	True

Transition T_2 is executed if there is a token in each of its input places P_1 and P_3. It is defined as follows:

$T_2 = <\{P_1, P_3\}, \{P_5\}, t_{21}, t_{22}, r_2, *, \wedge(P_1, P_3)>$, and

r_2 =	P_5
P_1	W1
P_3	¬W1

where

W1: The characteristic of the token in P_1 is 0 $\wedge$

The characteristic of the token in P_3 is 1

As a result of the transition execution, if W1 holds, the token from place P_1 transfers to place P_5 and keeps its characteristic. Otherwise, the token from place P_3 transfers to place P_5 and a characteristic with value 0 is received.

Transition T_3 is executed if there is a token in each input place In_2, In_3 and In_4. It is defined as follows:

$T_3 = <\{In_2, In_3, In_4\}, \{P_3, P_4\}, t_{31}, t_{32}, r_3, *, \wedge(In_2, In_3, In_4)>$, and

r_3 =	P_3	P_4
In_2	W2	¬W2
In_3	¬W2	W2
In_4	False	False

where

W2: h_2 is 1 $\wedge$ h_3 is 1 $\wedge$ h_4 is 1

As a result of the transition execution, if W2 holds, the tokens from places In_2 and In_3 transfer into places P_3 and P_4, respectively, and keep their characteristics. Otherwise, the tokens from places In_2 and In_3 transfer into places P_4 and P_3, respectively, and receive characteristics with values of 0. The token in place In_4 does not transfer.

Transition T_4 is defined as follows:

$T_4 = <\{P_2, P_4\}, \{P_6\}, t_{41}, t_{42}, r_4, *, \wedge(P_2, P_4)>$, and

r_4 =	P_6
P_2	¬W3
P_4	W3

where

W3: The characteristic of the token in P_4 is 1 $\wedge$

The characteristic of the token in P_2 is 0

As a result of the transition execution, if W3 holds, the token from place P_4 transfers to place P_6 and keeps its characteristic. Otherwise, the token from place P_2 transfers to place P_6 and receives characteristic of 0.

Lastly, the transition T_5 is defined as follows:

$T_5 = <\{P_5, P_6\}, \{Out\}, t_{51}, t_{52}, r_5, *, \wedge(P_5, P_6)>$, and

$r_5 =$	Out
P_5	W4
P_6	$\neg$W4

where

W4: The characteristic of the token in P_5 is 0 $\wedge$

The characteristic of the token in P_6 is 1

As a result of the execution of transition T_5, if W4 holds, the token from place P_5 transfers to place Out and receives characteristic of 0. Otherwise, the token from place P_6 transfers to place Out and receives characteristic of 1.

Next part of the article is dedicated to verification of the defined above GNs.

4 Verification of the designed for protection generalized net

Let the GN which will realize the software protection (we denote it as GNprot) be chosen to have the following characteristics:

k input places (k is the power of 2) – In_1, In_2, …, In_k,

m internal places P_1, P_2, …, P_m and

1 output place Out.

Its development was described in Part 2. Verifying GNprot means to prove that there is exactly one k-element variation of the elements 0 and 1 with repetition given as a sequence of input values of the characteristics of the tokens in In_1, In_2, …, In_k, which ends its execution at receiving a token with characteristic of 1 in its output place Out. In the case of the other one k-element variations of the elements 0 and 1 with repetition, what holds is that the execution of GNprot ends at receiving a token with characteristic of 0 in its output place Out.

Firstly, all GNs belonging to the sets G_1 and G_2 are verified. To this end, it must be proven that a random net belonging to the set G_1 ends its execution at code 1 in the case of a single 4-element variation of the elements 0 and 1 with repetition, whereas in the remaining 15 4-element variations of 0 and 1 of with repetition it is true that the execution of the net ends at code 0.

It must also be proven that a random net belonging to the set G_2 ends its execution at code 0 in the case of a single 4-element variation of the elements of 0 and 1 with repetition, whereas in the remaining 15 variations it is true that the execution of the net ends at code 1.

Next, structural induction is applied on the layers of upgrade.

Verification of the GNs belonging to the sets G_1 and G_2 is realized by tracking the transitions of the tokens between the places of these GNs. Examples of verification of the nets in fig. 4 and fig. 5 follow below.

Verification of the net in fig. 4

It will be proven that the execution of the net ends at code 1 only in case of input characteristics {1, 1, 0, 1} of the tokens in respective places {In_1, In_2, In_3, In_4}.

In order to do that, the changes in the characteristics of the tokens in places {In_1, In_2, In_3, In_4, P_1, P_2, P_3, P_4, Out} of the net will be tracked for all possible input values of the initial characteristics h_1, h_2, h_3 and h_4. The lack of a token in the respective place of the GN will be denoted by "-".

Let {In_1, In_2, In_3, In_4, P_1, P_2, P_3, P_4, Out} = {1, 1, 0, 1, -, -, -, -, -}. Execution is possible for the transitions T_1 and T_3. After their execution in parallel, the characteristics of the tokens in the net are of the following kind: {-, -, 0, -, 1, 1, -, 1, -}. At this stage, execution of transitions T_2 and T_4 is possible. As T_4 is of lower priority, transition T_2 is executed. The result after that (¬W2 holds) is: {-, -, -, -, 1, 1, 0, 1, -}. In the end, after executing transition T_4 (W3 holds), the result is: {-, -, -, -, 1, 1, 0, -, 1}, i.e. the execution ends at code 1.

In case of input characteristics: {0, 0, 0, 0}, {0, 0, 0, 1}, {0, 0, 1, 0}, {0, 0, 1, 1}, {0, 1, 0, 0}, {0, 1, 0, 1}, {0, 1, 1, 0}, {0, 1, 1, 1}, {1, 0, 0, 0}, {1, 0, 0, 1}, {1, 0, 1, 0},{1, 0, 1, 1} of the tokens in places {In_1, In_2, In_3, In_4} the execution of the GN ends by code 0, as ¬W3 holds (After executing T_1, the characteristic of the token in place P_2 is 0).

In case of input characteristics: {1, 1, 0, 0} and {1, 1, 1, 0} of the tokens in places {In_1, In_2, In_3, In_4} the execution of the GN ends by code 0, as ¬W3 holds (After executing T_3 the characteristic of the token in place P_4 is 0).

In case of an input characteristic: {1, 1, 1, 1} of the tokens in places {In_1, In_2, In_3, In_4} the execution of the GN ends by code 0, as ¬W3 holds. (After executing T_2 the characteristic of the token in place P_3 is 1).

Verification of the net in fig. 5

It will be proven that the execution of the net ends at code 0 only in case of input characteristics {0, 1, 1, 1} of the tokens in respective places {In_1, In_2, In_3, In_4}.

In order to do that, the changes in the characteristics of the tokens in places {In_1, In_2, In_3, In_4, P_1, P_2, P_3, P_4, P_5, P_6, Out} of the net will be tracked for all possible input values of the initial characteristics h_1, h_2, h_3 and h_4. Again, the lack of a token in the respective place of the GN will be denoted by "-".

Let {In_1, In_2, In_3, In_4, P_1, P_2, P_3, P_4, P_5, P_6, Out} = {0, 1, 1, 1, -, -, -, -, -, -, -}. Execution is possible for the transitions T_1 and T_3. As a result of their parallel execution, the characteristics of the tokens in the net become of the following kind: {-, -, -, 1, 0, 0, 1, 1, -, -, -}. Next, execution of transitions T_2 and T_4 is possible. As a result of their parallel execution, the characteristics of the tokens in

the net become of the following kind: {-, -, -, 1, -,0, 1, -, 0, 1,-}. After executing transition T_5, the characteristics of the tokens of the net become of the kind: {-, -, -, 1, -, 0, 1, -, -, 1, 0}.

In case of input characteristics: {0, 0, 0, 0}, {0, 0, 0, 1}, {0, 0, 1, 0}, {0, 0, 1, 1}, {0, 1, 0, 0}, {0, 1, 0, 1}, {0, 1, 1, 0}, {1, 0, 0, 0}, {1, 0, 0, 1}, {1, 0, 1, 0}, {1, 0, 1, 1}, {1, 1, 0, 0}, {1, 1, 0, 1}, {1, 1, 1, 0}, {1, 1, 1, 1} of the tokens in places {In_1, In_2, In_3, In_4}, the execution of the GN ends by code 1, as ¬W4 holds. (After executing T_4 the characteristic of the token in place P_6 is 0).

Verification of the whole GN

Structural induction is applied on the layers of upgrade.

Base case: (level 1; k = 4)

As the GN at level 1 is a random net belonging to G_1, the condition against which it is verified holds.

Inductive case (induction hypothesis):

Let the following holds for the GN resulting after n-th upgrade (with n layers) and possessing k/4 input places:

∃ only one k/4-element variation of the elements 0 and 1 with repetition {h_1, h_2, …, $h_{k/4}$}, which, when set as an input value of the characteristics of the tokens in {In_1, In_2, …, $In_{k/4}$} leads to completion of the GN execution at a token with characteristic 1 in output place. The GN execution ends at a token with characteristic 0 in output place in case of all remaining k/4-element variations of the elements 0 and 1 with repetition. We denote this net by $GN_{k/4}$.

Let GN_k is a net resulting of upgrading $GN_{k/4}$ once by the nets $GN_{k/4,i}$, i = 1, 2, …, k/4. This GN will be GNprot. The net $GN_{k/4,i}$ (i = 1, 2, …, k/4) belongs to the set G_1 if h_i equals 1, or to G_2 if h_i equals 0. For each net belonging to G_1 is true that its execution ends at code 1 in case of a single 4-element variation of the elements 0 and 1 with repetition, and at 0 in case of all 15 remaining 4-element variations of the elements 0 and 1 with repetition. Also, for each net belonging to G_2 is true that its execution ends at code 0 in case of a single 4-element variation of the elements 0 and 1 with repetition, and at 1 in case of all 15 remaining 4-element variations of the elements 0 and 1 with repetition. Therefore, for each net $GN_{k/4,i}$ (i = 1, 2, …, k/4) one of these two characteristics is true, i.e.

∃ only one 4-element variation of the elements 0 and 1 with repetition {$h_{i,1}$, $h_{i,2}$, $h_{i,3}$, $h_{i,4}$}, i = 1, 2, …, k/4, which, when set as an input value of the characteristics of the tokens in the input places of $GN_{k/4,i}$, leads to completing the execution of $GN_{k/4,i}$ at a token in the output place with code h_i.

Then, {$h_{1,1}$, $h_{1,2}$, $h_{1,3}$, $h_{1,4}$ $h_{2,1}$, $h_{2,2}$, $h_{2,3}$, $h_{2,4}$, ..., $h_{k/4,1}$, $h_{k/4,2}$, $h_{k/4,3}$, $h_{k/4,4}$} is the one and only k-element variation of the elements 0 and 1 with repetition, which, when

set as an input value of the characteristics of the tokens in the input places of $GN_{k/4,i}$, their execution is completed and returns the sequence of the characteristics of the tokens in the input places $\{h_1, h_2, \ldots, h_{k/4}\}$. After applying the induction hypothesis, the statement about GN_k with k input place is true.

5 Reliability of the method

Method reliability is determined by its correctness and by the length of the period of delaying the process of making the software freely available.

Method correctness is determined by the correctness of the defined GN, through which the correctness of the registration number is checked, as well as by the correctness of GN traversing by layers from the leaves to the root.

Experiments on the extent to which the popular methods of breaking protection will work in the case of this method have not been conducted yet. However, if the codebreaker is not familiar with the described method of programming code protection, the only way to find out the access code is by completely exhausting all possible values of the registration number. In case of a registration number with length k, $k \geq 256$, the number of different registration numbers is no less than 2^{256}.

6 Conclusion

The suggested method of software protection realizes complicated protection based on parallel traversing of a tree realized via a GN whose nodes are GNs. The main advantage of this method in comparison to the other anti-debugging methods are the tools of making the debugging more complicated: a large number of transitions included in the GN; parallel traversing of the tree-like GN structure; parallel execution of GNs at each node of the tree; a high level of multi-threading realized by the transitions of the GN.

References

1. Shields, T.: Anti-Debugging – A Developers View, Tech. rep. Veracode Inc., USA, 2009, http://www.secnews.pl/wp-content/ uploads/2011/05/whitepaper_antidebugging.pdf
2. Kim, D., Kwak, J., Ryou, J.: DWroidDump: Executable Code Extraction from Android Applications for Malware Analysis, International Journal of Distributed Sensor Networks, Volume 2015, Article ID 379682, http://dx.doi.org/10.1155/2015/379682
3. Gagnon, M., Taylor, S., Ghosh, A.: Software protection through anti-debugging, IEEE Security and Privacy, Vol. 5, 2007, pp. 82—84 (2007)
4. Atanassov, K.: On Generalized Nets Theory, Prof. Marin Drinov Academic Publishing House, Sofia (2007)
5. Trifonov T., Georgiev, K.: GNTicker – A software tool for efficient interpretation of generalized net models, Issues in Intuitionistic Fuzzy Sets and Generalized Nets, Vol. 3, Warsaw (2005)
6. Dimitrov, D. G..: GN IDE – A Software Tool for Simulation with Generalized Nets. Proceedings of Tenth Int. Workshop on Generalized Nets, pp. 70--75, Sofia (2009)
7. Dimitrov, D. G.: A Graphical Environment for Modeling and Simulation with Generalized Nets. Annual of "Informatics" Section, Union of Scientists in Bulgaria, Vol. 3, 2010, pp. 51--66 (In Bulgarian)
8. **Schneier, B.:** Applied Cryptography, John Wiley & Sons, ISBN 978-1-119-09672-6, 784 Pages, 1996

Embedded Architecture of Tolls Collecting System

Ioannis Patias, Vasil Georgiev

Faculty of Mathematics and Informatics
University of Sofia St.Kliment Ohridski“
ioannis.patias@gmail.com

Abstract. In this paper we present the system architecture of a Toll Collection System (TCS) based on Arduino Microcontroller. The system aims to reduce the traveling time, since there no need the car to stop in a queue to pay the tolls, but also the system is an advanced and affordable solution. RFID technology is used for vehicle identification. A unique RFID tag identifies each vehicle. The revenue authority assigns the tag. Following, our system stores all necessary information, like pre-paid amount. Strategically placed reader deducts the corresponding amount each time the vehicle passes the toll checkpoints. The system also updates the balance, and in case of insufficient balance, cameras placed on checkpoints can capture an image of the vehicle.

Key words: Embedded system architecture, business process modelling, control systems

1. Introduction

Toll Collection is an instrument for funding maintenance and rehabilitation of roads infrastructure. The processing time at checkpoints, operating costs, and traffic congestion caused by manual tolls collection can be avoided by using All-Electronic Tolls Collection based on RFID. The proposed RFID system consists of a tag transmitter, an antenna, and a computer. The transmitter consists of a microchip combined with an antenna. Tags can store up data and consist of microchip, and antenna, and also battery for the cases of active and semi-passive tags. All the components can be enclosed in plastic, or silicon. Passive RFID have no internal power source and use external power to operate. These tags are powered by the electromagnetic signal received from a reader. The received electromagnetic signal charges an internal capacitor on the tags, which in turn, acts as a power source and supplies the power to the chip. The RFID tag is used as a unique identification code for each vehicle. Thus, vehicle driving through a checkpoint, his RFID tag is automatically scanned. Vehicle’s ID number (tag’s number) is then crosschecked with the active ones stored in the system. In case the tag is not identified, we get a negative message from the system, and then a camera gets an image of car. In case of positive message we track the car until gets to an exit-tracking checkpoint. Then the amount calculated for tolls, based on the distance on the highway is deducted from vehicle’s account. The available amount can be sent to the vehicles owner by

different means, and for recharging the user can enter the system. After recharging the new amount is back stored for further use.

Problem Definition

The purpose is to develop a Tolls Collection System (TCS) Based on Arduino Microcontroller. The main objectives of this system are to reduce the processing time at checkpoints, the operating costs for the tolls operating authority, and the traffic congestion caused for the vehicles by manual tolls collection. By implementing an all-electronic tolls collection systems we propose a more efficient, environmentally friendly, and safer tolls collection system. Thus, we can increase tolls collection and provide a strong instrument for funding maintenance and rehabilitation of roads infrastructure, with an affordable cost.

Literature Review

Tolls Collection is the most widely used instrument to secure maintenance and rehabilitation of roads infrastructure. Apart of from highways and roads, also bridges and tunnels are usually maintained and rehabilitated, but also cover their operating costs collected in similar form either directly or indirectly.

The indirectly way means by increasing taxes on fuels for instance or other form of allocating the budget on the national revenue. The disadvantage of this method is the unfair weight allocation to taxpayers, which are not using this infrastructure. So, they have to pay the same as the ones directly benefited by the use of the infrastructure in place.

The direct way aims to charge tolls directly to the users benefiting of the use of the infrastructure. For instance directly charging tolls to the drivers crossing the bridge or tunnel. This way also allows the authorities to accelerate the infrastructure projects funding and thus implementation.

Financing projects by tolls collection also can be seen as an opportunity of outsourcing. The authorities can either build sooner the infrastructure using the collected tolls, instead of waiting first the income of taxes to accumulate, or mobilize private funds through PPP agreements.

According to European Commission (EC) those new financial schemes for European transport infrastructure projects, can mobilize significant private funds. With €3bn of EU funds in combination of grants and financial instruments, €30bn of investments on the ground and on-board could be generated [1].

There are various ways of tolls collection, which could be broadly categorized as [2]:

- Manual Tolls Collection

The manual tolls collection requires an operator, meaning a toll collector. Based on vehicles' classification the operator collects the cash. The operator does all the processing of the transaction, thus the processing time is long.

- Automatic Tolls Collection

Automatic tolls collection using of automated machines. Those machines usually accept coins. The processing time is shorter and the operating costs lower in comparison with the manual tolls collection.

- Electronic Tolls Collection

Electronic Tolls Collection (ETC) refers to a system, which automates the transactions processing. The system identifies, and classifies the vehicles equipped with the necessary encoded data tag while they move through the checkpoints. After the identification the system finalizes the transaction and the vehicles do not need to stop at all going through the checkpoint.

There are also various organizational systems of toll roads exist[3], which affect the respective tolls collection infrastructure:

- Open Tolls Collection

For open tolls collection, vehicles need to stop at checkpoints to pay the tolls. The advantage is the safe of money, since there is no need of infrastructure like checkpoints at every exit. The disadvantages are first the traffic congestion caused by the vehicles at every checkpoint along the highway, and second the ability the system allows drivers not to pay tolls, by avoiding the checkpoints, since they can exit and re-enter the highway.

- Closed Tolls Collection

In closed tolls collection systems, vehicles stop once entering the highway, to collect an entrance ticket, and again upon exit. The payment transaction takes place at the exit. The payable amount varies on the distance between the entrance and exit checkpoints.

- All-Electronic Tolls Collection.

All-electronic tolls collection systems are based on a no cash approach. The amount is deducted by the vehicles account automatically, or submitted to the owner in some form. The amount payable is calculated either on entering, or after exiting the highway. All-electronic tolls collection is the most efficient, safer, and fair tolls collection system.

In implementing All-Electronic Tolls Collection the most widely used identifying devices are the ones using Radio-Frequency Identification (RFID). Ubiquitous Computing [4] is called the idea of a post-desktop model of human-computer interaction. This is to have integration of information processing into everyday objects and activities, which in many cases the end-user uses more than one distributed systems and devices even simultaneously, without even being aware of their existence. RFID tags, or electronic labels represent an example of such implementations. They are used with objects to be monitored or tracked, like the vehicles on a highway. Using RFID we can identify and track the vehicles by using radio waves or sensing signals. There are tags, which can be tracked with range of tens or hundreds of meters. The syntax of RFID tags contains two major parts at least. The first is storing and processing information integrated circuit,

which is also modulating and demodulating a radio-frequency (RF) signal. The second part consists of an antenna, used for receiving and transmitting the radio signals.

There are three categories of RFID tags:

- active,
- semi-active, and
- passive

Tags can store up data and consist of microchip, and antenna, and also battery for the cases of active and semi-passive tags. All the components can be enclosed in plastic, or silicon. In general RFID tags help us in our everyday activities, since they are not expensive, and at the same time they can apply in almost any object.

2. TCS architecture

Functionality and Business Logic

The following Arduino microcontroller based Tolls Collection System flow chart diagrams, show the basic operations.

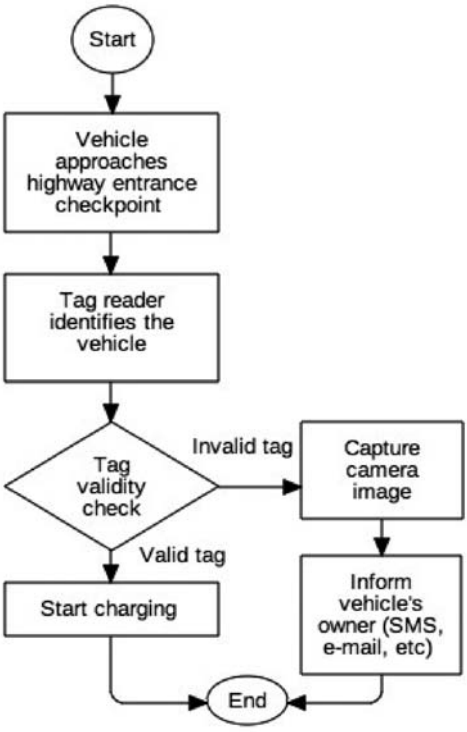


Fig. 1. TCS Flow Chart Diagram: Entrance checkpoint

As we can see in the figure above (Figure 1), once a vehicle approaches to a highway entrance checkpoint, it sends a identification tag. The system checks the ID tag validity. In case the ID tag is valid the system starts charging the vehicle. When the system cannot confirm the tag's ID validity, and as enforcement mesure, it captures a camera image and informs the vehicle's owner for the lack of a valid ID tag.

The tolls collection transaction takes place once a vehicle approaches a highway exit checkpoint (Figure 2). The system checks again the ID tag validity. In case the ID tag is valid the system checks the vehicle's account for funds sufficiency. In case there is sufficient amount deducts the tolls amount, updates the account's info, and informs the vehicle's owner. If there is not sufficient amount in the vehicle's account, then the system offers an option of eventually directly charging the vehicle owner's bank account, and again informs the owner for the transaction. When the system cannot confirm the tag's ID validity it captures a camera image and informs the vehicle's owner for the lack of a valid ID tag.

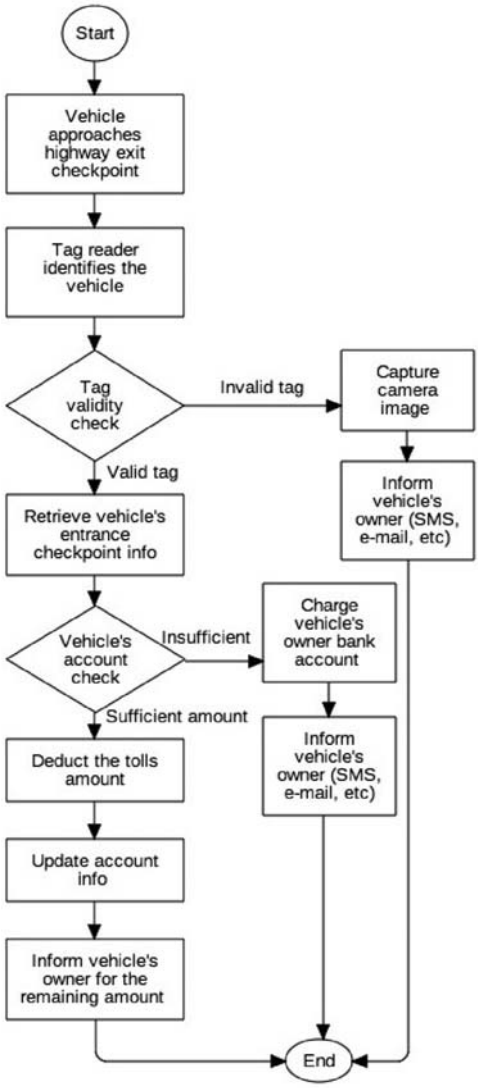


Fig. 2. TCS Flow Chart Diagram: Exit checkpoint

TCS deployment

When a vehicle approaches an entrance checkpoint, the entrance receiver captures an RFID signal sent by the vehicle transmitter. The receivers should be placed so to focus on signal captured at about 30m distance. The deployment diagram is as follows:

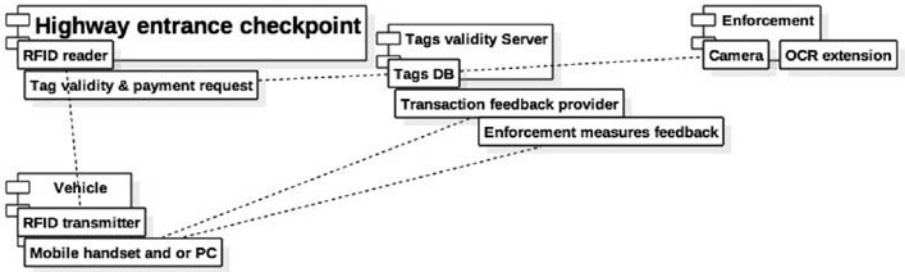


Fig. 3. Highway entrance checkpoint deployment diagram

Respectively, when a vehicle approaches a highway exit check point the following deployment diagram is in place:

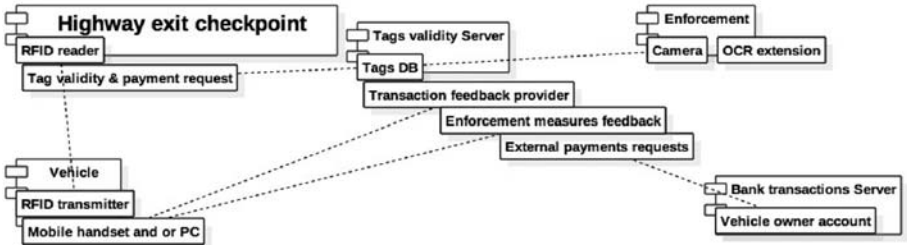


Figure 4: Highway exit checkpoint deployment diagram

The programming model of TCS including its control context and algorithmic transformations can be presented by the following pseudocode:

```
CASE: HIGHWAY ENTRANCE CHECKPOINT

LOOP () {
Read the RFID reader
If
The TAG_VALIDITY goes high
Then decode the ID
If
The ID is for VALID_TAG_VEHICLE
Then Enable the START_CHARGING module
Else
Enable the ENFORCEMENT module
}

CASE: HIGHWAY EXIT CHECKPOINT
```

```
LOOP () {
Read the RFID reader
If
The TAG_VALIDITY goes high
Then decode the ID
If
The ID is for VALID_TAG_VEHICLE
Then If
The AMOUNT is high
Then enable the TRANSACTION module
Else if the AMOUNT is low
Then Enable the LOW_AMOUNT_ENFORCEMENT  module
Else Enable the ENFORCEMENT module
}
```

CONCLUSIONS

The TCS based on Arduino microcontroller presented can really help tolls collection authorities increase the tolls collection, which is a strong instrument for funding maintenance and rehabilitation of roads infrastructure, with an affordable cost. The system reduces the processing time at checkpoints, the operating costs for the tolls operating authority, and the traffic congestion caused for the vehicles by manual tolls collection. As all electronic, the proposed tolls collection system is a more efficient, environmentally friendly, and safer tolls collection system.

REFERENCES

1 European Commision Infrastructure - TEN-T - Connecting Europe, http://ec.europa.eu/transport/themes/infrastructure/ten-t-guidelines/project-funding/doc/new_financial_schemes_for_european_transport_infrastructure.pdf

2 http://nptel.ac.in/courses/105101008/downloads/cete_46.pdf

3 International Journal of Electrical and Electronics Research ISSN 2348-6988 (online) Vol. 2, Issue 2, pp: (67-72), Month: April - June 2014, Available at: www.researchpublish.com

4 Kai Hwang, Geoffrey C. Fox, and Jack J. Dongarra, “Distributed and Cloud Computing From Parallel Processing to the Internet of Things”, 2012 Elsevier.

Automation Process By Means Of Proficy Machine Edition

Zh. Sartabanova, R Karassayev

Physics and Mathematics Faculty, K.Zhubanov Aktobe Regional State University, Kazakhstan

Abstract: This article discusses one of the modern universal application development tools - Proficy Machine Edition.

Keywords: Proficy Machine Edition, technology, machine interface, programming.

The modern production technology imposes high requirements on the automation of technological processes (Industrial control system), on the choice of optimum means of complex automation.

It is known that the production organization requires a set of various control devices, programs and systems that induce the enterprises to look for the single solution, which could provide all the needs of the existing infrastructure.

Proficy Machine Edition represents the universal environment for the development of applications for operator interfaces, traffic controls and automatic equipment. Providing the possibility of fast, effective object-oriented programming, the packet of Proficy Machine Edition uses all advantages of standard technologies, including XML, COM/DCOM, OPC and ActiveX. Machine Edition also includes Web functions, for instance, the built-in Web server, which provides data arrival in a real time and diagnostic messages to any employee of the enterprise [1].

All the components and appendices of the Proficy Machine Edition packet have the unified workspace and set of workbenches. The standardized user interface allows reduce the training period, and the integration of new applications doesn't require learning of additional operation principles. Combination of effective, user-friendly design makes the packet of Proficy Machine Edition the best choice for a man machine interface development, for programming of a PLC, for programs of traffic control and control on the basis of the PC.

Along with the general editing tools all the components of Proficy Machine Edition share the general objects in applications, including logical elements, scenarios, graphic panels and data structures. After creation of a variable that has several attributes, it could be used in other components of the project. [2]

Combining the best traditions of program and graphic applications with effective open technologies on the basis of industry standards, the packet of Proficy Machine Edition provides easy transition to the newest development workbenches. The interface of the software is shown in the Figure 1.

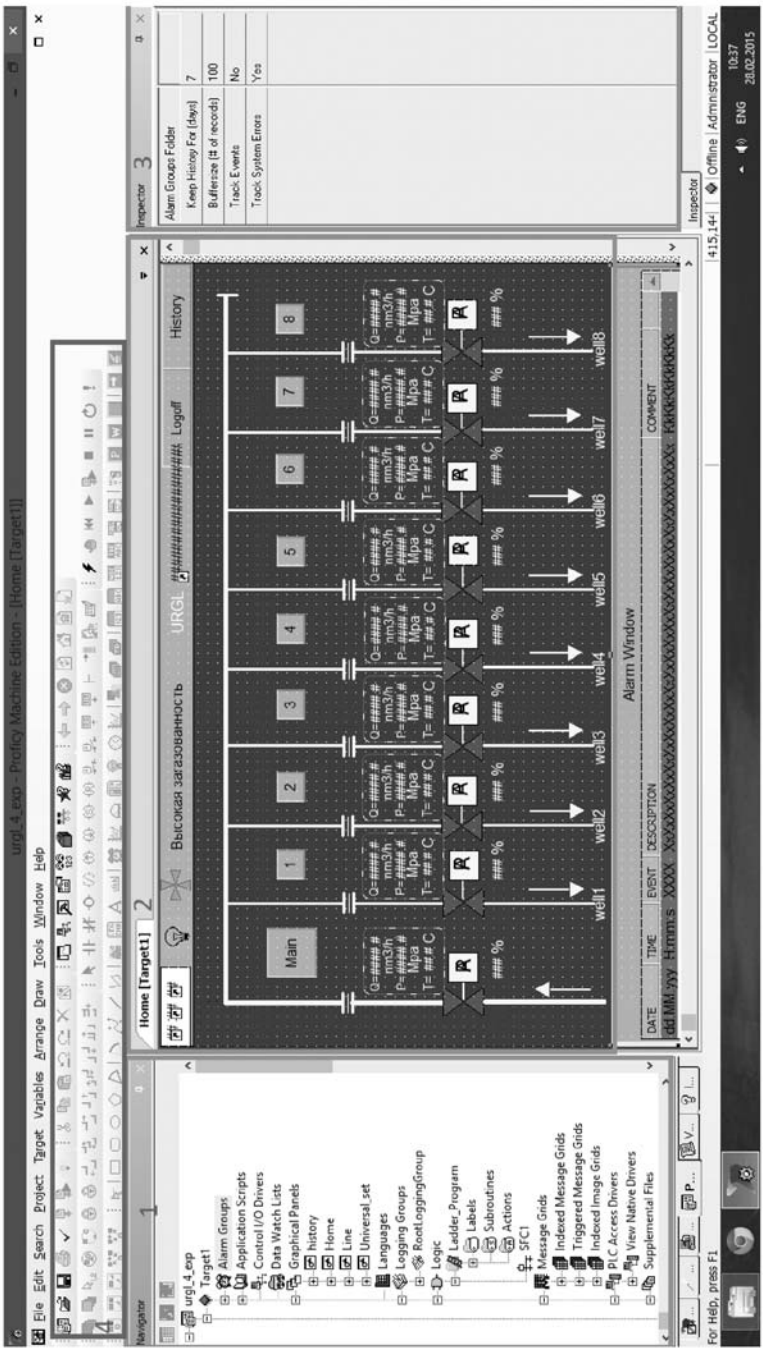


Figure 1. Program interface

1. Navigator (Project contents tree);
2. The principal screen (Opens the selected project element);
3. The inspector (Displays properties of the selected object);
4. Toolbar.

Scripts in this system are written in two languages, it is Visual Basic and View Script. An example of the simple script, which is responsible for copying assignment to variables corresponding value:

```
ft := Target2.temp
IF uni_Immitation = 1
     Target1.uni_Immitation_reverse := 0
ELSE
     Target1.uni_Immitation_reverse := 1
ENDIF
'-----------------------------------
IF Target1.Nomer = 0
     END
ENDIF
'-----------------------------------------------
IF Target1.Nomer = 1
     Target1.uni_alarm_val := Target2.alarm_val1
     Target1.uni_mode := Target2.mode01
     Target1.uni_mode1 :=Target2.mode2
     Target1.uni_rsp := Target2.rsp
     Target1.uni_POL := Target2.POL01
     Target1.uni_rsum :=Target2.rsum
     Target1.uni_h := Target2.h1
     Target1.uni_h12 := Target2.h12
     Target1.uni_h12_24 := Target2.h12_24
     Target1.uni_h24 := Target2.h24
     Target1.uni_mounth := Target2.mounth
     Target1.uni_year := Target2.year
     Target1.uni_rsumh :=Target2.rsumh
     Target1.uni_rsum12 :=Target2.rsum12
     Target1.uni_rsum12_24 :=Target2.rsum12_24
     Target1.uni_rsum24 :=Target2.rsum24
     Target1.uni_rsumm :=Target2.rsumm
     Target1.uni_rsumy :=Target2.rsumy
ENDIF
```

Flowcharts are built in the environment by help of the graphic tool. The example of the flowchart are provided in the Figure 2.

References

1. Logic Developer – PLC. Инструментальное программное обеспечение ПЛК
2. http://www.ingener.info/

Author Index