RJ10349 (A0506-007) June 20, 2005
Other

IBM Research Report

Levels of Business Structures Representation

K. Grigorova, P. Hristova, G. Atanasova
Department of Informatics and Information Technologies
University of Rousse
8 Studentska Str.
Rousse, Bulgaria

J. Q. Trelewicz
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

LEVELS OF BUSINESS STRUCTURES REPRESENTATION

K. Grigorova, P. Hristova, G. Atanasova. J. Q. Trelewicz

Abstract: *With the increasingly large number of software frameworks available to facilitate "business modeling", it is important to understand the implications of the level of abstraction provided by the frameworks. In this paper, we discuss three levels of abstraction of framework for business modeling, including the most typical and widely accepted representatives of each level. We show that XML is an emerging standard of data interchange and integration for business modeling. We discuss these frameworks in the context of database representation, which is important for storage and retrieval of models.*

Keywords*: Business Modeling, Modeling Languages, Modeling Techniques.*

Introduction

Competitive pressure, globalization, and the wide availability of Internet have made necessary the formal design of businesses. While in the past, business practices -- rules, routines, procedures, and processes -- could evolve in a piecemeal, isolated, and historical way, today, a rigorous and systemic design of such practices is needed, to ensure that customers' requests for products and services are processed at the satisfactory speed. Today's business modeling aims at the integration of the partial models that represent particular views on an enterprise. This means not only that models of distinctive and important parts of the enterprise should be created, but also that semantic relationships between partial models can be expressed.

The basic idea of business modeling is to offer different views on the business. The views should complement each other and thereby support a better understanding of complex systems by emphasizing appropriate abstractions. Therefore a corresponding modeling language is used based on specific terminology that is common within particular view. It provides intuitive concepts to structure the problem domain in a meaningful way.

The baseline views should be flexible in the sense that they can be applied to any business area. Then business modeling may provide concepts that can be reused and adapted in a convenient way to detailed models for businesses in a specific market. Specialized modeling languages are one example. Other examples include reference models for certain types of industry.

Business modeling is performed on different levels of detail according to the needs of designers. Sometimes it is sufficient to create a common picture of the enterprise, while other times the use of detailed concept is required. For this reason the business modeling methods should allow various levels of abstraction:

- The highest level in the hierarchy of abstraction is related to external description allowing the users to express their view of how a given business structure looks, a type of meta-modeling language.

Often, communication between people that belong to different professional communities will not require a high level of detail. Instead it is sufficient, and helpful, to seek a common understanding of the "big picture". On the other hand, there are also specific tasks, like the re-design of a business process or the design of an object model that require the use of detailed concepts. This confirms the existence of variety of modeling languages and techniques. They are used to create the meta-model of a given business structure.

- The next level in the hierarchy of abstraction considers internal representation of business structures, which is a kind of application level.

This level of representation corresponds to the requirements of a given application. The variety of modeling languages and techniques does not demand the multiplicity of internal descriptions. It is preferable to use some standard approaches in order to allow the successful exchange of models between different environments.

Lately the emergence of XML (eXtensible Markup Language) is a first step to solve the problem of the variety modeling languages. XML is now widely accepted and acknowledged a standard. XML allows representing information in a simple, readable format, easily parsed by software tools.

- The third, and final, level of the hierarchy of abstraction is related to the real (existing) storage.

The representation on this level may be viewed in some sense as on-line database. In some implementations, the user is not allowed to access it and its format may not even be known by the user. In other implementations, it might be useful to enable the system analysts to operate directly with database models. Additionally, the effective database design leads to successful performance.

In this paper we discuss both external and internal levels of business structures representation and indicate the corresponding database models as a proper subject of efficient analysis. Sometimes we refer to modeling techniques suitable mostly for representation of business processes as the most important part of business modeling.

External representation of business structures

There is a large variety of meta-languages commonly used for the representation of the highest level of business structures. Generally these meta-languages involve different graphical primitives for describing the objects and connections between them.

The most important aspect of all of the meta-languages is important part that the notation plays in any model – it is the glue that holds the process together. Notation has three roles:

- It serves as the language for communicating decisions that are not obvious or cannot be inferred from the core itself;
- It provides semantics that are rich enough to capture all important strategic and tactical decisions;
- It offers a form concrete enough for humans to reason and for tools to manipulate.

Here we will discuss some typical and wide spread representatives of modeling tools, used for external business structures description.

Flowcharting

Flowcharting is among the first graphical modeling techniques, dating back to the 1960s. The advantages of flowcharts centre on their ability to show the overall structure of a system, to trace the flow of information and work, to depict the physical media on which data are input, output and stored, and to highlight key processing and decision points [Schriber, 1969] [Jones, 1986].

Flowcharting was initially intended to provide computer program logic representation, but, because of its flexible nature, it has been used in many other application areas as well, including business modeling. Despite its advantages, namely familiarity and ease of use, flowcharting is no longer a dominant modeling technique because it can provide only basic facilities in representing processes. Therefore, in the area of business modeling, flowcharts nowadays are typically used primarily as a simple, graphic means of communication, intended to support narrative descriptions of processes, when the latter become complicated and difficult to follow.

Data flow diagrams

Data Flow Diagramming (DFD) is a technique for graphically depicting the flow of data amongst external entities, internal processing steps, and data storage elements in a business process. DFDs are used to document systems by focusing on the flow of data into, around, and outside the system boundaries. In that respect, DFDs are comparable to flowcharts, differing from them basically in the focus of analysis: DFDs focus on data, instead of activities and control [Yourdon, 1989].

DFDs have been widely used for data modeling purposes and have become an ad-hoc standard notation for traditional systems analysis and design.

DFDs used to model the system's data processing and the flow of information from one process to another. They are an intrinsic part of many analysis methods. They show the sequence of processing steps traversed by the data. Each step documents an action taken to transform or distribute the data. DFDs are easy to read, making it possible for domain experts to create or to validate the diagrams [Sommerville, 2003].

Entity-Relationship diagrams

Entity-Relationship (ER) diagrams [Yourdon, 1989] are another widely used data modeling technique. ER diagrams are network models that describe the stored data layout of a system. ER diagrams focus on modeling the data present in a system and their inter-relationships in a manner that is entirely independent of the processing that may take place on that data. Such separation of data and operations may be desirable in cases where the data and their inter-relationships are complex enough to necessitate such an approach.

For the purposes of business process modeling, ER diagrams share similar limitations with DFDs. More specifically, ER diagrams focus primarily on data and their inter-relationships and hence do not provide constructs for modeling other process elements. Even more importantly, ER diagrams, unlike DFDs, do not provide any information about the functions depicted that create or use these data. Finally, ER diagrams are entirely static representations, not providing any time-related information that could drive analysis and measurement.

State-Transition Diagramming

State-Transition (ST) diagrams originate from the analysis and design of real-time systems. ST diagrams attempt to overcome the limitations arising from the static nature of DFDs and ER diagrams by providing explicit information about the time-related sequence of events within a system. The notation being used by standard ST diagrams is very simple, consisting only of rectangular boxes that represent states and arrows that represent changes of state (transitions) [Quatrany, 2001].

Namely the possibility for a transition's depiction allows the usage of State-Transition diagrams as internal description tool. The explicit description of time-related sequence of data changes points out context relationship, which is in the base of the internal description.

Role Activity Diagramming

Role Activity Diagrams (RADs) uses diagrammatic notation that concentrates on modeling individual or group roles within a process, their component activities and their interactions, together with external events and the logic that determines what activities are carried out and when [Huckvale, 1995]. RADs differ from most other process diagrammatic notations in that they adopt the role, as opposed to the activity, as their primary unit of analysis in process models. Due to this focus, they are mostly suitable for organizational contexts in which the human element is the critical organizational resource that process change aims to address.

Business Process Modeling Notation

The Business Process Modeling Notation (BPMN) is the new standard for modeling business processes and web service process, as put forth by the Business Process Management Initiative (BPMI). BPMN is a core enabler of a new initiative in the Enterprise Architecture world called Business Process Management.

BPMN is only one of three specifications that the BMNI has developed – the other two are a Business Process Modeling Language (BPML) and a Business Process Query Language (BPQL).

BPMN specification provides a graphical notation for expressing business processes in a Business Process Diagram (BPD). The objective of BPMN is to support business process management by both technical users and business users by providing a notation that is intuitive to business users yet able to represent complex process semantics [Owen, 2003] [Stephen, 2001].

A BPD is made up of a set of graphical elements. These elements enable the development of simple diagrams that are intended to look familiar to most business analysts, resembling a flowchart-type diagram. The elements were chosen to be distinguishable from each other and to utilize shapes that are familiar to most modelers. For example, activities are rectangles and decisions are diamonds. It should be emphasized that one of the drivers for the development of BPMN is to create a simple mechanism for creating business process models, while at the same time being able to handle the complexity inherent to business processes. The approach taken to handle these two conflicting requirements is to organize the graphical aspects of the notation into specific categories. This approach provides a small set of notation categories, easier recognition of the basic types of elements and understanding of the diagram. Within the basic categories of elements, additional variation and information can be added to support the requirements for complexity without dramatically changing the basic look-and-feel of the diagram.

Internal representation of business structures

Business structure models must be capable of providing various information elements to its users. Such elements include, for example, what activities are carried out, who is performing these activities, when and where are these activities performed, how and why are they executed, and what data elements they manipulate. Modeling techniques differ in the extent to which their constructs highlight the information that answers these questions. To provide this information, a modeling technique should be capable of representing one or more of the following "perspectives" [Curtis, 1992]:

- Functional perspective: Represents what activities are being performed.
- Behavioral perspective: Represents when activities are performed (for example, sequencing), as well as aspects of how they are performed through feedback loops, iteration, decision-making conditions, entry and exit criteria, and so on.
- Organizational perspective: Represents where and by whom activities are performed, the physical communication mechanisms used for transfer of entities, and the physical media and locations used for storing entities.
- Informational perspective: Represents the informational entities (data) produced or manipulated and their relationships.

There are known some techniques and specific languages for internal description of business structures. All of the techniques discussed here posses the above perspectives. The main benefit of one of these techniques is to build a model of a business structure that is suitable for future data processing. We discuss standard approaches here, since it is appropriate to use some standard approach to ensure interoperability between environments.

It is important to point out that the internal models describe the context data dependence, the internal relationship between processes and subprocesses and the data flow in the scope of the presenting business structure.

The presentation of some wide spread techniques for business structure representation follows.

IDEF Techniques

The IDEF family of modeling techniques was developed as a set of notational formalisms for representing and modeling process and data structures in an integrated fashion. The IDEF suite consists of a number of independent techniques, the most well known being IDEF0 (Function Modeling), IDEF1x (Data Modeling), and IDEF3 (Process Description Capture).

The IDEF0 method is designed to model the decisions, actions, and activities of an organization or other system and, as such, it is targeted mostly towards the functional modeling perspective (Mayer). As a communication tool, IDEF0 aims at enhanced domain expert involvement and consensus decision-making through simplified graphical devices. Perhaps the main strength of IDEF0 is its simplicity, as it uses only one notational construct, called the ICOM (Input-Control-Output-Mechanism). IDEF0 supports process modeling by progressively decomposing higher-level ICOMs into more detailed models that depict the hierarchical decomposition of activities.

Despite its advantages, IDEF0 presents a number of limitations that may render the technique unsuitable for process analysis. More specifically, IDEF0 models are static diagrams with no explicit or even implicit representation of time. Even the sequence of ICOMs is not meant to depict the temporal relations between activities. As such, IDEF0 models cannot represent the behavioral or informational modeling perspectives. To overcome some of the limitations of IDEF0 models, IDEF3 has been developed. IDEF3 describes processes as ordered sequences of events or activities. As such, IDEF3 is a scenario-driven process flow modeling technique, based on the direct capture of precedence and causality relations between situations and events. The goal of an IDEF3 model is to provide a structured method for expressing the domain experts' knowledge about how a particular system or organization works (as opposed to IDEF0, which is mainly concerned with what activities the organization performs).

IDEF1x was designed as a technique for modeling and analysis of data structures for the establishment of database requirements. IDEF1x differs from traditional data modeling techniques in that it does not restrict the model to the data elements that are being manipulated by computers, but allows the modeling of manually-handled data elements as well. IDEF1x utilizes simple graphical conventions to express sets of rules and relationships between entity classes in a fashion similar to Entity-Relationship diagrams.

The power of IDEF1x diagrams for integrated databases can be harnessed when these diagrams are combined with IDEF0 and IDEF3 business models. Since they belong to the same "family" of techniques, IDEF models can complement each other effectively and, when combined, can provide a holistic perspective of a modeled system. However, this facility comes at a potentially high complexity of developing and maintaining many different models for a single system.

Petri Nets

Petri Nets do not provide a business process structure representing technique, since they have originated from and have been traditionally used for systems modeling. However, among the systems modeling techniques, Petri Nets is perhaps the one technique that has received the most attention as a potential candidate for business process structure representing as well [Reising, 1992]. Basic Petri Nets are mathematical-graphical representations of systems, intended for assisting analysis of the structure and dynamic behavior of modeled systems, especially systems with interacting concurrent components [Peterson, 1981]. A basic Petri Net graph is composed of a set of states and a set of transitions.

It has been recognized that basic Petri Nets are not succinct and manageable enough to be useful in modeling and representing high-level, complex business processes structures. To this end, a number of extensions to the basic Petri Net formalism (usually to include the notions of “colour”, “time”, and “hierarchy”) have been proposed [Jensen, 1996]. These extensions are collectively referred to as “high-level Petri Nets” and include, for example, Generalised Stochastic Petri Nets (GSPN), Coloured Petri Nets (CPN), and others.

The power of Petri Nets for internal description is the well-composed formalism consisting of the set of states and the state of transitions. Those familiar with this formalism can use the internal structure description for different intentions in any chosen language without environmental dependence.

Unified Modeling Language (UML)

Introduced in 1997 and supported by major industry-leading companies, the Unified Modeling Language (UML) has rapidly been accepted throughout the object-technology community as the standard graphical language for specifying, constructing, visualizing, and documenting software intensive systems [Booch, 1999]. UML utilizes a wide array of diagrammatic notations, including:

- Use case diagrams, which capture system functionality as seen by the users
- Class diagrams, which capture the vocabulary of the system.
- Behavior diagrams (for example state chart, activity and interaction diagrams).
- Implementation diagrams (for example, component and deployment diagrams).

The underlying reason for the development of the language is simple: although a wide variety of notational languages have long existed for the representation of software systems, most languages are typically aligned with a particular analysis and design method. This wide variety can be a source of complexity and problems of non-compatibility between languages. UML attempts to address this gap by being a “universal” language, covering everything from business process representation to database schema depiction and software components modeling According to its developers, UML “will reduce the degree of confusion within the industry surrounding modeling languages. Its adoption would settle unproductive arguments about method notations and model interchange mechanisms, and would allow the industry to focus on higher leverage, more productive activities” [UML, 1997].

As far as business structure representation and database modeling are concerned, UML is mostly targeted to systems modeling situations, although an “extension for business structure modeling” has also been developed. Some authors [Trelewicz, 2004] argue that UML is not appropriate for these applications because of its lack of context and structural complexity and resistance to the natural evolution of business structures. Furthermore, some may argue that the language is heavily based on the object-oriented paradigm and hold out very good possibilities for internal system representation without program environment and language dependence. There is no reason to be used in situations where the modelers want to follow only the system overview.

SADT (Structured Analysis and Design Technique)

An SADT model is a simple representation one aspect of business structure, which is adequate for a given purpose. To achieve the benefits of the principles of structuring, especially top-down, levels of detail and hierarchy the model should be graphic. Each SADT model consists of a set of related diagrams, which are organized in a top-down manner. Each diagram is either a summary (parent) diagram or a detailed (child) diagram of the parent [Vernadat, 1996].

There exist two types of SADT models. An Activity model is oriented toward the decomposition of activities whereas a Data model is oriented toward the decomposition of data. Each type of model contains both activities and data; the difference lies in the primary focus of the decomposition.

A SADT Activity Model is used to describe the decomposition of activities. Data is included in the activity model as inputs, outputs, controls, and mechanisms. The top-level diagram is detailed on separate diagrams. All data is related to a given activity is explicitly shown (usually in more detail) on the lower level diagrams.

When one develops a Data model, it is not a mirror image of the Activity model, but rather the Data model is used like a data dictionary to provide a more rigorous definition of data. Experience has shown that the development of an Activity model in and by itself does not force a precise decomposition of data.

The SADT usage for internal structure representation provides the ability for explicit depiction of data and process relationships without program language and environment dependence. Each structure represented with this technique may be used in different ways and for different purposes even after its storage.

Extensible Markup Language

Many applications use business descriptions. The problem is that these applications work with descriptions in their own internal representations. Therefore communication between them, a growing need for industry, is nearly impossible without some kind of translator. That is why some sort of exchange standard is needed in order to avoid a point-to-point translator for every pair of applications.

Extensible Markup Language (XML) is widely recognized as a rapidly emerging standard for moving data over the Internet. It is a scripting language for representing structured data in a text file. The structured data represented by XML can be virtually anything, for example, address books, configuration parameters, spreadsheets, Web pages, financial transactions, technical drawings, and so on. XML defines a set of rules for text formats for such data. By storing data in a structured text format, XML allows the user to read the data independent of the program that produced it. XML files are easy for computers to generate and read, they are unambiguous, and they avoid common pitfalls of text data formats, such as lack of extensibility, lack of support for internationalization and localization, and platform dependency.

XML offers many advantages as a general-purpose mechanism for representing data and communicating between applications [XML, 2002]:

- Flexibility

XML can be used for an enormous variety of different purposes just by defining element names and arrangements appropriate for the particular purpose. Since each element is clearly marked with begin and end tags, elements can grow and shrink as needed. Finally, the nesting property of XML elements makes it easy to combine smaller documents into larger documents.

- Portability

XML documents can be moved easily among machines or over the Internet because they are based on text, not binary representations. The text form used in XML is Unicode, which supports all of the world's languages, allowing the use of XML for applications that span national boundaries.

- Self-describing

Each XML document carries a structural description of its contents in the form of the element tags. This makes it much easier for one application to use an XML document created by a different application.

- General-purpose tools

Since all XML documents have the same basic form, general-purpose tools can be created that operate on any XML document, such as tools that create documents, display their contents, modify their structure, or record statistics about the flow of XML documents in a system. Several general-purpose XML parsers have already been created, which makes it easy to XML-enable applications.

- Robustness

Because XML documents are self-describing, XML-based applications can be built to tolerate errors and to evolve with changes in the content structure. The tags also allow graceful evolution of XML-based software. A new element can be added to a document without affecting existing software that uses the document: old software will simply ignore the new element.

- Human-readability

Although XML is intended for processing by computer programs, its textual form is also possible for humans to read. This can be useful when debugging XML-based applications and means that, if needed, a human can use an ordinary text editor to create or repair XML documents.

A software module called an XML processor is used to read XML documents and to provide access to their content and structure. It is assumed that an XML processor is doing its work on behalf of another module, called the application. This specification describes the required behavior of an XML processor in terms of how it must read XML data and the information it must provide to the application.

XML's real impact is in the area of data interchange and integration. Over the next few years XML promises to revolutionize the way that applications and enterprises exchange information. XML makes it much easier for applications to work together, even when they are in different organizations.

Database representation of business structures

Database models of business structures are usually hidden and the user does not access them directly, but rather through an interface on the software tools for capturing and analyzing the models. We include in our definition of "database representations" those structures that business modeling tools may utilize for file storage or analysis; i.e., we do not restrict to tables in recognized relational database middleware. In many cases the database design plays an important role for making possible the kinds of analysis that the business designer wishes to perform on the model. If the designer is authorized to have knowledge of the structure of the business model in the database, more efficient understanding, implementation and support may be possible. For example, this can allow the designer to structure the representation in such a way to facilitate faster analysis or higher degree of reuse of business process templates.

When the business designer needs to deal with only a small part of business model working with its database model is very convenient. There are a number of aspects to the enterprise, each giving a different view on it, presented by corresponding business structure.

When designing the business, the business designer will often create one aspect of the enterprise at a time, linking the aspects of the enterprise once each aspect is understood. A business designer interested in one aspect of the business can look just at that aspect of the business, without considering the others. However, having in mind the interdependence of the aspects and the affect they may have on each other, the business analyst have to take care about the connections between the different aspects.

Formally the business structures that we discuss are mostly graphs, comprising nodes and edges. Thus, the database approach is addressed to database representation of graphs. The database model must allow modifications with computational efficiency. Also, the database must support the structuring of queries appropriate for the business model. The database representation is scaleable, the memory usage associated with the business model increases linearly in respect with the of the business structures.

Conclusions and future works

Our current work addresses the co-representation of business structures using XML and the BPMN standards. We have chosen these representations of the suite of prior art discussed above, leveraging the wide acceptance and ad-hoc standardization that is provided by these options.

In spite of its advantages XML has three disadvantages, all of which are inevitable consequences of XML's flexibility [Bouret, 2002]:

- Size

XML documents occupy a lot of space due to the use of text for everything and the presence of the tags. Thus, XML documents will take more space on disk and may also take more time when transmitting over a network.

- Performance

It takes time to read and write XML documents. The tags must be read and processed, and information such as numbers will have to be converted from its textual form to the form that the application needs.

- Complexity

Reading an XML document is very complicated due to the tag processing that must occur.

Despite these limitations, our subsequent work will discuss the analysis of business models utilizing XML and leveraging the large number of tools available for its creation and editing, such as Microsoft Visio or Rational Software Modeler.

Acknowledgements

The paper present results of research project “Database representation of business architectures for efficient analysis and modification” supported by IBM.

Bibliography

[Booch, 1999] G. Booch, J. Rumbaugh, I. Jacobson, Unified Modeling Language User Guide, Addison-Wesley, Reading, MA, 1999

[Bouret, 2002] R. Bourret, XML and Databases, (http://www.rpbourret.com), 2002

[Curtis, 1992] W. Curtis, M. I. Kellner, J. Over, Process Modeling, Communications of the ACM, 35, 9, 1992, pp. 75-90.

[Huckvale, 1995] T. Huckvale, M. Ould, Process Modeling – Who, What and How: Role Activity Diagramming. In Grover, V. and Kettinger, W.J. (Eds.), Business Process Change: Concepts, Methods and Technologies, Idea Group Publishing, Harrisburg, PA, 1995, pp. 330- 349

[IDEF, 2003] IDEF Family of Methods, A Structured Approach to Enterprise Modeling and Analysis, Knowledge Based Systems, Inc., http://www.idef.com/, 2003

[Jensen, 1996] K. Jensen, Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Springer Verlag, Berlin, 1996

[Jones, 1986] J. L. Jones, Structured Programming Logic: A Flowcharting Approach, Prentice Hall, New Jersey, 1986

[Owen, 2003] M. Owen, J. Raj, BPMN and BPM. Introductions to the New Business Process Modeling Standard, PopkinSoftware, www.popkin.com, 2003,

[Peterson, 1981] J. L. Peterson, Petri Net Theory and the Modeling of Systems, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[Quatrani, 2001] T. Quatrani, Visual Modeling with Rational Rose 2000 and UML, Addison-Wesley Publishing Company, 2001

[Reising, 1992] W. Reising, S. S. Muchnick, P. Schnupp, (Eds.) A Primer in Petri Net Design, Springer Verlag, Berlin, 1992.

[Sommerville, 2003] I. Sommerville, Software Engineering, Addison-Wesley Publishing Company, 2003

[Schriber, 1969] T. J. Schriber, Fundamentals of Flowcharting, Wiley, New York, 1967

[Stephen, 2001] A. Stephen, Introduction to BPMN, IBM Corporation,
http://www.bpmn.org/Documents/IntroductionBPMN.pdf, 2001

[Trelewicz, 2004] J. Q. Trelewicz, J. L. C. Sanz, D. W. McDavid, A. Chandra, S. C. Bell, Informatics for business is more than process automation: i-BUSINESS > e-PROCESS, Int'l Conf on Automatics and Informatics, SAI'04.

[UML, 1997] UML Proposal to the Object Management Group, http://www.rational.com/uml, 1997

[Vernadat, 1996] F. V. Vernadat,. Enterprise Modeling and Integration: Principles and Applications, Chapman & Hall, 1996

[XML, 2002] XML Introduction, (http://www.tcl.tk/advocacy/xmlintro.html), 2002

[Yourdon, 1989] E. Yourdon, Modern Structured Analysis, Prentice Hall International, Englewood Cliffs, NJ, 1989

[Хохлова, 2004] Хохлова М. Н., Теория Эволюционного Моделирования, ФГУП Цниииа-то-минформ, Москва, 2004

Author information

Katalina Grigorova - Department of Informatics and Information Technologies, University of Rousse, 8 Studentska Str, Rousse -7017, e-mail: katya@ami.ru.acad.bg

Plamenka Hristova - Department of Informatics and Information Technologies, University of Rousse, 8 Studentska Str., Rousse -7017, e-mail: pamela@ami.ru.acad.bg

Galina Atanasova - Department of Informatics and Information Technologies, University of Rousse, 8 Studentska Str., Rousse -7017, e-mail: gea@ami.ru.acad.bg

Jennifer. Q. Trelewicz, - IBM Research Relationship Manager, Eastern Europe, Russia, and CIS, Research Staff Member,IBM Almaden Research Center, e-mail: trelewicz@us.ibm.com