LATVIJAS UNIVERSITĀTES RAKSTI

ACTA UNIVERSITATIS LATVIENSIS

Datorzinātne un informācijas tehnoloģijas

Computer Science and Information Technologies

669

ISSN 1407-2157

LATVIJAS UNIVERSITĀTES
RAKSTI
669. SĒJUMS

Datorzinātne un informācijas tehnoloģijas

SCIENTIFIC PAPERS
UNIVERSITY OF LATVIA
VOLUME 669

Computer Science and Information Technologies

SCIENTIFIC PAPERS
UNIVERSITY OF LATVIA
VOLUME 669

Computer Science and Information Technologies

Automation of Information Processing

LATVIJAS UNIVERSITĀTE

LATVIJAS UNIVERSITĀTES
RAKSTI
669. SĒJUMS

Datorzinātne un informācijas tehnoloģijas

Informācijas apstrādes automatizācija

UDK 004
Da 814

Visi krājumā ievietotie raksti ir recenzēti.
Pārpublicēšanas gadījumā nepieciešama Latvijas Universitātes atļauja
Citējot atsauce uz izdevumu obligata

ISSN 1407-2157
ISBN 9984-770-03-4

Contents

Preface

This volume is the first one in the new series of Acta Universitatis Latviensis in **Computer Science and Information Technologies**. Research in computer science and software engineering has been done at the University of Latvia since 1960s, and there are hundreds of publications by computer scientists of the University of Latvia in various scientific journals and conference proceedings worldwide. However, there have been only a few attempts to publish collections of the University computer science research papers in the Acta - there were three volumes in the mid 1970s in Russian.

The first volume in the new series – **Automation of Information Processing** contains recent results of young researchers, most of them doctoral students at the University of Latvia. Though the topics of the papers are quite different, they are all centered around the problem of providing theory, methodology, development tools and supporting environment for the development of information systems. All the papers in the volume are related to the most up-to-date issues in the respective area.

Theoretical problems are discussed in papers by Ģirts Linde and Kārlis Freivalds – both of them papers of high scientific quality. The paper by Linde is devoted to the formalization of semantics of class diagrams – the main UML notation for system design and to formalization of class diagram equivalence. Freivalds' paper provides new results and an efficient heuristic algorithm for a classical optimization problem in graph theory – finding the minimum ratio-cut.

Several papers are devoted to generic modeling and development tools for information systems. The Paper by Edgars Celms discusses an important aspect of metamodel based modeling tools – generic facilities for defining tables. The paper by Guntis Arnicans is devoted to a generic language interpreter implementing a new method for defining language semantics. The paper by Jānis Iljins discusses an experience in using a generic development environment – the IS Technology, where the design specification can be directly interpreted.

The papers by Jānis Benefelds and Laila Niedrite consider various aspects of data warehouses – an overview of data staging and an experience of application to the education domain.

Two papers discuss the software development management problems – Baiba Apine the issues of various software development estimation models and Martins Gills – the traceability issues in software testing.

Finally, the paper by Mara Gulbe and Arnis Gulbis consider the availability of IT services in Latvia.

All the papers in the volume have been reviewed by several members of the international Editorial Board of the series.

Prof. Audris Kalnins,
Deputy Editor-in-Chief,
University of Latvia

LATVIJAS UNIVERSITĀTES RAKSTI. 2004. 669. sēj.: DATORZINĀTNE UN INFORMĀCIJAS TEHNOLOĢIJAS. 7.–12. lpp.

Software Development Effort Estimation

Baiba Apine
PricewaterhouseCoopers
baiba.apine@lv.pwc.com

Formal analytical and analogy-based software development cost estimation models are analysed to find out what factors are considered to have an influence on the development process in each of the models, and to provide recommendations for the application of those models.
Key words: estimation, software development cost, effor estimation models.

Introduction

Software development effort estimation is a rather old, but still relevant, problem. Since the middle of the last century, when the very first formal software development effort and schedule estimation models appeared, the software development process, the subject of these estimates, has changed dramatically. For instance, [EXP02] highlights that productivity of the software development process increases by 10% annually due to the progress of software development technologies. The continuous improvement of the development process, and different factors which influence the process productivity and must be taken into account, create a nightmare for estimators.

The following experiment was carried out during classes on software cost estimation. A group of 200 students was asked to estimate the effort and schedule for the development of information system (IS) storing data about software development projects (name, development environment, start date, expected end date etc.), developers and customers (names, skills, office hours, phone numbers etc.) and documents produced during the project lifecycle (name, comments, author). All the students had the same input information about the IS. The results for this rather small project differed significantly: starting from 10 man-days (2 weeks schedule) to 12 man-months (1 year schedule) with an average of 3 man-months (schedule 3.5 months). The students themselves were very surprised about the differences in estimates. If this is acceptable with students in the classroom, in real-life such a situation might be rather painful and must be discussed in order to solve the problem and find consensus. Here formal cost estimation methods could help:

1. To provide a disciplined way of thinking during the estimation process, which was not available for students.
2. To provide a subject, subjects for discussions for all involved parties.

Formal software development cost estimation models are analysed to find out what factors are considered to have an influence on the development process in each of the models, and to provide recommendations for the application of those models.

Software Development Effort Estimation Models

To be honest, not always are the formal cost estimation models recommended for effort and schedule estimation. [COC2], [KEM93] do not recommend formal effort

estimation models for small projects (less than 2K lines of code [BOE91] or less than 10 developers work for 3-6 months [YOU97]). For small projects the development productivity depends on the individuals very much, hence the right way of estimating is asking the developer for an estimate.

Formal methods are welcome for average (20-30 developers' work for 1-2 years) and large (100-300 employees' work for 3-5 years) projects, because:

1. Individual experience of the estimator is limited, as even a very experienced project manager has worked for 5-7 average and 3-5 large projects during his/her career.
2. Typically there are more than two stakeholders in average and large projects, therefore it is very crucial to have a documented, step by step estimation process as a subject for discussions.

Different analytical and analogy-based software cost estimation models are developed. Analytical models are based on a negative exponential curve called the Rayleigh-Norden manpower approximation curve (see Figure 1). Functionality of the software product, usualy expressed in function points or lines of code, is used as an input for analytical models.

Manpower distribution for one real-life software development project is shown in Figure 1. This illustrates a rather typical fault in project management – as the development process seems to be stable (see SEP_98 to APR_99), some developers might be taken away from the project (see MAY_99). This yields an increasing response time to customer queries and lowers customer satisfaction. Additional developers must be added in the project urgently (see JUN_99). Otherwise the real new software development life cycle approximates the theoretical one quite closely.

Analogy based models use information about past projects. The estimation process basically means database browsing to find similar projects. The success of estimation using analogy-based models depends on how successfully the criteria for finding similar projects from past experience are found.

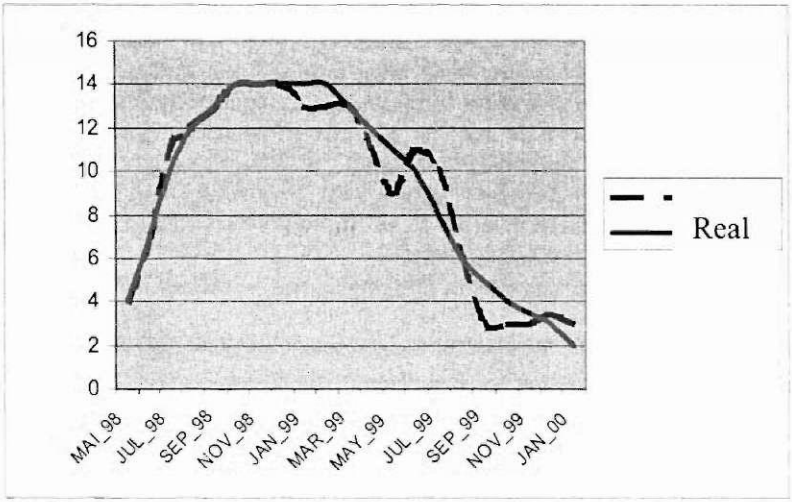

Figure 1. Rayleigh-Noren manpower approximation curve for new software development process (optimal) in comparison with the real one

Samples of Analytical Models

SLIM (Software Lifecycle Model) was developed in the middle of the 1970's [KEM87] and currently is maintained by QSM (Quantitative Software Management) [QSM02]. The Model was developed using data about 5000 software development projects. The SLIM model uses a productivity parameter and productivity index, which characterise the development environment (tools used, developers' skills etc.). In comparison to other analytical models, in the SLIM model a schedule must be set before the effort is estimated. However this is not a disadvantage of the model, because in real life very often the schedule is already predefined.

The Jensen model [JEN84] estimates the development effort and schedule based on the size of the product, technology index and 11 development environment factors, which characterize the development technology. A Current version of the Jensen model is implemented in the tool SAGE [JEN95].

COCOMO II [COC2] estimates the development effort and schedule based on the size of the product, 5 scaling drivers and 6 adjustment factors, which describe the product's quality and reliability requirements, technologies used, developers' skills etc.

At the end of last year a survey about the most popular development effort estimation method was carried out by [ISB01] in more than 100 software development organizations of varied sizes [CUT03]. More than a quarter of the respondents (27%) use various analytical models. A surprisingly low number (9%) report the use of COCOMO, which was considered by many to be a groundbreaking technique in the 1980s. One possible explanation for this is that the COCOMO applications are often cumbersome and require the use of other tools and techniques to generate input to COCOMO, such as lines of code estimators.

Samples of Analogy Based Models

The Checkpoint model is based on information about 8000 software development projects. This model is property of SPR (Software Productivity Research) [JON96]. The Checkpoint model uses a survey containing multiple-choice questions about the development process and the product must be developed. Information from the survey then is used to adjust the productivity of the development process.

The same principle as for the Checkpoint model is used in the ExperiencePro model and tool, which is property of Software Technology the Transfer [EXP02]. This model uses information about approximately 500 past projects. All the users of this model are joined in FiSMA [FIS03] and encouraged to collect information about projects to update the ExperiencePro model continously.

According to the survey by [ISB01], most organizations (65%) collect historical data and use it to help them estimate software development. 26% report that they do so diligently, by maintaining a central database that formally receives and stores data from every project.

Conclusions

Analytical models as well as analogy based ones use data about the software development process (measurements), consider different risks and their influence on the development process. A Set of risks is predefined and depends on the model. Table 1 shows software development risks found out in the survey of risk management [API02]. There are risks which are considered in each cost estimation model, for instance, an unrealistic project schedule and budget or lack of knowledge in software development technologies and environment. At the same time there are risks which are important and influence the development schedule and effort, but are not considered by any of the estimation models analysed. For instance, software development environment bugs or change of qualified personnel.

Table 1.

Risks considered in different software cost estimation models (+ - the model considers that)

Software development influencing factor	SLIM	Jensen	Checkpoint	ExperiencePro	COCOMO II
Lack of hardware on the customer side		+			
Difficult communication with customer	+			+	
Low quality of software requirements	+		+	+	+
Unstable software requirements	+		+	+	
Unrealistic schedules and budgets	+	+	+	+	+
Weak project management			+	+	
Software development environment bugs					
Lack of developer motivation				+	
Lack of hardware on developers side	+	+	+		+
Change of qualified personnel					
Lack of knowledge in software development technologies and environment	+	+	+	+	+

Models concentrate on effort and schedule estimation for new software development [PUT92], [JEN84], [JON96], [COC2], [EXP02], [PAR95], [TAU81]. There are add-ons developed for some models to apply them to more specific projects like software maintenance [COC2], [EXP02]. Taking into account that models rely on development process measurements, they are becoming out-of-date continuously as technologies develop. For instance, software development productivity increases by 10% annually. Hence using a year old cost estimation model, it overestimates the effort by 10% even if we exactly how know to use the model. To solve this problem, authors continuously gather data about projects to maintain their models [QSM02], [JON96]. This problem could not be solved for analytical models, but for analogy based ones it is not so actual if the project base is updated with information about new projects continuously, like it is done for [EXP02].

There are add-ons for development process operational planning developed for some of the models, which are based on the Rayleigh-Norden manpower curve [PUT92]. There are no such add-ons for analogy based models, because data gathering for such models is much more complex than approximation of the development process using analytical methods. The development of such add-on requires integration of the measurement and risk management processes.

What to do?

To estimate the software product development effort and schedule the following steps must be carried out.

1. Perform a risk analysis for the particular project to find out what factors will influence the development of the product.
2. Choose the development effort estimation model which takes into account all the risks pertaining to the particular project.

Such an approach is not feasible due to the following reasons:

1. There is no knowledge available about different models. Experience shows that most companies have knowledge about one or two effort estimation models.
2. Application of most of the models requires acquisition of specific, rather expensive tools (several thousand EUR for licence and approximately a thousand EUR for annual maintenance of the model).

Therefore one formal model must be chosen for usage in the company. There are some sources which reccomend usage of two formal models for estimation in parallel [ISB01], [JON96]. The second model is to verify the results given by the first one. Instead of the second formal model, development process measurements could be applied to adjust the formal results. Developers and managers must be trained to use the chosen formal model to avoid painful estimation mistakes.

The development process must be measured in order to use the gathered information for analysis of the results obtained from application of the formal model. Results must be communicated among the developers.

References

[EXP02] "Estimation and Measurement Methodology of Experience Pro 3.0". [Electronical resource],-Access: WWW, URL: http://www.sttf.fi/html.-Resource described 12th of June, 2002.

[COC2] C.Abts, B.Boehm, B.Clark, S.Devnani-Chulani. "COCOMO II Model Definition Manual". University of Southern California, 2000, 68 p.

[KEM93] Chris F. Kemerer. "Empirical studies of assumptions that underlie software cost estimation". Information and Softw. Technol., Vol.34 #4, 1992, pp. 211-218.

[BOE91] Barry W. Boehm. "Software Risk Management: Principles and Practices". IEEE Software, January, 1991, pp. 32-41.

[YOU97] Yourdon, E. "Death March. The complete Software Developer's Guide to Surviving 'Mission Impossible' Projects". Prentice Hall, 1997, 218 p.

[KEM87] Chris F. Kemerer. "An Empirical Validation of Software Cost Estimation". ACM Communication, May, 1987, pp. 416-429.

[QSM02] "SLIM-Estimate 5.0". [Electronical resource].-Access: WWW. URL: http://www.qsm.com/slim_estimate.html.-Resource described 3rd of January, 2003.

[JEN84] Randall W.Jensen. "A Comparision of the Jensen and COCOMO Schedule and Cost Estimation Model". Proceedings of the International Society of Parametric Analysis, 1984, pp. 96-106.

[JEN95] Randall W.Jensen. "A New Perspective in Software Schedule and Cost Estimation". [Electronical resource].-Access: WWW. URL: http://www.seisage.com/publications.htm .-Resource described: 17th of February, 2003.

[ISB01] International Software Benchmarking Standards Group. "Practical Project Estimation". 2001, 46 p.

[CUT03] "The Guru Method Prevails". [Electronical resource].-Access: WWW. URL: http://www.cutter.com/benchmark/index.html.-Resource described 12th of February, 2003.

[JON96] Caper Jones. "Applied Software Metries". McGraw Hill, 1996, 324 p.

[FIS03]"Finnish Software Metrics Association". [Electronical resource].-Access: WWW. URL: http://www.fisma-network.org.-Resource described 12th of February, 2003.

[API02] Baiba Apine. "Software Development Risk Management Survey". accepted paper for 11th International Conference on Information Systems Development, Riga, Latvia, September 12 - 14, 2002, http://www.cs.rtu.lv/isd2002/accepted.asp.-Resource described:2003.g. 3.jan.

[PUT92] Lawrence H. Putnam, Ware Myers. "Measures for Excellence". Yourdon Press Computing Series, 1992.

[PAR95] Naval Sea Systems Command . "Parametric Cost Estimating Handbook". [Electronical resource].-Access: WWW. URL: http://www.jsc.nasa.gov/bu2/PCEHHTML/pceh.htm.-Resource described: 2003. g. 17.feb.

[TAU81] Robert C. Tausworthe. "Deep Space Network Software Cost Estimation Model". Jet Propulsion Laboratory Publication 81-7, Pasadena, C.A., April, 1981.

LATVIJAS UNIVERSITĀTES RAKSTI. 2004. 669. sēj.: DATORZINĀTNE UN INFORMĀCIJAS TEHNOLOĢIJAS. 13.–33. lpp.

Description of Semantics and Code Generation Possibilities for a Multi-Language Interpreter

Guntis Arnicans
Faculty of Physics and Mathematics, University of Latvia
Raina Blvd. 19, Riga LV-1586, Latvia, garnican@lanet.lv

In this paper we describe the definition of semantics for a Multi-Language interpreter (MLI), which provides the execution of the given program, receiving and exploiting corresponding language syntax and the desired semantics. We analyze the simplest solution – the MLI receives the language syntax and the semantics descriptions, which have already been compiled to executable objects. Semantics is defined as a composition from several semantic aspects, considering the pragmatics of a language. Semantic aspects are translated to semantic functions by composing descriptions of the aspects. A traversing program's intermediate representation and the calling out of semantic functions similarly to the principle of the Visitor pattern perform the desired semantics. To simplify the semantic descriptions, we use abstract components that are joined by connectors at the meta-level. The implementation of these components and connectors can be very different. Examples of conventional and specific semantics are given for the simple imperative language in this paper.

Key words: interpreter, programming language specifications, tool generation.

1. Introduction

The number of new languages that are related to the IT sector has increased rapidly over the last several years (programming languages and data description languages, for example). Problems associated with the implementation and use of these languages has also expanded, of course. Kinnersley [Kin95] has reported that there were 2,000 languages in 1995, which were being put to serious use. Even back then specialists found that the new languages were mostly to be classified as domain-specific languages. Most of them are not easy to implement and maintain [ITSE99, DKV00 (DSL analysis, problems and an annotated bibliography)]. It is also true that we need not just a compiler or an interpreter, but also a number of supportive tools. Questions of programming quality are very important today, and these questions often cannot be answered without specialized and automated ancillary resources.

Computers are being used with increasing dynamism today: systems have been divided up in terms of time and space, the operational environment is heterogeneous, and we have to ensure the implementation of parallel processes while organizing cooperation among components and systems, adapting to changing circumstances without interrupting our work, etc. We are making increasing use of interpreters or of code generation and compilation just in time. The formal resources that are used to describe the semantics of a language, however, cannot fully satisfy our needs in the modern age, and they are starting to lose their positions [Sch97, Lou97, Paa95].

The basic problem that is associated with the formal specifications of programming languages is that these specifications are far too complex. It is not clear how they are administered, we cannot use them to explain all of our practical needs, and in the end we are still faced with a problem – who can prove that these complex specifications are really correct? The literature claims that the best commercial compilers (interpreters or other language-based tools) are written without formalism or are used only in the first

phases – scanning and parsing [e.g. Lou97]. Formalisms are elaborated and used mostly for research purposes in educational and scientific institutions at this time.

The development of semantics is gradually moving away from the development of languages and tools. One way to overcome this gap is to take a tool-oriented approach to semantics, making the definitions of semantics far more useful and productive in practice and generating as many language-based tools as possible from them [HK00]. We support this approach in principle, but our aim is to propose a different approach toward the definition of semantics, making room for far less formal records.

Those who prepared descriptions of semantics in the past have long since been looking for ways in which semantics can be divided up into reusable components, and it is not yet clear whether the formal or the partly formal methodology is the best in this case. We chose a less formal and more free form of description keeping from the theoretical perspective, and our empirical research showed that rank-and-file developers of tools understand this method far more easily.

2. The concept of a Multi-Language Interpreter

The concept of a multi-language interpreter was introduced in [AAB96]. A Multi-Language Interpreter (MLI) is a program which receives source language syntax, source language semantics and a program written in the source language, then performs the operations on the basis of the program and the relevant semantics. Conceptually, we parse an input token stream, build a parse tree and then traverse the tree as needed so as to evaluate the semantic functions that are associated with the parse tree nodes. Once an explicit parse tree is available, we visit the nodes in some order and call out an appropriate function. This approach is similar to the principle build a tree, save a parse and traverse it [Cla99] and to a Visitor pattern [GHJV95], except in terms of the methodology which we apply in obtaining semantic functions and organizing physical implementation. The idea of MLI is expressed in Figure 1.

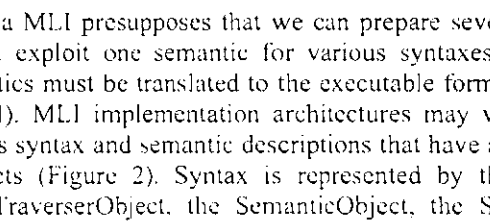

Figure 1. The concept of a Multi-Language Interpreter

The concept of a MLI presupposes that we can prepare several semantics for one syntax, and we can exploit one semantic for various syntaxes. The descriptions of syntaxes and semantics must be translated to the executable form (before or during the running of the MLI). MLI implementation architectures may vary. The one we use receives and exploits syntax and semantic descriptions that have already been compiled as executable objects (Figure 2). Syntax is represented by the SyntaxObject, and semantics by the TraverserObject, the SemanticObject, the SymbolTable, and the necessary volume of the Component (the components A, B, C in our figure). The MLI Kernel, which provides the initial bonding of all syntax and semantics objects, initializes the execution of the program.

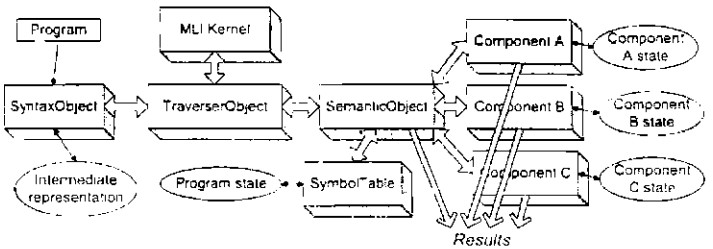

Figure 2. MLI runtime architecture

Each of the components can be implemented in various ways – with a different semantic assignment and physical implementation. Here we have a chance to combine syntaxes and semantics in both ways – in terms of architecture and in terms of implementation. Then, however, we immediately face the question of the compatibility of the syntax and semantics so as to avoid senseless interpretation.

The obtaining of an executable syntax and of semantic objects from their descriptions can be done before or during the actual program execution (analogue to a classical compiler and interpreter). Dynamic code generation is more difficult because all generation phases must be done automatically.

3. Language Specifications for MLI

Programming language is an artificial means to communicate with a computer and to fix the algorithms for problem solving. Like a natural language, a programming language's definition consists of three components or aspects: syntax, semantics and pragmatics [Pag81, SK95]. All of these aspects are significant in dealing with our problems. Usually exploited rarely, pragmatics deals with the practical use of a language, and this is an important element in defining semantics.

We can look at syntax and semantics from two perspectives – the definition or description phase and the runtime phase. Our goal is to achieve runtime components which can freely be exchanged or mixed together in pursuit of the desired collaboration. First we must look at the principles of syntax and semantics descriptions, and then we can view the target code generation steps.

Our basic principle is to divide syntax and semantics into small parts, and later, with a simple method, to combine these parts thus providing a mechanism to tie together the semantic parts and the syntax elements. Our method is close to some of the structuring paradigms of attribute grammars [Paa95]: The definition phase is similar to the relationship Semantic aspect = Module, but the runtime phase is similar to Nonterminal = Procedure. That means that we basically use the language pragmatics and divide the semantics into semantic aspects.

3.1. Syntax

The formalisms for dealing with the syntax aspect of a programming language are well developed. The theory of scanning, parsing and attribute analysis provides not only the means to perform syntactical analysis, but also a way to generate a whole compiler as well. Such terms, concepts or tools as finite automata, regular expression, context-free grammar, attribute grammar (AG), Backus-Naur form (BNF), extended BNF

(EBNF), Lex (also Flex), Yacc (also Bison), and PCCTS are well known and accepted by the computer science community.

We do not need to reinvent the wheel and it is reasonable to choose the existing formalisms and generators (lexers and parsers). The main task when dealing with syntax description for a given language is code generating which can transform the written program, which uses the syntax, into intermediate representation (IR). Additionally, we need to attach a library with functions, which provide the means to manipulate with the IR and to compile the whole code. The result is the SyntaxObject (Figure 2).

In this paper we concentrate mostly on the class of imperative programming languages, but our method is adaptable for other languages too, such as diagrammatic languages (e.g., Petri nets, E-R diagrams, Statecharts, VPL – visual programming languages, etc.), which exploit other formalisms (e.g., SR Grammars, Reserved Graph Grammar) and processing styles [FNT+97, ZZ97].

3.2. Semantics

The chosen principle for the runtime semantics parse and traverse states that the most important things are a traversing strategy and the semantic functions which must be executed when visiting a node (*Figure 3*). Therefore, the central components of the semantics are TraversalObject and SemanticObject (*Figure 2*).

The TraversalObject manages the node visiting order, provides semantic functions with information from the IR, and is the main engine of the MLI. The SemanticObject, for its part, contains all of the necessary semantic functions and provides for the execution environment. At the same time, we can also put into the semantic functions certain commands which force the Traverser to search for the needed node and to change the current execution point in the IR (traversing strategy changes and a transition to another node are problems in the Visitor pattern [e.g. Vis01]).

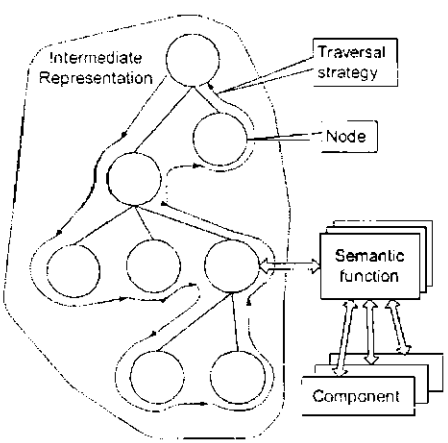

Figure 3. Runtime correspondence between syntax and semantics

Semantic functions have to be as simple and as small as possible. This can be achieved by using a meta-language and by employing high-level expression means, which allow for easy understanding and verification of the description. Following this

principle becomes more natural if we use abstract components so that the underlying semantic can be clear without additional explanations (in Figure 3, the abstract components already have a concrete implementation component – A, B and C). This statement may lead to objections from the advocates of formal semantics, because the components are not described with mathematic precision. At the same time, however, formal semantics sometimes use such concepts as Stack or Symbol table.

Let us introduce a conceptual syntax element, which is a grammar symbol with a name (e.g., a named nonterminal symbol or a named terminal symbol). Considering the various types of syntax elements and the traversing strategy, we separate various visitations and introduce the concept of the traversing aspect. For instance, we can distinguish the arriving into node from the parent node (PreVisit) as well the arriving into node from the child node (PostVisit). Thus we create the semantic functions and name them not only on the basis of the name of the syntax element but also on the basis of the arriving aspect (traversing aspect) into this element (Figure 3).

Runtime semantics or simply semantics for multi-language interpreters are a set of semantic functions. We represent the runtime semantic in Table 1. There is an executable code (⊏) or nothing (λ) for the syntax element, according to the traversing aspect. n depends on the size of syntax (e.g., the count of all nonterminal and terminal symbols), and m depends on the complexity of the traversing strategy (usually 1..3). We notice that the matrix mainly consists of empty functions (λ).

Table 1.

The matrix of syntax elements and semantic function correspondence

Syntax Element (SE)	Traversing Aspect (TA)				
	T A_1	T A_2	T A_3	.	T A_m
SE_1	⊏	⊏	λ	.	λ
SE_2	λ	λ	⊏	.	λ
SE_3	λ	λ	λ	.	⊏
...	...	...	...	.	λ
SE_n	⊏	λ	λ	.	λ

The identification of semantic functions is realized both by the syntax name and by the traversing aspect name. Technical implementation may differ, but it is very advisable that functions identification and calling be performed with constant complexity O(1).

Now we arrive at the most difficult and important problem – how can we obtain semantic functions and ensure correct collaboration between them, and how is it possible to create reusable semantic descriptions? Let us explain our ideas about how to define semantics and how to gain the matrix observed above, i.e., how to generate executable semantics from the semantic description.

4. Semantic Aspects and Abstract Components

4.1. Semantic aspects

In practice, programming languages are frequently presented through the pragmatics of the programming language, i.e., examples are used to show how the language constructs are exploited and what their underlying meaning is. Let us call these language constructs and their meaning like semantic aspects.

We have chosen to define the semantic as a set of mutually connected semantic aspects. Here are some examples for typical groups of semantic aspects for imperative programming languages: execution of commands or statements (e.g., basic operations, variable declaring, assigning of a value to the variable, execution of arithmetic expressions), program control flow management (e.g., loop with a counter, conditional loop, conditional branching), dealing with symbols (e.g., variables, constants), environment management (e.g., the scopes of visibility). Here, too, are examples of nontraditional semantic aspects: attractive printing of the program, dynamic accounting of statistics, symbolic execution, specific program instrumentation, etc.

We have chosen an operational approach to describe the semantic aspect – we define the computations, which a computer has to do to perform the semantic action.

4.2. Abstract data types and abstract components

The next significant principle to define the semantic aspect is using abstract data types (ADT) as much as possible. ADT is a collection of data type and value definitions and operations on those definitions, which behave as a primitive data type. This software design approach breaks down the problem into components by identifying the public interface and the private implementation.

In our case, typical examples of ADT are Stack, Queue, Dictionary, and Symbol table (in compiler construction theory [ASU86, FL88], in formal semantics [SK95]). In this way we hide most of the implementation details and concentrate mainly on the logic of the semantic aspect. Later we can choose the best implementation of ADT for the given task. Seeing that some exploited components can be complicated (E-mail, Graph visualization, Distributed communication, Transaction manager, etc.) and have no standards, we use another term – abstract component. Sometimes we want to utilize an already existing component, and the term abstract component seems more appropriate to us.

It is advisable to describe the semantic aspect through meta-language, even if one does not have a translator for this. Then one can translate or simply rewrite it by hand to the target programming language, select appropriate implementation for the abstract components, and use the needed interface, collaborating protocol and execution environment. For instance, Stack can be implemented in a contiguous memory or in a linked memory, Symbol table – as a list or as a dictionary with the hashing technique. Furthermore, instances of abstract components can be viewed as distributed objects in a heterogeneous computing network.

4.3. Examples of abstract components

Some abstract components and their operations are very popular, e.g., Stack (createStack, push, pop, top, etc.), Queue (createQueue, enqueue, dequeue, first, etc.), while some are guessed, e.g., E-mail (prepare, send, receive, open). Among the many specific components we would like to emphasize one that is useful for most of semantics - Symbol table (SymbolTable in Figure 2) or its analogue to provide the execution environment.

While building prototypes of the MLI, we have created an implementation of Symbol table – MOMS (Memory Object Management System) - that is appropriate for implementing the imperative programming languages. It is possible to define basic and user defined data types, to define base operations and functions, to operate with variables and their values, to manage the scope of visibility of all objects, etc. The most important data types, concepts, and operations of MOMS are listed in the appendix to this paper so as to give the reader a better idea about MOMS.

The second important component is Traverser (TraverserObject in Figure 2). Its main task is realizing the traversing strategy, to change the current execution point and to organize cooperation with the syntax object.

There is a depth-first left-to-right traversing strategy, which is used in the following examples (Table 2). This strategy has three visiting aspects: Visit (for tree leaves - terminals), PreVisit and PostVisit (for the other tree nodes - nonterminals). To define semantic functions for examples, we have used the following operations: NodeValue() returns a value for the current terminal or nonterminal symbol (value from the current IR node), and both goSiblForw(aName) and goSiblBackw(aName) provide for a changing of the current node, searching the node with the name aName between siblings going forward or backward.

Table 2.

A depth-first left-to-right traversing strategy

```
Traverse(node P)
  if IsLeaf(P)
    Visit(P)
  PreVisit(P)
  for each child Q of P,
  in order, do
    Traverse(Q)
  PostVisit(P)
```

It is possible to describe interfaces for SyntaxObject, TraverserObject and SemanticObject with domain-specific language. Then interfaces for obtaining the IR, manipulating with it and working with the symbol table can be compiled together, and it is possible to engage in high-level optimization and verification [Eng99].

The Traversing strategy can also be described with domain-specific language. This is important if the strategy is not trivial and depends on syntax elements and the program state [OW99 (traversing problems and solutions for Visitor pattern)]. The traversal strategy should be independent from syntax as much as possible and organized

(combined) by patterns [Vis01]. In addition to common traversing strategies there are also less traditional ones, e.g., the strategy for reverse execution of the program [BM99]

4.4. Defining the semantic aspect

It is more convenient to define the semantic aspect by using diagrams (as in Figure 7). We can write a meta-program or a program in the target language in textual form, too. Diagrams contain syntax elements that are important for the semantic aspect and are visualized with graphic symbols. We can use different graphic notations. If the visiting order of syntax elements is important, then we mark the order with arrows.

Let us call the operations that are performed during the aspect node visiting semantic action. Semantic action is similar to semantic function, but it is written at the meta-level and relates only to a given semantic aspect. Semantic action is shown as a box with the meta-code connected to the syntax element and takes into account the traversing aspect.

There are all kinds of abstract data types that are needed for the semantic aspect into the box with the key words IMPORT GLOBAL. For better perceptibility of the semantic aspect, it is permissible to use additional graphic symbols that are not needed in real execution. For instance, we use Other aspects to signal that we expect there to be a composition with the other semantic aspects.

4.5. Examples of semantic aspects

Let us look at some examples of semantic aspects (*Figure 4* - *Figure 8*) that are applicable for the simple imperative programming language Pam [Pag81]. Terminal symbols are denoted by a rectangle, while nonterminal symbols are indicated by rounded rectangles. The left circle in the nonterminals corresponds to the PreVisit semantic action, the right one – to the PostVisit semantic action, while for the terminals, the Visit semantic action is assigned.

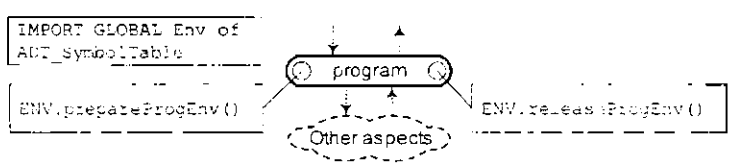

Figure 4. The semantic aspect PROGRAM. It prepares the program environment to manage variables, constants, etc. and operations involving them. The environment is destroyed at the end

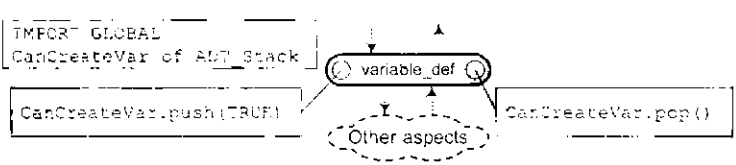

Figure 5. The semantic aspect VARIABLE DEFINITION. It allows for variable creation

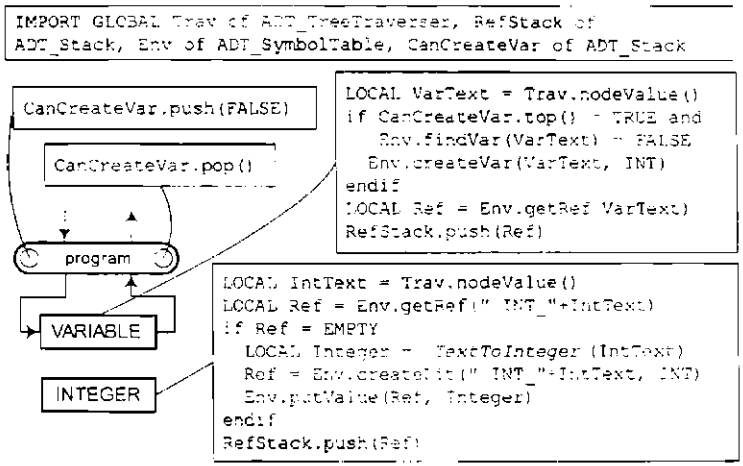

Figure 6. The semantic aspect ELEMENT. It provides for the pushing into the stack all references to each variable encountered while traversing. Variable creation is forbidden by default. The Trav provides for getting the values of the current terminal node in IR.

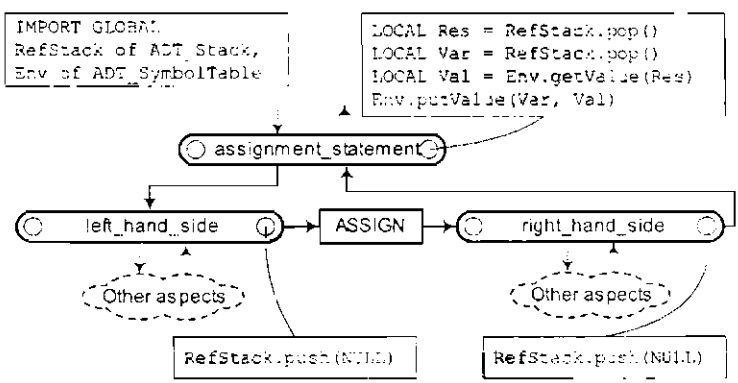

Figure 7. The semantic aspect ASSIGNMENT. It takes reference to the variable and reference to the value from the stack and assigns a value to the variable. Pushing of references is simulated

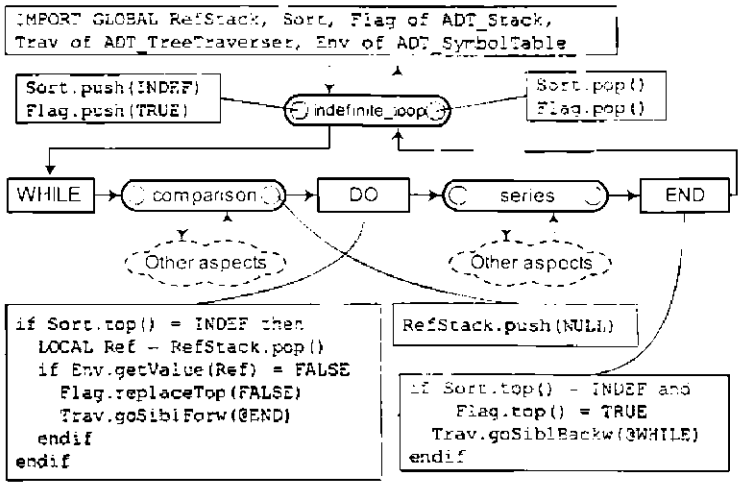

Figure 8. The semantic aspect INDEFINITE LOOP. It "goes through" the series and back to WHILE until the comparison sets a NULL reference or a reference with the value FALSE

5. Meta-semantics

Meta-semantics is a term which relates to a meta-program that describes the counterparts of which semantics consists and the way in which these counterparts are connected together. The conceptual scheme of meta-semantics is shown in Figure 9.

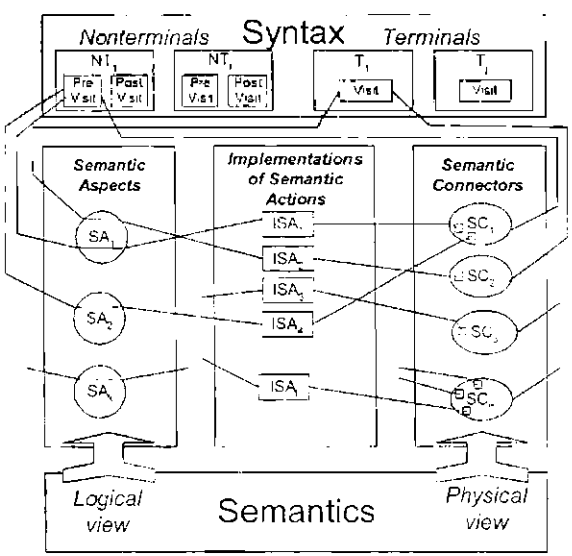

Figure 9. The conceptual scheme of meta-semantics. For instance, SA1 involves nonterminal NT1 and terminal T1, and SC1 is performed while previsiting. SC1 is a composition of ISA1 and ISA4

If we look at semantics from the definition side (the logical view) then semantics is formed by traversing strategy and by a set of semantic aspects that consist of semantic actions realized while visiting the appropriate node of the intermediate representation. If we look at semantics from the runtime side (the physical view) then while visiting one node it is possible that several semantic actions have to be realized, and we have to ensure correct collaboration between all involved instances of abstract components into the desired environment. How can we put all of this together correctly?

To solve this problem we introduce the concept of the semantic connector. This concept has been adapted from the concept Grey-box connector, which solves a similar problem: how to connect the pre-built components in a distributed and heterogeneous environment for collaborating work [AGB00]. The Grey-box connector is a meta-program that introduces a concrete communications connection into a set of components, i.e., it generates the adaptation and communications glue code for a specific connection.

6. From semantic aspects to meta-semantics and executable semantics

The obtaining of semantics for the fixed syntax is achieved in several steps: 1) select predefined semantic aspects or define new ones for the desired semantics, 2) rename syntax elements and traversing aspect in the selected semantic aspects with names from fixed syntax and traversing strategy, 3) rename instances of abstract components to organize collaboration between semantic aspects, 4) make composition from semantic aspects, 5) specify the runtime environment and translate the meta-code to the code of the target programming language, and 6) compile the semantics.

- Selection of semantic aspects (step 1)

If we have a library with previously created semantic aspects, then we can search for appropriate ones, i.e. reuse some parts of the semantics. The traversing strategy also has to be considered.

- Syntax element and traversing aspect renaming in semantic aspects (step 2)

Actually this is semantic action mapping. Matching to the fixed syntax elements is achieved by mapping syntax elements of semantic aspects to fixed syntax elements (rename with – simple mapping and duplicate to – mapping of a semantic aspect syntax element to several fixed syntax elements):

rename <name> <traversing aspect> with <target name > <target traversing aspect>

duplicate <name> <traversing aspect> to

<target name> <target visiting aspect> [,<target name> <target visiting aspect> ...]

For instance, rename left hand side PostVisit with VARIABLE Visit ;

duplicate COUNTABLENODE Visit to VARIABLE Visit, assignment_statement PostVisit

- Renaming of instances of abstract components (step 3)

Matching of components is necessary because there is no direct data exchange between semantic functions, and we need collaborative work. The program state is fixed by using the runtime states of components. At first we decide what instances they have

in common and what names they have to get, and then we rename instances in the semantic aspects:

replace <name> with <target name>

For instance, replace RefStack with DataStack ; replace Sort with LoopSortStack

- Composition of semantic aspects (step 4)

The goal of a semantic aspect composition is to bring together several semantic aspects into one more complicated aspect that nearly describes entire semantics. While composing, we stick together the meta-code of semantic actions that have the same name. The sticking principles can vary, for instance, sequential (one code is appended to the other one), parallel (codes can be executed simultaneously or sequentially in any order), free (the user can modify the code union as he likes). To achieve better results, we ignore some semantic actions or apply the sticking principle to the semantic aspects in the reverse order. At this time we have to be aware of conflicts between local variable names.

compose aspect <<new SA>> (<refined SA>) [[append | parallel, free] (<refined SA>) ...], where

<refined SA> <<old SA>> [ignore <name> <trav aspect> [,<name> <trav aspect>]] | [reverse]

The result is meta-semantics. An example of a meta-code fragment for meta-semantics is given in Table 3.

Table 3.

An example of meta-semantics

```
compatible with
  ir type ParseTree
  traverser type ParseTreeTraverser

syntax elements (program, expression, VARIABLE, ...)
semantic       actions       (<PROGRAM>       program       PreVisit
{ENV.prepareProgEnv()},
                  <PROGRAM> program PostVisit {...}, ...)

global Trav of ADT_TreeTraverser
global Env of ADT_SymbolTable
create DataStack, OperatorStack, CanCreateVar, LoopSortStack,
    LoopCounterStack, LoopFlagStack, IfFlagStack of ADT Stack
create InputFile, OutputFile of ADT_FILE

compose aspect <A1>    // composes semantic aspects from aspects given above
  (<PROGRAM>)          // semantic aspects PROGRAM remains the same
append (<ELEMENT>
  replace RefStack with DataStack // replaces stack for collaborating work
  rename INTEGER Visit with CONSTANT Visit)   // renames nonterminal according
to PAM syntax
append (<ASSIGNMENT>
  replace RefStack with DataStack
  rename left_hand_side PostVisit with VARIABLE Visit,
         right_hand_side PostVisit with expression PostVisit
  ignore left_hand_side PostVisit)      // ignore pushing of NULL reference
append (<INDEFINITE LOOP>
  replace RefStack with DataStack,
    Sort with LoopSortStack, Flag with LoopFlagStack)
append (<VARIABLE DEFINITION>
  rename variable_def PreVisit with assign statement PreVisit,
    variable_def PostVisit with ASSIGN Visit)
end compose aspect

compose aspect <A2>
(<A1>
  ignore expression PostVisit, comparision PostVisit)
  append ( ...
  /* Others aspect are appended such as <TYPE AND OPERATOR>,
<INPUT>, <OUTPUT>, <BASE BYNARY OPERATION>, <DEFINITE LOOP>,
<CONDITIONAL STATEMENT> */
  ...)
end compose aspect
```

- Meta-code translating (step 5)

Meta-code is translated to the target programming language, taking into account the target language (e.g. C++), the implementation of abstract components (e.g. Stack in linked memory), the operating system (e.g. Unix), the communications between components (e.g. CORBA), MLI components type (e.g. DLL), etc. The translation may be done by hand or automatically (desirable in common cases).

- Obtaining semantic objects (step 6)

By compiling the code we get executable objects that provide semantic performance, i.e. they contain the semantic functions that are called while traversing the program intermediate representation. The instances of the concrete implementation of abstract components are created, or the existing ones are dynamically linked via the selected communications protocols.

7. Examples of alternative semantic aspects

In this section we provide a short insight on how we can build nontraditional semantics. By adding new features to existing semantics we can create a specific tool that works with a given programming language. We would like briefly to survey two examples: 1) statistics accounting of program point visiting, and 2) storing of symbolic values for variables. Both aspects are added to the conventional semantics, and this is program instrumentation if we speak in terms of software testing.

7.2. Accounting of program point visiting

Our goal is to account for any visiting of a desirable program point. This means that we need to set counters at these points. At first we write the semantic aspect NODE COUNTER (*Figure 10*). We use the abstract component Dictionary where we can store, read and update records in form <key, value>.

```
IMPORT GLOBAL Trav of ADT_TreeTraverser,
Dict of ADT_Dictionary

        COUNTABLENODE

LOCAL key = Trav.getNodeID()
LOCAL record = Dict.getRecNum(key)
if record = 0
   Dict.createRec(key, 1)
else
   Dict.update(record, Dict.get(record) + 1)
endif
```

Figure 10. The semantic aspect NODE COUNTER

Defining meta-semantics, we add this semantic aspect to the others. For instance, we define accounting for the use of any variable and assignment operation:

duplicate countable_node Visit to VARIABLE Visit, assignment statement PostVisit

Another semantic aspect can be built which accounts for every concrete variable using statistics into an additional dictionary (the variable name serves as the key). We can improve this aspect further by accounting for an aspect of variable use - defined, modified, referenced, released, etc. In the program analysis and instrumentation area, our approach is similar to the Wyong system (based on the Eli compiler generation system and the ATOM program instrumentation system), because specific operations are attached to syntax elements, and in this way we obtain a specific tool with additional semantics [Slo97].

7.2. Storing of symbolic values for variables

The second example provides for the fixing of symbolic values for variables (*Figure 11*). To do this task in an effective way, we have additional operations in our MOMS (symbol table). The operation createSymbValue creates an entry for symbolic value, and with the operations addTextToSymbValue and addVarToSymbValue, we form the value while traversing all nodes in the desired subtree (we store all needed program symbols and symbolic values of variables). At the end we store accumulated value with the operation storeSymbValue.

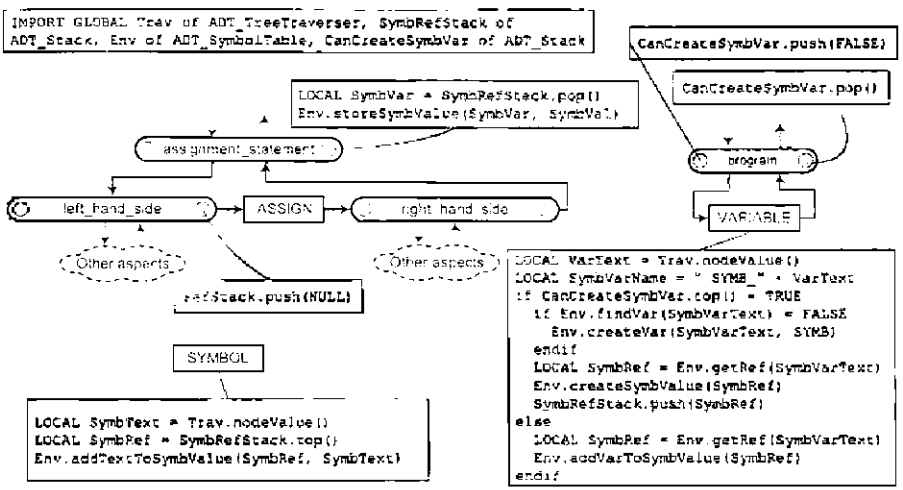

Figure 11. The semantic aspect SYMBOLIC VALUES

8. Conclusions

This method was developed with the goal of reducing the gap between practitioners (tool developers) and theoreticians (developers of formal specifications for semantics). Our experience shows that a significant achievement is attained if abstract components or abstract data types realize the greater part of semantics, because that way it is easier to perceive the full implementation of semantics.

The second acquisition of our method involves significant disjoining of syntax and semantics from each other. It allows us to combine various syntaxes and semantics and to find out the most desirable semantics for the given syntax. So, if we have written syntax for a new language, we can match several semantics to it in a comparatively short time. As a result, we can develop a wide spectrum of tools in support of our new language.

Our approach allows us to change semantics dynamically while the interpreter is running, i.e. replace semantics or execute various ones simultaneously. It is possible to reduce a derived parse tree (by deleting nodes with empty connectors) or to optimize it (tree restructuring statically and dynamically, considering performance statistics).

At this moment the environment for tool construction or semantics generation is not completely developed. Our experience shows that tools can be developed without significant investments, for example, by using Lex/YACC as a generator to create a syntactic object, which produces program intermediate representation. It is not too difficult to develop a Traverser and a simple SymbolTable. And as the last job, we have to work up semantic aspects on the basis of our method and compose them, thus obtaining connectors, which can be written in some common programming language (skipping meta-language use and its translation). The use of abstract components depends on target semantics.

We believe that the development of the serious tools demands a more universal implementation of the Symbol table. Our Symbol table implementation - MOMS - is not applicable only for imperative language implementations. It was also the basic object-oriented database for the commercial application Mosaik (Sietec consulting GmbH Co. OHG, graphical CASE tool for business modelling).

The weakness of our method lies in the semantic aspects composition stage. At this moment we have not analyzed all risks in terms of obtaining senseless or erroneous semantics. The problems are not trivial, and they are similar to problems in the proper collaboration of objects or components in object-oriented programming, too. [e.g. ML98]. Most name conflicts can be precluded automatically, but it is considerably harder to organize collaboration among the common components in semantic aspects (it is easier if the semantic aspects are mutually independent).

Another problem is that the language grammar is frequently not context free (this is true of our example above, too). In this case we have to introduce additional flags to memorize the context of syntax elements. It is advisable to rewrite the syntax and to use context-free grammars.

9. References

[AAB96] V. Arnicane, G. Arnicans, and J. Bicevskis. Multilanguage interpreter. In H.-M. Haav and B. Thalheim, editors, *Proceedings of the Second International Baltic Workshop on Databases and Information Systems (DB&IS '96)*, Volume 2: Technology Track, pages 173-174. Tampere University of Technology Press, 1996.

[AGB00] U. Aßmann, T. Genßler, and H. Bär. Meta-programming Grey-box Connectors. *Proceedings of the Technology of Object-Oriented Languages and Systems (TOOLS 33)*, pp.300-311, 2000.

[ASU86] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, 1986.

[BM99] B. Biswas and R. Mall. Reverse Execution of Programs. *ACM SIGPLAN Notices*, 34(4):61-69. April 2000.

[Cla99] C. Clark. Build a Tree – Save a Parse. *ACM SIGPLAN Notices*, 34(4):19-24, April 2000.

[DKV00] A. Deursen, P. Klint, and J. Visser. Domain-Specific Languages: An Annotated Bibliography. *ACM SIGPLAN Notices*, 35(6):26-36. June 2000.

[Eng99] Dawson R. Engler. Interface Compilation: Steps toward Compiling Program Interfaces as Languages. In DSL-99 [ITSE99], pp.387-400.

[FL88] Charles N. Fisher, and Richard J. LeBlanc, Jr. *Crafting A Compiler*. Benjamin-Cummings, 1988.

[FNT-97]F. Ferrucci, F. Napolitano, G. Tortora, M. Tucci, and G. Vitiello. An Interpreter for Diagrammatic Languages Based on SR Grammars. *Proceedings of the 1997 IEEE Symposium on Visual Languages (VL '97)*, pages 292-299, 1997.

[GHJV95]E. Gamma, R. Helm, R. Johnson, and J. Vlisides. *Design Patterns: Elements of Reusable Software*, pages 331-334. Addison-Wesley, 1995.

[HK00] J. Heering and P. Klint. Semantics of Programming Languages: A Tool-Oriented Approach. *ACM SIGPLAN Notices*, 35(3):39-48, March 2000.

[ITSE99] Special issue on domain-specific languages. *IEEE Transactions on Software Engineering*, 25(3), May/June1999.

[Kin95] W.Kinnersley, ed., The Language List. 1995. http://wuarchive.wustl.edu/doc/misc/lang-list.txt

[Lou97] Kenneth C. Louden. Compilers and Interpreters. In Tucker [Tuc97], pp.2120-2147.

[ML98] M. Mezini and K. Lieberherr. Adaptive Plug-and-Play Components for Evolutionary Software Development. SIGPLAN Notices, 33(10):97-116, 1998. *Proceedings of the 1998 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages & Applications (OOPSLA '98).*

[OW99] J. Ovlinger and M. Wand. A Language for Specifying Recursive Traversals of Object Structures. SIGPLAN Notices, 34(10):70-81, 1999. *Proceedings of the 1999 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages & Applications (OOPSLA '99).*

[Paa95]J. Paakki. Attribute Grammar Paradigms – A High-Level Methodology in Language Implementation. *ACM Computing Surveys*, 27(2):196-255, June 1995.

[Pag81] Frank G. Pagan. *Formal Specification of Programming Languages: A Panoramic Primer*. Prentice-Hall, 1981.

[Sch97] David A. Schmidt. Programming Language Semantics. In Tucker [Tuc97], pp.2237-2254.

[SK95] K. Slonneger and B. L. Kurtz. *Formal Syntax and semantics of Programming Languages: A Laboratory Based Approach*. Addison-Wesly, 1995.

[Slo97] A. M. Sloane. Generating Dynamic Program Analysis Tools. *Proceedings of the Autralian Software Endineering Conference (ASWEC'97)*. pp.166-173, 1997.

[Tuc97] Allen B. Tucker, editor. *The computer science and engineering handbook*. CRC Press, 1997.

[Vis01]J. Visser. Visitor Combination and Traversal Control. SIGPLAN Notices, 36(11):270-282, 2001. *Proceedings of the 1998 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages & Applications (OOPSLA '01).*

[ZZ97] D.-Q.Zhang and K.Zhang. Reserved Graph Grammar: A Specification Tool For Diagrammatic VPLs. *Proceedings of the 1997 IEEE Symposium on Visual Languages (VL '97)*, pages 292-299, 1997.

10. Appendix

Table 4.

MOMS types

Type	Description
Constructor	Handle to an object type description
Value	Handle to a byte stream that contains an object value
Reference	Handle to a memory object
MemoryMap	Main central object that organizes other objects (other MemoryMap also)
IdentifDict	Dictionary of all identifiers (variables, constants, etc.)
TypesDict	Dictionary of all types (basic types and user defined)
FunctDict	Dictionary of all functions (basic operators, basic functions, user defined functions)
NamesTable	Compact storing of all strings
ConstrTable	Compact storing of all constructor descriptions
ValuesTable	Storing and managing of all object values
MemoryBlock	Organizing scope visibility in all dictionaries and providing memory management
...Others	Stack, queue, collection, etc.

Table 5.

System initialization and global operations

Operation	Description
MemoryMap initialize(Uint memoryCount)	Creates MOMS with internal parallel but related memories (MOMS)
Uint switchMemoryTo(Uint memNum)	We can exploit only the specific internal memory
Uint getCurrentMemory(void)	Returns the number of the actual memory
defineCountOfBaseTypes(Uint countOfBaseTypes)	Defines a count of the basic types of MOMS
defineBaseType(char* typeName, Uint type, Uint typeSize)	Defines base types. This interface is in C and depends on previously defined types. Some examples: defineBaseType("long_", LONG_, sizeof (long_)); define Base Type ("boolean ", BOOLEAN_, sizeof (boolean_)); define Base Type ("date ", DATE_, sizeof(date_))
defineBaseFunction(char* langFunctName, char* internalName, Uint returnType, int paramCount, ...)	Defines the base operations and functions. This interface is in C and depends on previously defined types. Some examples: define Base Function ("+", "PLUS", LONG_, 2, LONG_, LONG_); define Base Function ("day", "day", LONG_, 1, DATE_)
prepareProgramEnv(Uchar scope)	Prepares a new MemoryBlock, defines the scope (visibility) of previously defined variables, types, functions
releaseProgramEnv(void)	Releases a current MemoryBlock and all related memory in other objects (dictionaries, tables) and restores a previously defined Memory-Block
defineAutomaticMemSwitching(Uint firstMemNum, Uint lastMemNum)	Provides for automatic switching in various functions. For instance, we look up the variable in a local memory and then in a global memory (if the variable is not founded yet).
... Others	

Table 6.

Defining of user defined data types

Operation	Description
putValue(Ref aRef, char* aValue)	Sets a new value for the object.
char* getValue(Ref aRef)	Returns a value for the object.
char* createDynamicValue(ConstrPtr ptrToConstr)	Provides dynamic memory allocation for the object given by type.
deleteDynamicValue(char* aValue)	Releases dynamically allocated memory.
setValueProtectionOn(char* aName)	Protects a value of the given object against modification, for instance, protects constants.
setValueProtectionOff(char* aName)	Takes off a value protection.
gotoArrayElementConstr(Ref& aRef, Sint index)	Sets a virtual mark to element constructor and to a given array element value. aRef is modified, it refers to the array element.
gotoNameConstr(Ref& aRef)	Moves the virtual mark to the name constructor.
gotoPointerConstr(Ref& aRef)	Moves the virtual mark to the pointer subconstructor and to the start of value.
gotoProductionLeftConstr(Ref& aRef)	Moves the virtual mark to the left subcon-structor and to the start of the corresponding value.
gotoProductionRightConstr(Ref& aRef)	Moves the virtual mark to the right subcon-structor and to the start of the correspondding value.
gotoRecordConstr(Ref& aRef)	Moves the virtual mark to the start of record.
gotoNameInList(Ref& aRef, char* aName)	Moves to the appropriate type and value (list of named types linked by productions) E.g., search a field in the user-defined structure.
... Other	

Table 7.

Operations with variables and similar objects

Operation	Description
ConstrPtr create Constr Array (Uint min Index, Uint maxIndex, ConstrPtr ptrTo ElemConstr)	Defines an array type with the given dimensions and element types. Here and in other functions we can use any previously defined (or partly defined) data type.
ConstrPtr createConstrFunct (ConstrPtr ptr To Return Constr, ConstrPtr ptr To Param Constr)	Defines a function type with the given parameters and return type.
ConstrPtr createConstrName(char* aName, ConstrPtr trToSubConstr)	Assigns a user-defined name for the given type.
ConstrPtr createConstrPointer(ConstrPtr ptrToSubConstr)	Defines a pointer type to the given type.
ConstrPtr createConstrProduct(ConstrPtr ptrToSubConstr1, ConstrPtr ptrToSubConstr2)	Creates a production of two types (establishes some relation between them). It is useful to construct a serious data structure.
ConstrPtr createConstrRecord(ConstrPtr ptrToSubConstr)	Defines a record data type (a set of pairs {name, type}).
... Other constructors	For instance, base data type constructors
constrArraySetMinIndex(ConstrPtr ptrToConstr, Uint minIndex)	Modifies the type description (attributes).
Uint constrArrayGetMinIndex(ConstrPtr ptrToConstr)	Provides details about data type attributes.
... Others	

Table 8.

Operations with value

Operation	Description
Create Var (char* aName, ConstrPtr ptrToConstr)	Creates a variable with the given name and type.
createVar(char* aName, char* typeName)	Creates a variable with the given name and type name.
createLiteral (char* aName, Constr Ptr ptrTo Constr)	Creates a literal (constant) with the given name and type.
Ref createRef (ConstrPtr ptrToConstr)	Creates an object without a name with the given type, for instance, internal loop counter, return value of function.
ConstrPtr getConstrRef (char* typeName)	Returns pointer to type with the given type name.
createSynonym (char* aName, Ref aRef)	Creates another reference by name to the existing object.
... Other	

LATVIJAS UNIVERSITĀTES RAKSTI. 2004. 669. sēj.: DATORZINĀTNE UN INFORMĀCIJAS TEHNOLOĢIJAS. 34.–52. lpp.

Data Staging in the Data Warehouse

Jānis Benefelds
Computer Science Ph.D. student, University of Latvia
Janis.Benefelds@unibanka.lv

This topic describes one of Data Warehousing components – Data Staging. Data staging ensures a) data extraction from the source systems; b) data transformation and standardisation; c) data transfer to the Data Warehouse. The Importance of such important things like data quality and data integrity is described in more details too.

This paper intends to be a survey based on practical experience in real projects and theoretical professional literature studies.

The Paper consists of a Foreword, six Chapters and Referenees.

The Foreword describes the scope or the subject to be analysed in this article.

The first Chapter is an introduction that gives definitions and common understanding about concepts used in this article.

The second Chapter is the largest one, and it deseribes three main parts of Data transformation Staging: data export, data transform and data load. Two types of Data Staging – dimensional and fact table staging – are described in more details. This Chapter is concerned with relative problems, like data quality, as well.

The third and fourth Chapters describe such an important component of Data Staging like Metadata and the mutual relation between it and other parts of Data Staging described in the previous Chapter.

The Fifth Chapter summarises the more important items and makes emphases on them to understand how essential they are to be successful in Data Staging.

The Sixth Chapter points out the directions that follow as logic and sequential continuation to be a subject for further research and analysis.

Referenees contains a list of information sources.

Key words: metadata, data warehouse, data staging, data integrity.

Foreword

This article is a summary about Data Staging as one of the components of Data Warehousing. There is an explanation of Data Staging – what it means and what it consists of. Some common definitions are given and the basic elements (data extraction, transformation, loading and metadata) are described. The content is additionally illustrated by many examples that are taken from the financial industry.

This Article points out the importance of the right understanding of the functions which Data Staging is responsible for, to be successful in designing and deploying projects in real life.

The following picture (Figure 1) shows the basic Data Warehousing elements and describes what part of it this article covers.

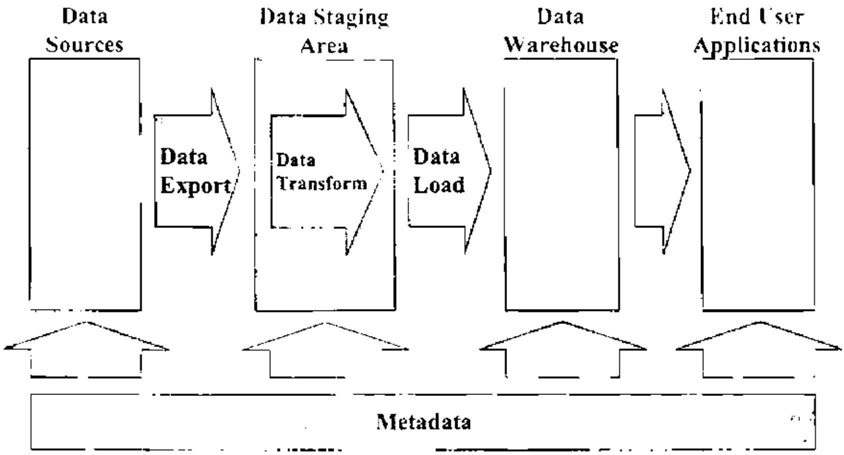

Figure 1: The basic elements of Data Warehousing.

1. Introduction

Data Staging is one of the first and one of the most important things, if we are discussing issues about developing a Data Warehouse.

In a lot of books, the Internet and other sources we can find numerous Data Staging definitions that are very similar. I would like to mention some of them that would be sufficient and describe all the main functions performed by Data Staging.

> Data Staging is a set of processes that cleans, transforms, combines, deduplicates, households, archives and prepares source data for use in the Data Warehouse.

> Data Staging Area is everything in between the source system and the presentation server where Data Staging processes are performed. The key defining restriction on the Data Staging area is that it does not provide query and presentation services to end–users.

> The Data Staging area is the Data Warehouse workbench. It is the place where raw data is loaded, cleaned, combined, archived and exported to one or more presentation server platforms.

One of the DWH gurus, Ralph Kimball, has mentioned that Data Staging adds to the data an additional value, e.g., the data quality and sense of use of these data in some analysis is higher than before. In general the Data Staging process consists of one common service component – metadata – and three functional components:

- Data Export from the source systems or data consolidation;
- Data cleaning, transforming and sorting or data standardization;
- Data import into the Data Warehouse, Data Mart or loading;

There may be some different abbreviations used to name these processes. ETL stands for Export, Transfer and Load, ETT stands for Extraction, Transformations and Transportation. In this article we'll use ETL.

It certainly does not mean that Data Staging consists of three strictly different modules, where the sequence and completeness of tasks to be performed is highly prescribed. Data Staging, depending on the situation, can be very a complicated process from the functional and technical point of view.

It is recommended to start the Data Staging design process with a very simple schematic of the pieces that you know: the sources and targets. Keep it very high-level and highlight where data are coming from and where data are going to, point out the major challenges that you already know about. Figure 2 shows an example of a high-level Data Staging design plan that came from the financial industry.

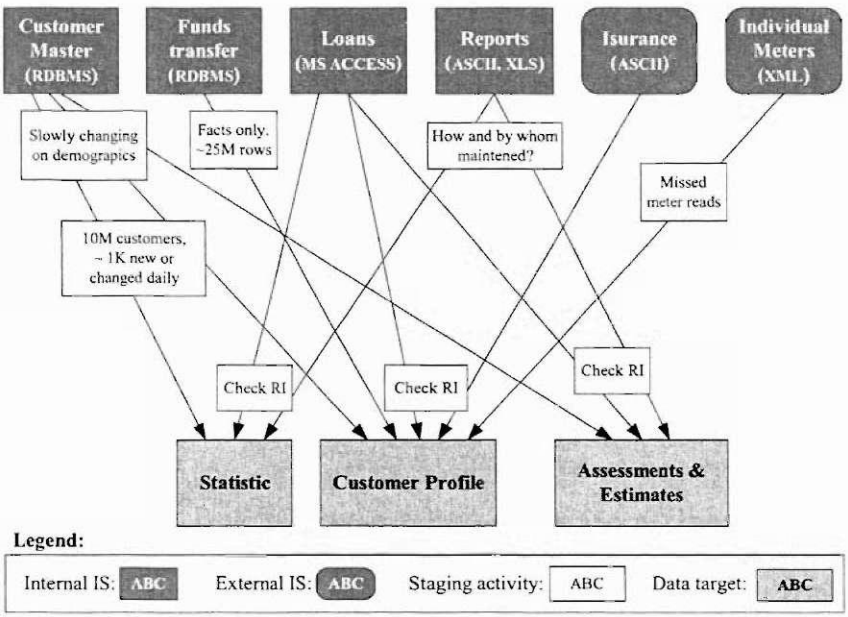

Figure 2: Example of the high-level Data Staging plan for some financial institution.

For a better understanding of the principles and mission of Data Staging, let's analyse each of the three steps mentioned before more closely.

2. ETL

2.1. Data export from transactional systems or data consolidation.

Practically every on-line transactional system (OLTP) used in organization is a potential data source for the Data Warehouse. However a data source for the Data Warehouse can be some external information systems and not classified information

(flat files etc.) as well. Everything that serves as a data source can be developed in different environments on different platforms and it can be "home-made" and delivered by external vendors. From the point of view of the Data Warehousing it should make no difference – the only thing we are interested in is data.

Normally, the scope of the data to be collected in the Data Warehouse is defined by the end users – usually business people. That means the scope is defined in business not Information Technology concepts and therefore high quality business analysts and system analysts are required at this stage of the Data Warehousing process.

To determine the right database objects (and their relationship) that potentially could be included in a data export processes, business analysts should very precisely define the requirements of data needed and the system analyst should be able to point out how and where these data physically can be allocated. Normally the system analyst uses the system documentation that should help to find out the right information about data within a system. Sadly, there may be situations when documentation does not help, or there is no documentation at all. Such a situation can arise because of poor management during system development, implementation and maintenance processes. Then the only source of information about the inside of a particular information system are people. Some kind of interviews should be organized and system documentation, should be redesigned accordingly to the actual status of the system.

One of the resulting documents of these interviews is a data dictionary and it is actually a part of the metadata (see chapter 3.Metadata). A Data dictionary is a collection of descriptions of the data objects or items and their relationships in a data model. A data dictionary can be consulted to understand where a data item fits in the structure, what values it may contain and basically what the data item means in real-world terms. A small example of that part, which explains what the data item means in real–world terms, is shown in the following table.

Table No.1

"Example of a Data dictionary from the financial industry"

Real-world term	Database object and its relationship
Account (Acct)	One record of the table ACCOUNTS
Active Acct	Accounts.Status = 1 AND Accounts.CloseDate IS NULL
Dormant Acct	Accounts.Status 2 AND <&today> – Accounts.LastTranDate > 3M
Closed Acct	Accounts.Status = 2 AND Accounts.CloseDate <= <today>
Client	One record of the table CLIENTS
Active client	Client that has at least one active account (see 'Active Acct')
Dormant client	Client with all accounts being dormant (see 'Dormant Acct')
Closed client	Client with all accounts being closed (see 'Closed Acct')

Using such a data dictionary improves the efficiency of communications between business and IT people. There should be no questions about what the business people are requiring and where it can be found.

When we have clarified what data is needed and where it can bee allocated, we should give an answer to the question "how" the data can be extracted or exported from the particular data source. Usually the first step[s] of the technical solution is very close

to the technical solution of the data source itself. It means a special program should be developed that extracts specific data from all data sources and puts it together in a Data Staging area. Figure 3 shows a small schema of data extracting from the different data sources and gathering the particular data together in the Data Staging area. Some financial figures of the customers are taken to build a customer profile.

At the Start of building the Data Staging application the primary goal is to work out the infrastructure issues, including connectivity, security and file transformation.

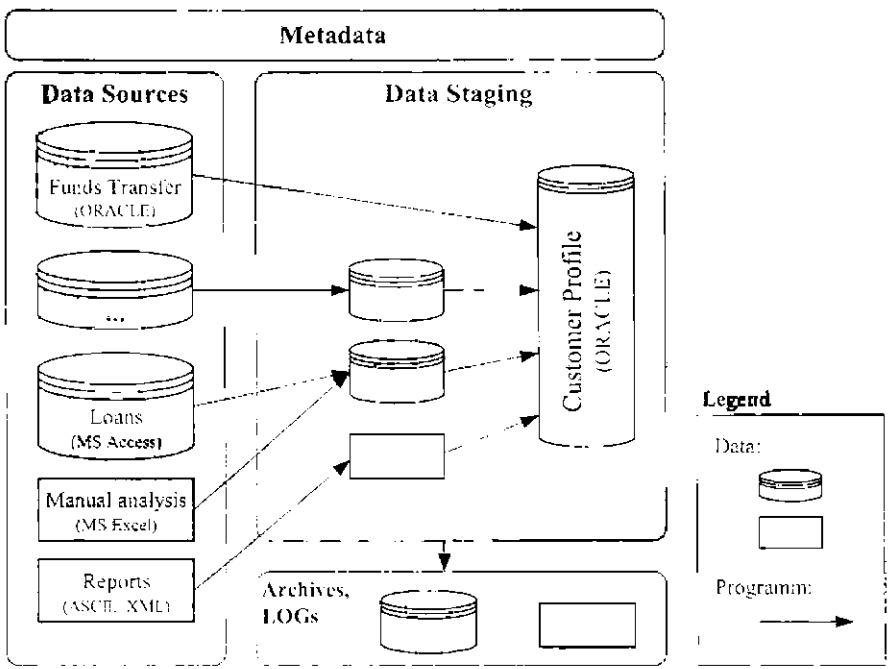

Figure 3: An Example of data extraction processes from different data sources and putting it together in the Data Staging area.

Certainly data in a Data Staging area are stored in some particular structure and therefore the way in which the data comes into these structures should be strictly predefined and completely documented. The name of such a document is data mapping. Data mapping shows a complete way in which the particular data from some source system is transported (and transformed in between if necessary) to a Data Staging area.

Corresponding to a dimensional data model (generally that is a data model for the Data Warehouse) there are two different types of Data Staging, each for one of two main components of the dimensional model: dimensional Data Staging and fact Data Staging.

Dimensional Model - a specific discipline for modelling data that is an alternative to entity-relationship (ER) modelling.

A fact table is the primary table in each dimensional model that is meant to contain measurements of the business. The most useful facts are numeric and additive. Every fact table represents a many-to-many relationship and every fact table contains a set of two or more foreign keys that join to their respective dimensional tables.

A dimensional table is one of a set of companion tables to a fact table. Each dimension is defined by its primary key that serves as the basis for referential integrity with any given fact table to which it is joined.

2.1.1. Dimensional table staging.

There are three types of processing dimensional data: overwrite, create a new dimension record, and displace the changed value into an "old" attribute field. The second one is the technique usually used for handling occasional changes in a dimension record. We shall actually focus on that type of dimensional data processing.

In terms of complexity the major problem could be the identification and extraction of data that has been changed since the last data extraction instead of extraction of all data every time. It is convenient to have a mechanism to select new or changed data only. Ideally there is an additional column in the source data that shows the last change date and time or just a yes/no stamp that indicates whether the particular data is or is not changed since the last data extraction. That means that only these data will be the subject of the Data Staging. Taking into account that usually some of the data source information systems are either delivered by an external vendor or long ago "in-house" systems without any knowledge about their contens and without full documentation, there may be some problems to change these systems for having such a stamp indicating a change status according to the last data extraction process. That means there should be another way to determine the existence of the new records and changes in the old ones. In this case the easiest way is to pull the current snapshot of the dimension table in its entirety and let the transformation process deal with determining what has changed and how to handle it. Change determination could be done by comparing the actual data snapshot either with yesterday's data snapshot or with the current Data Warehouse dimensional table. This way is more time-consuming and complicated. Figure 4 shows data flow for this type of dimensional Data Staging.

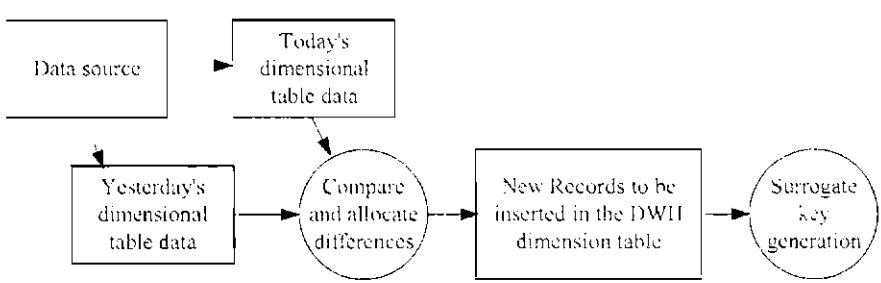

Figure 4: Allocation of new and changed records in dimension data.

Because of the large amount of data to be extracted, compared (or transferred), special attention to usage of software and hardware resources should be paid. Special comparison methods could be used for performance improvement, such as the Cyclic Redundacy Checksum (CRC). This method calculates a checksum for a particular string – it could be a record of a table or a whole text file or something else. The checksum calculated for the new data is then compared with the checksum of the yesterday's data that is already calculated. If these checksums are equal there is no need to compare details for particular details. The Usage of such methods substantially saves time and hardware resources.

2.1.2. Fact table staging

Facts usually represent a snapshot of a data source at that particular time point. Accordingly facts normally are the numeric attributes of the subject of the transactional system, like daily balance of an account, sales amount of a product, etc. Problems regarding extracting of such data from the data source normally are connected with the amount of data to be processed and with the changeability of data. Problems regarding data amounts usually can be solved on account of technical issues or data processing performance optimization. Almost every transactional system provides today on-line services and that means transactions are performed all the time (24hours/7days). The problem grows even bigger in case of a multinational system, where many time zones are involved. To get a snapshot of such a system in the particular time point, some technical tricks can help. There can be several solutions. In current data base management systems, like Oracle, a backup technology is provided without disconnecting users or shutting down a data base – a back-up is created taking into account the events in the system logs. Accordingly fact data could be extracted from a back-up or directly from the system using the same technology. Actually, normally every system has some housekeeping phase for supporting some internal functionality (like EOD – end of day) that makes a system constant for some time. Whether the system serves business transactions or not during this time-out depends on the particular system – there are some distinctions. Such time-out is a good place for fact data (and dimensional data as well) extracting.

Conclusion.

The Financial industry differs from the other major sectors of business with its non-stop business. Because of many channels (ATM, Internet, WAP, etc.), which banks provide, the customer has the possibility to do business all the time any day. A Similar situation exists in the telecommunication business as well.

That is the N°1 problem for data extraction, which should be solved. Normally every IS operating in the financial industry has already solved the problem of on-line transactions. There are some techniques that ensure on-line data processing and daily (weekly, monthly, etc.) data servicing at the same time. These data servicing procedures (housekeeping, calculations, etc.) could be populated with data extraction.

Problem N°2 is the large amount of heterogeneous data (funds transfers, loan agreements, securities, trust operations, insurance, etc.) to be processed. That could be solved by 1) using optimal data allocating algorithms to find data to be processed and 2) fast data extraction mechanisms. A Data allocating mechanism could be, for instance,

Cyclic Redundacy Checksum method or using different 'changed data' flags in the source systems. As fast a data extraction mechanism specific tools could be used or just making a copy and processing 90% of data instead of selecting 90% of data and processing all of the resulting data set.

2.2. Data cleaning, transforming and sorting or data standardization

This is the place where an additional value to the data actually is added.

Usually there are some activities between data extraction from data source systems and data import into the data target. These activities ensure that data in the right structure, format, and sequence, and that quality should be adequate for the data model of the data target. The Data should satisfy all logical rules provided by business analysts and all physical rules that come from the Data Warehouse data model and end user applications.

2.2.1. Data quality

Actually, most of all data transformation deals with data quality.

There is no reason to assume the quality of data in OLTP systems always will be acceptable for the Data Warehouse. Some data validations can be integrated directly in the source (OLTP) to preserve data from logical and structural mistakes. In any case, there are some problems. First – every data validation procedure makes the system slower and it conflicts with a definition of the OLTP "on-line (identical "fast") transaction processing". Second – even if we create all necessary data validations in OLTP, there always will arise some new requirements or conditions against data to be realized in the Data Warehouse. It is easier, cheaper and more secure to realize data cleaning and transformation procedures outside of the data source.

In the Situation when data are validated after they have been captured, another problem arises – how to edit them now? If a validation error occurs by data capture, they can be entered once more till they are correct. Whereas if a data validation error is discovered during a fully automated data cleaning and standardization process, there should be some previously described algorithms to manage the situation. Automatic data editing algorithms should be approved by business analysts and every data change should be stored in the LOG files.

Wherever and whenever data are modified before importing them into the Data Warehouse, they should be at the best quality possible. There are attributes that indicate the level of data quality.

Table No.2

"Attributes indicating the data quality."

Attribute	Description
ACCURATE	means data should be accordingly equal to the data extracted from data sources. All data modifications, standardizations and other processes performed by data transforming are accountable and an explanation of every difference can be found in audit trails - LOG files;
COMPLETE	means the particular data set (data mart) should contain absolutely all data and nothing over accordingly the Data Staging procedures. If one particular data mart should contain all active customer records then all customers with an 'active' status should be included there and all customers with a status not equal to 'active' are not included there;
CONSISTENT	means during data transformation process they should not lose their meaning and value. These are situations when one should solve problems regarding to rounding of numeric, interpretation of aggregated values and other similar issues;
UNIQUE	means data should not have duplicates. Especially when data are extracted from more than one source. The unique ID from business understanding must be allocated or generated for the same business unit and business analyst improved algorithms used for data joining from multiple sources;
TIMELY	means data should be current. Nobody is interested in out of date information. The data used in the Data Warehouse should be taken from and validated against 'fresh' data whenever possible. Of course, 'fresh' has many meanings in different businesses. If we are talking, for instance, about hotel chains, the 'fresh' data could be a turnover of the last day or even week. Whereas if we are talking about financial institutions, like money markets or stock exchanges, then minutes or even seconds is a very long time.

To ensure the data characteristics mentioned above many problems have to be solved. These are problems regarding the data itself, not performance problems of hardware or software.

Table No.3

"Types of problems to be solved regarding data quality"

Type Description
Inconsistent use of code or different look-up values The same value could have a different ok key in different information systems. Sex, for instance, in one source could have a look-up table like M – male and F – female, while in another system 1 – male and 2 – female. These values should be standardized during the data transformation process by unification of the particular values;
Unofficial or undocumented functionality Sometimes some of the fields are used for another reason than documented in the user manual or data in the particular field are unclassified. A big mistake could be the address fields that are not structured as an address usually is – five text fields of 25 characters length, for instance. From the business point of view the content of these fields is gold, but it is absolutely unusable for data processing;
Overloaded codes Usually overloaded codes occur in the systems that are developed a long time ago when every saved bite was very important. That means, a single code was used for multiple purposes. If the customer is an individual then a particular field could be used for birth date. If it is a corporate customer – for registration date;
Evolving data Systems that have evolved over time may have old data and new data that uses the same fields for different purposes or where the meaning of the codes has changed over time. It is very important if data are planned to load into Data Warehouse from historical data archives;
Missing, incorrect or duplicate values Names and addresses are the classic example of this problem. Transactional systems do not care about data that are not needed for performing the particular transaction. It does not matter whether the name of the customer is IBM or I.B.M. But they can end up looking like two different customers in the warehouse;

I would like to add one more type of problem from my own experience. That relates to the different data formats – data should be in the same format to be processed automatically.

Table No.3 (continued)

"Types of problems to be solved regarding data quality"

Different data formats When Gathering data from many sources, data could be in different formats. That refers to any data format: date ('dd.mm.yy'; 'YYYY-MON-DD'; ...), numeric ('0.99';'9.9999E'; ...) etc. To use such data for common data processing they have to be in one previously described format;

If it is possible a feedback from Data Staging back to the data source could be considered. This is also a way to improve the data quality in the data sources. However it should be estimated very critically, because data sources – analytical transactional systems – most of all are made for data processing captured by users. Data modifications made by another software, like Data Staging procedures, could not be the best way of improving data quality in the analytical system.

2.2.2. Data integrity

Besides data quality this part of Data Staging is responsible for data integration as well. Ensuring of data integrity means surrogate key (as primary key) generation for dimensional data tables and fact table population (or even creation) using just generated surrogate keys. All dimensional tables have single part keys, which, by definition, are primary keys. In other words, the key itself defines uniqueness in the dimensional tables. It is a significant mistake to use, for uniqueness of dimensional table record, an SQL-based date stamp, a set of various attributes in the dimensional table, production keys used in the data source. In most cases an integer makes a great surrogate key. A 4-byte integer, for instance, can contain 2^{32} values, or more than two billion positive integers starting with 1, and that is enough for just about any dimension.

Since fact tables are always associated with dimensional tables, surrogate keys should replace corresponding foreign keys there. Figure 5 shows a dimensional data model where generated surrogate keys (Time_key, Customer_key, Acct key) of dimensional tables are used as foreign keys in the fact table instead of production keys (Date, Customer_ID, Acct_ID).

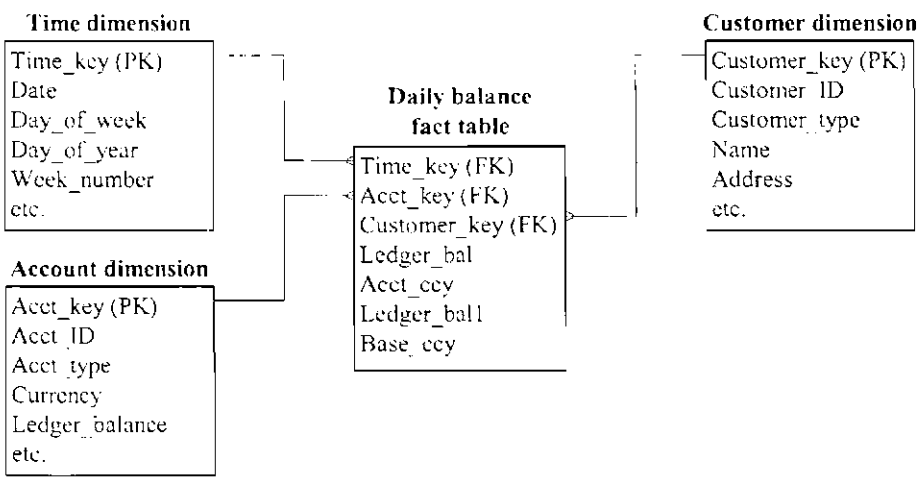

Figure 5: Example of using surrogate keys instead of production keys in the Daily balance fact table.

For better querying performance besides dimensional and fact data, the Data Warehouse has the aggregate date as well. Since the Data Warehouse is used most of all to get summarized data over any dimensional attributes, a significant data aggregation would be performed by every data request. Therefore data base management systems provide an opportunity to maintain aggregated data – materialized views or data where subtotals are already calculated up to a predefined level of granularity. Figure 6 shows how an aggregate data table is handled that ensures answer to data query about product sales subtotals for a particular week or a few weeks.

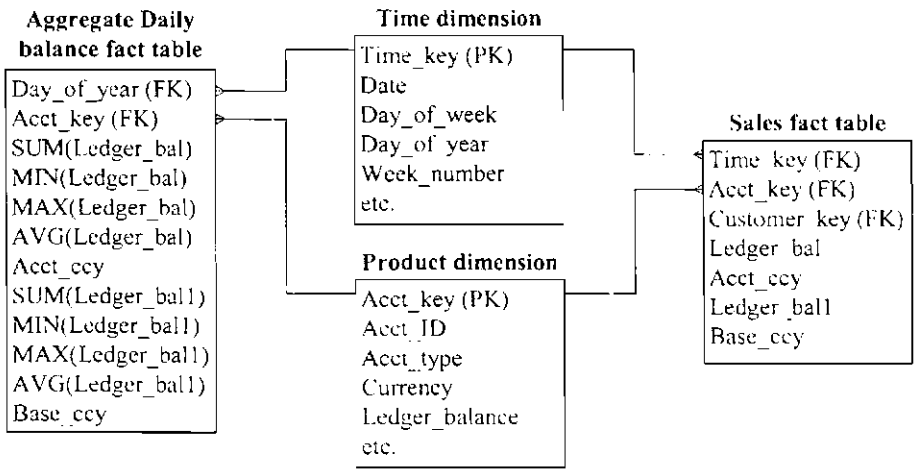

Figure 6: Example of the aggregate Daily balance fact table.

Conclusion.

I would say that data quality is one of the major units of measure (the second one is data accessing performance, but that is not the subject of this topic) for the Data Warehousing. The benefit that users obtain from the usage of the data warehouse is as large as how clean and good are the data. So, if data are inaccurate, the user gets the wrong picture of the business and a decision that is based on that picture could be wrong.

No recourses should be economized to improve data quality. Regarding the data that banks capture and maintain themselves – that is all their own business, and they do everything to avoid 'dirty' data. Sometimes a solution could be some administrative order to stop doing things in the wrong way. Data that we get from outside of the bank, or even the financial industry, should have a warranty, e.g., the data supplier should be trustworthy and secure enough. Usually these are some state institutions, but they could be some commercial organisations (insurance companies, travel agencies, etc.) as well – contract about providing data should contain requirements and warranties regarding data quality and security.

A separate Data Quality department within the organisational structure would help significantly as well. Such a department could deal with 'dirty' data – to ensure the certain level of the data quality by developing data transformation and cleaning algorithms as well as looking for some external data sources that could serve as a standard for data comparison.

2.3. Data loading into the Data Warehouse, Data Mart or data loading

Data loading into the Data Warehouse means physical data transportation from the Data Staging area (database A) to the Data Warehouse (database B). To populate an existing database with new data usually some data management problems should be

solved. This refers to such technical terms like table spaces, table partitioning, indexing, data load method (usually bulk load), audit action, memory, parallel processing, network architecture etc. The software, hardware and time resources can be significantly saved by configuring items like these. To speed up the load cycle the frequency of load can be changed, especially regarding some aggregate data. If some monthly totals should be accumulated, it does not mean that data load can be performed only once a month. It is possible to accumulate data on a daily basis by incremental load. Then some additional changes to the presentation layer should be added – whether users see 'month-to-date totals' (instead of 'totals per month') or it could be restricted by software to see just full months (the previous one will be the newest then).

Because of data integrity the sequence for loading data into the Data Warehouse is prescribed accordingly. First of all all dimensional data are loaded in the right sequence to ensure star schema data model data integrity. When all dimensional (look-up) data tables are populated, fact data and aggregated data are loaded and/or calculated. If necessary, some additional data validation can be performed.

For duplication and possible repetition processes data transportation (load) supports data archiving and back-up. That means we can repeat some data transformation and load process from any point of the staging process. That could be needed if some errors are found and removed, or if data load should be repeated because of some technical disaster. Such a duplicate table loading technique is shown in Figure 7. Back-ups of load data are needed for tomorrow's incremental load as well (see 2.1.1. Dimensional table staging).

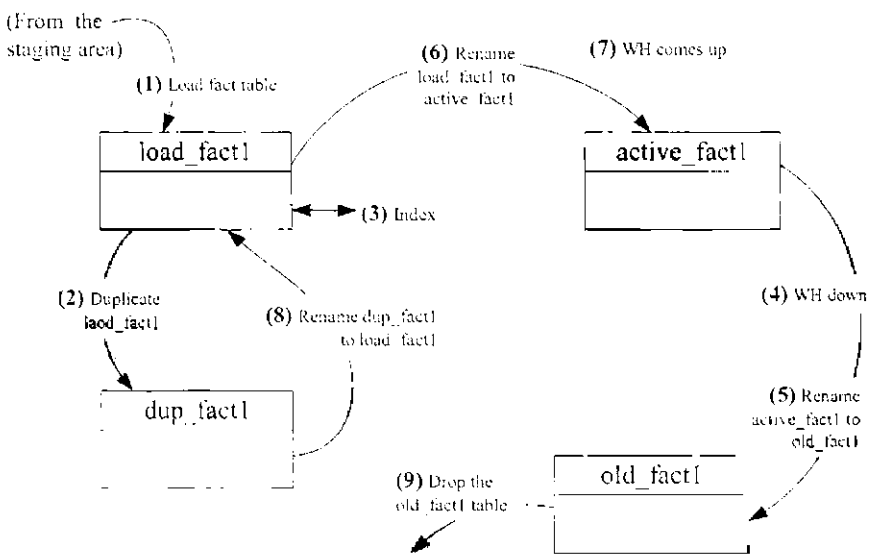

Figure 7: Duplicate table loading technique.

Step 0	The active_fact1 table is active and queried by users.
Step 1	The load_fact1 table starts off being identical to active_fact1. Load the incremental (different) facts into the load_fact1 table.
Step 2	Duplicate load_fact1 into dup_fact1 for tomorrows load.
Step 3	Index load_fact1 table according to the final index list.
Step 4	"Bring down" the warehouse, forbidding user access.
Step 5	Rename the active_fact1 table to old_fact1 (back-up).
Step 6	Rename the load_fact1 to active_fact1.
Step 7	Open the warehouse to user activity. Total downtime: seconds!
Step 8	Rename the dup_fact1 table to load_fact1 for performing of next load.
Step 9	Drop the old_fact1 table.

3. Metadata

Some tools that automate the transformation and load layer can actually reside on a third party platform and generate the code necessary to perform those activities (see Figure 8 as well). And the most important thing that ensures such a possibility is metadata.

The definition of metadata is really simple – metadata is data about data. Does it help anybody to understand what exactly this elusive information? In my opinion – no. More descriptive definition could be as follows. Metadata is all of the information (or descriptive information) in the particular environment that is not the actual data itself. Some authors are even talking about the "back room metadata" and "front room metadata". The back room metadata is process related and it guides the data extraction, cleaning and loading process. The front room metadata is more descriptive and it helps query tools report writers function smoothly. Of course, both metadata types overlap, but it is useful to think about them separately. In this case we are interested in the first one – back room metadata.

Metadata is like traditional systems documentation (in fact, it is documentation) and at some point the resources needed to create and maintain it will be diverted to other "more urgent" projects. Active metadata helps solve this problem. Active metadata is metadata that drives a process rather than documents it, while documentation of these processes is an important issue as well.

The following is the general metadata needed to get the data into the Data Staging area and prepare it for loading into one or more data marts.

Data extraction information

- Description of the source systems (schemas, formats etc.);
- Access methods, rights, privileges and passwords;
- Data transportation scheduling and results of them;
- File usage in the Data Staging area including duration and ownership;

Data transformation information

- Data model;
- Job specifications for joining sources, stripping out fields and looking up attributes;

- Data mapping between sources and Data Staging area;
- Yesterdays copy of production data to use as the basis for DIFF COMPARE
- Data update (refreshing) policies for each incoming attribute;
- Current surrogate key assignments and methodology of that;
- Data cleaning, formatting, filtering and sorting;
- Populating data for data mining (e.g., interpret nulls, scale numerics etc.);

Data load information

- Target schema design, source-to-target (staging area to data mart) data flows and target data ownership;
- DBMS load scripts;
- Aggregate definitions;
- Data (especial aggregate) usage statistics and potential improvements;

Audit, Job Logs and Documentation

- Audit records (where exactly did this record come from and when?);
- Data transform run time logs, success summaries and time stamps;
- Information about software (version numbers) and hardware (configuration);
- Security settings;

Conclusion.

Metadata is an important issue in terms of descriptions. It describes (it should at least) any object and any process dealing with these objects. It is very hard to define or to show – this is metadata and that is not. The content of metadata usually describes the structure or nature of other data (the real ones). Metadata, for instance is ER diagram, description of table structures, documentation of the program packages etc.

4. Relationships between Export, Transfer and Load.

As I have already mentioned data extraction, transformation and load are very closely related to each other on the one hand (usually in the small projects) and these parts could be very strictly separated on the other hand (large projects sometimes use separate software products to manage data extraction, transformation, loading and metadata in their entirety). Sometimes it is really impossible to strictly separate which piece of software or hardware belongs to one part of Data Staging and which one to another. For instance, let us say we have just one data source system, which operates in the Oracle, and data should be loaded in the Data Warehouse, which is developed in an Oracle environment too. If hardware and software configuration allows creating a database link between these two Oracle databases, then Data Staging could be just one SQL statement. Of course, it is a very simplified example, but it is true:

```
INSERT INTO CUSTUM_DIMENSION
(
SELECT NEXTVAL.cust_seq cust_pk,
cust_id,
cust_name,
TO_DATE(birth_date,'DD.MM.RRRR') birth_date,
DECODE(sex,'1','M','F') sex
FROM   CUSTOMERS@SOURCE
ORDER BY cust_id )
```

And there we can see:

Data extraction:	SELECT cust_id, cust_name, birth_date, sex FROM CUSTOMERS@SOURCE	
Data transformation:	NEXTVAL.cust_seq	– surrogate key generation
	TO_DATE(birth_date,'DD.MM.RRRR')	– data format adjustment
	DECODE(sex,'1','M','F')	– unification of code keys
	ORDER BY cust_id	– particular record order
Data transportation:	INSERT INTO CUSTUM_DIMENSION	

It actually means that from the technical point of view Data Staging looks like Figure 8.

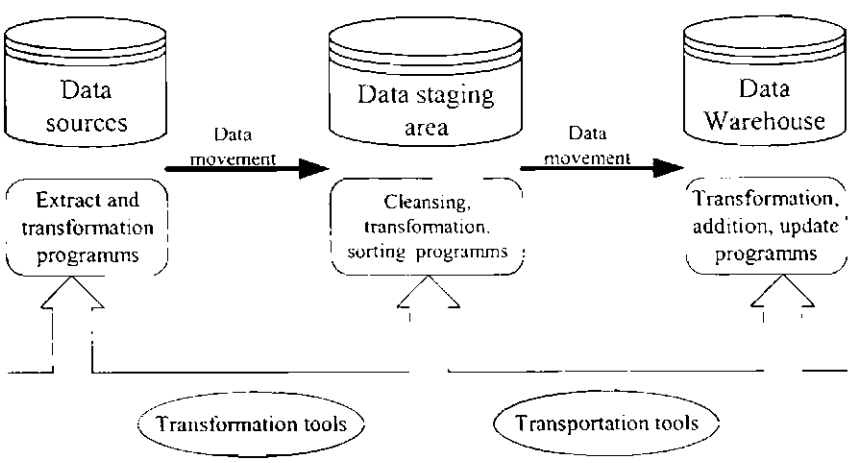

Figure 8: Role of data transformation and transportation tools within a Data Staging process.

5. Summary

As a conclusion I would say that Data Staging is one of the main issues of the Data Warehousing one must pay much attention to. Data Staging covers any action with data starting from extracting them from the source systems and finishing with data loading into the presentation server (Data Warehouse and/or Data Mart). Data Staging itself does not include data presentation, e.g., data cannot be accessed by the end users from the Data Staging area. Data Access by end users is performed by other tools. Actions that Data Staging performs are (at least, should be) very short in time, whereas the developing of these processes could take much of time and many other resources.

The main tasks of Data Staging are 1) fast particular data extraction from the source systems 2) fast data transformation, standardization and data quality assurance according to the business user requirements and 3) fast data loading into the target according to the data model (usually dimensional) it requires.

Nowadays technologies usually provide a range of possibilities to ensure fast physical data processing and transportation – I would say, that should solve all problems regarding physical data extraction, moving them through the hardware infrastructure and loading them into the data target, whatever that may be. Besides that, the logical side of data processing still should be taken into account. This part of the data processing brings out the importance of the human sense or intelligence that should be both the driver and supervisor of the manner in which data is managed. That means Data Staging development requires additional special skills and experience in the data management area. In the case of Data Warehousing these special skills would be dimensional data modelling and usage of and management of the metadata. It could be reasonable to look toward some vendors that provide tools for the functionality just mentioned (dimensional data modelling and metadata management). In any case, it is very good to have at least a theoretical background and understanding about how it should work. On the one hand there is no ready solution (software) that will satisfy you for 100% of your needs, and on the other hand you will never develop that functionality by yourself according to the budget, time and end user requirement restrictions. The truth is always in the middle!

The main deliverables (in terms of the metadata) of the Data Staging would be:

- Data dictionary
- Data mapping
    - Data source to Data Staging area
    - Data Staging area to data target
- Description of the data transportation (extract, back-up and load)
- Data quality checklist
- Data transformation algorithms and audit trails

Conclusion

Regarding the financial industry, data extraction and consolidation is very important in terms of completeness and accurateness. Only complete and accurate data give the right picture – common trends and aggregates.

Aggregated information is usually passed to a company's management, external supervisors and auditors. Each of them requires special and very similar information. They are different but comparable! Data requirements are both regular and on demand. This is a good way to show others that data is under control in one's company, especially if response times to irregular data requests are short enough.

Besides the common trends financial institutions focus on particular individuals or companies as well. Therefore data of the particular individual can be evaluated in relation with common trends only partly. The other, the larger part of an evaluation is based on a client's individual data – client profile. Actually, such approach to data analysis is part of Customer Relationship Management (CRM), which is very important for financial institutions. This is exactly the reason why data quality should be the best possible. Banks, insurers and other financial industry players are marketing themselves as ones who are professionals in whom you can trust. And clients are expecting an adequate attitude: "So, if you know everything about my finances, why do you offer me a Gold Credit card? My monthly income is by far insufficient to buy such a product!" Such a situation comes from unclean and in consistent data, and it leads to the customer that doubts whether the bank is dealing with his many at a level that is professional enough.

6. Further research

Sequentially the next part after Data Staging is Database – either Data Warehouse, Data Mart or by a special Data Mining tool predefined data structures.

Developing the right **Data** Model and overall data architecture strategy could be the direction of the next research. This includes understanding the difference between dimensional and traditional OLTP design practices.

Since Data Model is developed data is coming in. Data Management is a very important issue as well. By Data Management I mean such things like data storage solutions, fast data access problems and other problems similar to it. As data volumes accumulated by Data Warehouse generally are extremely large, a technical and architectural solution should be ready to process it. This part of the Data Warehousing definitely is a subject for following research as well.

As Data Warehousing products (reporting, analysing, data mining etc. tools) are required by business, it is very close to the particular end users. Of course, every company has its own specific requirements, but there are some common specificities from industry to industry. Financial industry specificities and possible solutions to them are good area for research and for having key knowledge too.

References

Kimball R., Reeves L., Ross M., Thornthwaite W. (1998), The Data Warehouse Lifecycle Toolkit, John Wiley & Sons, Inc.

Groth R. (1998), Data Mining, Prentice Hall PTR

Inmon W. H., (1999), Building the Operational Data Store – Second Edition, John Wiley & Sons, Inc.

Oracle Corporation, (1996 2000), Oracle Documentation Library, Release 8.1.7

Kimball R., Merz R. (2000), The Data Webhouse Toolkit. John Wiley & Sons, Inc.

Marco D. (2000), Building and Managing the Meta Data Repository, John Wiley & Sons, Inc.

Todman C. (2001), Designing a Data Warehouse, Hewlett-Packard Company

Greenberg P. (2001), CRM at the Speed of Light, McGrow-Hill Companies

Imhoff C., Loftis L., Geiger J.G. (2001), Building the Customer-Centric Enterprise, John Wiley & Sons, Inc.

Kimball R., Ross M. (2002), The Data Warehouse Toolkit – Second Edition, John Wiley & Sons, Inc.

LATVIJAS UNIVERSITATES RAKSTI. 2004. 669. sēj.: DATORZINĀTNE UN INFORMĀCIJAS TEHNOLOĢIJAS, 53.-60. lpp.

Generic Data Representation by Table in Metamodel Based Modelling Tool

Edgars Celms
Institute of Mathematics and Computer Science, University of Latvia
Raina bulv. 29, LV-1459, Riga, Latvia
Edgars.Celms@mii.lu.lv

The foundation of a metamodel based modelling tool is its flexible facility to declaratively define the modelling method, notation and tool support. One of the problems for such tools is generic data representation by table. The paper proposes a method for declarative definition of a table based on the logical metamodel. The most interesting aspect of this definition is the specification of element selection criterion. Some practical examples of table specification by the described method are also explained.

Key words: modelling tool, generic table editor, metamodel, editor definition.

1. Introduction

Today there are a lot of modelling tools on the market. Modelling tools are designed to provide all that is necessary to support major areas of modelling, including business process modelling, object-oriented and component modelling with UML[1], relational data modelling, and structured analysis and design, etc. Why is it not sufficient to use "hard-coded" modelling tools? Let us consider, for example, the situation in business modelling. On the one hand there exist several well-known business modelling languages (IDEF3[2], ARIS[3] etc.), each with a set of tools supporting it. But there are also Activity diagrams in UML, whose main role now is to serve business modelling. There is GRADE BM [4] a specialized language for business modelling and simulation. Thus for the area of business modelling there is no one best or most used language or tool, and each of them emphasizes its own aspects. For example, GRADE BM presents very convenient facilities for specifying performers of a task and its triggering conditions. However any new language feature does not come for free, and the language becomes more complicated for use. Therefore one universal business modelling language, which would support all wishes, would become extremely difficult for use in simple cases. UML for this situation offers one ingenious solution – stereotypes for adjusting the modelling language to a specific area. In many cases the idea works perfectly, and it is well supported in several tools including GRADE. The latest version of UML - 1.4 extends the notion of stereotype, by assigning tagged values to it and grouping stereotypes into profiles (thus actually extending the metamodel). But currently no tool fully supports it and already a new version of UML 2.0 is coming with significant changes, particularly in the area of Activity diagrams.

The problem with flexible modelling environment is even more urgent for domain-specific modelling, where countless special notations are used for separate domains.

There are probably several ways to solve such a problem.

You can develop a modelling tool specially for any specific modelling method but this way can be very time and cost consuming.

You can make a tool as universal as possible to support all needs. For example, there is ARIS tool by IDS prof. Scheer, whose "home notation" is the ARIS BM language. But it supports also the UML notation with Activity diagrams, as well as numerous modifications of the main process notation via eEPC (office processes, industrial processes etc). In general, ARIS tool can be characterised to be extremely "wide", with about 110 different types of diagrams, and frequently having about 100 different symbol types per diagram (most, in fact, are predefined stereotypes of the basic ones). At the same time, there are practically no facilities for defining new stereotypes. In practice such a universal tool is difficult for use in simple cases.

An alternative way is a completely metamodel based generic modelling tool (previously called metaCASE). Such a tool has no built-in modelling methodology. It has to be filled up with specific metamodel and additional information to start modelling something.

2. Metamodel Based Modelling Tool

In this paper some aspects of the metamodel based approach to building flexible modelling environments are explained. The metamodel concept has become popular in recent years especially due to the principle used in UML definition [1]. What is a metamodel in a metamodel based modelling tool? To put it in short, a metamodel is a class diagram containing all modelling concepts, their attributes and their relationships. A metamodel based tool has no built-in modelling methodology. It has to be filled up with specific metamodel and additional information to start modelling something.

An earlier alternative name for the approach is metaCASE. Let me mention the key research in this area. Perhaps, the first similar approach has been made by Metaedit [5], but for a long time its editor definition facilities have been fairly limited. The latest version named Metaedit+ now can support definition of most used diagram types, but via very restricted metamodelling features (non-graphical), the resulting diagrams corresponding to the simplest concept of labelled directed graph. The most flexible definition facilities (and some time ago, also the most popular in practice, but now the tool is out of the market) seem to be offered by the Toolbuilder by Lincoln Software. Being a typical metaCASE of the early nineties, the approach is based on ER model for describing the repository contents and on special extensions to ER notation for defining derived data objects which are in a one-to-one relation to objects in a graphical diagram. A more academic approach is that proposed by Kogge [6], with a very flexible, but very complicated and mainly procedural editor definition language. Other newer similar approaches are proposed by ISIS GME [7], DOME [8] and Moses [9] projects, with main emphasis for creating environments for domain-specific modelling in the engineering world. The richest editor definition possibilities of them are in GME. Several commercial modelling tools (ARIS by IDS prof. Scheer, System Architect by Popkin Software[10], etc.) use a similar approach internally, for easy customisation of their products, but their tool definition languages have never been made explicit.

The approach explained in this paper is in a sense a further development of the above mentioned approaches. In this approach at first the domain metamodel of the modelling area (logical metamodel) is built. Then the modelling method, notation and tool support are defined declaratively, by means of a special metamodelling

environment. These activities also require extension of the metamodel (adding some presentation classes), but it is not relevant to the purposes of this paper.

A very significant part of the tool support in metamodel based modelling tools is flexible data representation facilities. There are some main principles how data can be represented in modelling tools:

- Diagrammatic,
- Hierarchical (e.g. "tree views"),
- Data dictionaries,
- Tables,
- Object editors.

Obviously the most popular data representation form (modelling notation) in any domain now is diagram based. Therefore a very important part of the metamodel based approach is the Editor Definition Language (EdDL) for a simple and convenient definition of a wide spectrum of diagrammatic graphical editors [11, 12]. Nevertheless not all information is convenient to represent by diagrams. There is also the necessity to have a possibility in metamodel based modelling tools to define such facilities as flexible model content browsing and flexible definition of an editor for a single metamodel class instance. Perhaps one of the most undervalued data representation manner in modelling tools today is data representation by table. For example, in Rational Rose [13] UML tool, in real size models it is complicated to browse through packages and classes in the model tree. Frequently a package contains more than ten class diagrams with ten to thirty classes in each. That leads to a situation where there are hundreds of classes at one tree level and it is not easy to find the right one. It would be reasonable to represent such uniform information not in hierarchical (tree) form but in some easily configurable tabular form. Some tools (System Architect) have something like tables but with a very restricted functionality (reports only). More or less usable in practice tables are implemented in GRADE tool but they are "hard-coded".

Although all of the data representations principles mentioned here are interesting problems in relation to metamodel based modelling tools, this is out of scope for this paper to explain all of them. The paper introduces a method for declarative definition of a table, based on the logical metamodel.

Partly the described approach has been developed within the EU ESPRIT project ADDE [14], see [15] for a preliminary report.

3. Generic Data Representation by Table

The foundation of a metamodel based modelling tool is its flexible facility to define declaratively the modelling method, notation and tool support. One of the problems for such tools is generic data representation by table.

The paper proposes a method for table editor definition based on the logical metamodel. Certainly, the logical metamodel for the selected modelling method should be defined before we start defining any of the editors used for the tool. This logical structure is described by a UML class diagram [1]. Fig.1 shows an example of a logical metamodel for UML class diagram (here the UML class metamodel is described by a

UML class diagram). This example will be used in the paper to demonstrate the table editor definition features.

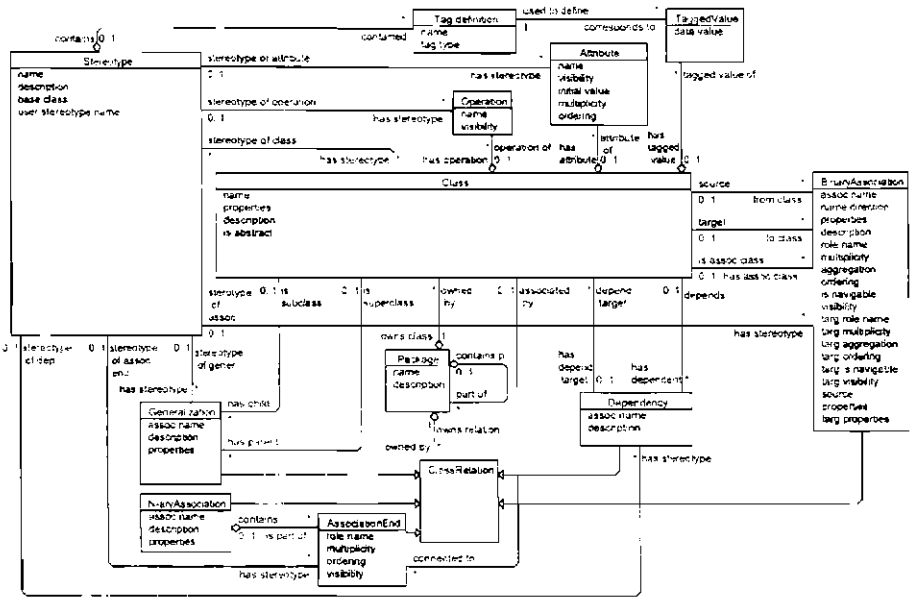

Figure 1. Logical metamodel example (fragment)

A table editor definition for a new table starts with the selection of the domain objects to be represented by the table. It should be emphasized that the logical metamodel itself is never modified during this process. Our goal is to define the corresponding table editor, which in this case should be able to present the selected set of metamodel class instances in a tabular form. For example, Fig.2 shows a table which represents all instances of Class in some model. Classes are grouped by the association owned_by ("Is in Package" column in the table) and ordered by the Name attribute.

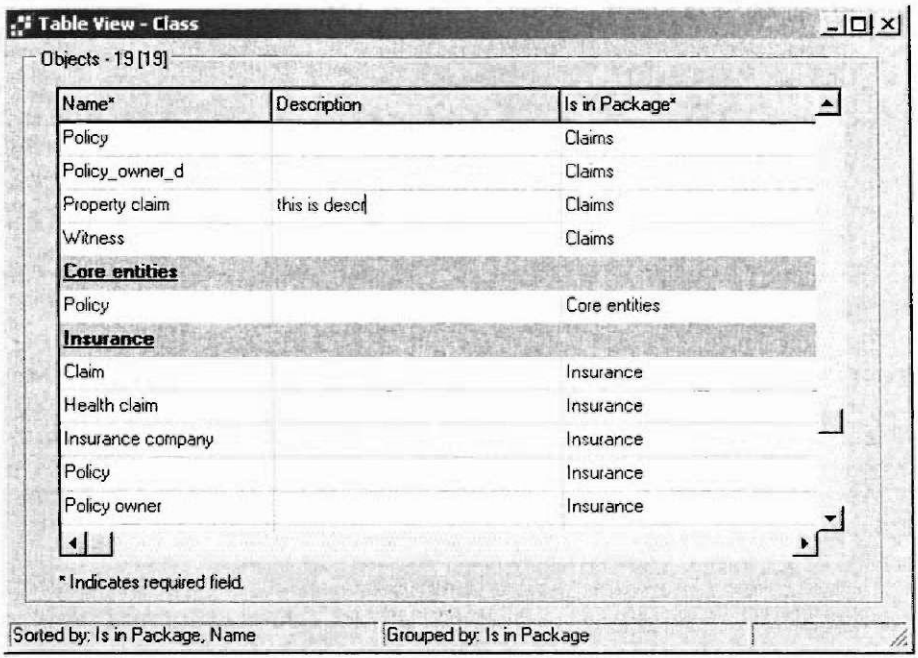

Figure 2. Class table example

Definition of a table consists essentially of two sections: how and what data should be represented. How – it means to define such things as:

- Table columns properties – the user should be allowed to define which columns are visible in the table (based on the metamodel elements). For example, the Class table in Fig.2 is configured to show three columns, which correspond to two attributes (*name* and *description*) and one association (*owned_by*) of the class *Class* in the metamodel. Other attributes and associations of the *Class* are not visible in this table. The user should be also able to define for each column in the table such properties as column title, column edit mode (either it is allowed to do in-place editing or some "native" object editor should be invoked), column ordering and grouping (by which column(s) the table is ordered or grouped), etc.
- Prompting/display form for associations – it is a very important feature to make the table more usable for end users. For example, the association column *owned_by* for the table in Fig.2 is defined to be shown as the value of the attribute *name* of *Package*. In other words, it is the facility to define the text assembly rules for columns, which correspond to associations in the metamodel. The user can define the text to display as a concatenation of segments, each consisting of a constant prefix, a variable part and a constant suffix. Each variable part is defined as the object attribute value located at the end of the chain defined by associations (for example, *Class.owned_by.name*). The Association chain may be empty, or it must start at the metamodel class corresponding to the objects in the table. Attribute must belong to the metamodel class at the end object of the associations chain.

- Navigation facilities for objects, attributes and associations – what to do in response to user activities. For example, what to do on single or double mouse click on some object attribute or association.
- Popup menu configuration – when defining popup menus for table objects the user should be allowed to define menu items at least for the following actions: *create* a new object (for example, "New Class"), *delete* the object represented in the table, *open* an object viewer/editor ("Properties..."), *copy* and *paste*, *execute* any other program, etc.

The above described facilities for table definition allow the user to define nearly any reasonable table form for practical use.

The second part of the definition of the table is used to specify what data should be represented by the table. The specification of the element selection criterion (selection rule) is the most non-trivial and the most interesting aspect of the table definition. A selection rule can be specified, saying which of the possible instances actually must appear in the table. To be short, a selection rule is a Boolean expression (AND, OR, NOT) on link conditions (specifying that a certain sequence of metamodel associations must link the given object to another object) and attribute value conditions (defined by an association chain, attribute at its end and a fixed attribute value). The actual expression language contains some more details (including simple existential quantifiers).

Examine one more example of the table. Let's assume that we need to define a table which contains Stereotypes of Classes, which are contained in one concrete Package and these Stereotypes do not have Tag_definition (see Fig.1). There are languages where it is possible to define selections or assertions of such a kind. Such languages, for example, are Object Query Language (OQL[16]), which is used for querying an Object Database or Object Constraint Language (OCL[1]), which is used in the UML to specify the well-formedness rules of the metaclasses comprising the UML metamodel. In OQL and OCL the above mentioned selection criterion can be written as (where the concrete Package is given by the parameter curr_package):

```
In OQL: select st from Stereotype st
            where curr_package in st.stereotype_of_class.owned_by
              andthen is_undefined(st.contains)
In OCL: self.stereotype_of_class.owned_by->exists(pck:Package |
              pck = curr_package) and self.contains->IsEmpty()
```

However in practice both these languages are difficult to use for declarative definition of the tables. OQL is a very powerful and extensive language. OCL is a very concise and rather complicated to use language and it is not intended for querying but for specifying static constraints in a metamodel. In order to use OQL or OCL you need to know precise semantics of the languages.

Let's say, that the paper proposes a method to specify a selection rule for a table, which is a "reasonable subset" of OQL or OCL with a graphical realization (see Fig.3). "Reasonable subset" in this case means – such a subset, which is sufficient for practical use and allows also an efficient implementation.

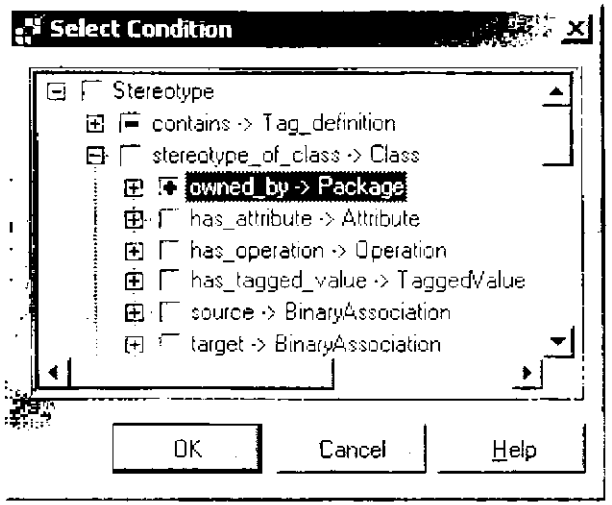

Figure 3. Selection criterion definition example

The definition of the selection rule is done by tree. The tree is some view (fragment) of the metamodel where the root node corresponds to the metamodel class for which we want to define the table (in our example the *Stereotype* class). Every next level of the tree contains nodes (classes) which are linked in the metamodel with associations to the parent node (class) in the tree. Fig.3 gives an example of such a tree where the above mentioned selection criterion is defined. The tree row marked by the red "minus" sign means that there might not be a link to a *Tag_definition* instance.

In other words, according to this approach it is possible in an easy, usable manner to compose textual selection expression, which is based on metamodel elements. We can also specify some additional constraints for attribute values for classes.

It should be emphasized, that unlike OQL or OCL, in the proposed method it is not important to "know" such technical issues as cardinalities of associations or kind of collections (bag, set, list in OQL or bag, set, sequence in OCL). Therefore the proposed method is easy to use in practice. Practical experiments have demonstrated that the described metamodel based table definition method has a realistic implementation and can reach an industrial quality of the defined editors.

4. Conclusions

The approach to table definition described in this paper permits us to define nearly any reasonable table for practical use, and the described method also allows an efficient implementation (in the sense of run-time necessary to build the tables from real amounts of data). Practical experiments have demonstrated that the table editors obtained by the described definition method can reach an industrial quality. However this approach can be made significantly more universal and applicable not only to tables but also for other similar data representations, since it is generally accepted that a metamodel (class diagram) can be used for describing the logical structure of nearly any system.

References

[1] Rumbaugh, J., Booch, G., Jackobson, I.: The Unified Modeling Language Reference Manual. Addison-Wesley (1999)

[2] Mayer, R., Menzel, C., Painter, M.: IDEF3 Process Description Capture Method Report http://www.idef.com/Downloads/pdf/Idef3_fn.pdf, Knowledge Based Systems Inc (1995)

[3] Scheer, A.-W.: ARIS Business Process Modeling. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (2000)

[4] Barzdins, J., Tenteris, J., Vilums, E.: Business Modeling Language GRAPES-BM (Version 4.0) and its Application. Dati, Riga (1998)

[5] Smolander, K., Martiin, P., Lyytinen, K., Tahvanainen, V-P.: Metaedit – a flexible graphical environment for methodology modelling. Springer-Verlag (1991)

[6] Ebert, J., Suttenbach, R., Uhe, I.: Meta-CASE in Practice: a Case for KOGGE. Proceedings of the 9th International Conference, CAiSE'97, Barcelona, Catalonia, Spain (1997)

[7] Ledeczi A., Maroti M., Bakay A., Karsai G., Garrett J., Thomason IV C., Nordstrom G., Sprinkle J., Volgyesi P.: The Generic Modeling Environment. Workshop on Intelligent Signal Processing, Budapest, Hungary, May 17, 2001 (available in PDF at http://www.isis.vanderbilt.edu)

[8] DOME Users Guide, http://www.htc.honeywell.com/dome/support.htm

[9] Esser, R.: The Moses Project. http://www.tik.ee.ethz.ch/~moses/MiniConference2000/pdf/Overview.PDF

[10] Popkin Software: System Architect. http://www.popkin.com/products/system_architect.htm

[11] Audris Kalnins, Janis Barzdins, Edgars Celms, Lelde Lace, Martins Opmanis, Karlis Podnieks, Andris Zarins.: The First Step Towards Generic Modelling Tool. Proceedings of Baltic DB&IS 2002, Tallinn, 2002, v. 2, pp. 167-180.

[12] Audris Kalnins, Karlis Podnieks, Andris Zarins, Edgars Celms, Janis Barzdins.: Editor Definition Language and its Implementation. Proceedings of the 4th International Conference "Perspectives of System Informatics", Novosibirsk, 2001, LNCS, v.2244, pp. 530 – 537

[13] Quatrani, T.: Visual Modeling with Rational Rose 2002 and UML. 3rd edn. Addison Wesley Professional, Boston (2002)

[14] ESPRIT project ADDE. http://www.fast.de/ADDE

[15] Sarkans, U., Barzdins, J., Kalnins, A., Podnieks, K.: Towards a Metamodel-Based Universal Graphical Editor. Proceedings of the Third International Baltic Workshop DB&IS, Vol. 1. University of Latvia, Riga, (1998) 187-197

[16] Cattel, R.G.G., Barry, D.: The Object Database Standard: ODMG 3.0. Morgan Kaufmann (2000)

LATVIJAS UNIVERSITĀTES RAKSTI. 2004. 669. sēj.: DATORZINĀTNE UN INFORMĀCIJAS TEHNOLOĢIJAS. 61.–79. lpp.

Nondifferentiable Optimization Based Algorithm for Graph Ratio-Cut Partitioning

Kārlis Freivalds

University of Latvia, Institute of Mathematics and Computer Science
Raina blvd. 29, LV-1459, Riga, Latvia
Karlis.Freivalds@mii.lu.lv, Fax: +371 7820153

We propose a new method for finding the minimum ratio-cut of a graph. Ratio-cut is an NP-hard problem for which the best previously known algorithm gives an $O(\log n)$-factor approximation by solving its dually related maximum concurrent flow problem. We formulate the minimum ratio-cut as a certain nondifferentiable optimization problem, and show that the global minimum of the optimization problem is equal to the minimum ratio-cut. Moreover, we provide strong symbolic computation based evidence that any strict local minimum gives an approximation by a factor of 2. We also give an efficient heuristic algorithm for finding a local minimum of the proposed optimization problem based on standard nondifferentiable optimization methods and evaluate its performance on several families of graphs. We achieve $O(n^{1.6})$ experimentally obtained running time on these graphs.

Key words: graphs, heuristic graph algorithms, minimum ratio-cut.

Introduction

Balanced cuts of a graph are difficult computation problems that are important both in theory and practice. Ratio-cut is the most fundamental one since most of the others including minimum quotient cut, minimum bisection, and multi-way cuts, can be easily approximated using it [11, 17]. Also several other important approximation algorithms, such as crossing number and minimum cut linear arrangement are based on the ratio-cut [11]. Ratio-cut has many practical applications, the most important being VLSI design, clustering and partitioning [20, 12, 1].

Since ratio-cut is an NP-hard problem [13], we must seek approximation algorithms to solve it in practically reasonable time. Many purely heuristic algorithms were developed [20, 22, 18, 6] most of them relying on simulated annealing, spectral methods or iterative movement of nodes from one side of the partition to the other. A common idea exploited by several authors [22, 2, 7, 18, 19] to improve their quality is using multi-scale graph representation usually obtained by edge contraction. At first a partition at the coarsest scale is obtained and then refined to a more detailed one by one of mentioned algorithms. Although these algorithms may perform well in practice, no optimality bounds are known for them.

The best previously known algorithm with proven optimality bounds finds an $O(\log n)$-factor approximation, where n is the number of nodes in the graph. It is based on reduction of the ratio-cut problem to the multi-commodity flow problem, which can be solved with polynomial time linear programming methods. Unfortunately, this method is not practical, since the resulting linear program is of quadratic size of the number of nodes in the graph and cannot be solved efficiently. Then, approximation algorithms

[15, 16, 10, 21] were discovered for the multi-commodity flow problem itself making this approach usable in practice. Several heuristic implementations [15, 9, 21] are based on this idea, some of them quite effective and practical. The most elaborate one [9] can deal with up to 100000-node graphs in reasonable time.

In this paper we propose a new way of finding the minimum ratio-cut of a graph. We construct a nondifferentiable optimization problem whose minimum solution equals the minimum ratio-cut value and use nonlinear programming methods to search for it. Since the problem is non-convex, we may find only a local minimum. However, we show that any strict local minimum gives us a factor of 2 approximate cut. For that purpose we introduce a notion of locally minimal ratio cut for which no subset of nodes taken from one side of the cut and moved to the other side decrease the cut value. We establish a one-to-one correspondence between strictly locally minimal cuts and strict local minima of the proposed optimization problem.

The reduction of an NP-hard discrete problem to a continuous one is not a novel idea. For example the maximum clique problem of a graph can also be stated as an optimization problem [24] and numerical optimization methods for finding the optimum may be used. However, for the maximum clique problem no optimality bounds of a local minimum are known.

To show that our method is practical we present an efficient heuristic algorithm for finding a local minimum of the proposed problem, which is based on the standard methods of nondifferentiable optimization and analyze its performance on several families of graphs. With the proposed method we can find a good partition of 200000-node graphs in less than one hour.

Problem Formulation

We are dealing with an undirected graph $G = (V, E)$, where V is its node set and E is its edge set. The nodes of the graph are identified by natural numbers from 1 to n. Each node i has a weight $d_i \geq 0$, and each edge (i, j) has a capacity $c_{ij} \geq 0$, satisfying the properties $c_{ij} = c_{ji}$, $c_{ii} = 0$. We define $c_{ij} = 0$ when there is no edge (i, j) in G. We assume that there are at least two nodes with non-zero weights. (A, A') denotes a cut that separates a set of nodes A from its complement $A' = V \setminus A$. The capacity of the cut $C(A, A')$ is the sum of edge capacities between A and A'. The *ratio* of the cut $R(A, A')$ is defined as follows

$$R(A, A') = \frac{C(A, A')}{\sum_{i \in A} d_i \cdot \sum_{i \in A'} d_i} \tag{1}$$

We will focus on finding the minimum ratio-cut i.e. the cut (A, A') with the minimum ratio over all non-empty A, A'.

Definition 1. A cut (A, A') is locally minimal if for all non-empty $U \subset A$, $U \neq A$: $R(A, A') \leq R(A \setminus U, A' \cup U)$ and for all non-empty $U' \subset A'$, $U' \neq A'$: $R(A, A') \leq R(A \cup U', A' \setminus U')$. Similarly we call a ratio-cut strictly locally minimal if strict inequalities hold in this definition.

Ratio-Cut as an Optimization Problem

We can assign a variable $x_i \in \mathbf{R}$ to each node i and consider the following optimization problem over x from $\mathbf{R}^n$.

$$\min F(x) = \sum_{i,j \in V} c_{ij} |x_i - x_j| \tag{2}$$

$$\text{subject to } H(x) = \sum_{i,j \in V} d_i d_j |x_i - x_j| = 1. \tag{3}$$

$$\sum_{i \in V} x_i = 0. \tag{4}$$

We are going to show that this optimization problem is equivalent to the ratio-cut problem in the sense described below.

Definition 2. A characteristic vector x^A for a cut (A, A') is defined such that its components

$$x^A_i = \begin{cases} a, & \text{if } i \in A \\ b, & \text{if } i \in A' \end{cases}, \text{ where} \tag{5}$$

$$a = -\frac{1}{2\sum_{i \in A} d_i \cdot \sum_{i \in A'} d_i} \cdot \frac{|A'|}{|V|}, \; b = \frac{1}{2\sum_{i \in A} d_i \cdot \sum_{i \in A'} d_i} \cdot \frac{|A|}{|V|}.$$

It is straightforward from this definition that for a cut (A, A') x^A satisfies the constraints (3, 4), and $F(x^A) = R(A, A')$.

Definition 3. For some feasible x and some $p \in \mathbf{R}$ we call the cut (P, P') positional, if $P = \{i \mid x_i \le p\}$, $P' = V \setminus P$, and both P and P' are non-empty. The ratio of this cut $R_p(x) = R(P, P')$. For a fixed x we can speak of minimum positional cut $R_{\min}(x)$ over all possible positional cuts obtained for different p:

$$R_{\min}(x) = \min_{p \in \mathbf{R}} R_p(x).$$

Theorem 1. For each feasible x^* of (2, 3, 4) $F(x^*) \ge R_{\min}(x^*)$. Also $F(x^*) > R_{\min}(x^*)$ if there are at least two positional cuts with different values.

Proof. Let us partition all nodes into three sets U_1, U_2, U_3 as follows.

$$y_1^* = \min_i x_i^*,$$

$$y_2^* = \min_{i: x_i^* > y_1^*} x_i^*,$$

$$U_1 = \{i \mid x_i^* = y_1^*\}$$

$$U_2 = \{i \mid x_i^* = y_2^*\}$$

$$U_3 = \{i \mid x_i^* > y_2^*\}$$

If U_3 is empty there is exactly one positional cut, x^* is in the form of characteristic vector and $F(x^*) = R_{\min}(x^*)$ that concludes this proof, otherwise define

$$y_3^* = \min_{i:x_i^* > y_2^*} x_i^* .$$

We create a sub-problem of (2, 3, 4) by reducing the original one to a new variable $y = (y_1, y_2, y_3)$.

$$\min F_1(y) = 2 \sum_{i\in U_1, j\in U_2} c_{ij}\left|y_2 - y_1\right| + \sum_{i\in U_1, j\in U_3} c_{ij}\left|y_3 + l_j - y_1\right| + \sum_{i\in U_2, j\in U_3} c_{ij}\left|y_3 + l_j - y_2\right| + K, \tag{6}$$

subject to

$$H_1(y) = 2 \sum_{i\in U_1, j\in U_2} d_i d_j\left|y_2 - y_1\right| + \sum_{i\in U_1, j\in U_3} d_i d_j\left|y_3 + l_j - y_1\right| + \sum_{i\in U_2, i\in U_3} d_i d_j\left|y_3 + l_j - y_2\right| + P = 1, \tag{7}$$

$$\left|U_1\right| y_1 + \left|U_2\right| y_2 + \left|U_3\right| y_3 = 0, \tag{8}$$

where constants $l_i = x_i^* - y_3^*$,

$$K = \frac{1}{2} \sum_{i\in U_3, j\in U_3} c_{ij}\left|x_i^* - x_j^*\right|$$

$$P = \frac{1}{2} \sum_{i\in U_3, j\in U_3} d_i d_j\left|x_i^* - x_j^*\right|$$

Next, we further restrict it with $y_1 \le y_2 \le y_3$ and can drop the absolute value signs:

$$\min F_2(y) = 2 \sum_{i\in U_1, j\in U_2} c_{ij}(y_2 - y_1) + \sum_{i\in U_1, j\in U_3} c_{ij}(y_3 + l_j - y_1) + \sum_{i\in U_2, j\in U_3} c_{ij}(y_3 + l_j - y_2) + K, \tag{9}$$

subject to

$$H_2(y) = 2 \sum_{i\in U_1, j\in U_2} d_i d_j(y_2 - y_1) + \sum_{i\in U_1, j\in U_3} d_i d_j(y_3 + l_j - y_1) + \sum_{i\in U_2, j\in U_3} d_i d_j(y_3 + l_j - y_2) + P = 1 \tag{10}$$

$$|U_1|y_1 + |U_2|y_2 + |U_3|y_3 = 0, \tag{11}$$

$$y_1 \le y_2 \le y_3. \tag{12}$$

We have obtained a locally equivalent linear program in the sense that for y^* the constraints (10, 11, 12) are satisfied and $F(x^*) = F_2(y^*)$. Also, if we can find a better solution for (9–12) we can substitute the result back to the original problem giving a better feasible solution for it.

From the linear programming theory it is known that we can find the optimal solution by examining the vertices of the polytope defined by the constraints – in our case that means when one of the inequalities in (12) is satisfied as equality. Let us examine both cases:

case $y_1 = y_2$:

$$H_2(y_1, y_1, y_3) =$$

$$2\left[(y_3 - y_1)\left(\sum_{i\in U_1, j\in U_3} d_i d_j + \sum_{i\in U_2, j\in U_3} d_i d_j\right) + \sum_{i\in U_1, j\in U_3} d_i d_j l_j + \sum_{i\in U_2, j\in U_3} d_i d_j l_j + P\right] =$$

$$= 2(y_3 - y_1) \sum_{i\in U_1\cup U_2, j\in U_3} d_i d_j - L - 1,$$

where $L = 2 \sum_{i\in U_1\cup U_2, j\in U_3} d_i d_j l_j + P$,

$$F_2(y_1, y_1, y_3) = 2\left[(y_3 - y_1) \sum_{i\in U_1\cup U_2, j\in U_3} c_{ij} + \sum_{i\in U_1\cup U_2, j\in U_3} c_{ij} l_j + K\right].$$

By substituting $y_3 - y_1$ from previous equation we get

$$F_2(y_1, y_1, y_3) = \frac{1 - L}{\sum_{i\in U_1\cup U_2, j\in U_3} d_i d_j} \sum_{i\in U_1\cup U_2, j\in U_3} c_{ij} + \sum_{i\in U_1\cup U_2, j\in U_3} c_{ij} l_j + K =$$

$$(1 - L)\, R(U_1 \cup U_2, U_3) + B,$$

where $B = 2 \sum_{i\in U_1\cup U_2, j\in U_3} c_{ij} l_j + K$.

case $y_2 = y_3$:

Similarly we get $F_2(y_1, y_2, y_2) = (1 - L)\, R(U_1, U_2 \cup U_3) + B$.

We chose the solution y with $y_1 = y_2$ if $R(U_1 \cup U_2, U_3) < R(U_1, U_2 \cup U_3)$, otherwise choose the solution with $y_2 = y_3$. It is evident that if both these ratio costs are non-equal we get a strictly smaller function value. We substitute the solution back into the original problem obtaining a new x. x is a feasible solution of (2, 3, 4) with a smaller or equal function value and the set U_2 merged to U_1 or U_3. We repeat the described process until U_3 is empty.

Let us analyze the resulting x and sets U_1 and U_2. We have $F(x) = R(U_1, U_2)$, where (U_1, U_2) is some positional cut of x^* (in fact the minimal one), hence $F(x^*) \ge F(x) \ge$

$R_{min}(x^*)$. If we had some non-equal cuts compared during the process, we would have a strict decrease in the function and hence the second statement of the theorem holds.

□

Definition 4. A feasible x^* is a local minimum of (2, 3, 4) if there exists $\varepsilon > 0$ such that $F(x^*) \le F(x)$ for each x satisfying (3, 4) and $|x - x^*| < \varepsilon$.

Definition 5. A feasible x^* is a strict local minimum of (2, 3, 4) if there exists $\varepsilon > 0$ such that $F(x^*) < F(x)$ for each $x \neq x^*$ satisfying (3, 4) and $|x - x^*| < \varepsilon$.

Theorem 2. Each strict local minimum x^* of (2, 3, 4) is in the form $x^* = x^A$ for some cut (A, A').

Proof. We need to prove that in a strict local minimum the expression (5) holds for some a and b, and the correct values for a and b are guaranteed by the constraints (3) and (4). Assume to the contrary that there are more than two distinct values for the components of x^*. We form the reduced problem as in Theorem 1 obtaining equations (9 – 12). We are able to do that since U_3 is non-empty due to our assumption.

From the linear programming theory a strict local minimum can only be on the vertex of the polytope defined by the constraints (10 – 12), however in our case y^* cannot be on the vertex since the constraints where equality holds at y^* are less then the number of variables. Consequently, x^* cannot be a strict local minimum for the original problem. Hence our assumption is false and the theorem is proven.

□

There is one-to-one correspondence between strictly locally minimal cuts and strict local minimums of (2, 3, 4) as stated in the following theorem.

Theorem 3. x is a strict local minimum of (2, 3, 4) if and only if x is a characteristic vector of some strictly locally minimal cut (A, A').

Proof. First we prove that the characteristic vector x^A of a strictly locally minimal cut (A, A') corresponds to a strict local minimum of (2, 3, 4). Assume to the contrary that x^A is not a strict local minimum point; then by Definition 5 there exists an arbitrary small vector $t \neq 0$ that leads to smaller or equal feasible solution $x = x^A + t$, $F(x) \le F(x^A)$. By the nature of the constraints (3, 4) there will be at least two positional cuts of x, and since t was sufficiently small, one of them is (A, A').

There are two cases:

(i) there exists a positional cut $R_p(x)$ with value not equal to $R(A, A')$. Then by Theorem 1 there is a positional cut with value $R_p(x) < F(x) \le F(x^A) = R(A, A')$ which divides the nodes into sets U_1 and U_2. Since t was sufficiently small, both U_1 and U_2 cannot contain nodes from both sets A and A' i.e. either $U_1 \subset A$, or $U_1 \subset A'$, or $U_2 \subset A$, or $U_2 \subset A'$ contradicting the local minimality of (A, A') by Definition 1.

(ii) all positional cuts of x are of the same value. Let us choose some positional cut $(U_1, U_2) \neq (A, A')$. And again, since we can choose t small enough, either $U_1 \subset A$, or $U_1 \subset A'$, or $U_2 \subset A$, or $U_2 \subset A'$ which contradicts strict local optimality of (A, A') by Definition 1.

Secondly, we prove that each strict local minimum x of (2, 3, 4) is in the form of a characteristic vector of a strictly locally minimal cut. From Theorem 2 we have that each strict local minimum is in the form of a characteristic vector x^A of some cut (A, A'). We need to show that this cut is strictly locally minimal. Assume to the contrary that there exist $S \subset A$, $S \neq 0$, $A \setminus S \neq 0$ for which $R(A \setminus S, A' \cup S) \le R(A, A')$. Denote $U = A \setminus S$.

Let us repeat the construction from the proof of Theorem 1 taking the sets U_1, U_2 and U_3 as S, U, A' correspondingly. We obtain equations (9 – 12) and y with $y_1 = y_2$. From Definition 5 each small move increases the function. Let us take such feasible move t that does not destroy the groups U_1, U_2 and U_3 but separates U_1 from U_2. Let t' be the projection of t onto the sets U_1, U_2, U_3 i.e. $t' = (t_k \mid k \in U_1, t_p \mid p \in U_2, t_r \mid r \in U_3)$. Let us denote $y''' = y + t'$, $y'' = y + \alpha t'$ where α is chosen such that $y''_2 = y''_3$. Since the constraints (10, 11) are linear and they have the same value for y and y''', they will have the same value for y'', which is a linear combination of y and y'''. Obviously y'' satisfies the constraint (12). Similar arguments hold for the function: since $F_2(y''') > F_2(y)$, also $F_2(y'') > F_2(y)$ from the linearity of (9). From y'' we can go back to the original problem and obtain x'', which is a characteristic vector of the cut $(A \setminus S, A' \cup S)$. Since $H(x'') = H_2(y'') = 1$ and $F(x'') = F_2(y'') > F_2(y) = R(A, A')$. $R(A \setminus S, A' \cup S) > R(A, A')$ which contradicts our assumption.

□

We shall note that also for each non-strictly minimal ratio cut, its characteristic vector x^A gives a local minimum of the function, however the converse is not true. There exist non-strict local minima of (2, 3, 4) with the function value not equal to any locally minimal cut value. In Fig. 3 a graph and their node coordinates in a local minimum of (2, 3, 4) are shown. It is a local minimum because no small move gives a decreased positional cut or a better function value. But none of its positional cuts are locally minimal.

Theorem 4. The minimum ratio-cut R is equal to the optimum solution of (2, 3, 4).

Proof. From Theorem 1 $F(x) \geq R_{\min}(x) \geq R$. For the characteristic vector x^A of the minimum ratio cut $F(x^A) = R$. The claim follows immediately. □

Theorem 5. Each locally minimal cut (A, A') is not greater than two times the minimum ratio cut.

Proof. Let us denote the optimal cut (B, B'). We form four sets $A \cap B$, $A \cap B'$, $A' \cap B$, $A' \cap B'$ shown in Fig. 1 From Definition 1 the ratio of each of the cuts $C_1 = (A \cap B, V - A \cap B)$, $C_2 = (A \cap B', V - A \cap B')$, $C_3 = (A' \cap B', V - A' \cap B')$, $C_4 = (A' \cap B, V - A' \cap B)$ is at least as large as ratio of $C_{12} = (A, A')$ and $C_{23} = (B, B')$.

We form a full graph by taking each of the sets $A \cap B$, $A \cap B'$, $A' \cap B$, $A' \cap B'$ as the nodes of the graph and assign edge capacities as the sum of all edge capacities in the original graph between the corresponding sets. We obtain

$$R_1 = \frac{c_1 + c_4 + c_6}{d_1(d_2 + d_3 + d_4)}, \qquad R_2 = \frac{c_1 + c_2 + c_5}{d_2(d_1 + d_3 + d_4)},$$

$$R_3 = \frac{c_2 + c_3 + c_6}{d_3(d_1 + d_2 + d_4)}, \quad R_4 = \frac{c_3 + c_4 + c_5}{d_4(d_1 + d_2 + d_3)},$$

$$R_{12} = \frac{c_2 + c_4 + c_5 + c_6}{(d_1 + d_2)(d_3 + d_4)}, \qquad R_{23} = \frac{c_1 + c_3 + c_5 + c_6}{(d_2 + d_3)(d_1 + d_4)},$$

$R_1 \geq R_{12}$, $R_2 \geq R_{12}$, $R_3 \geq R_{12}$, $R_4 \geq R_{12}$.

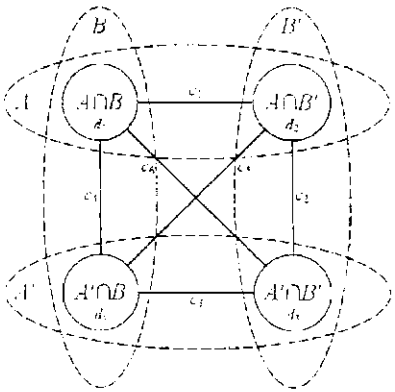

Fig. 1. Illustration of the proof of Theorem 5.

And the statement of the theorem to be proven translates to $\frac{R_{12}}{R_{23}} \le 2$. We verified this using symbolical computation in Mathematica 4.0. See Appendix A for details. □

Corollary 6. Each strict local minimum of (2, 3, 4) is not greater than two times the minimum ratio cut.

The proof is straightforward from Theorem 3 and Theorem 5.

The bound of Theorem 5 is tight; the graph with 4 nodes shown in Fig. 2 achieves this bound. One can make larger examples easily by substituting any connected graph with sufficiently high edge capacities in place of the nodes of the given graph.

The Algorithm

In this section we present a heuristic algorithm based on standard nondifferentiable optimization methods for finding a local minimum of (2, 3). Finding the minimum of a nondifferentiable function is one of several well-explored nonlinear programming topics [5, 23]. One of the possibilities is to approximate the nondifferentiable function with a

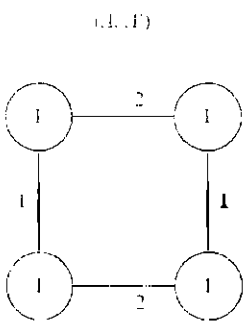

Fig. 2. A graph achieving the boundary of Theorem 5.

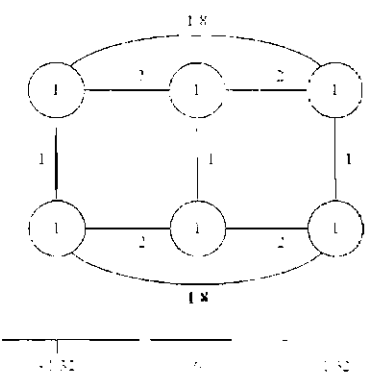

Fig. 3. A graph with assigned node coordinates in a local minimum of (2, 3, 4) where no positional cut is locally minimal.

smooth one and apply one of the well-known algorithms to find its local minimum [5, 3]. Often a better approach is to handle it directly. Indeed, in our case we obtain a very simple and fast algorithm.

Most of the optimization theory deals with convex problems for which algorithms with proven convergence can be developed. Many of these methods also work for non-convex functions finding a local minimum. However, then the convergence cannot be shown or can be shown only in a local neighborhood of some local minimum which is not satisfactory in our case. The very basic algorithm of nondifferentiable optimization is a *subgradient* algorithm [5, 23]. We will adopt it for solving our problem. Since we apply it to a non-convex problem, it should be considered mostly as heuristic, however practice shows that it actually converges to a local minimum of our problem.

The algorithm is an iterative one. The iteration of the algorithm consists of going in the negative direction of a subgradient of the function by a fixed step and then performing a projection onto the constraint (3). The constraint (4) was introduced for technical reasons required in proofs and may be not considered in the algorithm.

An appropriate subgradient q of F can be calculated with the following equation for its components

$$q_i = \sum_{j \in V} c_{ij} \operatorname{sign}(x_i - x_j),$$

$$\text{where } \operatorname{sign}(x) = \begin{cases} 1, x > 0 \\ 0, x = 0 \\ -1, x < 0 \end{cases}.$$

Choosing the right step size is crucial for the convergence speed. Our heuristic observation is that it should be proportional to the node spread. We choose the step equal to `stepFactor`$*(x_{\max} - x_{\min})$, where `stepFactor` is some parameter. Initially `stepFactor` $=$ `1/14`. Its update during the algorithm is be discussed later. The projection is performed by going in the direction of a subgradient of the constraint until the constraint is reached. For the constraint H we can write its subgradient r in a different form to allow its faster evaluation

$$r_i = d_i \sum_{j \in V} d_j \operatorname{sign}(x_i - x_j) = d_i \Big(\sum_{j: x_j < x_i} d_j - \sum_{j: x_j > x_i} d_j \Big).$$

If we sort the x_i values and consider them in increasing order, then the needed sums can be updated incrementally leading to linear time evaluation (not counting the sorting). To perform the projection we need to calculate the step length towards the constraint. To simplify the calculations we will assume that the ordering of x_i will not change during the projection. Then the constraint function H becomes linear and the desired step can be easily calculated from the linear equation defined as the point of value 1 on the ray defined by the subgradient from the current point. In the case of unit node weights our assumption is fulfilled. To see this we must observe that the step in the function will always lead to a point with $H < 1$. Then, if we have unit node weights, $r_i \leq r_k$ for all i and k satisfying $x_i \leq x_k$, so after the projection the distance between them can only increase thus keeping the same ordering. For the case of arbitrary node weights such an algorithm gives a usable approximation of the projection step length.

The whole algorithm starts with a random initialization. We assign a random position for each node such that $H < 1$ and then perform a projection to obtain a feasible starting position. We experimented also with several other initializations, but obtained significant improvements only for tree graphs. Since the optimal cut for trees can be found in linear time, we can construct a starting position that reveals it and the algorithm will only perform a few iterations to confirm that the solution does not improve. Hence to get a comprehensive picture of the algorithm behavior we decided to consider random initialization only.

After the initialization we perform iterations until convergence. To tell when the process is converged we track the minimum positional cut values obtained at each iteration. The same values will also tell us how to change the step size parameter `stepFactor`. If the new cut value is lower than the previous one, then we are making progress and we should continue with the same step size. If the value is higher than the previous one, then the step size is too large. If the cut does not change, then we have a clue that the process has converged. Of course we do not hurry to make decisions only from one iteration. Instead we wait for a certain number of iterations controlled by the constants `MAX_OSCILLATIONS` and `MAX_EQUAL` before making the decision. Such a delay also improves the convergence speed by allowing to iterate longer with a larger step size.

To determine the positional cut value at each iteration, we proceed as follows. We consider the sorted sequence of nodes, calculate the positional cut in each interval between two consecutive nodes, and take the minimum one. We can do it incrementally on the sorted sequence in time $O(n + m)$ provided by the sorting, where m is the number of edges in the graph.

As suggested in one of the exercises in [5], the performance of this algorithm can be improved by taking the previous directions into account. We add the previous direction to the current one reduced by some constant `REDUCTION_FACTOR` between 0 and 1. It models a heavy ball motion in the presence of a force in the direction of the subgradient. In our experiments such a modification with `REDUCTION_FACTOR = 0.95` performed substantially better.

All the steps described before can be implemented to run in time $O(n + m)$. Adding the time needed for sorting the nodes one iteration takes time $O(m + n \log n)$. The number of iterations is hard to estimate so we will provide experimental data in the next sections. The constants `MAX_OSCILLATIONS` and `MAX_EQUAL` have the most impact on the iteration count and also on the quality of the obtained cut. So we must select them carefully. After some experimentation we chose `MAX_EQUAL = 200` and `MAX_OSCILLATIONS = 30`.

```
algorithm ratio-cut
calculate a random feasible initial position
acum = 0
oscillationCounter = 0
equalityCounter = 0
stepFactor = 1/14
while (equalityCounter < MAX_EQUAL)

        x = x + acum
        d = direction(x)
```

```
        x = x - d
        acum = acum * REDUCTION_FACTOR - d

        if (minimum positional cut value has increased in this
iteration)
                equalityCounter = 0
                oscillationCounter ++
        or else (minimum positional cut value has decreased in
this iteration)
                equalityCounter = 0
        or else
                equalityCounter ++

        if (oscillationCounter > MAX_OSCILLATIONS)
                stepFactor /= 1.3
                oscillationCounter = 0

endwhile
end

function direction (x)
        d = subgradient(x)
        step = (x_max - x_min) * stepFactor
        x_1 = x + d*step
        x_2 = projection of x_1 on the constraint
        return x_2-x
end
```

The Data

We evaluated the proposed algorithm on three families of graphs: random cubic graphs, random geometric graphs and random trees. We considered only graphs with unit weight nodes and edges.

Random cubic graphs are potentially hard for ratio-cut algorithms, because in [11] it was shown that there is actually an $O(\log n)$ gap between the minimum ratio-cut and the maximum concurrent flow on these graphs. We generate them using the algorithm provided in [14].

Random geometric graphs are standard test suite for balanced cut problems used in several papers [9, 22]. To generate a geometric graph we place the nodes of the graph randomly in the unit square. Then we include an edge between each of two nodes that are within distance δ in the graph, where δ is the minimum value such that the resulting graph is connected.

Tree graphs are seemingly easy graphs because their optimal ratio-cut can be calculated in linear time. Also it is not hard to show that only one local minimum exists for the corresponding optimization problem. Nevertheless, it is an interesting family since our experiments indicate a slow convergence of our algorithm on these graphs. Also, we can compare our result with the optimal one. We generate random trees with the classical algorithm, where each tree is produced with the same probability. This

algorithm produces long and skinny trees, which are particularly difficult for our algorithm.

Experimental Results

We implemented the algorithm in C++ and evaluated its performance on a computer equipped with a Pentium III 800 MHz processor and 256 Mbytes of RAM. For each graph family we measured the running time in seconds, the number of iterations and the quality of the produced cuts. Since we did not know the exact cut values for random and geometric graphs, we evaluated how much the ratio-cut value decreases when we continue the algorithm for the same number of iterations as performed before termination. Measuring the decrease of the ratio-cut value we can estimate how far the result is from the optimal.

The algorithm was run on a series of graphs of exponentially increasing size from 100 up to 204800 nodes. Ten graphs were generated for each size and the results were averaged. The average node degree for all graph families is constant. Although we cannot specify the degree explicitly for geometric graphs, due to their nature it was about 10 on all instances. The experimental results are given in tables 1–3. For each graph size tables show the iteration count, the running time, the obtained ratio-cut value and the improved ratio-cut value when the iteration count is doubled. For tree graphs the optimal ratio-cut value is shown also. Let us discuss the results separately for each graph family.

Cubic Graphs

Although these graphs were suggested as difficult, the algorithm performed very well on them. It took on average about 35 minutes to partition the 204800 node graphs. The running time dependence on the graph size is shown in Fig. 4. When we approximated the running time with a function in a form $O(n^r)$ we get the asymptotical running time about $O(n^{1.6})$ on these graphs.

The algorithm seems to find a very close to optimal cut since after doubling the iteration count, the quality increased only by less than 0.4%. Even more, the quality improved for the larger graphs approaching 1 (see Fig. 7). Such behavior is not surprising since it is not hard to prove that the ratio of a cut (A, A') of a random cubic graph is $\frac{3}{n-1}$ on average independently of the sizes of A and A' (a similar proof for general random graphs is presented in [20]). Then it is unlikely that the minimum ratio cut will be much different from this average value.

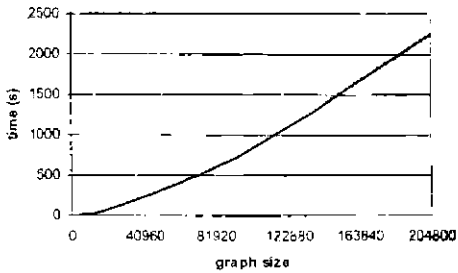

Fig. 4. Running time for cubic graphs.

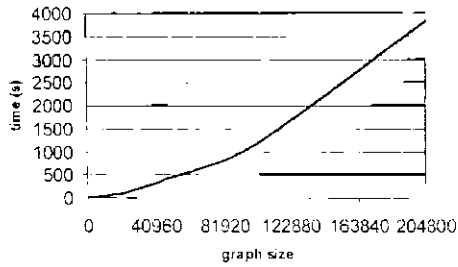

Fig. 5. Running time for geometric graphs.

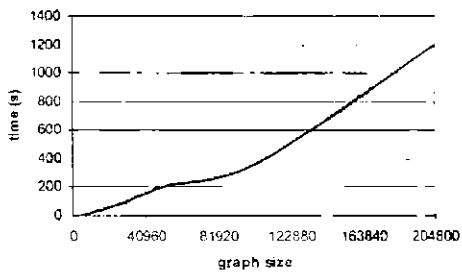

Fig. 6. Running time for tree graphs.

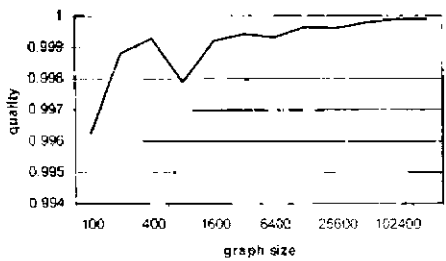

Fig. 7. Quality for cubic graphs.

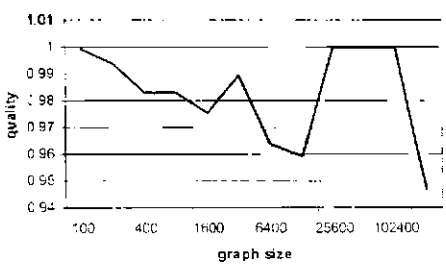

Fig. 8. Quality for geometric graphs.

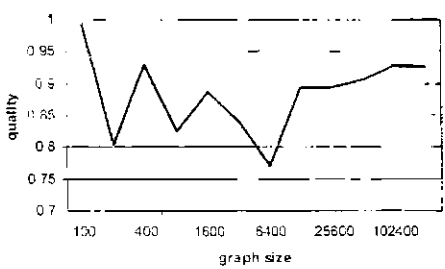

Fig. 9. Quality for tree graphs.

Geometric Graphs

The algorithm performed very well on these graphs both in terms of speed and quality. It took on average about 1 hour to partition the 204800 node graphs. The running time dependence on the graph size is shown in Fig. 5. The asymptotical running time behavior on these graphs was about $O(n^{1.64})$. After doubling the iteration count the quality increased by less than 5% (see Fig. 8). Also visually the cuts seemed the best possible. Fig. 10 shows a typical cut of a 10000-node graph.

Tree Graphs

The running time for trees was better than for other families. The largest graphs were partitioned in about 20 minutes. The running time dependence on the graph size is shown in Fig. 6. The asymptotical running time on these graphs was about $O(n^{1.45})$. However the quality was poor. As shown in Fig. 11 the obtained cuts were far from the optimal and the quality decreased with increasing graph size. Also doubling the iteration count showed 10% to 20% quality improvement (see Fig. 9).

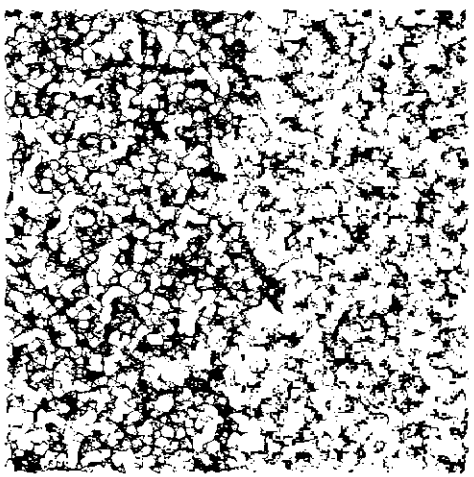

Fig. 10. The obtained cut for a 10000-node geometric graph.

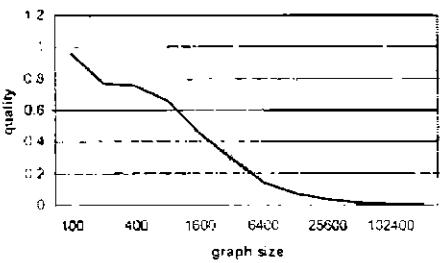

Fig. 11. Optimality for tree graphs

When we explored further the reason for the poor behavior, we found out that convergence is much slower than for other graph families and the stopping criterion does not work correctly in this case. When we allowed the algorithm to run for a sufficiently long time, it always found the optimum solution. However we did not find a robust stopping criterion that correctly works with tree graphs and does not increase running time much for other graph families. As already mentioned, a smarter initialization can be used to improve the quality of the partition if such tree or tree-like graphs are common for some application.

Conclusions and Open Problems

We have proposed a nondifferentiable optimization based method for solving the ratio-cut problem and presented a heuristic algorithm implementing it. We have shown that any strict local minimum is 2-optimal. The presented algorithm, however, in certain cases can find a non-strict minimum, but we can easily transform the obtained x vector into the characteristic form. Then the algorithm can be run again from this starting position and this process can be iterated until the result does not change giving a locally minimal cut, which by Theorem 5 is 2-optimal.

The obtained algorithm is simple and fast and uses an amount of memory that is proportional to the size of the graph. Its running time and quality are verified

experimentally. Its practical running time is about $O(n^{1.6})$ on our test data. The algorithm produces high quality cuts on random cubic and geometric graphs. On trees and other very sparse graphs the quality can be significantly improved by choosing a better starting position than a random one. We evaluated the algorithm on artificially generated data. As further study it would be important to evaluate its performance on real-life problems.

Although the algorithm performed well on most graphs, it is heuristic anyway. Is it an open question whether we can find a local minimum of (2, 3, 4) in polynomial time?

Acknowledgements

The author would like to thank Paulis Ķikusts and Krists Boitmanis for valuable discussion and help in preparation of this paper.

References

1. C. J. Alpert and A. Kahng, Recent Directions in Netlist Partitioning: a Survey, Tech. Report, Computer Science Department, University of California at Los Angeles, 1995.
2. C. J. Alpert, J.-H. Huang, and A. B. Kahng. Multilevel circuit partitioning. *Proc. Design Automation Conf*, pp. 530 533, 1997.
3. R. Baldick, A. B. Kahng, A. Kennings and I. L. Markov. Function Smoothing with Applications to VLSI Layout, *Proc. Asia and South Pacific Design Automation Conf.*, Jan. 1999.
4. J. W. Berry and M. K. Goldberg. Path Optimization for Graph Partitioning Problems. Technical report *TR: 95-34*, DIMACS, 1995.
5. D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.
6. L. Hagen and A. B. Kahng. New Spectral Methods for Ratio Cut Partitioning and Clustering. *IEEE Trans. Computer-Aided Design*. 11 (1992), pp. 1074 1085.
7. T. Hamada, C.-K. Cheng, P. M. Chau. An Efficient Multilevel Placement Technique using Hierarchical Partitioning. *IEEE Trans. on Circuits and Systems*, vol. 39(6) pp. 432 439, June 1992.
8. P. Klein, S. Plotkin, C. Stein, E. Tardos. Faster Approximation Algorithms for Unit Capacity Concurrent Flow Problems with Applications to Routing and Sparsest Cuts. *SIAM J Computing*, 3(23) (1994), pp. 466–488.
9. K. Lang and S. Rao. Finding Near-Optimal Cuts: An Empirical Evaluation. *4th. ACM-SIAM Symposium on Discrete Algorithms*, pp. 212 221, 1993.
10. T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, S. Tragoudas. Fast Approximation Algorithms for Multicommodity Flow Problems. *Journal of Computer and System Sciences*, 50(2), pp. 228 243, April 1995.
11. T. Leighton, S. Rao. Multicommodity Max-Flow Min-Cut Theorems and their use in Designing Approximation Algorithms. *Journal of the ACM*, vol 46 , No. 6 (Nov. 1999), pp. 787 – 832.
12. T. Lengauer. Combinatorial Algorithms for Integrated Circuit Layout. Stuttgart, John Wiley & Sons 1994.

13. D. W. Matula, F. Shahrokhi. Sparsest Cuts and Bottlenecks in Graphs. *Journal of Disc. Applied Math*., vol. 27 (1990), pp. 113–123.
14. A. Steger and N. C. Wormald. Generating Random Regular Graphs Quickly. *Combinatorics, Probab. and Comput.* vol. 8 (1999), pp. 377–396.
15. F. Shahrokhi, D. W. Matula. On Solving Large Maximum Concurrent Flow Problems. *Proceedings of ACM 1987 National Conference*. pp. 205–209.
16. F. Shahroki, D. W. Matula. The Maximum Concurrent Flow Problem. *Journal of the ACM*, vol. 37, pp. 318–334, 1990.
17. M. Wang, S. K. Lim, J. Cong, M. Sarrafzadeh. Multi-way Partitioning using Bi-partition Heuristics. *Proc. Asia and South Pacific Design Automation Conf.*, pp. 667–672, 2000.
18. Y. C. Wei, C. K. Cheng. An Improved Two-way Partitioning Algorithm with Stable Performance. *IEEE Trans. on Computer-Aided Design*, 1990, pp. 1502–1511.
19. Y. C. Wei, C. K. Cheng. A Two-level Two-way Partitioning Algorithm. *Proc. Int'l. Conf. Computer-Aided Design*, pp. 516–519, 1990.
20. Y. C. Wei, C. K. Cheng. Ratio Cut Partitioning for Hierarchical Designs. *IEEE Trans. on Computer-Aided Design*, vol. 10, pp. 911–921, July 1991.
21. C.-W. Yeh, C.-K. Cheng, and T.-T. Y. Lin. A Probabilistic Multicommodity-flow Solution to Circuit Clustering Problems. *IEEE International Conference on Computer-Aided Design*, pp. 428–431, 1992.
22. C.-W. Yeh, C.-K. Cheng, T.-T. Y. Lin. An Experimental Evaluation of Partitioning Algorithms. *IEEE International ASIC Conference*, P14-1.1–P14-1.4, 1991.
23. N. Z. Shor. Methods of Minimization of Nondifferentiable Functions and their Applications. Kiev, "Naukova Dumka" 1979. (in Russian).
24. I. Bomze, M. Budinich, P. Pardalos, and M. Pelillo. The Maximum Clique Problem. Handbook of Combinatorial Optimization, Volume 4. Kluwer Academic Publishers, Boston, MA, 1999.

Appendix A – Justification of Theorem 2.

Unfortunately all attempts proving this theorem analytically failed. We succeeded only in some special cases when all nodes or all edges have unit weight. Therefore we resorted to a "computer assisted proof".

The equalities can be equivalently written as:

$$R_1 = \frac{c_1 + c_4 + c_6}{g_1 + g_4 + g_6}, \qquad R_2 = \frac{c_1 + c_2 + c_5}{g_1 + g_2 + g_5},$$

$$R_3 = \frac{c_2 + c_3 + c_6}{g_2 + g_3 + g_6}, \qquad R_4 = \frac{c_3 + c_4 + c_5}{g_3 + g_4 + g_5},$$

$$R_{12} = \frac{c_2 + c_4 + c_5 + c_6}{g_2 + g_4 + g_5 + g_6}, \qquad R_{23} = \frac{c_1 + c_3 + c_5 + c_6}{g_1 + g_3 + g_5 + g_6},$$

$g_5 g_6 = g_2 g_4 + g_1 g_3$

$R_1 \geq R_{12}$, $R_2 \geq R_{12}$, $R_3 \geq R_{12}$, $R_4 \geq R_{12}$.

prove: $\dfrac{R_{13}}{R_{23}} < 2$

Since all the inequalities do not change when we multiply all c_i with some constant, we can choose this constant such, that $R_{12} = 1$. Solving this equality for c_5 and substituting the result into the inequalities we obtain.

Given:

$c_1 + c_3 - c_6 - g_1 - g_4 - g_6 \geq 0$ (1)

$c_1 + c_4 - c_6 - g_1 - g_4 - g_6 \geq 0$ (2)

$c_3 - c_4 - c_6 - g_3 - g_4 - g_6 \geq 0$ (3)

$c_3 - c_4 + c_6 - g_3 - g_4 - g_6 \geq 0$ (4)

$g_6 g_6 - g_2 g_4 - g_1 g_3$ (5)

prove:

$2(c_1 + c_3 + c_6) - g_1 - g_3 - g_6 \geq 0$ (6)

This is the same as proving that (1)–(5) together with the negation of (6) can never be satisfied for any values of the involved variables. Let us also remember that the variables come from node weights and edge capacities that are nonnegative. We can further require that g_i are strictly positive; in the opposite case the proof is easy. We verified that these inequalities have no solution with several numerical solvers, however that does not guarantee the correctness due to numerical errors. Therefore we used the Cylindrical Algebraic Decomposition method implemented in Mathematica 4.0 to verify the inequalities using symbolic computation. The running time of the Cylindrical Algebraic Decomposition algorithm is inherently doubly exponential; hence we had to transform the inequalities and write supporting functions to obtain results in practically reasonable time. Using symbolic computation we can be certain, if Mathematica has no implementation bugs, our conjecture holds.

The Mathematica 4.0 notebook to verify these inequalities follows.

```
In[]:= (* We introduce auxilary function to speed up the calculations. It
        returns False if and only if the expression is always false *)

       Verify[a_ || b__] := (Examine[a] || Verify[Or[b]]);
       Verify[a_] := Examine[a];
       Examine[a_] := Experimental`CylindricalAlgebraicDecomposition[
          a, {g1, g2, g3, g4, g5, g6, c5, c6, c1, c3, c4}];

       $RecursionLimit = 10000;

       eq1 = c1 + c4 + c6 - g1 - g4 - g6;
       eq2 = c1 - c4 - c6 - g1 + g4 + g6;
       eq3 = c3 - c4 - c5 - g3 + g4 + g5;
       eq4 = c3 + c4 + c5 - g3 - g4 - g5;
       eqimp = 2 (c1 + c3 + c5 + c6) - g1 - g3 - g5 - g6;

       (* Do the verification in two steps. Doing it all together
         takes too much time *)

       tmpResult = Experimental`CylindricalAlgebraicDecomposition[
          c5 >= 0 && c6 >= 0 && eq2 >= 0 && eq4 >= 0 && eq1 >= 0 && eq3 >= 0 &&
           eqimp < 0 && g1 > 0 && g3 > 0 && g5 > 0,
          {c5, c6, g1, g2, g3, g4, g5, g6, c1, c3, c4}];


       Verify[LogicalExpand[tmpResult && g1 * g3 == g2 * g4 == g5 * g6]]

Out[]= False
```

Table 1.

Experimental results for tree graphs.

Node count	Iterations	Time (s)	Cut value	Improved cut	Optimal cut
100	257	0.01755	0.000424058	0.000421019	0.000407809
200	377	0.05105	0.000131803	0.00010598	0.000101407
400	512	0.1377	3.38841E-05	3.14657E-05	2.55342E-05
800	739	0.43965	9.62836E-06	7.94314E-06	6.32158E-06
1600	906	1.14515	3.49721E-06	3.10424E-06	1.58787E-06
3200	987	2.72245	1.3683E-06	1.14917E-06	3.97123E-07
6400	1085	8.336	7.15262E-07	5.51447E-07	9.88485E-08
12800	1294	27.96925	3.34456E-07	2.98915E-07	2.46653E-08
25600	1343	79.48075	1.78731E-07	1.59801E-07	6.19941E-09
51200	1374	199.9305	1.03187E-07	9.352E-08	1.53453E-09
102400	1115	359.2426	6.66622E-08	6.17942E-08	3.89972E-10
204800	1734	1197.7344	2.81771E-08	2.60658E-08	9.6445E-11

Table 2.

Experimental results for cubic graphs.

Node count	Iteration count	Time (s)	Cut value	Improved cut
100	199	0.013	0.005185495	0.005166088
200	290	0.04205	0.002513133	0.002510172
400	383	0.11065	0.001214463	0.001213595
800	570	0.36905	0.0005994	0.000598147
1600	727	1.02045	0.000296	0.000295773
3200	835	2.69475	0.000147637	0.000147552
6400	1012	8.92785	7.36747E-05	7.36234E-05
12800	1195	30.71865	3.67635E-05	3.675E-05
25600	1455	107.70535	1.8375E-05	1.83679E-05
51200	1672	320.36415	9.15616E-06	9.15434E-06
102400	1998	834.6924	4.56971E-06	4.56927E-06
204800	2500	2254.4636	2.28649E-06	2.28627E-06

Table 3.

Experimental results for geometric graphs.

Node count	Iteration count	Time (s)	Cut value	Improved cut
100	129	0.016	0.002848427	0.002845499
200	146	0.04305	0.001631237	0.001620948
400	204	0.1282	0.000592549	0.0005825
800	266	0.4532	0.000338845	0.000333014
1600	389	1.9732	0.000116216	0.000113355
3200	430	5.28105	5.88979E-05	5.82837E-05
6400	530	14.3365	2.34042E-05	2.25542E-05
12800	681	45.96155	1.13718E-05	1.09091E-05
25600	800	149.0739	5.18003E-06	5.18003E-06
51200	886	461.85255	2.88345E-06	2.88345E-06
102400	976	1169.0154	1.4157E-06	1.4157E-06
204800	1417	3797.3582	3.32351E-07	3.14612E-07

LATVIJAS UNIVERSITATES RAKSTI. 2004. 669. sēj.: DATORZINATNE UN INFORMACIJAS TEHNOLOĢIJAS. 80.–88. lpp.

Review of Traceability Models for Software Testing Processes

Martins Gills
Riga Information Technology Institute
Kuldigas iela 45, Riga, LV-1083, Latvia
martins.gills@riti.lv

A considerable part of software development activities is devoted to testing. The scale of testing may depend on quality criteria for the particular development project. As soon as development and testing is performed in some organized manner, there exist some relations and dependencies between all types of project items. The relations outline some path of traceability that starts from the origins of a particular item, and follows to its derivatives in a form of other item types. This paper reviews various traceability models for software projects, and analyses the sub-models of testing processes. The common properties of these models are identified.

Key words: traceability, problem reporting, testing process.

Introduction

The property of traceability for software development projects has been known for long. It usually assures a better control over software development artefacts, provides a more organized project life-cycle and, consequently, reduces the risk of developing a low quality product [6]. Testing as one of software processes is extensively dependent on the established traceability principles or, more formally, a traceability model within the given project. A vital problem in software testing is to establish an appropriate coverage of requirements. Typically, the formal coverage could be achieved according to some criteria, e.g. stating that each requirement must have at least one appropriate test, or in more advanced situations using standard requirements like ones of BS 7925-2 [3]. It is especially important to evaluate the impacts of requirement's changes or to analyse the updates within the test suite. The previously mentioned issues are related to traceability property within the software development life cycle and to the testing process in particular.

Software testing process properties

A large part of the effort devoted to software development is due to software testing [11]. Software testing can be regarded as one of the processes presenting within any software development life cycle. Although it may not appear explicitly within the formal list of processes (like e.g. in ISO/IEC 12207 [8]), it may be derived as a tailored process.

Typically, the testing is organized at various levels, depending on what is the focus of the specific testing activity. The concept of test levels (test activities with a common direction) is mainly derived from the software development V-model. The model itself originally comes from the Software Lifecycle Process Model [4], more often it is represented in a very simplified form [10], and as a more generic model is shown in Fig.1. In principle, the V-model is close to the more classical Waterfall software development model [11]. Every project has a different testing process, and not all projects do have tasks corresponding to

every test level - some may have very specific ones. Main differences can be due to different naming and due to the combination or subdivision of particular test levels. In each case various organizational and methodological issues play a role in defining the way these test levels are implemented. For example, a unit test may be called a component test, but acceptance testing may be subdivided as factory acceptance testing and user acceptance testing, or this level may be supplemented by qualification testing. Every test level has a different object to be tested, and the tests themselves are based on different project artefacts.

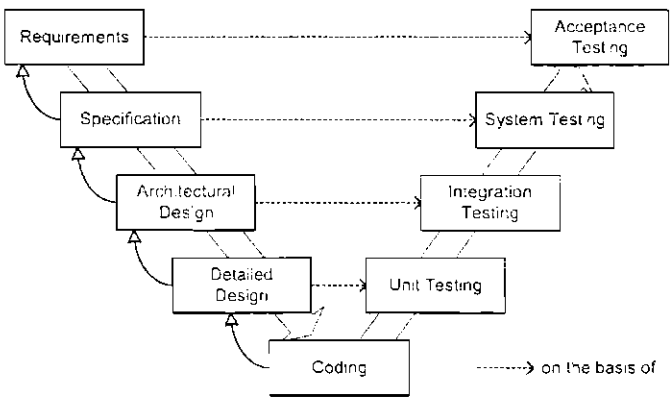

Figure 1. The V-model.

The Author has found out that in rare cases the projects can immediately say what kind of life cycle model they use. In most cases such a model exists, but it is not formally described in an explicit form, rather it is hidden within the project plan, procedures or standards. Although the notion of test levels is explicitly used in the V-model, they can be a part of other models as well. It could be extremely difficult to indicate a development process where testing is just a once-through activity with only one testing level. Even the simplest approach of quality requires a product (e.g. software element) checking activity prior to passing it further for integration into a larger product or prior to delivery. As the typical software development consists of numerous developers, project stages (typically, with cycles), there are at least two test levels present.

Traceability models for development process

According to the concept of the traceability model [5], it is formed from project items and relations between them. The most convenient form of representation is graphical where boxes are item types and arrows indicate relations. To analyse the testing process items, the author compiled preliminary information from 14 projects. Each of those represented a slightly different development approach, but at the same time, each project from this group could be mapped to the typical life cycle building blocks defined by the standard J-STD-016 [7]. For this paper, four different models were selected, each representing some possible class of traceability models. Three of them represent custom information system development projects, and one represents development of small utility software.

Project A (see Fig.2A) represents the most popular class of traceability relations. Formal requirements are derived from customer proposals, tests are based on these requirements,

and problems are documented in problem reports. The test execution results are kept together with the test information. Items directly related to testing are: test and problem reports.

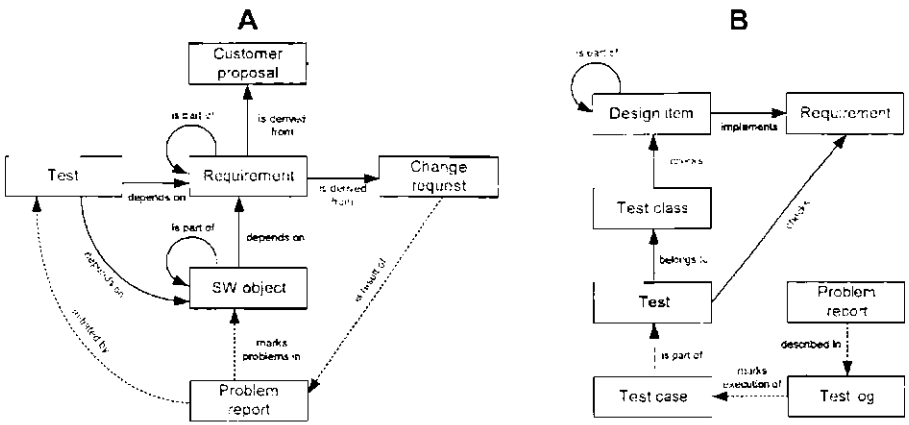

Figure 2. Traceability models for Projects A and B.

Project B (see Fig.2B) shows the project where the testing process is highly refined – the testing process items are test class, test, test case, test log and problem report. This project may correspond more to a situation where requirements are not undergoing changes, and the project stage represented in the model is more oriented to implementation and testing rather than designing or developing specifications.

Two other projects had more relaxed requirements for the development. Project C (see Fig.3A) is a sample of the model that includes quite detailed implementation description. The particular model was present in a project, developing a web-based application. Testing process items were: test, test case, test log and problem report. Testing was well organized, and in the particular case these tests were mostly at unit and integration level.

A slightly different situation from all the previous ones was in Project D (see Fig.3B). This was a development project of a software utility with no particular external requirements in mind, but they were rather defined iteratively and served at the same time as tests. Something similar may happen when special development methodologies like Extreme Programming are practiced. The Project D case is not dominant, but it characterizes a semi-formal approach in projects with a rapid and evolutionary development.

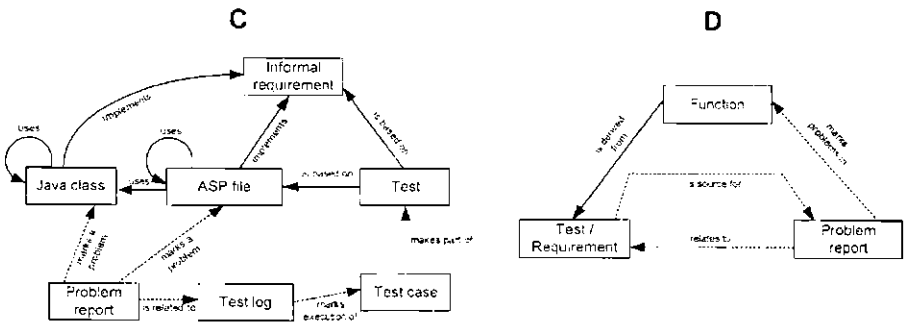

Figure 3. Traceability models for Projects C and D.

The Author found out that it was possible to map the traceability models of all 14 reviewed projects to the four models previously presented. The main difference could be that specific names are given for particular item types in each particular project. For example, requirement could be a particular diagram, use case, scenario, instruction, feature, etc.

Typical item types of testing process

The identification of testing items within the development process traceability model allows defining the traceability for the testing process. The typical constituents of the testing process are project items shown in Table 1.

Table 1.

Testing-related project items.

Test	A set of objectives, methods and criteria to be applied in verification/checking purposes to a certain software item.
Test case	A set of inputs, execution preconditions, and expected outcomes developed for a particular test objective. In some cases the test case may be created in the form of a test script for automated tests.
Test class	A set of tests with similar test objectives, methods or test levels.
Test configuration	A set of additional test attributes to be applied during execution for certain test cases.
Test suite	A set of test cases to be executed within a certain test session, scenario or by the particular personnel. Test suite can be implemented in the form of a test script for automated tests.
Test log record	Information about the status of test execution.
Problem report	A detailed description of test incidents, a record describing all events requiring an additional exploration.

The identification of testing process items within project traceability models shows that the relations between these items are not random. There are typical directions of dependencies. In every project, the testing process items form some subset of a more generalized model. Figure 4 shows a generalized model of the testing process. Each individual project may include a subset of these, or define some specific additional testing-

related items. For example, the test script as an item type can be added to the model for projects with a high level of test automation.

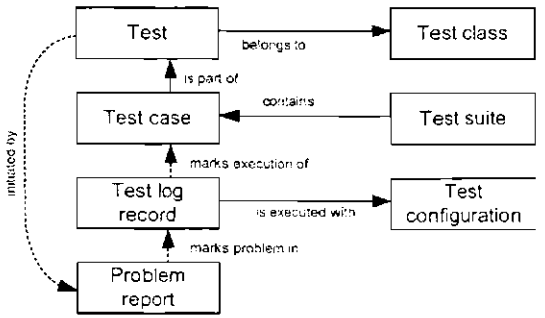

Figure 4. A generalized traceability model for testing process.

Items shown on the left side (Fig. 4) represent the typical constituents of the information set for testing that could be derived from software engineering standards like J-STD-016 [7]. The information set of the testing process has to provide answers to the following three question areas that are vital for project management and product quality:

- what is to be (or was) tested and how?
- when and how were the tests executed?
- what problems are encountered?

In Fig. 4, the project items on the right side do have a more classifying role - in some cases they could not be regarded as separate items but rather as attributes to the project items on the left side.

The most common approach in industrial projects is not to distinguish between tests and test cases. All are called tests, and the item could be either very detailed (like the test case), or it could outline just the main issues (like the test). Some other optimisations could take place, sometimes reducing the minimal test item set to only two project item types. Fig. 5 shows two possible cases that were observed in the projects. In case (a) there are separate tests defined, but all testing results are maintained as one log. This can function in small teams where it is not important to establish a formalized communication between testers and developers, but at the same time testers recognize the need to document the progress and problems. Case (b) formalizes the situation in which attention is paid mainly to problem registering, and probably tests are not even in every case formally documented, but rather briefly described while they are executed. This may be typical for larger project teams than in case (a).

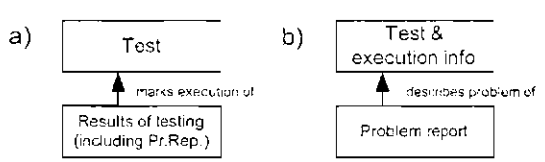

Figure 5. Possible optimisations of traceability models of the testing process.

In some projects, intentions have been seen to omit writing the tests, or tests are written as an exact copy of requirements list with words like "make sure that", "check" or "verify" added. Sometimes the main motivation for this is the need to minimize the effort spent on documenting (due to economy reasons). This formalism is quite widespread in software engineering, it assures a good traceability, but it is not efficient in terms of good testing [1]. The Overall evaluation shows that the base element, the item type for test-related project items is the Test - it has to be present in the project if the testing is made. All other testing process items mainly depend on this. The dependence of the test on the problem report does not play a significant role - it indicates the case where new tests may be added as a result of found problems.

External relations of testing process items

The Previous chapter defined the generalized traceability model for testing process. At least two items -- the test and the problem report - within this model have relations with project items that are not part of the testing process.

Dependencies of the test

Items externally directly related to the test (relative to the testing process) can be classified into three groups:

- software system defining items (requirements, design items),
- the code,
- co-products (e.g. user's manual).

The test is usually strongly dependent on these items. There is a correlation between test levels and the dependencies of the test (see Table 3). The test can also be derived from the problem report (see Fig. 4), but this is not dependent on the test level, rather on test process organization.

The relation to the initial or refined requirement could be considered as a norm for functional testing. This provides in the most direct way the answer to the question whether the expectations of the customer are met. Design items and code elements are referenced primarily in developer tests, because in a traditional setting neither users nor independent testers would have the availability to exploit internal development information.

Table 3.

Items related to the test and the corresponding test levels.

Project item	Test levels
Initial requirement	acceptance
Requirement	system
Design item	integration
Code element	unit
User's manual	system, acceptance
Problem report	unit, integration, system, acceptance

The relation of the test to the user's manual is quite interesting. It could be applied in two cases - either when the documentation is under test, or when there are no formally specified requirements. Of course, there may be some defined requirements, but at the time of testing they are completely outdated, and there is no practical feasibility to base the test on these requirements. An interesting alternative approach with case studies has been proposed - to use the user's manual as a requirements specification [2]. Such an approach requires systematic writing of the manual, but for many real-life projects this method could be less expensive than writing and maintaining two documents. A Special advantage is gained when the customer does not request these two. Relation with a problem report is typical for the tests added during the test execution phase to check the problem areas in later testing cycles.

Relations of the problem report

The problem report is an event-provoked item, and therefore it is tied with a weak relation to other items -- no changes in other related items can provoke change into the problem report itself. The problem report can be related to almost any project item, but the most typical are the same as for the test. It may have numerous indirect relations (e.g. through an interface element problem it may point to some specific requirement). The existence of particular direct relations depends on the test level (see Table 4). In each particular project, only part of the above mentioned items are usually used. The problem report's relation to the test or test log indicates what kind activity led to the given problem report. Design, code, interface elements or the user's manual or do indicate the location of the problem observable by the tester. Reference to the requirement could be used either to indicate problem in requirements or into its derivatives.

Table 4.

Items related to the problem report and the corresponding test levels.

Project item	Test levels
Test	unit, integration, system, acceptance
Test log record	unit, integration, system, acceptance
Design element	integration, system
Interface element	system, acceptance
Code element	unit, integration
Requirement	acceptance
User manual	system

Additionally, there is an external item that depends on the problem report – the change request. Usually, it is further related to the requirements.

Related work

A comprehensive analysis of traceability models within software development projects is made in [12]. At the same time, it is more focused on generalization of the problem rather than analysing individual relations of project items. In [9] a model of trace links within testing activities is shown. At the same time it describes a particular case of traceability links with generalized, bi-directional relations. The Main focus is on coverage-oriented questions. Some papers (e.g. [13]) express the need for increased traceability for software development

quality techniques to be more similar to traditional measurement principles. No reports were found that are focused on traceability and software testing relations.

Summary

The traceability models of software testing processes belonging to different development life cycle models indicate similar properties. There are considerable similarities between every testing process, and no contradictory principles were found. These particular testing process models can be regarded as a subset of a generalised traceability model of the testing process. The Main differences could be found within the terminology - how each item type is called in a particular project, and the level of optimisation - which items are skipped or regrouped. The quality of testing can be considerably improved if traceability information is taken into account. This increases the level of maintainability of tests, and could reduce effort and time spent to carry out a change.

The Author has indicated possible research areas concerning the traceability within the projects. From the application point of view, a potential interest could be in finding influence on the properties of the traceability model caused by the duration of the project, its particular life cycle model, project team size, project (sub-)contract conditions, project scope and tasks, development methodology and development environment.

Acknowledgement

The author would like to thank the Riga Information Technology Institute for making it possible to analyse the software development projects and the Latvian Council of Science Grant (grant assignation decision No. 3-2-1, 2002.07.09) for partial financing of the research.

References

1. Bach J. Risk and Requirements-Based Testing. Computer, June 1999, pp. 113-114.
2. Berry, D.M., Daudjee, K., Dong, J., Nelson, M.A., and Nelson, T. User's Manual as a Requirements Specification. Technical Report, CS 2001-17, University of Waterloo, Waterloo, ON, Canada, May, 2001.
3. BS 7925-2 Software component testing, BSI 1998.
4. DSK RR413320010. Allgemeiner Umdruck 250/1. BWB IT 1997. Entwicklungsstandard für IT-Systeme des Bundes, Vorgehensmodell. (Teil 1: Regelungsteil, Teil 3: Handbuchsammlung).
5. Gills, M., The Concept of a Universal Traceability Tool for Software Development Projects. Scientific Proceedings of the Riga Technical University, Computer Science, Applied Computer Systems, 2st thematic issue, Riga, 2001, pp 33-41.
6. Gotel O. C. Z.; Finkelstein A. C. W., An Analysis of the Requirements Traceability Problem. 1994, p.8 http://citeseer.nj.nec.com/78573.html
7. IEEE, J-STD-016-1995, Standard for Information Technology Software Life Cycle Processes. 1995.

8. ISO. Standard ISO/IEC 12207. Standard Information technology Software life cycle processes. 1995.
9. Linde S.. Relationships between quality and testing - Kvalitātes un testēšanas attiecības (in Latvian). Scientific Proceedings of the Riga Technical University, Computer Science, Applied Computer Systems, 2st thematic issue, Riga, 2001. pp 68-74.
10. Pol, Martin; van Veenendaal, Erik. Structured testing of information systems - an introduction to TMap. Kluwer Bedrijfslnformatie, 1998. p 177.
11. Pressman, Roger S. Software engineering: a practitioner's approach. McGraw-Hill, Inc. 1997 (4th ed.) p. 852.
12. Ramesh, Balasubramaniam; Jarke, Matthias. Toward Reference Models for Requirements Traceability. IEEE Transactions On Software Engineering, vol. 27, no. 1, January 2001. pp 58-93.
13. Wakid, Shukri A.; Kuhn, D. Richard, Wallace, Dolores R.. Toward Credible IT Testing and Certification. IEEE Software, July 1999. pp. 39-47.

LATVIJAS UNIVERSITATES RAKSTI. 2004. 669. sēj.: DATORZINĀTNE UN INFORMĀCIJAS TEHNOLOGIJAS. 89.–98. lpp.

Benchmarking Problems of Topic Telecommunications and Access in Latvia

Mara Gulbe*, Arnis Gulbis**

*University of Latvia, Aspazijas blv. 5, Riga, mgulbe@latnet.lv

**Riga Technical University, Azenes 12, Riga, arnisg@di.lv

The problems related to benchmarking of telecommunications and access area in Latvia are analyzed. The topic under consideration is important because it represents the backbone of all Information Society characterizing issues. The availability of data on T&A from national statistical sources is analyzed and indicators used for benchmarking of the topic T&A in Latvia are considered.

Key words: telecommunication, benchmarking.

Introduction

Taking into account the fact that Latvia is going to become a member state of EU the priorities of EU also become actual in Latvia. The eEurope+ action plan [1] directed towards the acceleration of the formation of Information Society in NAS, serves as proof of this statement. All NAS countries have agreed to take part in the implementation of this action plan in their respective countries. Up to now our governments have supported this process in Latvia. As an example we can mention National program "Informatics" [2] and socio-economic program e-Latvia [3]. At the same time the attention paid to benchmarking of action results is unsatisfied, though the 2003 action plan of central statistical bureau includes the data collection in accordance with the needs of eEurope+ activities. These data will be available only in 2004. Nevertheless EC has significant interest about the present state of IS (Information Society) characterizing areas in NAS countries. Therefore, independent investigations are supported by EC. Here we can mention the SIBIS project [4], which also extends the benchmarking process of IS to NAS countries. The level achieved in the development of Telecommunications and Access serves as the backbone for IS activities in different areas of society and individual lives. The present paper reflects the investigation of the topic T&A in Latvia carried out in framework of SIBIS project (supported by EC as FP5 investigation).

Telecommunications Infrastructure

To facilitate the better understanding of the topic some insight into the telecommunications and access infrastructure in Latvia is necessary. It includes:

- Fixed public exchange network;
- State significance data transmission network and other specialized networks;
- Mobile telephone network;
- Cable TV network.

Fixed Public Exchange Network

After the renewal of independence in 1991, the government owned a rather backward fixed public exchange network infrastructure where there was an almost hundred percent dominance of analogue telephone lines. Very often one telephone line was shared by two users making simultaneous calls impossible. The access demand was higher then the supply. Under these circumstances quick measures were necessary to accelerate the progress in the field under consideration. To stimulate the implementation of digital technologies in the area of telecommunications and to satisfy the access needs of the population, foreign capital was involved. The ownership of the fixed public exchange network was shared between Lattelekom (representative of the state) and one foreign telecommunication company (after the change of its shareholder it is now the Finnish Sonera) under conditions specified in the agreement between both parties. The agreement foresees the complete digitalization of the fixed public exchange network in Latvia by Lattelekom according to a time schedule fixed in the agreement. The agreement also established the Lattelekom (together with a second owner) monopoly on the fixed public exchange network (supervision, development, maintenance) for a time period of twenty years (up to 2013). This means that the fixed telecommunication market was forbidden for other parties who wanted to play in that market up to this date. This, in turn, acts against the price reduction in the telecommunication market. Later it turned out that this monopoly also makes a serious barrier for Internet penetration and usage by households in Latvia due to inappropriately high price when compared with the average salary of employees in this country. After this became clear to the government, efforts were undertaken to get free from that monopoly. This was also stimulated by EU recommendations demanding the liberalization of the telecommunication market. In case of a successful agreement between both players the liberalization of the fixed telecommunication market should be achieved in 2003. This is stated in the law "On telecommunications" and is what is expected by the government.

Now Lattelekom Ltd is the largest telecommunication company in Latvia. It offers to clients different telecommunication services including voice telephony, Internet, lease of telephone lines, etc. Through its subsidiary Apollo Lattelekom also acts as an Internet provider. The following Internet access possibilities are offered:

- Dial-up;
- ISDN;
- xDSL.

Depending on the purpose (business or household) different types of ISDN and xDSL are offered. Some types of xDSL can not be considered as broadband.

One more point must be mentioned here. Lattelekom does not keep its promises regarding digitalization of telephone lines everywhere in Latvia according to the time schedule set in the agreement with the government.

The argumentation is based on the statement that in rural regions, digitalization is unprofitable. At the same time the quality of data transmission by dial-up connection through analogue lines is sometimes very bad. This could become a reason for digital divide.

On the other hand other Internet service providers can not extend their services through the fixed telephone network to rural regions and must try to find an alternative (radio-link or Internet through satellite if possible).

State Significance Data Transmission Network and Other Specialized Networks

There is a state significance data transmission network (Latvian abbreviation VNDPT) in Latvia. This network is independent from the fixed public exchange network (though some lines are leased from Lattelekom) and is supervised by the state agency VITA (Latvian abbreviation from the Agency of Government Information Networks). VITA is a Nonprofit Organization State Joint-stock Company established in 1997 according to order of Cabinet of Ministers No 44. The aim of building up of the agency was to provide the development and maintenance of a joint governmental information system. As provider of State significance data transmission network services VITA maintains the corporative (intranet) computer network covering all the territory of Latvia. In this network the Frame Relay packet switching technology is used. The advantage of such a technology is the possibility of several logical channels in one physical channel and the foreseeable requirements of both data transmission speed (28.8 kbps – 2Mbps) and query response time in the address space governed by TCP/IP protocol. Besides, the logical channel of one user is not available for others.

Services provided by VITA are available in all of Latvia for both governmental establishments and state registers.

The VNDPT network is a closed network available for governmental institutions on-line 24 hours a day.

Access to State significance information systems from outside (Internet) is controlled through a fire-wall system. Access to public servers (containing e.g. home page information of Ministries) is open.

In order to improve client needs regarding an increased data transmission rate a transition to optical fibre connections is in progress. Five ministries and some state significance registers are connected through optical cables.

A new service offered by VITA is IP telephony, which allows the reduction of telecommunication costs for their clients.

In general benefits from VITA services for governmental institutions are as follows:

- Lower access and traffic costs when compared with those offered by other providers;
- Increased security of access and data transmission.

In European context the future objective for the VNDPT network is closer integration with IDA networks.

Besides the network supervised by VITA there are some other specialized telecommunication networks in Latvia, those of Latvia's Railroad, the Ministry of the Interior, Latvenergo (the biggest energy supplier in Latvia), Latvia State Center of Radio and Television. Due to restricted range of usage they are less important.

Mobile network

There are two mobile telephone networks in Latvia:

- LMT (Latvian abbreviation from Latvia's Mobile Telephone);
- TELE2 (subsidiary of the Tele2 AB (at the time "NetCom AB")).

Both operators are providing telecommunication services in GSM900 and GSM1800 frequencies and offering the following set of services:

- Voice telephony,
- SMS;
- Fax transmission;
- Data transmission;
- Internet;
- E-mail;
- WAP.

Types of connection used to initiate data transmission are:

- Analogue;
- ISDN.

LMT services support wider possibilities for mobile connection with PC when compared with those offered by TELE2. These include also Bluetooth technology.

The data transmission rate 9.6 Kbps is provided by both operators, but LMT also offers another more advanced technology HSCSD (High Speed Circuit Switched Data) with a data transmission rate up to 38.4 Kbps.

LMT services also include a new data transmission technology for GSM networks - GPRS (General Packet Radio Service) based on packet switching and offering additional possibilities for data transmission services.

Historically the first mobile operator in Latvia was LMT. The appearance of a second operator leads to reduction of costs for mobile telephone services.

One more point must be mentioned here. There is little hope that in the nearest years the services of the fixed telephone network will be available for many inhabitants of rural regions. This is due to the lack of telephone lines of the fixed network in many places. The alternative is the usage of services provided by mobile operators. The services offered by LMT are available almost in all of the territory of Latvia. The connection possibilities of the Tele2 network are not so good. Due to circumstances described many people of rural regions are using mobiles but mainly for voice telephony. Of course, Internet access through mobile is also available for them but only with a rather low data transmission rate provided by GSM. In the context of an increased data transmission rate the possibilities offered by 3G mobile services seems to be very attractive for inhabitants of rural regions, of course, only in case of acceptable service costs. The Internet via satellite is also possible but the exploitation of this service is less probable.

In order to increase the number of players in the telecommunication market and facilitate its liberalization the bid of three UMTS licenses was organized by the government in 2002. Both mobile operators LMT and TELE2 succeeded in that bid. The third license was not sold and, therefore, the bid must be repeated once more.

Cable TV Networks

There are several cable TV providers in Latvia, mainly in the capital Riga (one well-known in Riga is Baltkom (www.baltkom.lv), but there are also others). The service is available in areas with a high density of inhabitants where it is profitable. Up to now the barrier for development of cable TV services was the Lattelekom monopoly on underground cables. To overcome this difficulty very often TV programs were sent through the air from the base station to some antenna located on a definite building connected with neighboring buildings by cables also going through the air. The same solution is also used by Internet providers offering Internet access via radio-link. In fact, under the conditions of the Lattelekom monopoly, Internet access via radio-link is a very popular and widely used service.

As could be expected, some cable TV providers also offer Internet services, of course, only in the places where the cable TV network is available. For this the cable modem is necessary. The service offered is on-line connection with rather good parameters (broadband).

The barrier for Internet through cable TV network is rather high service prices. The cable modem is also not cheap.

Internet Via Satellite

Access to Internet is also offered by satellite TV providers (e.g. Unisat in Riga). Two possibilities for Internet via satellite are offered. These are:

- Europe Online (the query to specialized European proxy server and the query result through satellite to user);
- DiRECWAY (specialized equipment for up-link to satellite and cable modem is necessary).

The last possibility is convenient not only for individual access but also for collective access (for not very large Intranets), especially in places where other kinds of access are not available or are of bad quality.

The barrier for Internet through satellite is a rather high price for the service.

Policy Documents on Telecommunications and Access

This chapter covers a variety of policy documents at the national level, which contain information on national policy directions and priorities, regulation and legislation. These documents contain those which must be highlighted as particularly important or central to the topic T&A. From the T&A infrastructure considered above and the analysis of policy documents and national data sources given below, we shall try to identify the indicators which must be applied to the benchmarking of main issues of the topic T&A.

National Program "Informatics" 1999—2005, Ministry of Transport, 1998.

This document is an action plan which outlines the Latvian way towards the Information Society. It includes the large set of tasks which must be solved to achieve the target. It is a complex program covering the time period 1999-2005 and consisting of 13 subprograms. It foresees the realization of more then 120 individual projects

directed mainly to the implementation of information and communication technologies in the different areas of society and individual lives as well as wide international cooperation in integration of European data transmission networks and services. Regarding the topic T&A the following subprograms must be selected:

- telecommunication networks and services (9 projects),
- information and data transmission networks (15 projects),
- development of information and telecommunication networks (3 projects),
- state significance statistics (4 projects).

The goal of the subprogram "Telecommunication Networks and Sservices" is the reform of telecommunication sector in Latvia according to principles accepted by EU. A plan of such reform is considered in detail.

In the subprogram "Information and Data Transmission Networks" the architecture of existing data transmission networks in Latvia is considered and actions necessary for further development of these networks are outlined.

In the subprogram "Development of Information and Telecommunication Networks" a survey on perspective Information and communication technologies is given and a special role of network technologies (Intranet, Internet) is emphasized. The state's role in the development of ITC is outlined.

The goal of the subprogram "State Significance Statistics" is the development of a modern ICT-based statistical system in Latvia which is in accordance with standards of such a system in EU.

The laws related to the topic T&A and considered above are results of the implementation of national program "Informatics", or more precisely, the legislation part of this program.

Conceptual Guidelines of Socio-Economic Program "e-Latvia", Ministry of Economics, 2000

The document outlines the main objectives of the socio-economic program e-Latvia which appeared as a response to e-Europe initiative and in fact can be considered as a further development of the national program "Informatics". The program e-Latvia represents the Latvian contribution into e-Europe. The main objectives of e-Latvia are as follows:

- significantly improve the access possibilities to Internet for citizens,
- based on improved computer literacy and training possibilities, to provide for everybody the use of modern information technologies,
- provide for everybody the access to global and national information resources,
- stimulate the development of e-commerce and e-government.

Analogously to e-Europe action evaluation e-Latvia is structured along three key objectives:

- A cheaper, faster and secure internet;
- Investing in people and skills;
- Stimulating the use of internet.

For T&A the following issues from e-Latvia are of importance:

- Full digitalization of the telecommunication network.
- Full liberalization of the telecommunication market, opening of the market of leased lines,
- Regulation of Internet sector and services, Internet services provider's responsibility regarding data security,
- Installation of public Internet access terminals in every local government, school and library,
- Provision of personal data security.

Thus the conceptual guidelines propose actions stimulating both the development of telecommunications and access possibilities to Internet. Regarding the topic T&A two projects involved in the e-Latvia action plan must be highlighted. These are a Latvian education informatization system and a Unified information system for Latvia's libraries. Both projects are interesting from the point of view of public Internet access points (PIAP). Access to Internet in schools is available for pupils and teachers and not available for people from outside (restricted public access possibilities). In libraries such access is available for everybody. Thus the implementation of both projects is very important to extend the Internet access possibilities for the public.

Relevant National Statistical Sources Identification and Documentation

At first it must be noted here that Information Society statistics in Latvia are in their beginning stage. Up to 2002 there was no systematical collection of statistical data on Information society, though some aspects of such statistics were covered. The development in the area is greatly stimulated by e-Europe+ initiative. The 2003 action plan of central statistical bureau (www.csb.lv) (governmental institution) serves as a proof of this statement though the respective data will be available only at the beginning of 2004. Therefore the main issues from this plan related to topic T&A are not included in the following table covering the main national and international data sources.

Every year data collected by Central statistical bureau are published in "The Statistical Yearbook". Important information regarding prices of corresponding telecommunication services are given in the websites of the largest fixed and mobile network operators. It should be noted that exactly the service price is the most important barrier to wide penetration of Internet into households. It is also clear that there is a strong correlation between the number of computers and the number of Internet connections (if we exclude WAP). Without a computer there is no need for an Internet connection.

Name of data source (Acronym)	Main publication(s) of interest for SIBIS	Description (incl. target, survey unit)	Responsible
Computer usage in enterprises, Internet usage (number of computers, number of Internet connections, etc.), e-commerce	Data of Central statistical bureau	Response on e-Europe+ initiative, data collection is based on Eurostat module 491, survey questionnaires are used	Central statistical bureau
Cable TV network usage (number of customers, employees, financial figures, etc.)	Data of Central statistical bureau	Response on UNESCO initiative, data collection based on Eurostat module 493, survey questionnaire is used	Central statistical bureau
The statistical yearbook yyyy	Data of Central statistical bureau	Data on all topics characterising the development of country	Central statistical bureau
Report on development of Latvia's national economy	Ministry of economics	Data on all topics characterizing the development of country	Ministry of economics
eEurope+2003, progress report, June 2002	Joint High Level Committee	eEurope+ survey	Joint High Level Committee
Information society statistics, Data on candidate countries	Statistics in focus, Theme 4-17/2002	Information society statistics	Eurostat, Author Richard Deiss
Information society statistics, Rapid growth of Internet and mobile phone usage in candidate countries	Statistics in focus, Theme 4-37/2001	Information society statistics	Eurostat, Author Richard Deiss
Market of IT and telecommunications (percentage of digital network, number of mobile network customers, number of ISDN lines, etc.)	Department of Informatics at Ministry of Transport, website	Control of development of ICT sector	Ministry of transport
Equipment used to benefit from ICT services (number of telephones, modems, computers in households and business, etc.)	Department of Informatics at Ministry of Transport, website	Control of development of ICT sector	Ministry of transport
Website of Lattelekom	Data on largest fixed network services provider in Latvia	Available data important for topic T&A	Lattelekom
Website of LMT	Data on one (of two) mobile services provider in Latvia	Available data important for topic T&A	LMT
Website of Tele2	Data on one (of two) mobile services provider in Latvia	Available data important for topic T&A	Tele2
Website of Latvia's Internet association	Data on Internet services customers	Available data important for topic T&A	LIA

Here we shall focus our attention on conventional indicators used to characterize the topic T&A in Latvia. A list of such indicators is as follows.

1. Percentage of households that have fixed telephone service
2. Fixed lines per 100 inhabitants

3. Mobile phone subscriptions
4. Percentage of households with Internet access
5. Percentage of population regularly using Internet
6. Internet access costs
7. Number of personal computers
8. Internet users
9. Number of Public Internet Access Points per 1000 inhabitants
10. Number of Internet hosts
11. Computerized enterprises
12. Enterprises with access to the Internet
13. Type of Internet connection in enterprises
14. Enterprises with a home page on the Internet
15. Number of employees that at their work place regularly use a computer with internet connection
16. Number of computers with access to the internet used by enterprises
17. Number of cable TV stations
18. Number of cable TV customers
19. Expenditure for cable TV business
20. Revenue from cable TV business
21. Information technology expenditure
22. Telecommunication expenditure

Glossary

CSB	Central statistical bureau
DSL	Digital Subscriber Line
EC	European Commission
EDI	Electronic Data Interchange
ESS	European Statistical System
EU	European Union
Eurostat	Statistical Office of the European Commission
FP5	5th Framework Program
ICT	Information and Communication Technologies
IDA	Interchange Data between Administration
IS	Information Society
ISDN	Integrated Services Digital Network
MS	Member States
NAS	Newly Accessing States
SIBIS	Statistical Indicators Benchmarking the Information Society
SMS	Short Message Service
T&A	Telecommunications and Access
UMTS	Universal Mobile Telecommunications System
WAP	Wireless Application Protocol

References

[1] eEurope+ 2003, Action Plan, prepared by the candidate countries with the assistance of the European Commission, June 2001.

[2] National program "Informatics", Ministry of Transports, 1998 (in Latvian).

[3] Conceptual guidelines of socio-economic program e-Latvia, Ministry of economics, 2000 (in Latvian).

[4] www.sibis-eu.org.

LATVIJAS UNIVERSITĀTES RAKSTI. 2004. 669. sēj.: DATORZINĀTNE UN INFORMĀCIJAS TEHNOLOĢIJAS. 99.–106. lpp.

Design Interpretation Principles in Development and Usage of Informative Systems

Jānis Iljins
University of Latvia, Raiņa 29, Riga, Latvia
Janisj@lanet.lv

The paper summarizes the experience and ideas of the last five years in developing Informative Systems by means of Informative System Technology (IST) and shows the application of these ideas in practice.

One of the main methods discussed in the paper allows exact implementation of the system compatible to the design – design interpreter method. In the paper the system design formalization method interpreted during the operation of the system is suggested. The IST environment acts as the design interpreter.

Other advantages of IST are described – modularity and simple maintenance, the possibility to automate the system prototype development, to generate the system documentation automatically and other advantages.

All the described methods are evaluated according to their practical application.

Key words: informative systems, IST, system prototype, maintenance.

Introduction

In order to develop and operate the Informative System (IS) according to the classic IS "waterfall" life cycle it is necessary to describe the problem statement or design from the needs of the real world. It is the base of system implementation [1]. Typical problems occurring in this approach are design incompliance to actual requirements and implementation incompliance to design. The last most often occurs in the phase of system implementation and maintenance when errors are stated in the implemented system; they are eliminated without any changes in the design. In the practice often systems the design specification and system documentation of which are older then implementation are met. They do not contain information on the latest changes and innovations in the system.

There are well-known ways how to solve the above-mentioned problems. The system prototype is usually used to check the compliance of the design to the requirements. Before implementation of the system the prototype is tested and changes can be easily done. As often the system prototype is automatically generated from the design specification, it exactly complies with the design. To achieve the system implementation compliance with the design used code generation is usually. In such a case the design should be formalized by means insuring code generation. Such means are offered by the specification language GRAPES and its environment GRADE elaborated in Latvia [2][3]. Many other tools like RATIONAL ROSE, ORACLE Designer 2000 [4] etc. are widely successfully used in the world. As the design specification does not contain information on implementation details, code generation makes programming easier but does not eliminate it. It is possible to change the generated system code, so causing danger that the changed code may not comply with the design.

At the Bank of Latvia in 1995 development of the Currency Operation Management Informative System (VOPIS) was started under the leadership of Professor M. Treimanis. It was clear at the beginning that the system will have to be able to operate in an environment in which requirements are changing very fast and it will be necessary to make many changes during its maintenance. Therefore special attention was paid to the fact how to develop the system so that it is easy to change it in the future and how to achieve that the changes are easily reflected in the system documentation. Professor M. Treimanis offered to develop the IST containing the design method and the environment supporting the method [5]. The specified design is stored in the database by means of the method. IST environment can interpret the design. Such an approach allows us to:

- combine design and implementation as the system implementation will directly use the design definition, so implementation will exactly comply with the design;
- change the applied system easily, for to make changes in the system functionality, in most cases, it is necessary to make changes only in the data stored in the database;
- maintain the system documentation corresponding to implementation as it can be obtained automatically from the design definition.

IST was successfully used in VOPIS development and operation. Later IST became the technology of all IS in the Bank of Latvia where it is still used. IST is also successfully used in Datorikas Instituts Ltd. elaborated IS.

Following paper chapters show the IST offered design formalization methods and describe the project interpreter. Different IST possibility applications in real projects, their advantages and disadvantages are analyzed.

System Design Method

The IS developed based on this design method is operated by means of the design interpreter – IST.

To develop a system by IST method the work consists of three independent parts:

- database design,
- technology definition,
- development of specific applications.

The essence of these parts will be described further.

Database Design

Target system database design is developed by means of the classic ER model.

Usually, in actual IST designs in database ER model diagrams, Object Model notation is used [6].

IST also envisages the registration of the entities and relations of ER model in the database, although it is not compulsory. If the developed ER model is registered in the technology database, it gives the following advantages:

- automatic data structure (table) formation (definition script generation) is possible;

- development of the system prototype even before development of the actual data structure is possible;
- automatic system documentation generation;
- technology definition work is made easer.

Several real systems have been developed in which the target system database is developed by means of classical relation database tools. In such cases these advantages are not used and ER model description (entities and relations) is not registered in the technology database. Nevertheless, the system is successfully operated by means of IST interpreter.

Technology Definition

The basic principles and metamodel of technology definition are described in [5].

To develop IS it is necessary to define the organization work technology. It is done by combining various methods already known described in [6][7][8][9].

In order to define the technology the following questions should be answered:

- **"Why?"** – the tasks of the organization should be understood. IST envisages defining the tasks, to define the subtasks for the tasks that can be described in more detail. In the result a hierarchic network is formed. IST task module helps programmers understand the target system, but it is not necessary during operation. Therefore the design interpreter does not use it.
- **"Who?"** – it should be clarified what structural units and what staff of the organization are involved in implementation of the task. The design interpreter uses the information on the staff from IST organization module in order to give them Login rights and control the staff rights to work with IS.
- **"Where?"** – according to the posts and responsibilities of the organization's staff members the necessary work places and what rights and appearance are needed for every work place should be specified. Starting a work session with the design interpreter in IST environment the user identifies and chooses one of the available work places where he will work during the work session. The user will be offered those IS possibilities which are envisaged in the design for the particular work place.
- **"What?"** – the design determines which database objects the user may process at the particular work place. The IST view concept is introduced – IST view is a set of objects of a single object class of the target system. In the context of a work place a hierarchic view network may be formed. As a view is a set of objects a sub view can be defined as the set of objects related to the selected view object. The project interpreter offers the user to see the available views and the objects in them, to find and select them for implementation of operation.
- **"How?"** – in the context of a work place an operation can be implemented for the view or object in view. Calling the operation the design interpreter starts another application shown in the design. The application receives a parameter from IST environment – the selected object in the view network.
- **"When?"** – for particular target system object classes state-transition diagrams can be designed. The technological process for each object of this class is started during the work and the object is set in a definite state of the state-transition diagram. IST environment offers the possibility to change the object state according to the state transition diagram. The view network should be designed so that the views containing

the objects in the needed state could be available in the particular work places. The operation should be linked to the views that change the object state.

The descriptions of the system users, work places, work place view network and operations linked to the views as well as state transition diagrams together form an interpreted technology that at the same time is IS design. In order to operate IS only the design description has to be interpreted.

Application Development

Specific target system applications are developed separately. They differ for every system. Therefore they are designed separately and independently of the organization technology used by IST.

The principle of modularity is observed in development of applications – every specific function of the system has its own application. Typically every target system object class has its own application to register the objects of this class and to edit their attributes.

In IST operation it is indicated which specific application and which specific mode to call it. For all applications it is necessary to use a predefined command line interface by means of which the IST environment conveys information to application on the user's activities – what objects the user has selected in the IST environment and wants to indicate as parameters for application.

Design Interpreter Implementation

Implemented Metamodel

The interpreted design description is formalized by the IST metamodel work place module [5] (Figure 1.)

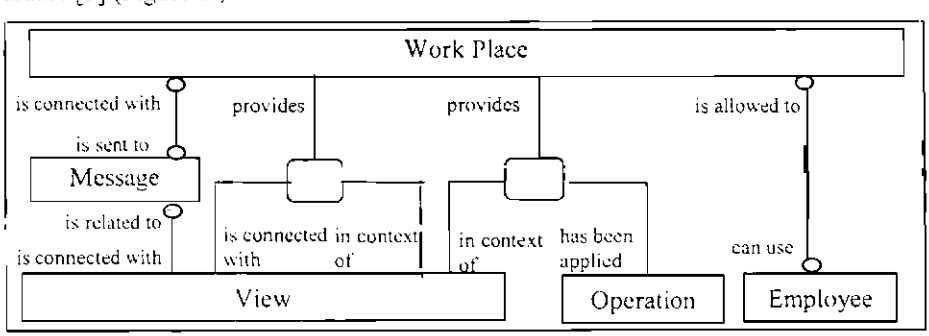

Figure 1: Part of the IST Workplace Module

The notion of the Workplace is used to define the employee's rights and duties. More formally, the IST Workplace Module defines

- Workplace
  - The Workplace entity attributes are the name and the comment. This entity is used as a placeholder to define the employees' rights and duties.

- Message
    - The Message entity main attributes are the text (for instance, Warning: FX deals are waiting confirmation!), the icon-representing message and SELECT statement. It is possible to use arbitrary SELECT statements with a set of predefined variables. ISTE executes this statement after defined time intervals. The message text appears, if SELECT statement, when executed, finds some object.
- Set of technological relations
    - the relation between the Employee and the Workplace entities allows employees to participate in the organization's technological process with a predefined set of rights, which will be defined below;
    - the relation between the View and the Workplace entities allows the employee who works in a particular Workplace to get access only to predefined sets of Views and Objects;
    - the relation between the View, the Operation and the Workplace entities allows the employee to execute a predefined set of Operations with Views and objects in a particular Workplace;

The relation between the Message and the Workplace entities allows the employee to receive Messages about predefined events, which is connected with the Workplace. Usually these messages are related to a necessity to process some object that "arrives" in the Workplace in some View. The relation between the Message and the View entities allows the employee to find those objects easily.

IST interprets these technological relationships and thus enforces employees to perform the organization's tasks according to predefined OTM.

Functional Shell

The Functional Shell is a program – design interpreter operating according to the above described metamodel and interprets the system design description coded in it. The program controls the user approach to the database, thus so controlling operation of the system. The program insures work in a particular work place available to the user. In windows the view network – views and the objects they contain which may have linked sub views. So it is controlled what the user sees and can process from the database. Selecting a particular object or view operations by means of which the selected object can be processed are shown. This indicates how the processing is done. After a definite time interval the message window is refreshed. In it the user can see when to do the appropriate activities. The main window of the functional shell as it is seen by the system user is shown in *Figure 2.*

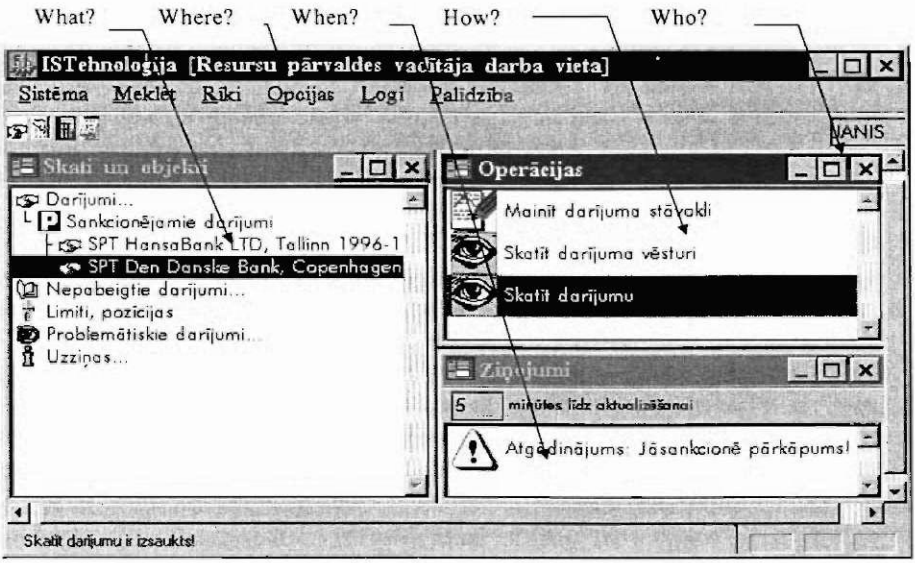

Figure 2. Main window of the Functional Shell.

IST Additional Possibilities

Besides the Functional Shell it is also possible to realize other possibilities using the system design entered in the database. One of such possibilities is data structure generation. If in the technology description the system database entity types and relations between them as well as entity attributes are defined it is possible to develop a program that transforms the design description coded in the database into SQL CREATE TABLE statements, so creating a real database structure.

It is possible to enter in the technology database objects-examples for every type of entities. The system design can be made including work places and the views they contain so that the objects shown in the views are objects-examples from the technology database. Thus it is possible to develop a system prototype even before the real database structure is developed. The prototype will be operated by the real design interpreted by means of which the implemented system will be operated later. So the prototype will precisely coincide with the system implementation.

The design description that can be interpreted or used for data base structure generation can be printed in a prepared documentation template. For it a program is needed that reads the design description in the database and passes it to a tool like VISIO or MS Word.

Information on reports necessary to print from the target system can be entered into the technology database. The technology database stores information on the data needed for the report (SQL SELECT statement returning the data) and information on how the data should be shown in the report templates. So a program – report interpreter can be implemented by means of which most of the system reports can be developed.

IST Application Examples

Experience

ISTechnology currently is used in the Bank of Latvia as the standard framework for building Information Systems. Besides the Bank of Latvia between 1996 and 2001 ISTechnology framework was used in many other organizations to build and maintain complex Information Systems (Figure 3).

Information System	Organization
ISTechnology (as described in this paper)	Computer and Software Engineering Institute Ltd.
Salaries Management IS	Bank of Latvia
Employees Management IS	Bank of Latvia
Commercial Banks Management IS	Bank of Latvia
Trade and Treasury Management IS for Central Bank	Bank of Latvia
Trade and Treasury Management IS for Commercial Bank	Unibank of Latvia
Trade and Treasury Management IS for Investment fund	Optimus fund
Patient Register IS	Health Department of Riga Municipality
Pension Capital Management IS for Private Pension Fund	Unipensija

Figure 3. Examples of ISTechnology usage.

Application Evaluation

All IST possibilities were most completely applied in the Bank of Latvia. All systems where deigned by the IST method and operated by the design interpreter – IST Functional Shell. Several system development environments were created in one IST environment. A part of the IST modules was assessed as successful. But for a part of the implemented modules the application in practice did not prove to be useful. The functions of these modules were carried out by other tools and they were not actually used. So the data structures were not generated by means of IST, as the applied SQL Server Software includes sufficiently easy to use structure creation tools, and there was no need to use IST tools. System documentation generation and system prototyping before data structure generation also was not justified in practice because too much configuration work was needed for it.

The main reason why IST modules were not used is the complicated configuration work needed to enter the technology definition in the database. Handy editors are lacking. It would be worth developing such, but it is a time consuming process.

The IST module that proved to be the most efficient is the design interpreter described in this paper. The method for the technology definition by which work place configuration is described indicating the work place view network and the linked

operations and interpretation of the technology by the functional shell makes IST unique. Systems developed and operated in this way are easy to use and maintain. Therefore IST is used in more and more new projects.

In some systems developed and operated by IST, especially in Unibank of Latvia, the system dynamic model is successfully used, the IST state-transition and message module is based on it and the report generation module. Both these modules are additional to the Functional Shell helping to make the system maintenance even easier.

Future Perspective

IST is worked out in client-server architecture by means of which client-server IS is developed. Nevertheless, today other technical solutions are found. A possibility to develop IST as an Internet-based IS development and implementation tool is considered. New IST versions are created built in 3-level architecture – database, middleware and user interface. So the IS data security would be improved, and as well as there would be a possibility for one system to use different user interfaces, for example, users could work simultaneously with one system from MS Windows environment and Internet.

New data presentation possibilities to substitute the trees seen in the windows in the current implementation are also being searched for. One of the solutions is to approach maximally the appearance of the IST windows to those of generally accepted MS Windows Explorer.

Conclusion

The paper describes a new approach to system design implementation – the design interpretation. This approach ensures exact compliance of system implementation with design description and makes the system easy to maintain. Implementation of the above mentioned method is also discussed.

The method is applied in practice in real projects. It should be developed according to today's updated technologies, and new easy to use versions in which application of all IST modules would justify themselves should be elaborated.

References

Bruce I. Blum. Software Engineering a Holistic View. Oxford University Press, 1992.

Treimanis M., Krasts O., Nazartchik I. GRADE. GRAPES Development Environment. University of Latvia, 1992.

Bicevskis J. Specification Language GRAPES/PLUS. University of Latvia, 1992.

Oracle Designer'2000. Product Overview. Oracle Corporation.

Treimanis M. ISTehnology – Tehnology Based approach to Information System Development. University of Latvia, 1997.

James Ruabaugh and Co. Object-Oriented Modeling and Design. Prentice-Hall, 1991.

Sowa J.F., Zachman J.A. Extending and Formalizing the Framework for Information Systems Architecture. IBM Systems Journal, Vol. 31, No. 3, pp. 590 – 616, 1992.

David R.Hampton. Contemporary Management. McGraw-Hill, 1981.

Capers Jones. CASE's Missing Elements. IEEE Spectrum, June pp. 38-41, 1992.

LATVIJAS UNIVERSITĀTES RAKSTI. 2004. 669. sēj.: DATORZINĀTNE UN INFORMĀCIJAS TEHNOLOĢIJAS. 107.–116. lpp.

Semantics and Equivalence of UML Class Diagrams

Ģirts Linde
IMCS, University of Latvia,
glinde@acm.org

One and the same "real world" can be modeled by different UML class diagrams, which in such a case can be considered "intuitively equivalent". A new approach to the formalization of this "intuitive equivalence" of class diagrams is proposed, based on the formal object-oriented cognitive process. Two theorems are proved to support this approach. The new formalization can be used to construct algorithms for class diagram analysis.
Key words: UML, class diagrams, equivalence.

Introduction

As object modeling gains popularity, starting from the pioneering work [1], a need emerges for methods of handling and evaluating models. Object models are used in various stages of system development – from definition of requirements to system design. One problem is to validate the model as it gets transformed and refined in the lifecycle of system development. Research is already being done on this topic [2–5]. A more general problem is to compare several alternative object models of the same "real world".

Our goal is to formalize this "intuitive equivalence" for object models represented with UML class diagrams. Consider the class diagrams modeling directed graphs shown in Figure 1.

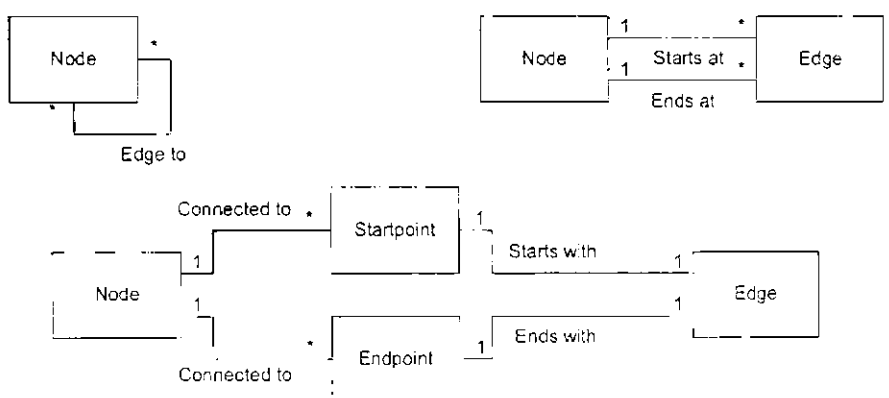

Fig. 1. Three class diagrams of a directed graph

Formally, we have here 3 totally different class diagrams. Each of them describes different instance diagrams. But intuitively we know that all three of these diagrams describe the same "real world" – directed graphs.

In [6] a formal definition of the "intuitive equivalence" of class diagrams is proposed called reduction. It is defined on the instance diagram level and uses composite classes to define the transformation between the class diagrams. In this paper a new approach to the definition of the "intuitive equivalence" of class diagrams is proposed, called semantical implication. It is based on the formalization of the object-oriented cognitive process, introduced in [7]. Two theorems are proved confirming that the new approach is more general and intuitive.

The paper is structured as follows. In section 2 a restricted formalization of the cognitive process is outlined. In section 3 the new definition of the "intuitive equivalence" is presented. In section 4 the reduction of class diagrams is briefly outlined. And finally in section 5 the difference between the two definitions of the "intuitive equivalence" is examined, proving two theorems.

The formalization of the cognitive process

The definition of the class diagram semantical implication is based on a restricted version of the formal cognitive process (for the full version see [7]) happening in the mind of a modeler when he analyzes a fragment of the real world and builds an object model for it. The fragment of the real world to be modeled, together with the object-oriented structure that gets created in the mind of the modeler we represent with a labeled directed graph and call it the cognition state (Fig. 2). During the cognitive process, as the modeler learns new concepts from the fragment of the real world, his mind is populated with new elements that correspond to objects, classes, links and associations. With every addition of a new element a new cognition state is created. The resulting sequence of cognition states we call the cognition path. The cognition path ends with a cognition state that contains the final object model as a part of it (Fig. 3).

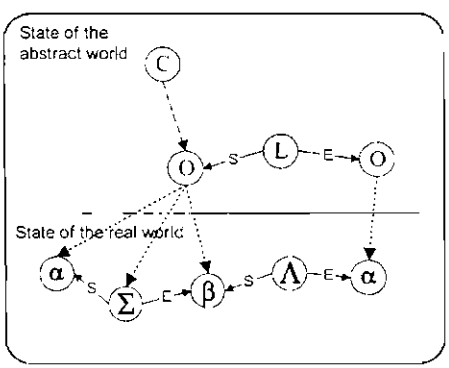

Fig. 2. A cognition state

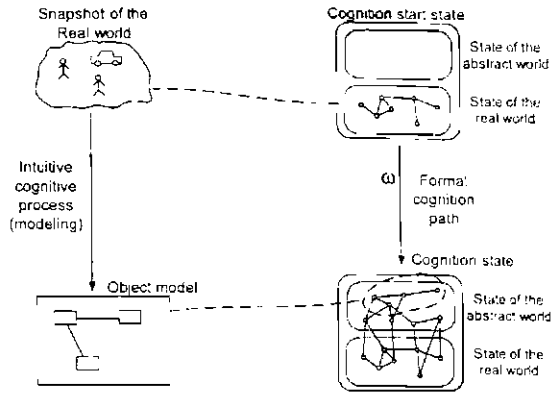

Fig. 3. Overview

A more precise description of the cognition state and the cognition path follows in the next two subsections.

Cognition state

The cognition state is defined as a labeled directed graph that consists of two interconnected parts:

- the state of the real world, which formally represents the fragment of the real world that is being modeled.
- the state of the abstract world, which formally represents the object model created in the modeler's mind on a fixed moment in time.

Each node of the state of the abstract world has one of the four labels, according to the element of the object model the node represents: O - object node, C –class node, L –link node, A – association node. (Note that links and associations are represented as nodes in the graph.) The following predefined edges are used to connect the nodes (they have a graphical notation, as shown in Fig. 4):

- startpoint (S) - connects a link or association node to its startpoint node,
- endpoint (E) – connects a link or association node to its endpoint node.
- *instance-of* – connects a class or association node to the nodes representing its instances,
- *part-of* – connects an object node to nodes representing the parts of the object.

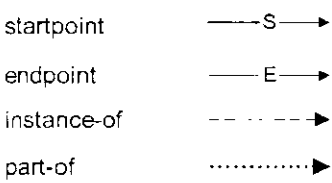

Fig. 4. Notation of predefined edges

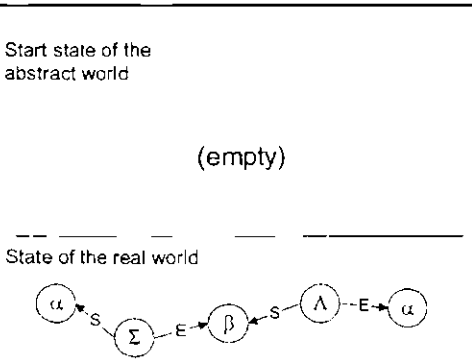

Fig. 5. Λ cognition start state

We define the cognition start state as a cognition state that consists of a state of the real world and an empty state of the abstract world (Fig. 5).

Cognition path

During the cognitive process, a cognition path is incrementally created. It is a sequence of cognition states, beginning with a cognition start state. Given a cognition state, the next cognition state is produced by extending the state of the abstract world in it – by adding a new node or by adding one of the predefined edges between the nodes. As in [6], in this paper we work with the basic class diagrams containing classes, binary associations and the four most popular multiplicity constraints (0..1, 1..1, 1..*, 0..*). So, the cognitive process can be restricted to use only the needed steps. We call the resulting cognition path a constructive cognition path.

Definition. We call a cognition path constructed using the following steps a constructive cognition path (CCP):

- "create a new aggregate object" (Fig. 6), which represents a subgraph of the real world state: create a node with label O, and connect it to all nodes of the subgraph by the edge part-of;
- "create a new class": create a node with label C (a class node);
- create an edge instance-of from an existing class node to any existing object node that has no instance-of edges connected to it;
- "create a new association": create a node with label Λ (an association node), connect it to two existing class nodes by the edge S and the edge E;
- "create a new link" (Fig. 7): create a node with label L (a link node), connect it to two existing object nodes by the edge S and the edge E;
- create an edge instance-of from an existing association node to any existing link node that has no instance-of edges connected to it; this is allowed only if the nodes connected by the link (nodes to which edges S and E go) are connected by instance-of edges to the class nodes, connected by the association (with edges S and E, respectively).

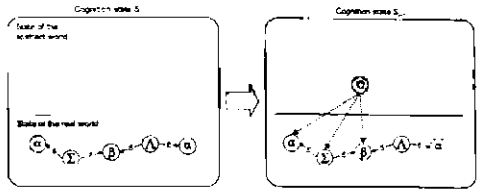

Fig. 6. Adding an object

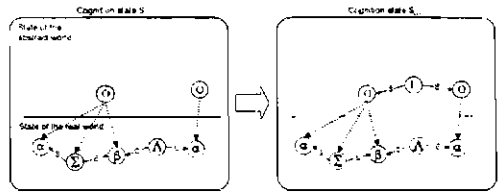

Fig. 7. Adding a link

If we assign names to the class and association nodes of a given CCP then we can construct the corresponding instance diagram.

The new definition of the "intuitive equivalence"

Consider a real world state R, and two CCPs P1 and P2 starting from this real world state R. If P2 can be obtained from P1 aggregating some of the objects and classes while preserving some properties of the structure, then we assume that these cognition paths model the same concept in different levels of detail. A more precise definition follows.

Definition. We say that P2 is less detailed than P1, if the following 4 conditions hold:

1) if two nodes of R in P1 are connected by part-of edges to one and the same object node (Fig. 8 a), then in P2 they are also connected by part-of edges to one and the same object node (Fig. 8 b),

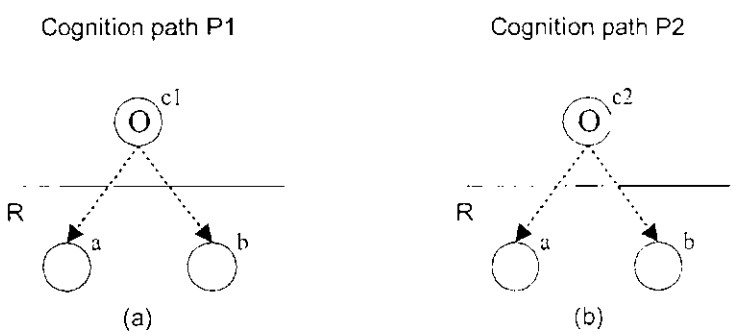

Fig. 8. Condition for object nodes

2) if there are two nodes a and b in R that are in P1 connected by part-of edges to object nodes c1 and d1, respectively, and there is a link node connected by S edge to c1 and by E edge to d1 (Fig. 9 a), then in P2 they are connected by part-of edges to one object node (Fig. 9 b) or they are connected by part-of edges to object nodes c2 and d2, respectively, and there is a link node connected by S edge to c2 and by E edge to d2 (Fig. 9 c).

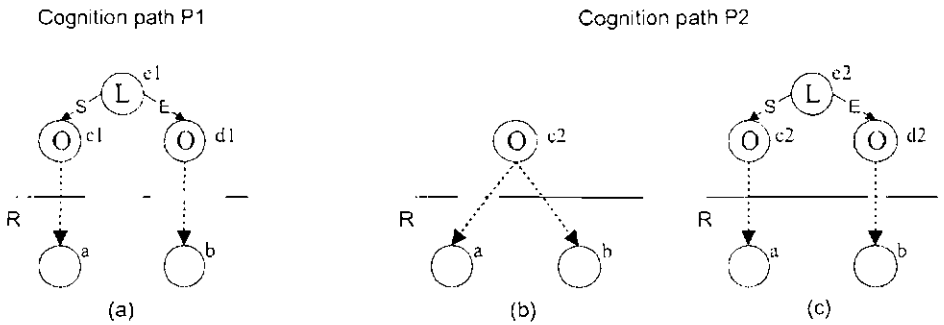

Fig. 9. Condition for link nodes

3) if there are two nodes a and b in R that are in P1 connected by part-of edges to object nodes c1 and d1, respectively, and the object nodes c1 and d1 are connected by instance-of edges to the same class node (Fig. 10 a), then in P2 they are connected by part-of edges to one object node (Fig. 10 b) or they are connected by part-of edges to object nodes c2 and d2, respectively, and the object nodes c2 and d2 are connected by instance-of edges to the same class node (Fig. 10 c).

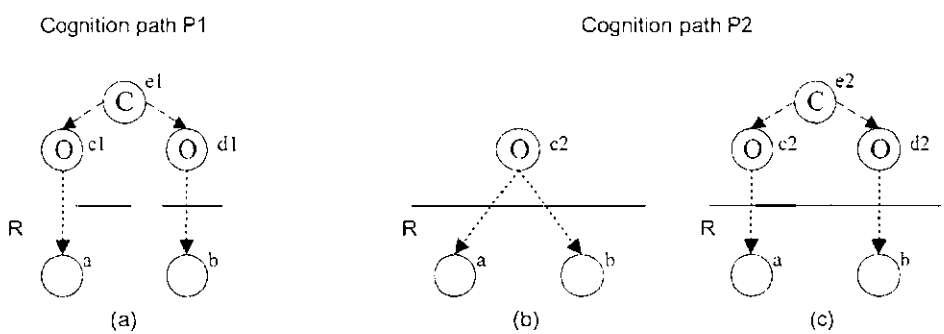

Fig. 10. Condition for class nodes

4) if there are 4 nodes a, b, c and d in R, and P1 contains 4 object nodes a1, b1, c1 and d1, 2 link nodes e1 and f1, and an association node g1, connected as in (Fig. 11 a), then in P2 there are two cases possible:

- a and b are connected by part-of edges to one and the same object node, and c and d are connected by part-of edges to one and the same object node (Fig. 11 b)
- or there is the same configuration as in P1 - there are 4 object nodes a2, b2, c2 and d2, and 2 link nodes e2 and f2, and an association node g2, connected as in Fig. 11 c.

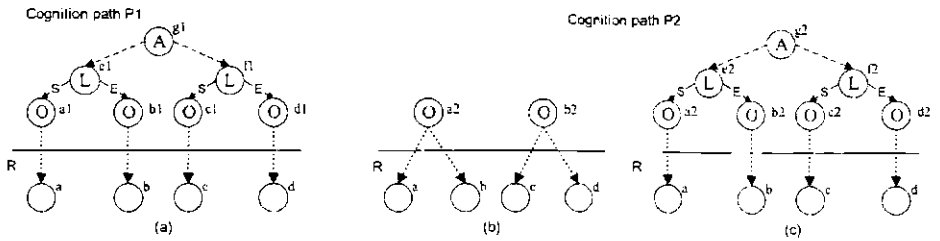

Fig. 11. Condition for association nodes

Now, using this method of comparing cognition paths we define the "intuitive equivalence" of class diagrams, in this paper called semantical implication.

Definition. We will say that a class diagram CD1 semantically implies a class diagram CD2, if for every state of the real world R:

- if there exists a CCP P1 starting from R whose corresponding instance diagram ID1 satisfies CD1,
- then there exists a CPP P2 starting from R that is less detailed than P1 and whose corresponding instance diagram ID2 satisfies CD2.

Reduction of class diagrams

In [6] the "intuitive equivalence" of class diagrams is formalized as a reduction. The reduction of the more detailed diagram to the less detailed one is defined introducing a set of concepts. Every concept is represented with a composite class containing some of the existing classes and associations, and assigning multiplicity constraints to the enclosed classes. The set of new concepts must cover all the classes of the class diagram.

The multiplicity constraints assigned to the contained classes must guarantee that the composite classes have only connected instances. When reducing an instance diagram with a set of concepts, each group of objects and links that forms an instance of one of the concepts is replaced with a new object, and so a new (less detailed) instance diagram is constructed.

Definition. We will say that a class diagram D1 can be reduced to a class diagram D2, if there exists such a set of new concepts S, using which every instance diagram of D1 can be reduced to an instance diagram of D2, and all instance diagrams of D2 can be constructed this way from instance diagrams of D1.

The relation between the two definitions

In this section we show that the semantical implication is more general than the reduction.

Theorem 1. If a class diagram D1 can be reduced to a class diagram D2, then D1 semantically implies D2.

Proof.

Consider

- a class diagram D1 that can be reduced to D2 by a set of concepts S,
- a state of the real world R, for which a constructive cognition path P1 exists starting from R, whose corresponding instance diagram ID1 satisfies D1,
- an instance diagram ID2 that is reduced from ID1 using a set of concepts S.

To prove the theorem we show that we can construct a constructive cognition path P2 starting from R that is less detailed than P1 and whose corresponding instance diagram is ID2.

First we construct the last cognition state X of P2 whose corresponding instance diagram is ID2 and which is based on the same state of the real world R:

- group the object and link nodes and the class and association nodes into subgraphs according to the new concepts in S,
- replace each object-link subgraph with an object node, and each class-association subgraph with a class node,
- replace multiple instance-of edges between two nodes with one instance-of edge,
- replace multiple link nodes (together with S and E edges) that are between the same object nodes, and that are connected by instance-of edges to the same association node, with one link node (and one pair of S and E edges).

Now it is easy to construct a constructive cognition path P2 that starts with the state of the real world R and ends with the state X. The cognition path P2 is constructed in the following order:

1) create aggregate object nodes with part-of edges (in any order),

2) create link nodes with S and E edges (in any order),

3) create class nodes (in any order),

4) create instance-of edges between object nodes and class nodes (in any order),

5) create association nodes with S and E edges (in any order),

6) create instance-of edges between link nodes and association nodes (in any order).

The set of new concepts S by definition is constructed so that every class and association of D1 belongs to exactly one new concept. Keeping that in mind it is easy to see that the new cognition path P2 is less detailed than the cognition path P1 (all four conditions are satisfied).

Theorem 2. There exist two class diagrams D1 and D2, such that

- D1 semantically implies D2, but
- D1 cannot be reduced to D2.

Proof.

Consider the two class diagrams in Fig. 12. They both describe directed graphs. The difference is that D1 allows us to model the structure of graphs, but for D2 every object instance is a whole graph.

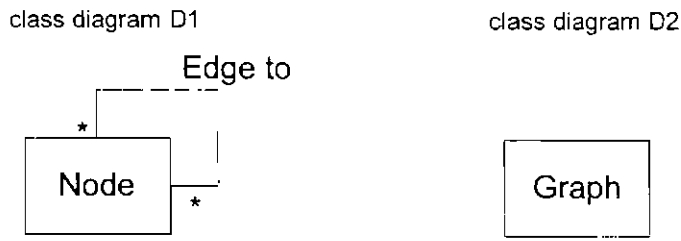

Fig. 12. Two class diagrams of a directed graph

Let's prove that D1 semantically implies D2. Consider an arbitrary state of the real world R, for which there exists a CCP P1 whose corresponding instance diagram satisfies D1. We can construct a CCP P2 that starts from the same state of the real world R and contains one object node, which is connected by part-of edges to all the nodes of R, and which is connected by an instance-of edge to a class node. The corresponding instance diagram of P2 satisfies D2. It is easy to see that P2 is a less detailed CCP than P1.

Now, let's see if D1 can be reduced to D2. It can be proved that the only new concept that can be constructed for the diagram D1 is a trivial composite class that contains the class Node, but doesn't contain the association Edge-to. The association Edge-to cannot be included in a new concept together with Node, because it forms a cycle, and cycles are not allowed inside the concept (that comes from the requirement that a new concept must have only connected instances). Therefore, D1 can be reduced only to D1, but cannot be reduced to D2.

Conclusion

In the paper a new formalization approach of the UML class diagram "intuitive equivalence" is presented. It is enabled by the semantics of class diagrams that is based on the formalization of the object-oriented cognitive process. Two theorems are proved to compare the two definitions of the class diagram "intuitive equivalence". The theorems confirm that the new definition is more general – the reduction is a special case of the semantical implication.

In fact, Theorem 2 can be generalized to prove that any class diagram semantically implies the trivial class diagram containing one class. That is a fundamental property of system modeling. These theoretical results show that the semantical implication is now very close to the concept of the class diagram "intuitive equivalence". The next problem that must be researched is the algorithmical decidability of the class diagram semantical implication, as it is already done for the reduction in [6].

Finally, it must be noted that this paper gives another confirmation of the sufficient generality of the object-oriented cognitive process formalization. It is shown that the formalization can be successfully used to reason about the problems of the class diagram equivalence.

References

J.Rumbaugh, M.Blaha, W.Premerlani, F.Eddy W.Lorsen. Object-oriented modeling and design. Englewood Cliffs, NJ: Prentice Hall, 1991.

A.S.Evans, R.B.France, K.C.Lano, B.Rumpe. The UML as a Formal Modelling Notation. UML'98. LNCS, Vol. 1618, Springer, 1999

K.Lano, J.Bicarregui. Semantics and Transformations for UML Models. UML'98. LNCS, Vol. 1618, Springer, 1999

A.S.Evans. Reasoning with UML class diagrams. WIFT'98. IEEE CS Press, 1998

K.C.Lano. A.S.Evans. Rigorous Development in UML. FASE'99, ETAPS'99. LNCS, Vol. 1577, Springer, 1999

G. Linde. Reduction of UML Class Diagrams. In: proceedings of the Fifth International Baltic Conference on Databases and Information Systems. Tallinn, 2002.

G. Linde. The Cartoon Method of Semantics Definition for Modeling Languages. Submitted to: Proceedings of the Latvian Academy of Sciences. Riga, 2003.

LATVIJAS UNIVERSITĀTES RAKSTI. 2004. 669. sēj.: DATORZINĀTNE UN INFORMĀCIJAS TEHNOLOĢIJAS. 117.-125. lpp.

Requirements and Options for Data Warehouses at Universities

Laila Niedrite
Lecturer at the University of Latvia
Raina blvd. 19, Riga, Latvia, +371 7034439
lnied@lanet.lv

This paper deals with the problems of the changing role of traditional universities on the educational market, and how data warehousing could help the universities solve their problems. It describes the experience of different universities and the issues of feasibility study of the data warehousing project at the university.
Key words: data warehouses, requirements, universities.

Introduction

Data warehousing is traditionally used in business. Higher education is trying to follow the growing trend in development and usage of data warehouses; however, the reasons for doing it are often more concerned with scientific and educational issues than with financial benefits. Higher educational institutions have grown in recent years into large businesses and the management of these institutions has changed and become more business-like. The management of higher educational institutions can benefit from using data warehousing just as other traditional business organizations.

The topic of data warehousing [5], [13], [4], and [7] comprises architectures, algorithms, models, tools, organizational and management issues for integrating data from several operational systems in order to provide information for decision support, e.g., using data mining or OLAP tools. The data warehousing technology provides integrated, consolidated and historical data. A data warehouse can be realized as a separate database containing these integrated data.

A data warehouse is built by extracting data from different data sources, transforming them and then loading into a separate database where specialized data modelling concepts, e.g. star schema, and database structures like materialized views [14] are used. Finally, the data are accessed with different data analysis tools as shown in Figure 1.

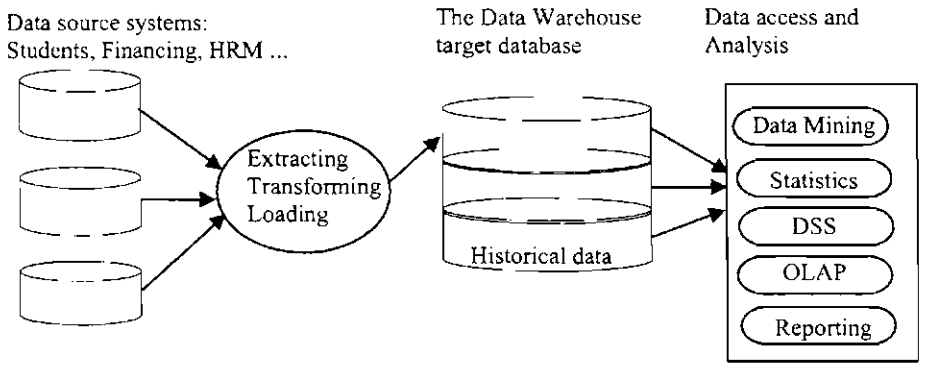

Figure 1. Common data warehousing environment

This paper will give further insight into the business processes of higher educational institutions, and some key benefits of having a data warehouse in business are examined concerning higher education.

The following section presents the business process model for higher educational institutions and university management issues in the new competitive environment. An overview of possible areas of application of data warehousing in education is given in Section 2. Section 3 presents the possible system architecture for the new application area, and the relevant experience of universities using data warehousing. Section 4 draws conclusions on the data warehouse feasibility study at the University of Latvia.

The Education Process as a Business Process

The starting point for finding out the necessity for higher education institutions to build and use data warehousing could be their traditional management and business processes. The two main university activities are education and research, and the most important support processes are human resource management and financial resource management [11].

In the last decade universities have experienced significant changes in the education area. One of the important issues is the growing competition between educational institutions.

Colleges and universities rarely express their policies, intentions, and practices in competitive terms. However, the pressure on traditional resources doubled with the emergence of technology-based education delivery systems will force competitive thinking [6].

If we consider education as business; we have to define the key terms in this context. The universities as "education providers" have to find out, what is the product? What are the resources? What is the price of the product? The students as „customers" have different important questions: from which institution to buy, which product exactly to buy? Finally, the customer will have to know about the quality and price of the product to be able to make the right decision.

There are many examples of using the market terminology in higher education now. One of the best known cases in this area is the University of Phoenix, which focuses on educational needs of adults. The new trend is the formation and growth of corporate educational institutions, many of which have become universities with accredited study programs in recent years. For example, Motorola University co-operates with universities around the world to develop and deliver courses for the Motorola Corporation employees.

The most challenging issues for university management in the new competitive environment are [12]:

- Public relations,
- Competition, including universities abroad,
- Market analysis,
- Strategic planning,
- Revenues/ expenditure analysis,
- Calculation of expenses,
- Cooperation with traditional businesses.

Alongside developing their policies in the new competitive environment, universities are looking for methods to help them evaluate the answers and to support them and their customers to make the right decisions.

Higher education is facing many problems in the next years and IT will play a major role in determining how institutions resolve and solve these problems [15]. Many universities are looking at data warehousing with the hope that it can help them to solve their problems.

The reasons why data warehouses are developed and successfully used in management and decision support in traditional business areas like sales, banking and others are the following:

- the data warehouse makes the information in the organization more accessible,
- integrates data from many operational data sources and thus ensures quick access to the information about business activities of the whole organization
- the data are user-friendly and consistent.

As it has been mentioned above, the universities' main activities are education, research and management, and also the new business activities, for example, processes which become important for the existence and development of universities. Therefore, we can assume that universities, like businesses, can benefit from data warehousing.

The following section deals with some scenario examples for data warehouse development at universities, which are connected with the traditional higher education institution's processes (scenarios 3 and 4) and also with the new business processes (scenarios 1 and 2).

Specifications for the University Data Warehouse

There are some attractive university data warehouse implementation scenarios. Components of them have been implemented at European and American universities [1], [2], [3].

Data warehouse implementation scenario 1.

The first specification is about attracting good students. With the ever escalating costs and competition associated with higher education, universities and colleges are very interested in attracting and retaining high-quality students [8].

A data warehouse can enhance the existing information system that handles admissions to university study programmes, providing access to the required information. One example data warehouse star schema for the admission process is shown in Figure 3. The star schema reflects the current administrative processes for student admission in the IS of the University of Latvia.

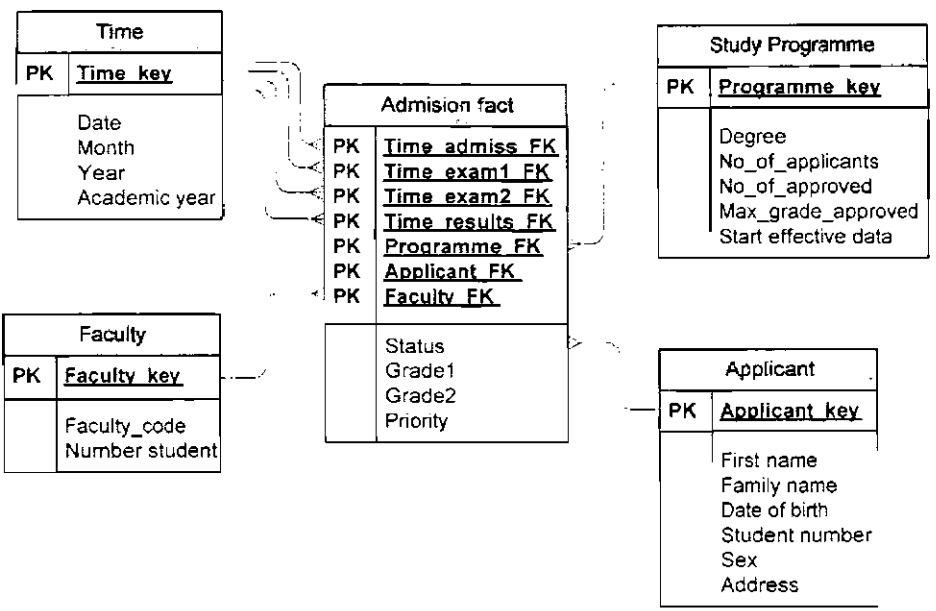

Figure 3. Example data warehouse star schema for admissions.

For example [1], if the admission algorithm is based on school grades and selected number of study programmes in priority order, the information about the admission process in previous years could help applicants avoid mistakes in choosing the programmes, because they do not really comprehend the selection algorithm.

Data warehouse implementation scenario 2

The next specification is to support customer relationship management (CRM) strategies.

'CRM aligns business processes with customer strategies to build customer loyalty and increase profits over time.' (Harvard Business Review, 2002).

Each year colleges and universities confront the challenge of meeting higher levels of service demanded by an increasingly sophisticated and competitive marketplace and consumer base. At the same time there is an ever present need to cut costs and tighten budgets. Therefore, it becomes more and more important for each service interaction to be effective in both cost and outcome. In order to do this, a greater understanding of the customer is critical [9].

Universities are now challenged with the needs of many different groups of learners. The universities are working with multi-generational students, and also the correspondence students are expecting new services. Another important issue is the feature of learners in the information age to get information when and how they choose.

Universities must now evaluate their services taking into account that the students are customers. Universities are starting to develop new strategies to manage their customer relationships in various stages of the student's lifecycle. The universities have to think about new services in each phase of the lifecycle to support students' different needs on the way towards graduation.

There are some additional outcomes with effective CRM strategy: reduced costs, if the self-service is supported with web–applications; improved document flow in the institution and influence on new technologies and investments.

Data warehouse implementation scenario 3

This scenario focuses on the improvement of the study process [2], for example, the assessment of study courses concerning the student and course performance. The evaluation of detailed statistics such as the number of students enrolled, evaluated and approved, the average mark, and the average approved mark can help to find out the potential problems for a particular course before they arise. It helps better understand the student needs, or in business terms, what are the students buying?

Another issue is the quality assessment of the study programme. Statistics on admissions, new students, drop-outs and graduates and average studying duration before graduation can give the faculty management the necessary feedback and can influence how the programme works and changes.

Another possible way to conduct course and study programme evaluation is by getting information from student questionnaires and by publication of aggregated data for all participants: students, lecturers and managers.

Data warehouse implementation scenario 4

This scenario describes resource planning at the university, for example, planning the teaching service. If the managers of the faculty plan the necessary number of classes for a particular course, they must know the statistics about the previous year. It can help to avoid empty or full classes and to find out the workload of teaching staff.

Another example is Human Resources, when saving all transactions made with university personnel records can help plan the carrier of employees and make the right decisions about new recruitments.

The financial resources of a university, in context with the size of the student body, as scientific results over several years are also very significant issues to analyze.

One less obvious example [3] for this scenario aims to measure the international attractiveness of the institution. The purpose is to control the institution's policy on its international reputation and exchanges, with budget involved.

Data Warehouse and University Information Technology Architecture

Because many universities' data warehousing scenarios mentioned above are concerned with Customer Relationship Management (CRM), it is necessary to understand the role of the data warehouse in the university IT architecture and its connection with CRM applications

The Connect Enterprise Architecture [9] also fits for universities and illustrates as in Figure 4, how CRM integrates many existing applications into a unified customer facing strategy.

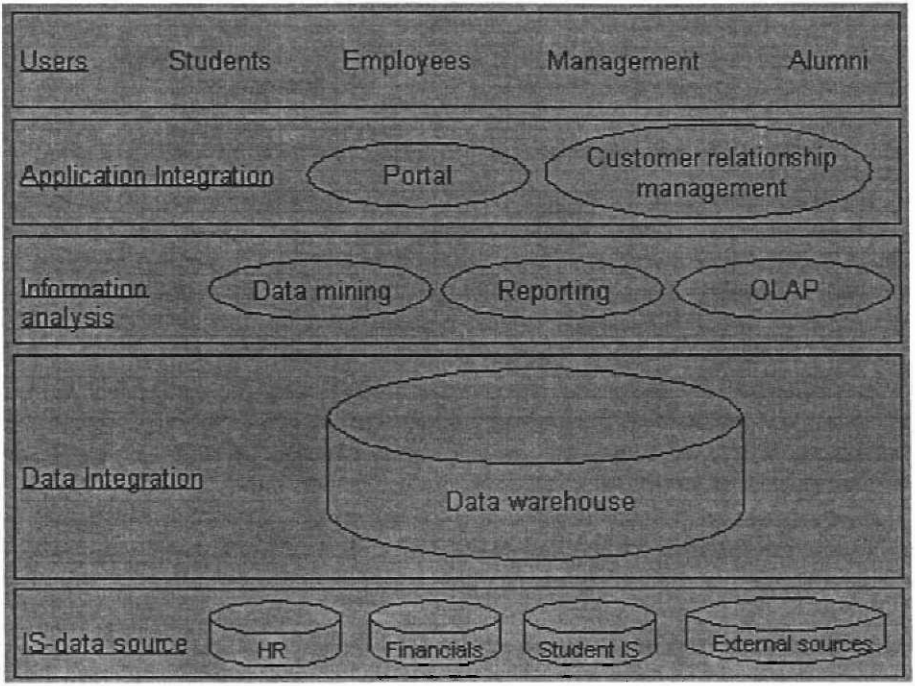

Figure 4. Data warehouse connections with CRM applications.

The layer of Enterprise Application Integration provides access to Back Office applications (ERP, Student IS ...) from various solutions: data mining, CRM applications, Portals, Knowledge Management applications.

The integration of Back Office Applications' data can be solved by using a data warehouse. The data warehouse also serves the reporting and analytical needs of an

institution, so the data warehouse can be a part of CRM solution which focuses on front office activities, e.g., customer service and support.

The present developments in universities in the field of data warehousing are very different in their decisions, which architecture and tools to use, and also which business processes to support with data warehousing solutions.

Among higher education institutions in the USA, less than 25% have or are planning a data warehouse [10]. Only a limited number of database tools is used.

The choice mostly depends on existing environments in other projects at the university. This is also true for client tools. The most popular is the ORACLE database and Web based client side tools.

Only a very small number of universities have all their information sources integrated into a corporate warehouse. Most are starting with students' data, human resources or finance data.

Many data warehousing projects in universities have been initiated in the IT department, and not all of them have obtained management sponsorship yet.

No similar survey has been done in Europe, and to asses the situation with data warehouses in European universities one has to look up the information on the Internet and the annual conference 'European Universities Information Systems' (EUNIS). The number of participating universities from different countries is usually around 100 every year, but during the last 5 years, the number of presentations on data warehouses was less than 10. For example, in 1999 there were 2 presentations from Ljubljana in Slovenia and from Porto in Portugal. The most significant data warehouse project in Europe among universities is in France, where the project is developed on a national scale. All French universities which had previously implemented the university information system APOGEE (also a national-scale project) can now participate in the data warehouse project.

The Data Warehouse Feasibility Study at The University of Latvia

As an example of a university starting the development of the university data warehouse we can consider the University of Latvia. The following issues are important at the starting point of the project:

The existence and quality of data sources

In the case of the University of Latvia, an information system was developed based on the Oracle database and Oracle Application Server. During the 5 years of the system development and implementation process, a large amount of various data have been collected. It is now possible to provide access to operational data, but the possibility to analyze historical data in different ways is limited. One of the questions for the evaluation of the feasibility of the data warehouse is the technical feasibility, which means the existence of data for expected data warehouse and the quality of existing data. At the University of Latvia the management information system (LUIS in Latvian) has been developed since 1996. The starting year of the students' module was 1997, of the study fees module - 1998, of the courses and courses' enrolment module - 2000, and the starting year of the admission module was 1998. The data accuracy differs not

depending on the module, but on the faculty, in some cases it depends on a particular program of study. The existing data are accurate enough, but in some cases the lack of data is obvious. The second possible data source for the data warehouse at the University of Latvia in addition to LUIS is the accounting system.

Reasons for having a data warehouse

For example, one reason for starting a feasibility study of a data warehousing project at the University of Latvia is burdened access to the necessary information in the operational systems. Another reason is the growth of competition among universities and the new roles of traditional universities. One more question regarding readiness is the business motivation of the institution. The business motivation of the University of Latvia is based on the changing situation in education, which was described earlier in this paper. The management of the University of Latvia is interested in a more flexible reporting facility, in the analysis of the integrated information, and the most actual need is a new portal. The idea of the portal fits CRM solutions and also fits the data warehouse as a data integrator.

Users' acceptance

Prior to starting a data warehouse project it is important to understand the readiness of the institution to have a data warehouse [7]. One of the questions for evaluation is the existence of data for the expected data warehouse. Also an important question is the readiness of people to use the data warehouse information. In the University of Latvia the users have long experience in usage of operational systems, and if the data warehouse will satisfy also the reporting needs of the users, the acceptance of the data warehouse will depend on the correctness of extracted data and reports. The users were also involved in the requirements analysis.

The decision between different development scenarios and the right choice for the first development stage.

The possible scenarios regarding specifications have to be presented to the university management and discussed. The information about users' needs and priorities help to make the right decision. In our case study we collected information about the number of potential users and about the purposes of the usage. We evaluated this information to get quantitative measurements expressing the more urgent needs. The earlier mentioned 4^{th} implementation scenario's subset of problems concerning the human resources and the financial resources of the university in context with the number of students, and scientific results were accepted as the starting point of the implementation.

Conclusions

Universities are not the typical owners of a data warehouse system, because the data warehouse projects need financial investments, which in the case of universities, are not obviously comparable with gained results in terms of money. The reasons why universities have or are starting the development of the data warehouse are only in the long term the expected return on investment, or the growing competition. According to

their prior business functions and goals, universities build data warehouses to improve the quality of education and communication with their customers – students. The improvement of the existing reporting facility of operational systems and improvement of data accessibility can be mentioned as a secondary reason.

References

[1] Bajec,M., Rupnik,R., Krisper,M. Using Data Warehouses in University Information Systems, Proceedings of conference EUNIS99, Espoo, Finland, 1999.

[2] David,G., Ribeiro,L. Getting Management Support from a University Information System. Proceedings of conference EUNIS99, Espoo, Finland, 1999.

[3] Desnos,J.F. A National Data Warehouse Project for French Universities, Proceedings of conference EUNIS2001, Berlin, 2001.

[4] Hammer,J., Garcia-Molina,H., Widom,J., Labio,W., Zhuge,Y. The Stanford Data Warehousing Project.

[5] Inmon,W.H. Building the Data Warehouse, John Wiley, 1996.

[6] Katz,R.N. Competitive Strategies for Higher Education in the Information Age. Proceedings of conference EUNIS99, Espoo, Finland, 1999.

[7] Kimball,R., Reeves,L., Ross,M., Thornthwite,W. The Data Warehouse Lifecycle Toolkit, John Wiley, 1998.

[8] Kimball,R., Ross M. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, John Wiley, 2002.

[9] KPMG Consulting, Managing Customer Relationships in Higher Education, 2002

[10] NCHEMS, Information Architecture: the Data Warehouse Foundation, CAUSE/EFFECT , Volume 20, Number 2, Summer 1997, pp. 31-33, 38-40, 60.

[11] Sinz, E. J.; Böhnlein, M.; Ulbrich-vom Ende, A.: Konzeption eines Data Warehouse-Systems für Hochschulen. In: Workshop "Unternehmen Hochschule" (Informatik'99, 29. Jahrestagung der Gesellschaft für Informatik, Paderborn, 5.-9. Oktober), 1999, S. 111-124.

[12] Stonis J., Augstākā izglītība kā pakalpojumu tirgus dalībnieks, LU konference, 2003.

[13] Widom,J. (ed). Special Issue on Materialized Views and Data Warehousing, IEEE Data Engineering Bulletin, June 1995.

[14] Yang,J., Karlapalem,K., Li,Q. Algorithms for Materialized View Design in Data Warehousing Environment, Proceedings of the 23rd International Conference on Very Large Data Bases, Athens, Greece, August 1997.

[15] Zastrocky, M.R. The Information- and Technology- Enabled University of 2004, Proceedings of conference EUNIS99, Espoo, Finland, 1999.