ITHEA

# International Journal INFORMATION THEORIES & APPLICATIONS

2009 Volume 16 Number 2

# International Journal
# INFORMATION THEORIES & APPLICATIONS

Volume 16 / 2009, Number 2

IJ ITA is official publisher of the scientific papers of the members of
the ITHEA® International Scientific Society

IJ ITA welcomes scientific papers connected with any information theory or its application.

IJ ITA rules for preparing the manuscripts are compulsory.
The **rules for the papers** for IJ ITA as well as the **subscription fees** are given on *www.ithea.org* .
The camera-ready copy of the paper should be received by http://ij.ithea.org.
Responsibility for papers published in IJ ITA belongs to authors.

General Sponsor of IJ ITA is the **Consortium FOI Bulgaria** (www.foibg.com).

International Journal "INFORMATION THEORIES & APPLICATIONS" Vol.16, Number 2, 2009

Printed in Bulgaria

Edited by the Institute of Information Theories and Applications FOI ITHEA®, Bulgaria,
in collaboration with the V.M.Glushkov Institute of Cybernetics of NAS, Ukraine,
and the Institute of Mathematics and Informatics, BAS, Bulgaria.

Publisher: ITHEA®
Sofia, 1000, P.O.B. 775, Bulgaria. www.ithea.org, e-mail: info@foibg.com

ISSN 1310-0513 (printed) ISSN 1313-0463 (online) ISSN 1313-0498 (CD/DVD)

# METHODS FOR AUTOMATED DESIGN AND MAINTENANCE OF USER INTERFACES

Valeriya Gribova

*Abstract: Methods for automated development and maintenance of intellectual software user interfaces are proposed. They rest on an ontology-based approach and intended for decreasing effort and time for user interface development and maintenance. A survey, principal conception of the approach, project components and methods of automated development of the project as well as comparison the methods with their analogues are described.*

*Keywords: ontology, interface project, automated generation*

*ACM classification: I.2.2 Automatic Programming*

## Introduction

An important task of software development is user interface development according to users' requirements. A major tendency is user interface complication. It grows out of increasing functionality of software and different conditions of its application which are changing constantly. As a result, development, implementation and modification of user interfaces are costly and time-consuming tasks because the developer has to take into account a lot of interrelating but often conflicting factors. According some reviews (for example, [1]) the user interface development takes 50% time required to software and it's the source code aggregates 48% of the total code.

To decrease labour-intensiveness and time for user interface development, an ontolology-base approach to its development, automatic generation and maintenance is proposed. The basic ideas of the approach are:

- to form a user interface project by ontologies that describe features of every component of the project;
- based on this project to automatically generate the user interface code to a programming language;
- to link the interface code with the application code (a programming language, an interaction type with application, local or distributed, is determined by the developer).

Therefore, development, implementation, maintenance of a user interface is added to development and maintenance of its project.

Application of the tool based on the ontological approach has shown that design and maintenance of the presentation component in the user interface project remains difficult for complicated and complex interfaces due to high requirements to developers. They are to know usability principals, various standards of development, take into account users' requirements, conditions and environment of using software, and so on. As a result, development of methods and tools for automation of the presentation component design in the user interface project is an urgent task.

## Present state in the field of user interface design automation

User interface development is a very complicated task. Experts in different fields take part in its design. To create a user interface in accordance with tasks to be solved the following information is required for the developer [2]:

- A domain model for choosing appropriate user interface elements for input/output data;

- A Style Guide for the target platform (for example, Apple Human Interface Guidelines [3], Windows Vista User Experience Guidelines [4], Osf/Motif Style Guide; SAP - R/3 Style Guide [5], Style Guide and Certification Checklist [6] etc.) for providing uniform for a target platform interface style, choosing interface elements and including in a user interface additional features specific for the platform;
- A task model for determining a set of windows according to a scenario dialog;
- A user model for design the user interface suitable for a user of the interface;
- An application model for linking the application with the user interface.

At present user interfaces are developed, as a rule, with specialized tools – Interface Builders and Model-Based Interface Development Environments.

Fig. 1 shows the basic scheme of the user interface development using Interface Builders.

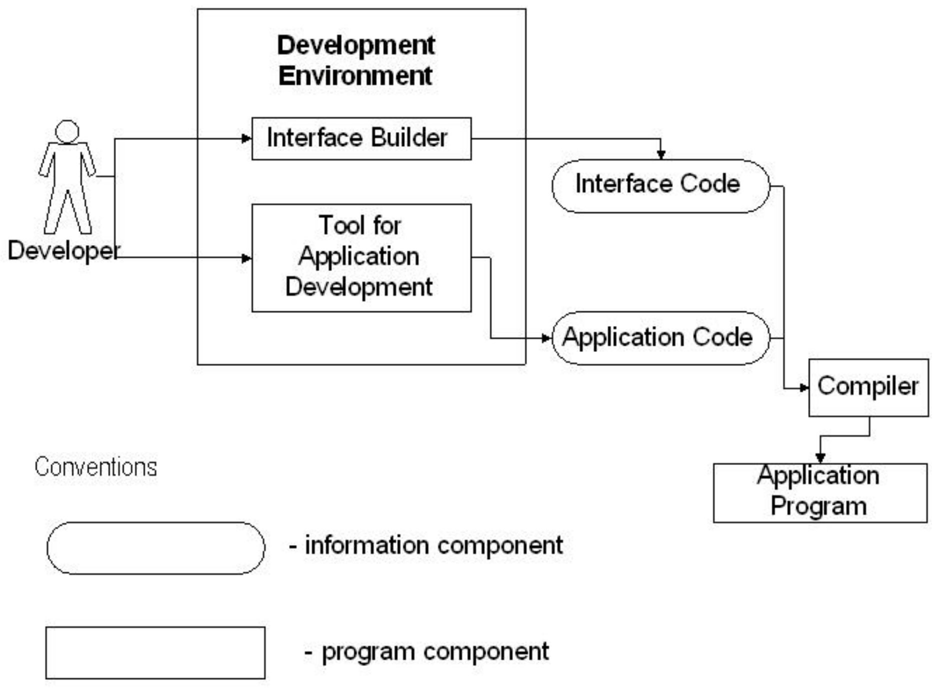


Fig 1. The basic scheme of the user interface development using Interface Builders.

Interface Builders do not have special tools for development of domain and task models [2] so developers use third-party software or an application specification. Interface Builders support design of only standard interface elements and designers should know rules of choosing and locating them. Solutions of these tasks depend on designer experience and qualification. For linking the user interface and the application a program code in a programming language (C++, Pascal, Java, C# etc.) should be written.

As an alternative to eliminating problems with Interface Builders, Model-Based Interface Development Environments (MB-IDE) are used. The key difference between Interface Builders and MB-IDEs is a declarative model interface, including all required information about the user interface [7-10]. Fig. 2 shows the basic scheme of the user interface development with MB-IDEs. Using modeling tools the developer forms components of the user interface model. Then, using this model, the source code is generated. First, the developer forms a domain model and a task model; after that based on these models, the user interface designer forms a presentation and scenario dialog models. The next step is to form a model of linking the interface with the application. These models are necessary for automated generation of the user interface code.

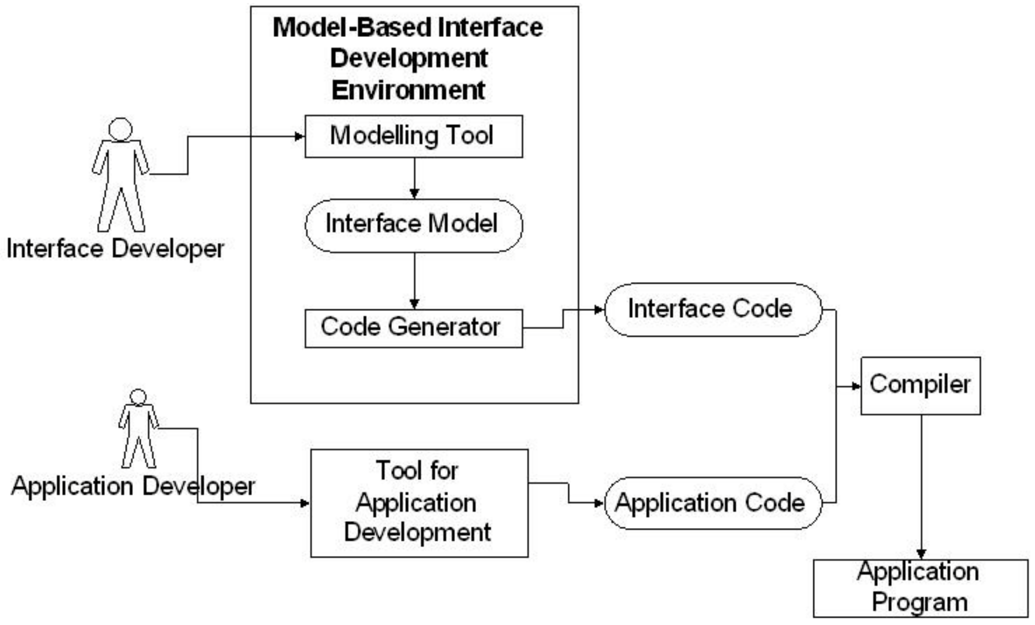


Fig. 2. The basic scheme of the user interface development using MB-IDEs

Some MB-IDEs have modeling tools. They assist the developer to build the components of the user interface model [10-13]. The main goals of the modeling tools are to hide the syntax of the modeling languages from developers, and to provide a convenient interface for developers for specifying large quantities of information stored in the model. A wide range of modeling tools has been developed, often specified different levels of the model. These tools may be text editors to build textual specifications of models (ITS [13, 14], Mastermind [15]), form-based tools to create and edit model elements (Mecano [16]), and specialized graphical editors (Humanoid [12, 17, 18], FUSE [11], and many others).

As shown in Error! Reference source not found., automated design tools usually use a repository of design knowledge or design guidelines to control the behavior of the design tool.

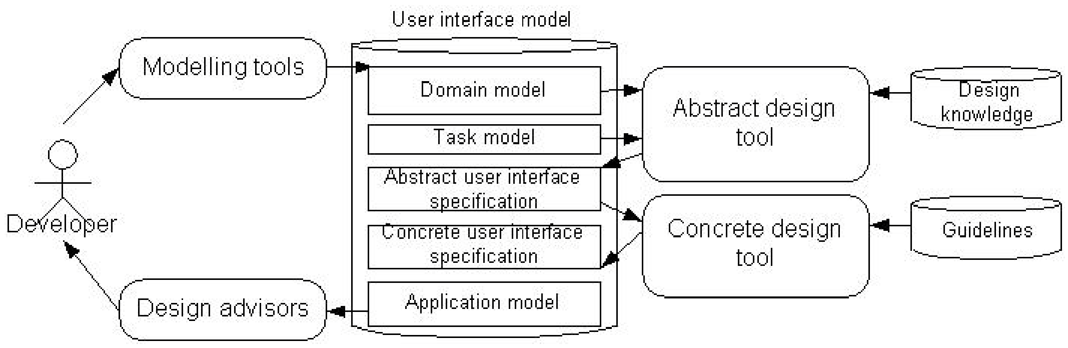


Fig. 3. The basic scheme of user interface model development using modeling tools

The primary goal of many MB-IDEs is to automate the design and implementation of a user interface. Experts develop a domain model (it describes the structure and attributes of the information that the application provides) and a task model (it describes the tasks that users need to perform) and then the abstract and the concrete user interface specifications are automatically generated from these models.

Most MB-IDEs make the following steps to automatically design an interface:

- To determine the presentation units; this step determines the windows that will be used in the interface, and information that will be shown in each window;
- To determine the navigation between the presentation units; this step computes a graph of the presentation units;
- To determine the abstract interaction objects for each presentation unit; the abstract interaction objects specify the behavior of each element in a presentation unit in an abstract way (for example, “to select one from a set”).
- To map abstract interaction objects into concrete interaction objects; the concrete interaction objects represent the widgets available in the target tool;
- To determine the window layout; this step determines the size and position of each concrete interaction object;

The first three steps build the abstract user interface specification, and the last two build the concrete specification.

Even though automated design MB-IDEs help to considerably decrease development efforts, the quality of the generated interfaces is low. The main shortcomings of MB-IDEs are:

- MB-IDEs and their modeling tools realize the only style guide and do not have any facilities to add other style guides;
- Usability requirements are often changed, style guides are often updated (for example, rules of appropriate interface element choosing), but MB-IDEs do not allow developers to keep up with the changes;
- All interface elements and their attributes are rigidly inserted in the modeling tool code so there it is impossible to extend attributes interface elements or insert new elements;
- All MB-IDEs use their own declarative models and specification languages incompatible with each other so developers can’t use advantages of different tools;
- There are no modeling tools which take into account all aspects necessary for user interface development: user requirements, the context of using application, and domain and user task structures.

One possible way to increase the quality of automatically generated interfaces, and it has been applied in MB-IDEs, is to allow developers to make post-editing. However, many MB-IDE’s developers are faced with the challenge:

How should the changes performed during the post-editing in all components of the interface model be recorded and reapplied (for example, if the developer deleted an interface element, how should he save this information in components of the model and change all links between them).

Another solution made in MB-IDEs is to specify steps 1 and 2 in a structure called a dialogue graph by the developer. Steps 3 and 4 are generated by the system in a table with default entries, but developers can edit these tables and delete any entry. Step 5 is done automatically. However, this solution, as mentioned in literature, increases the labor of development [10].

## User interface project

A user interface project defines a set of dialogs with the user for input/output parameters, control of the application program, and providing context-sensitive help for users [19, 20]. The interface project consists of a domain project, a task project, a presentation project, an application program project, a scenario dialog project, and a mapping project.

The domain project (DP) describes a set of domain terms consisting of input/output parameters of an application program, control information, and intellectual support of users activities. This component is specified by the domain ontology and is expressed by the pair: DP=( G, $\sigma$ ), where G is an oriented graph of the DP, $\sigma$ is a graph labeling. The oriented graph G=<Vertices, Edges, RootVertex>, where Vertices is a set of graph vertices, Edges is a set of graph edges, RootVertex is a root vertex.

The set of graph vertices Vertices=$\{Vertex_i\}_{i=1}^{vertexescount}$ consists of two subsets – a set of terminal vertices and a set of nonterminal vertices, $Vertex_i \in$ Nonterminal_Vertices $\cup$ Terminal_Vertices, where Nonterminal_Vertices is a set of nonterminal vertices, Terminal_Vertices – a set of terminal vertices.

The set of graph edges Edges = $\{Edge_i\}_{i=0}^{arcscount}$, $Edge_i$=<$VertexFrom_i$, $VertexTo_i$ >, where $VertexFrom_i \in$ Vertices, $VertexTo_i \in$ Vertices. The root vertex RootVertex is nonterminal vertex, RootVertex$\in$Nonterminal_Vertices. The graph labeling $\sigma$ is transformation of the vertex set and the edge set to a name set $\Omega$, where $\Omega$ = DomainName $\cup\{GroupName_i\}_{i=1}^{termgroupcount} \cup \{TermName_j\}_{j=0}^{termcount} \cup \{TermAttribute_i\}_{i=1}^{attributecount} \cup \{QualityValue_i\}_{i=2}^{valuecount} \cup\{((RealMin, RealMax), RealMeasure)_i\}_{i=0}^{floatcount} \cup\{((IntegerMin, IntegerMax), IntegerMeasure)_i\}_{i=0}^{integercount} \cup\{N_i\}_{i=0}^{stringscount} \cup$\{TermGroup, Term, Attribute, Joint, Disjoint, RealValue, IntegerValue, SrtingValue\}. The name set $\Omega$ is defined by names of the domain ontology, and graph labeling is defined by the vertex type $VertexFrom_i$ and $VertexTo_i$.

The task project (TP). The propose of an application program is to solving user tasks. Tasks could be divided into subtasks. Subtasks are steps for solving the main task. There are some relations among tasks. They determine conditions and the order of task performance [21]. TP=(T, $\sigma$), where T is a task tree, $\sigma$ is a labeling tree. The task tree T=<Vertices, Edges, RootVertex>. The tree labeling $\sigma$ is a transformation of the vertex set to a name set $\Omega$, where $\Omega$ =$\{TaskName_i\}_{i=1}^{taskcount} \cup$ SetType. The name set $\Omega$ consists of task names and set names. SetType ={«choice», «interleaving», «enabling», «deactivation»}. Labeling $\sigma$ is defined as RootVertex$\rightarrow$ CommonTaskName, the root vertex transforms to the common task name. Every vertex $Vertex_i \rightarrow$ ($Label_{1i}$,$Label_{2i}$), where $Label_{1i} \in \{TaskName_i\}_{i=1}^{taskcount}$, $Label_{2i} \in$SetType (semantics of sets is defined in [21]).

The presentation project describes the structure and parameters of WIMP-interfaces. This component is specified by the WIMP-presentation ontology and is defined as a set of windows: Windows=$\{Window_i\}_{i=1}^{windowcount}$, where $Window_i \in$ Controls | Controltype = ContainerWindow. According to the WIMP-presentation ontology every interface element $Control_i$ is defined by a type, sets of parameters, functions, and events. The type, functions and events of an interface element have been defined in the ontology so the presentation project describes values of interface parameters for the interface project. Other interface elements could also be parameters of a window $Window_i$. In this case $Param\_{Type_k}$ = $Control_m$, where $Control_m \in$Controls.

The application program project describes a method of interaction between the user interface and the application program, and program interfaces to provide this interaction. This component is specified by the corresponding ontology and is defined as a set of Interfaces=$\{Interface_i\}_{i=1}^{interfacecount}$. Every element of the set consists of the program interface definition assigned by the application program, $Interface_i$ = < $Interfacename_i$, $InteractionModel_i$, $Functions_i$>, where $Interfacename_i$– a program interface name. $InteractionModel_i$ is an element from the set IModels={Local, Distributed}; a function set Functions describes program interfaces assigned by the application program.

**The scenario dialog project** describes a first window StartWindow, specifying the beginning of dialog with users, StartWindow=$Window_i$, where $Window_i \in$ Windows, and so a set of possible conditions of dialog States $= \{State_i\}_{i=1}^{statecount}$. A state $State_i$ consists of a definition of the event $Event_i$, the set of variables $Variables_i$ for definition of a instruction sequence called $Instructions_i$ and the instruction sequence $Instructions_i = \langle Instruction_{ij} \rangle_{j=1}^{instructioncount}$. The instructions can be:

- program interface calls of the application program;
- system function calls;
- compound elements (“if” and “cycle” constructions) consisting of a sequence of the mentioned above elements.

The scenario dialog project is specified by the scenario dialog ontology.

**The mapping project** describes a mapping set among different elements of the interface project: elements of the domain and the task projects are mapped on parameters of interface elements in the presentation project, parameters of interface elements on output parameters of the application program in the scenario dialog project, and so on.

## Methods for specifying the presentation project

Interface element parameters of the presentation project are specified by the domain and the task projects, that is for every domain term (groups of terms) and every task (tasks) the interface developer chooses its (their) presentation in the interface, a set of interface elements with specified properties (Fig. 4).

The following problems arise in this case:

- Time-consuming development of presentation projectы for large size of the domain or the task projects;

- Labour-consuming maintenance of the interface project (modification of the domain project requires modification of the presentation project);

- High-level professional requirements to developers (they must know usability principals, requirements of standards, users, a domain for information presentation, and so on).

To meet the main requirement to the interface project, namely, easy modification and maintenance, some methods for specifying the presentation project are suggested. The choice of one of them is determined by the developer and depends on features of components of the interface project (for example the size of the presentation or task project, the domain features specifying type of information, and so on).

**The method for direct specifying the presentation project.** In this case values of interface element parameters are direct references (Fig. 4) either to domain project elements or task project elements, that is, for an interface element of the presentation project $Control_i \in$Windows, where Param_Type=String, Param_Value=Value, Value$\in$ $\{TaskName_i\}_{i=1}^{taskcount} \cup$ DomainName $\cup$ $\{GroupName_i\}_{i=1}^{termgroupcount} \cup$ $\{TermName_j\}_{j=0}^{termcount} \cup \{AttributeName_i\}_{i=1}^{attributecount} \cup \{QualityValue_i\}_{i=2}^{valuecount}$. For example, in Fig. 4 parameter values “Text” of the list box elements are references on attribute values “localization” from the domain project. The number of list box elements is determined by the number of “localization” attribute values; modification of the number of this attribute values results in modification of the number of list box elements; modification of themselves values of this attribute in the domain project results in modification of the “Text” parameter.

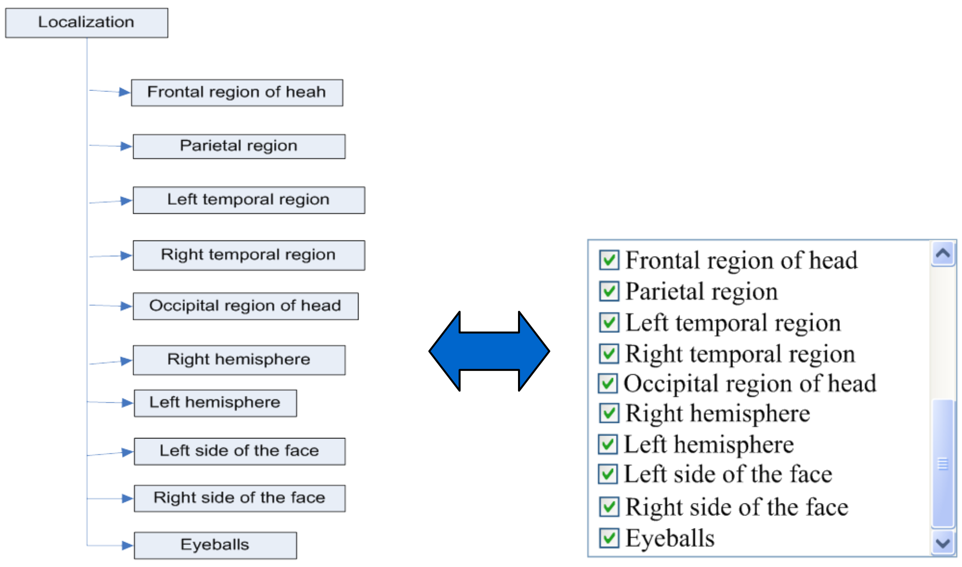


Fig. 4 The example of direct specifying of the presentation project

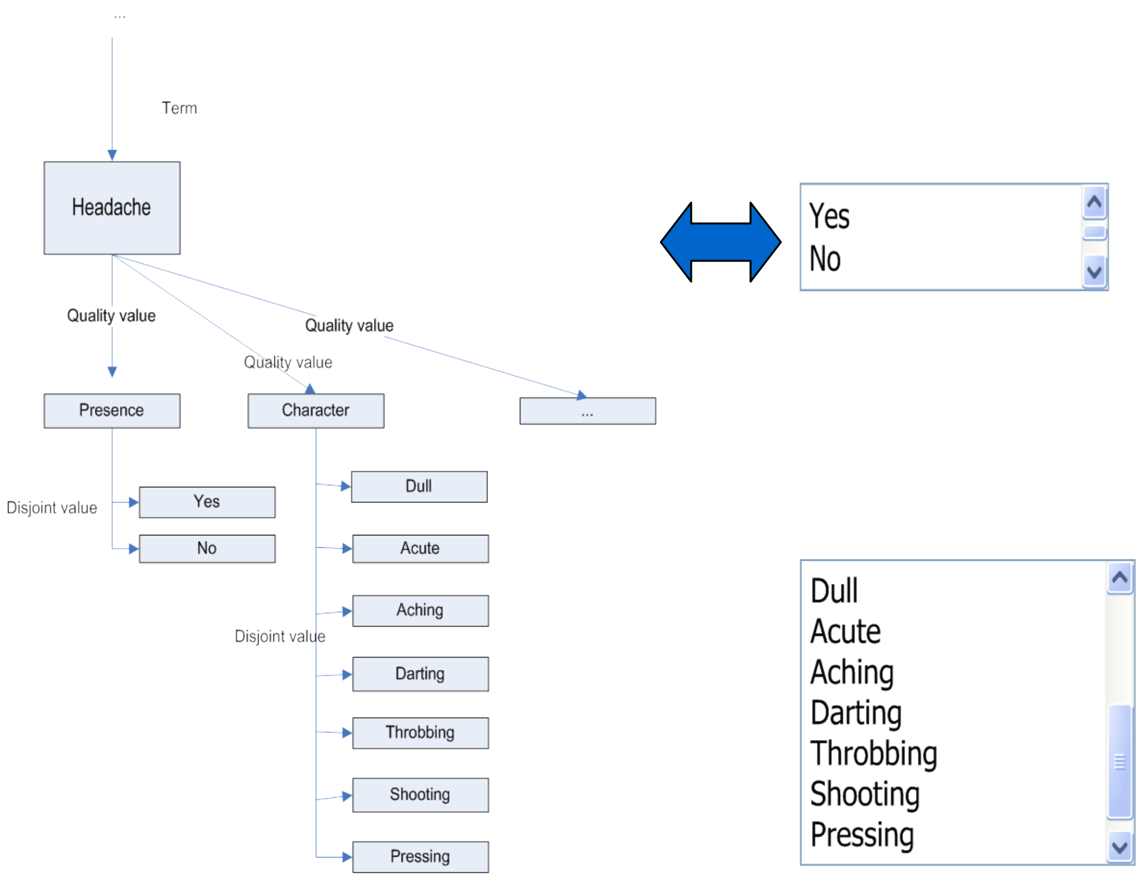


Fig. 5 The example of regular specifying of the presentation project

This method of specifying of the presentation project is convenient for small size of the domain or the task projects and so if accurate presentation domain terms in an interface is required (for example if the user has special requirements).

**The method of regular specifying of the presentation project.** In this case values of interface elements parameters are references on components the domain ontology (but no on components of the domain project): groups of terms, terms, values, attributes. That is for an interface element of the interface project $\mathrm{Control}_i \to$ $\mathrm{Element} \mid \mathrm{Control}i \in \mathrm{Windows}, \mathrm{Element} \in \{\mathrm{TermGroups, Terms, Attributtes, QualiyValues}\}\}_{i=1}^{controlcount}$.

As a result every term group (terms, values, attributes) included in the domain project is represented by an interface element which the developer choses. In this case the value of a string parameter(s) is (are) elements of the domain project.

For example, the parameter value "Elements" = $\{\mathrm{ListBoxElement}_i\}_{i=1}^{elemcount}$ of list box elements are corresponded with all quality joint values from the domain ontology. It means that all quality joint values from the domain project are presented by the same interface elements (Fig.5). At the same time their quantity could be arbitrary large and is determined by the domain.

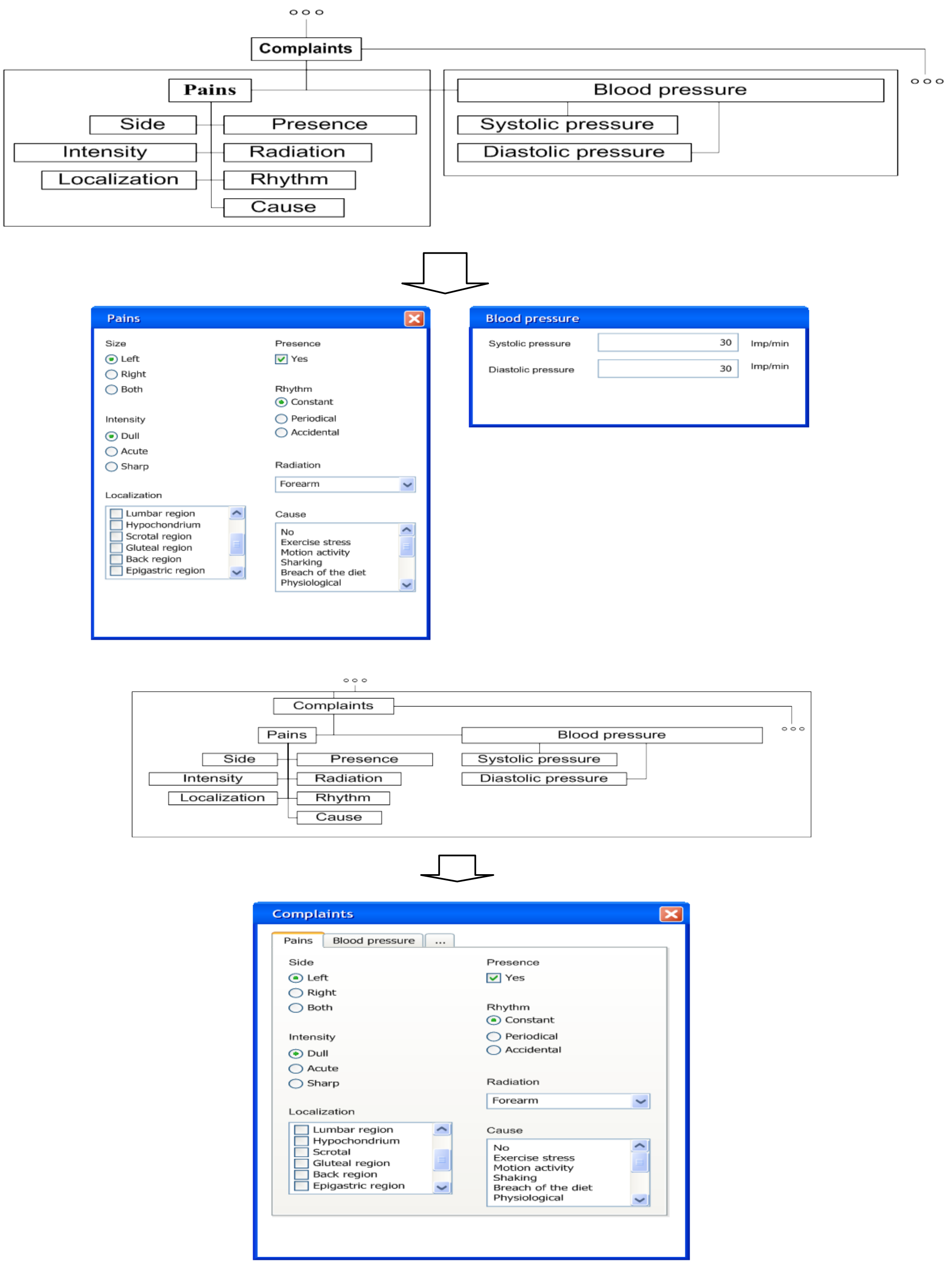
This method of specifying of the presentation project is convenient for large size of the domain or the task projects (hundreds and thousands of terms).

**The method of fragmentary specifying of the presentation project.** In this case the developer divides the presentation project on a set of fragments based on user requirements, functionality and semantic of the application program; every fragment is corresponded with their representation in the interface: $\{\mathrm{Fragment\_Presentastion} \to \mathrm{Fragment\_Terms} \mid \mathrm{Fragment\_Presentastion} \in \{\mathrm{Control}_i\}_{i=1}^{controlcount}, \mathrm{Control}_i \in$ $\mathrm{Windows}, \mathrm{Fragment\_Terms} \subset G\}_{i=1}^{fraggmentcount}$.

The basic statements of this method are:

- Every fragment of the domain project is automatically corresponded with a set of possible representations.
- Corresponding are based on usability principals.
- If a fragment has some possible representations in according to usability principals then the developer has the case:
    - He or she choose the only representation from the set of possible representations;
    - The representation is automatically chosen in according to default rules.
- The explanation containing the protocol of executing is given to the developer. The protocol consists of the set of used criteria with description of accepted and unaccepted criteria (criteria in chosen in according to usability principals).
- A set of criteria must be expanded (because usability principals are grown and modified).

To provide the expanding set of possible representations of a presentation mapping ontology is suggested. In the terms of the ontology rules of corresponding domain terms with their possible representations (in according to usability principles) is determined. In this way the set of mappings (the knowledge base) is formed. A mapping is described as a pair – a mapping condition and a set of possible representations. The condition could be consisted of a set of conditions, every of them determines parameter values. Conditions are verified by the domain ontology. Parameters of conditions are determined in the terms of the domain ontology. For example, a condition could be quantity of joint values of a term from the domain project and a method of their selection. (joint or disjoint). Based on various values of these conditions a set of representations are chosen: a set of radio-buttons, check boxes, a list box, a combo box, etc.). Two various fragmentations for the same domain project fragment and corresponding to it's the presentation project fragment are presented at Fig. 6.



Pains

Size
- [✓ Left]
- Right
- Both

Intensity
- [✓ Dull]
- Acute
- Sharp

Localization
- Lumbar region
- Hypochondrium
- Scrotal region
- Gluteal region
- Back region
- Epigastric region

Presence
- [✓ Yes]

Rhythm
- [✓ Constant]
- Periodical
- Accidental

Radiation: Forearm

Cause
- No
- Exercise stress
- Motion activity
- Sharking
- Breach of the diet
- Physiological

Blood pressure

| | | |
|---|---|---|
| Systolic pressure | 30 | Imp/min |
| Diastolic pressure | 30 | Imp/min |

Complaints

Pains | Blood pressure | ...

Side
- [✓ Left]
- Right
- Both

Intensity
- [✓ Dull]
- Acute
- Sharp

Localization
- Lumbar region
- Hypochondrium
- Scrotal
- Gluteal region
- Back region
- Epigastric region

Presence
- [✓ Yes]

Rhythm
- [✓ Constant]
- Periodical
- Accidental

Radiation: Forearm

Cause
- No
- Exercise stress
- Motion activity
- Shaking
- Breach of the diet
- Physiological

Fig. 6 The example of fragmentary specifying of the presentation project

As you see from the Fig. 6 different fragmentations of the domain project lead to different presentation projects. In the first case the fragment has two fragmentations. Every fragmentation is corresponded with the window ("Pain" and "Raising blood pressure"). In the second case the fragment has the only fragmentation corresponded with the window "Complaints".

This method of specifying of the presentation project is convenient (as the previous method) for large size of the domain or the task projects (hundreds and thousands of terms).

## Discussion of results

At present the methods described above have been implemented and integrated in a tool for development, automatic generation and maintenance of user interfaces based on the ontology-based approach. These methods have been used for implementation and maintenance interfaces for a number of application programs.

In discussing results obtained it is reasonable to compare them with Model-based interface development environments (MB-IDE) for development and automatic generation of user interfaces [22-24]. The other class of specialized tools – Interface Builders does not realize such methods.

The method for direct specifying the presentation project is implemented in some MB-IDE in the following way: string parameter values of interface elements are domain terms or users' tasks. However, any modification of domain terms or users' tasks constrains to modify string parameter values of interface elements also, and then to recompile the interface. This results in laborious interface maintenance. In the suggested method string parameter values of interface elements are references to domain terms or users' tasks, so modification of domain terms or users' tasks result in automatic modification of the string parameter values of interface elements and no recompilation is necessary.

The method for regular specifying the presentation project is suggested for the first time. It allows the developer to considerably decrease laboriousness of development and maintenance. For example, the domain project of the System for intellectual support of medical activity for urologists [25] consists of 700 terms and groups of terms and about 5000 possible values. In this case the method of regular specifying the presentation project provides fast specification and automatic maintenance of the presentation project.

The method for fragmentary specifying the presentation project has been used in some MB-IDE. With this method developers determine a window (a set of windows) and information for every window (from the domain and the task projects). Then a presentation for this information is chosen automatically and the developer only enhances the presentation. On the one hand, such an approach decreases the time of presentation project development; on the other hand, the main problem for developers is difficult maintenance of the project, because modification of terms and tasks result in automatic representation of these information and all enhancements are lose. Attempts to save results of enhancements were realized in some MB-IDEs. Nevertheless, they failed in case of addition or deletion of new terms or tasks. Using references solves this problem. Another drawback of this method in some MB-IDE is strict embedded of mappings (a set of presentations) to the tool. As a result, the developer is not aware of these mappings and they could not be expanded. According the method suggested all mappings are in the knowledge base and could be viewed and modified.

## Bibliography

1. Myers Brad, Rosson Mary. Survey on user interface programming // Tech. Rpt. CMU-CS-92-113, Carnegie-Mellon, School of Comp. Sci., February 1992. http://citeseer.ist.psu.edu/myers92survey.html
2. Baar de D., Foley J.D., Mullet K.E. Coupling Application design and User interface design // CHI'92 Conference Proceedings, Monterey, ACM Press, 1992, pp.259-266
3. Apple Human Interface Guidelines – Electronic resource – http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/index.html

4. Windows Vista User Experience Guidelines – Electronic resource – http://msdn2.microsoft.com/en-us/library/aa511258.aspx
5. SAP - R/3 Style Guide – Electronic resource – http://www.sapdesignguild.org/resources/MiniSG/index.htm
6. Common Desktop Environment: Style Guide and Certification Checklist - Electronic resource – http://docs.sun.com/app/docs/doc/802-6490?q=Motif+Style+Guide
7. Eisenstein J., Puerta A. Adaptation in Automated User-Interface Design // Intelligent User Interfaces, 2000, pp. 74-81
8. Myers B.A. User Interface Software Tools // ACM Transactions on CHI'95. Vol. 2, No. 1, 1995, pp 64-103
9. Puerta A.R., Eriksson H., Gennari J.H., Musen M.A. Beyond Data Models for Automated user Interface Generation // British HCI'94 Confirence Proceding, Glasgow, Cambridge University Press, 1994, pp.353-366
10. Szekely P. Retrospective and Challenges for Model-Based Interface. 1996. http://citeseer.nj.nec.com/szekely96retrospective.html
11. .Lonczewski, F., Schreiber, S.: The FUSE-System: an Integrated User Interface Design Environment. In: Vanderdonckt J. (ed.): Proceedings of CADUI'96. Namur: Presses Universitaires de Namur 1996 (pp. 37-56).
12. Luo, P., Szekely, P., Neches, R.: Management of Interface Design in Humanoid. In Ashlund S., Mullet K., Henderson A., Hollnagel E., White T. (eds.): Proceedings of INTERCHI'93. New York: ACM Press 1993 (pp. 107-114)
13. 48 Wiecha, C., Bennett, W., Boies, S., Gould, J., Green, S.: ITS: A Tool for Rapidly Developing Interactive Applications. ACM Transactions on Information Systems, Vol. 8, No. 3, 1990, p.204-236
14. Wiecha, C., Bennett, W.: Generating Highly Interactive User Interfaces. In Bice K., Lewis C. (eds.):.Proceedings of CHI'89. New York: ACM Press 1989 pp. 277-282.
15. Szekely P., Sukaviriya P., Castells P., Muthukumarasamy J., Salcher E. Declarative interface models for user interface construction tools: the Mastermind approach. In Bass L., Unger C. (eds.) // Engineering for Human-Computer Interaction, Proc. of EHCI'95. London, Chapman & Hall, 1995. P. 120-150.
16. Puerta, A.: The Mecano Project: Comprehensive and Integrated Support for Model-Based Interface Development. In: Vanderdonckt J. (ed.): Proceedings of CADUI'96. Namur: Presses Universitaires de Namur 1996 (pp. 19-36)
17. Szekely, P., Luo, P., Neches, R: Facilitating the Exploration of Interface Design Alternatives: The Humanoid Model of Interface Design. In Bauersfeld P., Bennett J., Lynch G. (eds.): Proceedings of CHI'92. New York: ACM Press 1992 (pp. 507-514).
18. Szekely, P., Luo, P., Neches, R.: Beyond Interface Builders: Model-Based Interface Tools. In Ashlund S., Mullet K., Henderson A., Hollnagel E., White T. (eds.): Proceedings of INTERCHI'93. New York: ACM Press 1993 (pp. 383-390).
19. Gribova V, Kleshchev A. Control of user interface development and implementation based on ontologies // Control Sciences, 2006. №2. P.58-62.
20. Gribova V., Kleshchev A. From an ontology-oriented approach conception to user interface development International //Journal Information theories & applications. 2003. vol. 10, num.1, p. 87-94
21. Gribova V. Automatic generation of context-sensitive help using a user interface project // Proc. of XШth Intern. Conf. "Knowledge-dialog-solution" – Varna, 2007. V.2. P. 417-422.
22. Puerta A. A model–based interface development environment IEEE Software, 14(1), July/August 1997. P. 41–47.
23. Puerta A.R. Issues in Automatic Generation of User Interfaces in Model-Based Systems. Computer-Aided Design of User Interfaces, ed. by Jean Vanderdonckt. Presses Universitaires de Namur, Namur, Belgium, 1996. P. 323-325.
24. Szekely P. Retrospective and Challenges for Model-Based Interface. 1996. http://citeseer.nj.nec.com/szekely96retrospective.html
25. Gribova V, Tarasov A, Chernyakhovskaya M. A system for intellectual support of patient examination controlled by the // Software & systems, 2007. №2. P. 49-51

## Author information

***Gribova Valeriya**, Prof., Head of the Intelligent System Laboratory, Institute of Automation & Control Processes, Far Eastern Branch of the Russian Academy of the Sciences: Vladivostok, +7 (4323) 314001 gribova@iacp.dvo.ru, http://www.iacp.dvo.ru/is.*

# INVESTIGATION ON COMPRESSION METHODS USED AS A PROTECTION INSTRUMENT OF FILE OBJECTS

Dimitrina Polimirova, Eugene Nickolov

*Abstract: This report examines important issues related to different ways to influence the information security of file objects, subjected to information attacks by the methods of compression. Accordingly, the report analyzes the relationships that may exist between selected set of known by the moment of exploration attacks, methods and objects.*

*A methodology for evaluation of information security of objects exposed to attacks is proposed. This methodology reports the impact of different methods of compression which can be applied to these objects. A coefficient of information security for each relation attack—method—object is created. It depends on two main parameters TIME and SIZE, which describe, respectively, the attack and the object. The parameters are presented as two separate relations variables before and after the impact of methods of compression.*

*Since one object can be processed by more than one method of compression, different criteria and methods are used for evaluating and selecting the best method of compression with respect to the information security of the investigated objects. An analysis of the obtained results is made. On this basis are made recommendations for choosing methods of compression with the lowest risk with respect to the information security of file objects, subjected to information attacks.*

*Keywords: Information Security, Information Attacks, Methods of Compression, File Objects, Co-efficient of Information Security, Risk Assessment.*

*ACM Classification Keywords: D.4.6 Security and Protection: information flow controls*

## 1. Introduction

The development of information systems and technologies are increasingly expanding need for processing, transferring and saving of volume sizable information flows, which are in network TCP/IP environment. These information flows in the form of file objects, are subject of non-stop attacks according to their information security, which determines the significant necessity for research of methods and means for their protection.

A common strategy for the protection of the file objects could include applying of methods of compression to objects to achieve decrease in the size of information flow. In addition, the use of password with fixed minimum and maximum length can be used. The possibility for encryption of objects, which are preliminarily compressed and protected by password, can be included as the last stage of the strategy for protecting.

Within the framework of mentioned above possibilities for investigations can be estimated that is expediently to investigate this problem on separate stages.

For the purposes of this paper the following reservation can be made: it is enough to investigate only the influence of compression methods on objects exposed to one or more attacks, as the difference in their behavior before and after the attacks when standard and not corporate (government) requirements are used, is taken into consideration.

Since the 70-th of XX century the problem for security and protection of information flows has drawn developers' and constructors' attention in the area if information technology [5]. With the first malattack in the 60th of last century [39], a progress in the area of object protection is observed and requirements for information security of objects are increased. Later the problem for creation of maximum protection for information flows arisen.

This cause the necessity to conduct targeted research to clarify and successfully solving various tasks related to improving security in the processes of transferring, processing and storing of different types of information flows. Research on methods to enhance the information security of different types of file objects, became topical.

Information flows subjected to different attacks, are characterized with their large volume size. Different methods for compression were developed and their use became necessity in order to reduce their size. Compressing objects, however, may be used as a means of increasing their security.

## 2. The Problem

### 2.1. Actuality

The aim that can be placed within this paper is related to the investigation of publicly known by the moment of exploration information attacks, methods of compression and file objects.

The main hypothesis is linked with the ability to analyze and evaluate the effectiveness of methods of compression, applied as e means to protect these objects from attacks.

The approach for achieving the aim in accordance with the basic hypothesis is to use matrix transformations, applied on initially created base of relations between attacks, methods and objects.

Future analyses and investigations can be carried out in the direction of precise planning of the economic costs when customizing the security policy of different configurations of computers, systems and networks.

Investigation can be carried out also in cases of government computers, systems and networks, where encryption has a significant impact on information flow, especially when this process is applied to the already compressed objects.

### 2.2. The goal and main task

The investigations, which can be planned, have to be related to the analysis of the condition and perspective for development of known information attacks, methods of compression and file objects. Their scientific generalization in the form of three-way relation is necessary because only in their mutual relation the best analysis and thus the best decisions with respect to the effectiveness of methods of compression applied as a means to protection the file object from information attacks, can be achieved.

Main goal: research and analyze the change in the information security of file objects located in a TCP/IP environment, subject to information attacks, recording the impact of methods of compression.

Therefore, the main tasks resulting from the above defined purpose are as follows:

1) to propose a method for reducing *maximum* three-way relationships among certain attacks, methods and objects to *real* three-way relationships that can exist among them;
2) to propose a methodology for evaluation of information security of an object under attacks by recording the effect of the applied method of compression. Furthermore, using this methodology, the task is to define the methods of compression achieving the highest values of the coefficient of information security for each object for a respective attack, and for all objects for respective attacks;
3) to propose a procedure for selecting methods of compression with the lowest risk with regards to the coefficient of information security of the respective objects for all attacks;

4) to conduct experiments proving the accuracy of the approach that entails the use of matrix transformations applied to an initially created base of two-way relationships among attacks, methods and objects which are to be transformed into attack-method-object three-way relationships.

## 2.3. Work definitions

For the aim of this paper the following work definitions are proposed [6], [26],[35]: 1) as information security we will note the protection of the information in an object from a random or purposeful access aimed at reading, transferring (coping), modifying or destroying the information in it; 2) as file object we will note the whole interconnected data or program records, saved under one name (http://www.answers.com/file); 3) as information attack we will note an attack in connection with the content of the current information stream; 4) as method of compression we will note the procedure for data encoding aimed at shrinking their volume during the processes of transfer and storage; 5) as a data compression we will note transforming of input data into output codes. The decision for the correspondence *input data—output codes* is based on preliminarily selected model. In case of effective compression, the obtained flow of codes is smaller in volume than the input data, but even though the compression is not effective, the file object will have a better protection against different attacks, because the output data will be presented by codes.

During the investigation of the information security of file objects, they will be exposed to different information attacks. The attacks have been provisionally divided into malware and malattacks. In case of malware the direct participation of a user at the moment of the attack is missing, while in case of malattack the user's presence is required [30], [22].

When investigating the methods of compression, applied not only for reducing the object's size, but also for protecting them from information attacks, they will be divided in two main categories: lossless methods of compression and lossy methods of compression. Lossy methods of compression achieve better results with respect to the level of compression, but part of the information is removed [17]. In the group of lossless methods of compression are included those methods, which can guarantee the absolute repetition of output data with input data during the process *compression—decompression* [12].

Methods of compression can be applied over file objects represented by different file formats. Over 23000 file formats are known by the moment of exploration [37]. As file format we will note the way used for presenting the file information [23]. Different file formats for the different type of information, saved in file exist. The file objects are divided in two main categories: directly executable and indirectly executable. The directly executable objects can be used directly while indirectly executable objects need to be processed additionally to become directly usable.

## 2.4. Short review of attacks, methods and objects

Information flows have been subject to various information attacks, and that has attracted the attention of scientists starting all the back in the 60s during the previous century [39]. Chris Rodgers has explored computer and network attacks in a TCP/IP environment featuring viruses, worms, Trojan horses and DoS attacks [25]. Daniel Klein has researched one of the most frequently used attacks for accessing systems or file objects – an attack through a password [16]. Marco de Vivo and David have examined the effect of various network attacks [19], [36]. In 2004, attacks on mobile phone become extremely popular, and those have been researched by Martin and Hsiao [20]. World organizations like CERT/CC, SANS and OIS research and analyze attacks and regularly publish related reports and bulletins.

Another group of scientists have been trying to find ways to reduce the size of the file objects by designing different methods of compression. For example, Cokus and Winkowski use methods of compression applicable to XML objects [3]. Butner, Iddan, Meron work on compressing images used in medicine and wireless telecommunications which also have a higher rate of compression [2], [11]. Gilbert and Haffner have conducted studies in the field of compression of complex images and video both with and without a loss of information [8], [9].

## 3. Method for Reducing the Maximum Triple Relations Between Attacks, Methods and Objects to Real Triple Relations Between Them

### 3.1. Maximum triple relations between attacks, methods and objects

For achieving the tasks of the paper, the set of *maximum* number of attacks, methods and objects has to be determined.

The set of *maximum* number of attacks can be collected from the current information base of National Laboratory of Computer Virology of Bulgarian Academic of Sciences. It collects information for the information attacks, which were carried out to a separate personal and/or corporate computers, and/or networks, and/or systems at the moment of the investigation. This is a generalization of attacks, implemented in Bulgaria, Balkan Peninsula and south-east Europe.

Known methods of compression including lossless and lossy methods of compression, will be described.

The objects, included in the set of *maximum* number of objects are representative of different file formats. They belong to two main categories – directly executable and indirectly executable.

In the set of maximum number of attacks ($A_{max}$) are included 89 different attacks [38], divided in 33 main groups. 20 of these are in the category “Malicious software (Malware)” and 13 – in the category “Malicious attack (Malattack)”.

59 methods of compression take part in the set of *maximum* number of methods of compression ($M_{max}$). They are divided in 9 groups: 5 of them belong to the category “Lossy methods of compression” [26] and 4 – to the category “Lossless methods of compression” [28].

42 file objects, representing over 23000 file formats are organized in 10 main groups. 7 of these belong to the category “Directly executable” and 3 – to the category “Indirectly executable”. They form the set of *maximum* number of objects $O_{max}$.

For the purposes of the investigation, the current attacks will be denoted as $a_i$, where the index $i$ changes from 1 to $n$ (maximum number of known attacks), the current method of compression will be denoted as $m_j$, where the index $j$ changes from 1 to $k$ (maximum number of methods), and the current object will be denoted as $o_f$, where the index $f$ changes from 1 to $l$ (maximum number of objects).

### 3.2. Real triple relations between attacks, methods and objects

To turn out the unreal relations from the determined *maximum* sets, attacks, methods and objects have to be singled out by reducing. They will form the sets of *potential* attacks, methods and objects.

A group of experiments are carried out to determine the sets of *potential* numbers of attacks, methods and objects. The experiments have two stages: determination the *sets* of possible relations and determination the sets of *real* relations.

The first stage investigates the sets of *possible* relations which can exist between attack—object, method—object and attack—method. For each relation pair will be composed two matrixes. The first matrix will include the result of *expert* assessment for the corresponding relation. The second one will include the results from carried out *experiments* for the same relation.

The *expert* assessment will provide the possibility to exclude from the analysis the attacks, methods and objects about which: 1) there is no sufficient information; 2) the information is not public; 3) the information is rapidly changing; 4) there is not enough authentic hardware and software.

The expertly determined sets of relations will be put to a partial test by means of a number of planned simulative experiments. Thus the set of *possible* attacks, methods and objects will be singled out.

## Determination of *possible* relations attack—object ($\Omega$)

Let $O_t$ be the set of file objects and $A_t$ is a set of attacks that in a discrete moment of time *t* can gain access to the objects *O*. The elements of the set $O_t$ and $A_t$ are the apex of oriented graph $G_t$, determining the ability to access $P$ to the object *O* in this way: the arc $A \xrightarrow{\rho} O$, where$\rho \subseteq P$ (as $\rho$ will denote the different type of access: read, write, execute and delete), belongs to $G_t$ then and then only when at the moment *t* the attack $a_i \in A_{max}$ accomplishes an access $\rho$ to the object $o_f \in O_{max}$.

Let $\{G_t\}_{t=1}^{T}$ denote the set of states of the object from the point of view of the possibility for the attack *A* to obtain access to the object. $\Psi$={G} is the set of graphs for access of *A* to *O*. In the common case the sets of states in the relation attack—object ($\Phi$), describing the successful/unsuccessful access of *A* to *O*, belong to the set $\Psi$.

The next stage is to determine the set $\Omega \subseteq \Phi \subseteq \Psi$, which includes those conditions, where the access of the attack $a_i \in A_{max}$ to the object $o_f \in O_{max}$ is possible.

Two matrices are composed to determine the set $\Omega$: $Y_{(l,n)}$ and $E_{(l,n)}$. By the vertical of the matrices are included all attacks $A_{max}$, separated in *n* varieties $a_1, a_2, \dots, a_i, \dots, a_n$, $\bigcup_{i=1}^{n} a_i = A_{\max}$, $\bigcap_{i=1}^{n} a_i = \varnothing$. By the horizontal of the matrices are included all objects $O_{max}$, separated in *l* varieties $o_1, o_2, \dots, o_f, \dots, o_l$, $\bigcup_{f=1}^{l} o_f = O_{\max}$, $\bigcap_{f=1}^{l} o_f = \varnothing$. A research is conducted, where an attack $a_i$ is trying to get access $\rho$ to the object $o_f$, where $\rho$ will present the different type of an access (read, write, execute and delete). Graphically this process can be illustrated in this way:

$$U_1 \longrightarrow a_i \xrightarrow{\rho} o_f$$

where $U_1$ is a user, who uses an attack $a_i \subseteq A_{max}$, to get access $\rho$ to the object $o_f \subseteq O_{max}$, *i=1,2,…,n*, *f=1,2,…,l*.

In the matrix $Y_{(l,n)}$ for each cell the function of truth is produced [4] on the base of expert assessments: $J_x(K)\begin{cases} 1, x \in K \\ 0, x \notin K \end{cases}$ for the attack and object: $J_a(a_1), J_a(a_2), \dots, J_a(a_i), \dots, J_a(a_n)$, $J_o(o_1), J_o(o_2), \dots, J_o(o_f), \dots, J_o(o_l)$. For each oriented graph $A \xrightarrow{\rho} O$ the result from the following logical expression $J_a(a_i) \wedge J_o(o_f)$ is filled out. If the obtained result is 1 (true) (*x=1*), then an attack $a_i \in A_{max}$ can get access to the object $o_f \in O_{max}$ ($A \xrightarrow{\rho} O$) and the relation can be investigated and analyzed. Otherwise with result 0 (false) (*x=0*), the attack $a_i \in A_{max}$ cannot get an access $\rho$ to the object $o_f \in O_{max}$.and the relation drops off for further investigation.

From the obtained results for the relation attack—object the set of *expert* relations (*E*) can be singled out, which includes all attacks and objects the following logical expression is realized for: $J_a(a_i) \wedge J_o(o_f) = 1$.

In the second matrix $E_{(l,n)}$ for each cell a logical processing is made with result logical 0 or logical 1 by the so called function of truth $I_x(B)\begin{cases}1, x \in B \\ 0, x \notin B\end{cases}$ respectively for the attack and object: *I*$_a$(*a*$_1$),*I*$_a$(*a*$_2$),...,*I*$_a$(*a*$_i$),..., *I*$_a$(*a*$_n$), *I*$_o$(*o*$_1$),*I*$_o$(*o*$_2$),...,*I*$_o$(*o*$_f$),...,*I*$_o$(*o*$_l$) and in the corresponding cell of the matrix is filled out the result from the logical expression: $I_a(a_i) \wedge I_o(o_f)$ on the of the experiments, which were carried out. If the obtained result is 1 (possible) (*x=1*) then during the experiments the attack $a_i \in A_{max}$ had gotten an access to the object $o_f \in O_{max}$ ($A \xrightarrow{\rho} O$), otherwise with result 0 (impossible) (*x=0*), the attack's access is not accomplished.

On the base of the obtained results the set of *experimental* relations (*B*) can be singled out, which include all attacks and objects, for which is completed the following condition: $I_a(a_i) \wedge I_o(o_f) = 1$.

Crossing the set *B* with *E* the set of possible relations attack—object (Ω) can be singled out, where $\Omega = \left[I_a(a_i) \wedge I_o(o_f)\right] \wedge \left[J_a(a_i) \wedge J_o(o_f)\right] = 1$.

From the set Φ will be picked out the set of *possible* relations attack—object Ω={Ω$_i$}, where *i* has one of the value from 1 to *T*, which covers those conditions of the object toward the attack, which will take a part in determination of the sets of *real* relations.

By analogy the other sets of possible relations method—object (Ξ) and attack—method (Θ) are determined. Figure 1, 2, 3, 4, 5, 6, 7, 8, and 9 are graphically presented the obtained result for the sets of *expert* relations (in red), the sets of *experimental* relations (in blue) and the sets of *possible* relations (in green).

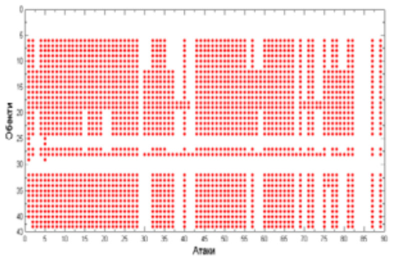


Figure 1 Set of *expert* relation attack—object (*E*)

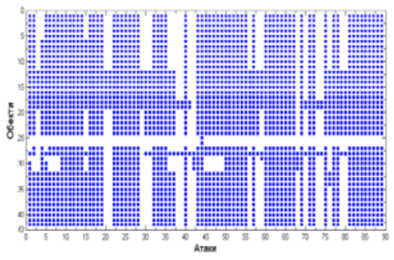


Figure 2 Set of *experimental* relation attack—object (*B*)

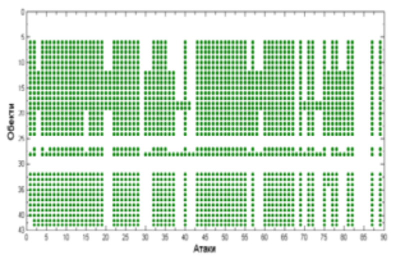


Figure 3 Set of *possible* relation attack—object (Ω)

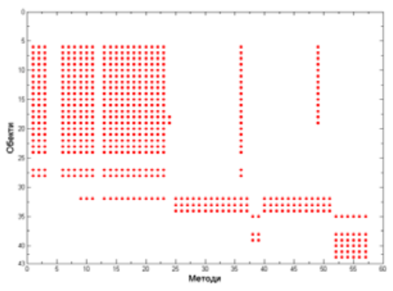


Figure 4 Set of *expert* relation method—object (*P*)

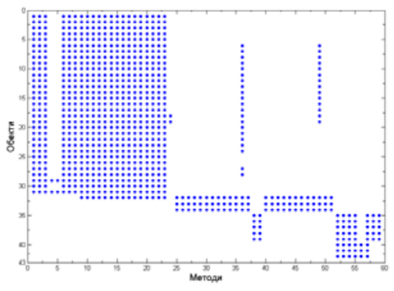


Figure 5 Set of *experimental* relation method—object (*C*)

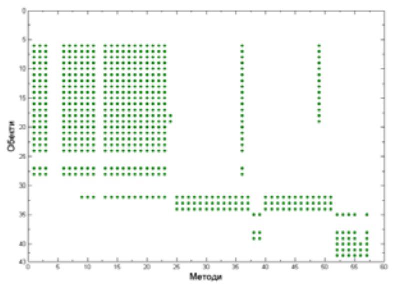


Figure 6 Set of *possible* relation method—object (Ξ)

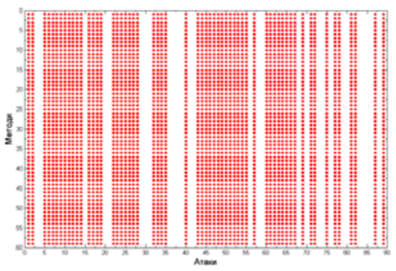


Figure 7 Set of *expert* relation attack—method (*Y*)

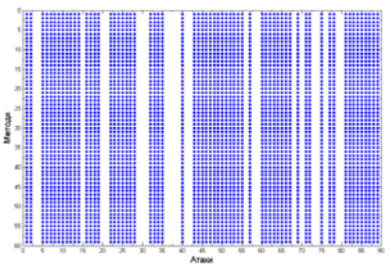


Figure 8 Set of *experimental* relation attack—method (*D*)

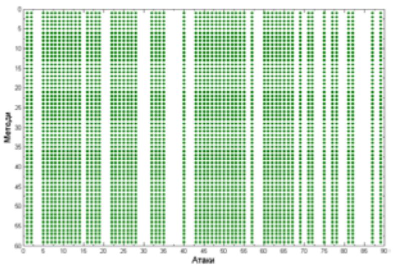


Figure 9 Set of *possible* relation attack—method (Θ)

The next stage of reducing the attacks, methods and objects includes investigation of the relations Ω—Ξ, Ω—Θ и Θ—Ξ.

Let the elements of the set Ω and Ξ form a totality of elements $\alpha \subseteq \mathrm{A}$ then and then only, when for the elements $\omega_z \subseteq \Omega$ and $\xi_h \subseteq \Xi$ the following logical expression is realized for (Expression 1):

$$\mathrm{A} = \left\{\left[I_a(a_i) \wedge I_o(o_f)\right] \wedge \left[J_a(a_i) \wedge J_o(o_f)\right]\right\} \wedge \left\{\left[I_m(m_j) \wedge I_o(o_f)\right] \wedge \left[J_m(m_j)\right] \wedge J_o(o_f)\right\} = \\ = \Omega \wedge \Xi = 1 \qquad \text{Expression 1}$$

The elements of the set Ω and Θ form a totality of elements $\beta \subseteq \mathrm{B}$ then and then only, when for the elements $\omega_z \subseteq \Omega$ and $\theta_c \subseteq \Theta$ the following logical expression is realized for (Expression 2):

$$\mathrm{B} = \left\{\left[I_a(a_i) \wedge I_o(o_f)\right] \wedge \left[J_a(a_i) \wedge J_o(o_f)\right]\right\} \wedge \left\{\left[I_a(a_i) \wedge I_m(m_j)\right] \wedge \left[J_a(a_i)\right] \wedge J_m(m_j)\right\} = \\ = \Omega \wedge \Theta = 1 \qquad \text{Expression 2}$$

The elements of the set Ξ and Θ form a totality of elements $\gamma \subseteq \Gamma$ then and then only, when for the elements $\xi_h \subseteq \Xi$ and $\theta_c \subseteq \Theta$ the following logical expression is realized for (Expression 3):

$$\Gamma = \left\{\left[I_m(m_j) \wedge I_o(o_f)\right] \wedge \left[J_m(m_j) \wedge J_o(o_f)\right]\right\} \wedge \left\{\left[I_a(a_i) \wedge I_m(m_j)\right] \wedge \left[J_a(a_i)\right] \wedge J_m(m_j)\right\} = \\ = \Xi \wedge \Theta = 1 \qquad \text{Expression 3}$$

The set of *real* relations between attacks, methods and objects is $\mathrm{X} = \mathrm{A} \cap \mathrm{B} \cap \Gamma$ or (Expression 4):

$$\mathrm{X} = (\Omega \wedge \Xi) \wedge (\Omega \wedge \Theta) \wedge (\Xi \wedge \Theta) = \mathrm{A} \wedge \mathrm{B} \wedge \Gamma = 1 \qquad \text{Expression 4}$$

Using the Matlab™ 33811 *real* three-way relations attacks—methods—objects from total 220 542 *maximum* three-way relations are determined.

The set of *potential* number of attacks ($A_{pot}$) is expressed as <u>real</u> attacks (received from the *real* relations) in relation to the <u>maximum</u> number of attacks ($A_{max}$). The real attacks are <u>expert</u> and <u>experimental</u> estimated for the corresponding object/objects with respect to the corresponding method/methods and described with the technique of matrix transformation. The set of *potential* number of attacks $A_{pot}$ can be denoted as $A_{pot} = \{a_1, a_2, ..., a_i, ..., a_p\}$, where *p* is the index the <u>potential</u> attacks alter for, where *p*≤*n* (Expression 5).

$$A_{pot} = \bigcup_{i=1}^{p} a_i, \qquad a_i \subseteq \mathrm{X} \qquad \text{Expression 5}$$

The set of *potential* number of methods ($M_{pot}$) is expressed as <u>real</u> methods (received from the *real* relations) in relation to the <u>maximum</u> number of methods ($M_{max}$). The real methods are <u>expert</u> and <u>experimental</u> estimated for the corresponding object/objects with respect to the corresponding attack/attacks and described with the technique of matrix transformation. The set of *potential* number of methods $M_{pot}$ can be denoted as $M_{pot} = \{m_1, m_2, ..., m_j, ..., m_q\}$ , where *q* is the index the potential methods alter for, where *q*≤*k* (Expression 6).

$$M_{pot} = \bigcup_{j=1}^{q} m_j, \quad m_j \subseteq \mathrm{X} \qquad \text{Expression 6}$$

The set of *potential* number of objects ($O_{pot}$) is expressed as real objects (received from the *real* relations) in relation to the maximum number of objects ($O_{max}$). The real methods are expert and experimental estimated for the corresponding attack/attacks with respect to the corresponding method/methods and described with the technique of matrix transformation. The set of *potential* number of objects $O_{pot}$ can be denoted as $O_{pot} = \{o_1, o_2, ..., o_f, ..., o_r\}$, where *r* is the index the potential objects alter for, where *r*≤*l* (Expression 7).

$$O_{pot} = \bigcup_{f=1}^{r} o_f, \quad o_f \subseteq \mathrm{X} \qquad \text{Expression 7}$$

## 4. Methodic for Evaluation of the Information Security of an Object, Exposed to Attacks with Considering the Influence of Methods of Compression

### 4.1. Information security and its evaluation about file object

The methodology for evaluation of the information security of an object will meet the following limitations:

- only the potential sets of attacks, methods and objects will be analyzed;
- the experiments are conducted at standard users', non-corporations' (governments') requirements;
- in order to simplify the computations the lossy methods of compression are except;
- in conducting the experiments for determining the co-efficient of information security, the objects used have equal or similar starting size (1 MB).

Studies and analysis can be made in the following three directions:

- evaluation of the *success of the attack*, made on an object processed with a method of compression;
- evaluation of the *protection by method of compression*, applied on an object, exposed to an attack;
- evaluation of the *security of an object* exposed to an attack and processed by a method of compression.

From the mentioned above three directions in this paper will pay attention only to the evaluation of the security (information security) of objects, exposed to information attacks noting the influence of the methods of compression.

The information security of an object can be determined as a quantitative value, which depends on several fundamental parameters. For the purposes of this paper only the parameters *TIME* and *SIZE* will be studied and analyzed by marking the difference in the objects behavior before and after applying the method of compression.

The parameter *TIME* (*T*) reflects the evaluation of time for attack at an object BEFORE and AFTER the influence of the method of compression. The parameter *SIZE* (*S*) reflects the evaluation of the size of an object BEFORE and AFTER its processing with a method of compression.

After determining the main parameters, which will be analyzed and evaluated with regard to the information security of an object, is necessary to determine the basic characteristics, which have influence on the evaluation of the main parameters.

The basic characteristics, which have influence on the evaluation of the parameters BEFORE applying a method of compression to the object, are:

- for the evaluation of the parameter *TIME* the following characteristics can be taken into consideration: *time for examination* and *time for processing*;
- for the evaluation of the parameter *SIZE* will pointed characteristics depending of the category to which file objects belong to. Two basic categories are: DIRECTLY USED (these are objects, which have to be used directly) and NON-DIRECTLY USED (these are objects, requiring secondary processing to become directly used):
- the characteristics, which have influence on the evaluation of the parameter *SIZE* for objects belonging to DIRECTLY USED category, are: *characters' size, image's size, video's and audio's size* and *official information's size*;
- the characteristics, which have influence on the evaluation of the parameter *SIZE* for objects belonging to NON-DIRECTLY USED category, are: *resolution of the image*, *bit depth, official information's size* (for representatives of the group "graphical objects"); *sample size*, *sample rate*, *official information's size* (for representatives of the group "music and sound").

The basic characteristics, which have influence on the evaluation of the parameters AFTER applying a method of compression to the object, are:

- for the evaluation of the parameter *TIME* the characteristic *time for restoration* is added to these, mentioned above <u>before</u> applying a method of compression to an object;
- for the evaluation of the parameter *SIZE* are specified characteristics, depending of the method of compression applied over the object:
- when *statistical methods* of compression are applied, the characteristics (in addition to these mentioned above for DIRECTLY USED objects), which have influence on the evaluation, are: *entropy of the message*, *information redundancy*, *level of compression*, *bits of information after compression*, *size of the model for decompression*;
- when *dictionary methods* of compression are applied, the characteristics (in addition to these mentioned above for DIRECTLY USED objects), which have influence on the evaluation, are: *size of the dictionary*, *entropy of the message*, *information redundancy, level of compression*;
- when *image methods* of compression are applied the characteristics (in addition to these mentioned above for NON-DIRECTLY USED graphical objects), which have influence on the evaluation, are: *average number of pixel repetitions, average number of sequenced pixels*, *level of compression*;
- when *audio methods* of compression are applied the characteristics (in addition to these mentioned above for NON-DIRECTLY USED objects from the group "sound and music"), which have influence on the evaluation, are: *level of sample size*, *level of sample rate*, *average number of sequenced zero samples, level of compression*.

Each characteristic is defined evaluation (*V*) with respect to the information security of an object subjected to attack BEFORE and AFTER applying a method of compression. To establish these evaluations is taken into consideration additional factors affecting the evaluation of the respective characteristic. Then each characteristic is examined by conducting experiments. Thus, the relationship between the result obtained after the examination and evaluation of the characteristic with respect to the information security of an object, is determined. At the end the valuation (*V*) of the respective characteristic is determined.

The next stage is to determine the weighted co-efficient of each characteristic. The weighted co-efficient (*W*) determine the level of influence which each valuation of the respective characteristic have influence on the general evaluation of the parameter to which it belongs to. For determining the weighted co-efficient of the characteristic is used the AHP (Analytic Hierarchy Process) method [10], [33], which consists of four basic stages: 1) determining the characteristics which have to be evaluated; 2) arranging the chosen characteristics in a AHP matrix; 3) comparing each couple of characteristics by preliminarily selected bipolar measurement scales for evaluation; 4) determining the respective weights of the characteristics by consecution of mathematical operations.

The estimating of the general evaluation of the parameter consists of the following stages: 1) determining the evaluation of the characteristics, which have influence on the basic evaluation of the selected parameter $V_{(\text{charact.}_n)} = [0 \div 1]$, where *n* is the number of the characteristics; 2) setting the weighted co-efficient of each characteristic $W_{(\text{charact.}_n)}$, like $\sum_{i=1}^{n} W_i = 1$; 3) determining the evaluation of the parameter as $V_{(\text{parameter}_1)} = \sum_{i=1}^{n} \left( V_{(\text{charact.}_i)} . W_i \right)$.

## 4.2. Determination of the coefficient of information security

A co-efficient of information security ($K^{IS}$) is compounded to analyze the information security of the objects. It is presented as a variable, formed from the examined above parameters *TIME* and *SIZE*, reflecting the condition of the object before and after applying methods of compression.

The co-efficient of information security of an object reflects the total estimation of the parameters TIME and SIZE for each set of real relations between attack ($a_i$,), methods ($m_j$) and object ($o_f$), where $a_i \in A_{pot}$, $m_j \in M_{pot}$, $o_f \in O_{pot}$.

The determination of $K^{IS}$ for each relation attack—method—object proceeds over the following stages:

1) determining the co-efficient of information security for evaluation of the parameter TIME and SIZE;

To determine the co-efficient of information security of an object with respect to the parameter *TIME* ($RV_{(T)}$) and *SIZE* ($RV_{(S)}$), relatively valuation of the parameter TIME and SIZE is determined. It presents the number of increases of the value $V_{(T)}$ and $V_{(S)}$ of an object after processing it with method of compression. $RV_{(T)}$ and $RV_{(S)}$ can be represented as a ration of the *valuation-delta* ($\Delta V_{(T)}$ respectively $\Delta V_{(S)}$) and *valuation-prim* ($V'_{(T)}$ respectively $V'_{(S)}$) for the security of the object with respect to the time (Formula 1 and 2):

$$RV^{(T)} = \frac{\Delta V_{(T)}}{V'_{(T)}} \tag{1}$$

$$RV^{(S)} = \frac{\Delta V_{(S)}}{V'_{(S)}} \tag{2}$$

where $\Delta V_{(T)} = V''_{(T)} - V'_{(T)}$, $\Delta V_{(S)} = V''_{(S)} - V'_{(S)}$ like $V'_{(T)}$ and $V'_{(S)}$ is the determined valuation of information security of an object in regard to the time BEFORE applying the method of compression and $V''_{(T)}$ and $V''_{(S)}$ is the determined valuation of information security of an object in regard to the time AFTER applying the method of compression;

Thus for each set of real relations between attacks, methods and objects a relatively valuation of the object with respect to the parameters TIME and SIZE is determined.

For the parameters *TIME* and *SIZE* is formed a coefficient of the information security ($K^{IS(p)}$) for evaluation of the parameter ($p$) as (Formula 3):

$$K^{IS(p)} = \frac{RV_{(p)}}{\max RV_{(p)}} \tag{3}$$

where: $RV_{(p)} = \frac{V''_{(p)} - V'_{(p)}}{V'_{(p)}}$ is the average value of the parameter, which show with how many times is increased the value of the corresponding parameter after the application of a method of compression on an object; $V'_{(p)}$ is the value of the information security of the object corresponding to the parameter before the application of method of compression; $V''_{(p)}$ is the value of the information security of the object corresponding to the parameter after its treatment with method of compression; max$RV_{(p)}$ is the maximum average value corresponding to the parameter, achieved from the same object in one of the others *real* relations, in which it takes part.

The coefficient of the information security of an object ($K^{IS}$) can be represented as average value of the coefficients of evaluation of the parameters (Formula 4):

$$K_z^{IS} = \frac{1}{n}\sum_{p=1}^{n} K^{IS(p)} \tag{4}$$

where $K^{IS(p)}$ represents a coefficient of the information security of an object corresponding to a given parameter, $n$ is the number of tested parameters corresponding to the information security of an object, and $z$ varies in the boundaries of the multiplication of $a_p$, $m_q$ and $o_r$.

On Figure 10 a), b), c), d), e), f) is shown graphical interpretation of the obtained values of $K^{IS}$ for some of the most commonly used objects.

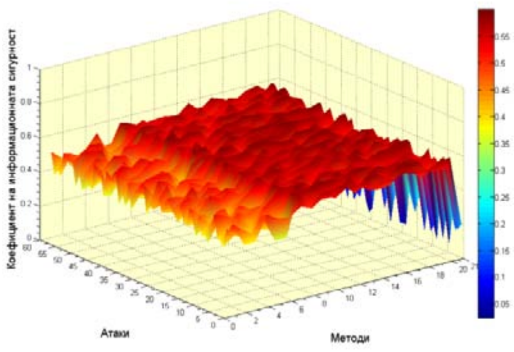


Figure 10 a) Geographic Information System object

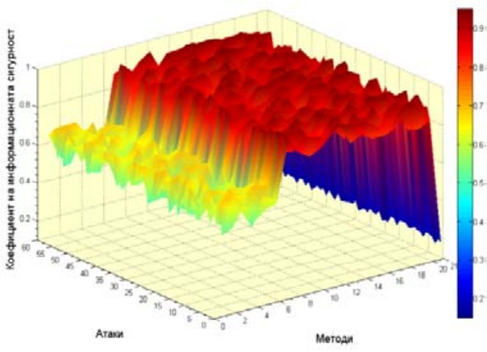


Figure 10 b) Text/Document object

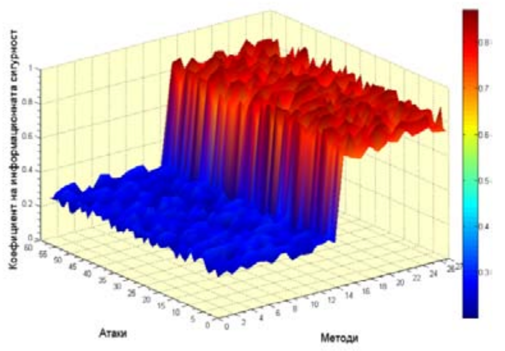


Figure 10 c) Raster graphic

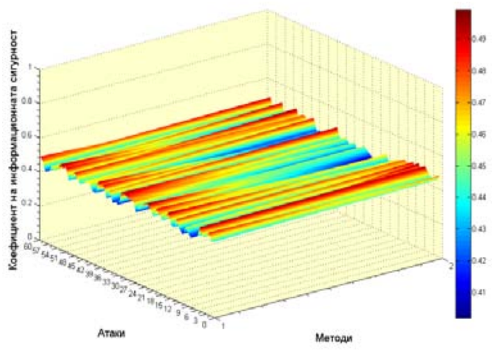


Figure 10 d) Uncompressed sound

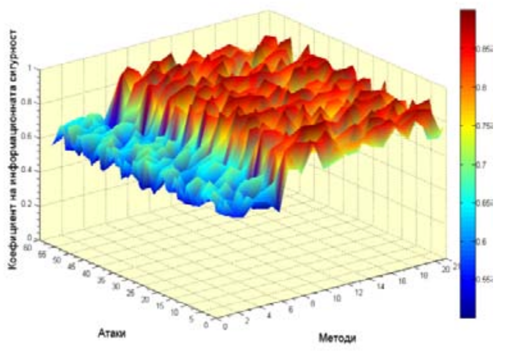


Figure 10 e) Dynamic web page

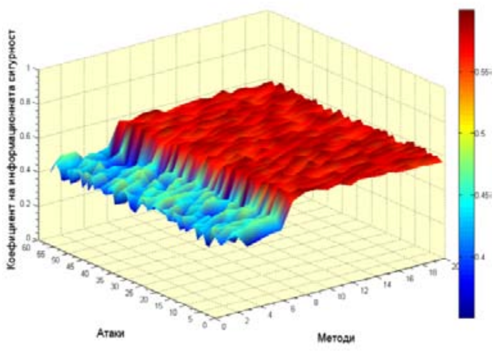


Figure 10 f) Source code

Figure 10 Graphic interpretations for determined values of the co-efficient of information security for different file objects

After determining $K^{IS}$ for each object we can determine which is the method with the highest value of $K^{IS}$ for the given object and attack. On Figure 11 a), b), c), d), e), f) we can see a graphical presentation of the change in the co-efficient of information security for given objects in regard to given attacks, determined after applying the given methods for compression.

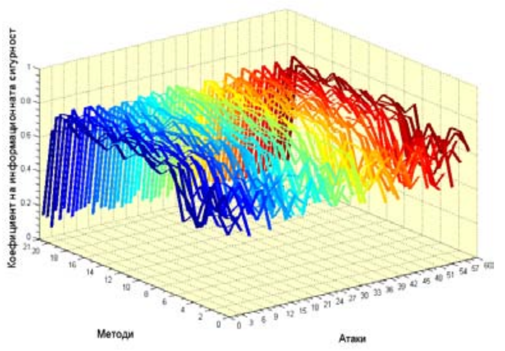


Figure 11 a) Geographic Information System fail object

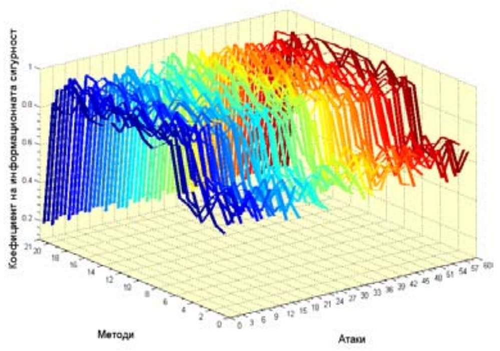


Figure 11 b) Text/Document object

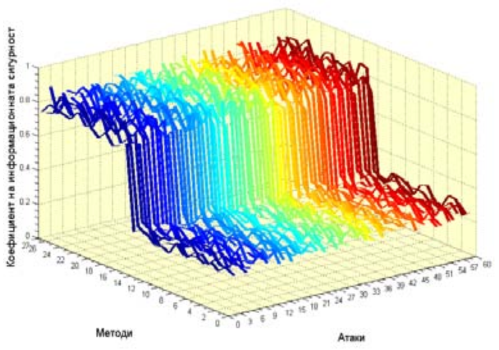


Figure 11 c) Raster graphic

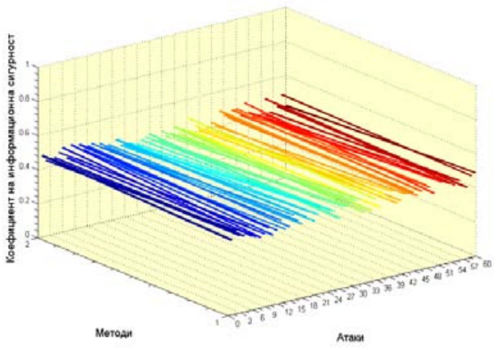


Figure 11 d) Uncompressed sound

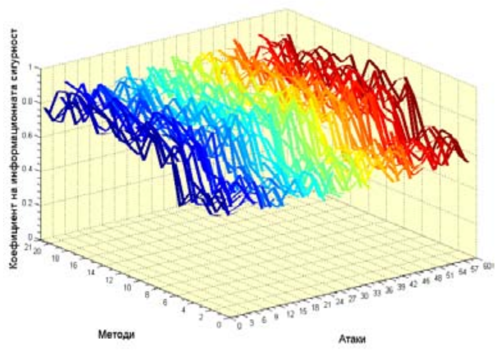


Figure 11 e) Dynamic web page

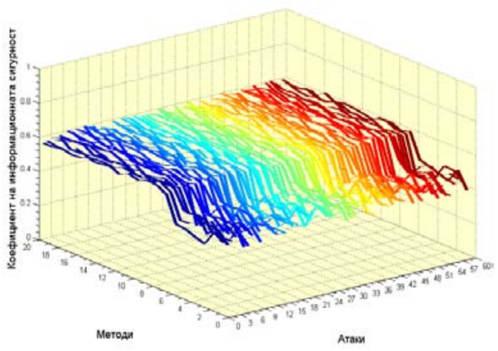


Figure 11 f) Source code

Figure 11 Distribution of the co-efficient for informational security for given object and attack, when a given method of compression is applied

In practice though, one object can be subjected of several attacks simultaneously. For each objects can be formed a group of methods, achieving highest values of $K^{IS}$ towards all attacks, to which it might be exposed. They are used to define the methods with lowest risk for compression in relation to the information security of the objects.

## 5. Selecting Procedure for Methods of Compression with Lowest Risk with Respect to the Coefficient of Information Security

### 5.1. Model for choosing of alternatives

The considered model is related to the examination of the possibility to influence on information security of objects, exposed to attacks by methods of compression. The final goal of the model is to choose the best alternatives (variants) for the decision making person by calculation several problems for multi-criteria evaluation [31]. Finally for each object a method of compression will be chosen which will reduce to the lowest risk with respect to the information security of selected object towards to all attacks to which the object can be exposed.

Stage one from the construction of the model is connected with the definition of the objects which will be explored and the different alternatives, compiling of the different compression methods, which can be applied to the corresponding object.

Stage two is connected with the selection of the different characteristics/situations compiling of the different attacks, which can attack the corresponding compressed object.

Stage three is connected with definition of the weight of the different characteristics/situations, e.g. to define for each attack the possibility to attack the chosen compressed object.

On stage four with the help of chosen criteria and methods for selection of alternatives is made a selection of a alternative with lowest risk (compression method), and the rest alternatives are sorted in descending order in relation to the information security of the object.

The last stage five from the construction of the model is connected with the selection of the best alternative (compression method) for each object, which is with lowest risk in relation to the information security of the object in question towards all attacks, to which it can be exposed.

In the model are included attacks ($a_i \in A_{pot}$), methods ($m_j \in M_{pot}$) and objects ($o_f \in O_{pot}$), which had been determined by means of matrix transformations, applied on initially build base of relations between maximum number of attacks ($A_{max}$), methods ($M_{max}$) and objects ($O_{max}$). After dropping out the sets of *real* relations, where lossy methods of compression take part, are analyzed 60 attacks (from the set of $A_{pot}$) from total 89 attacks (from the set of $A_{max}$), 36 methods (from the set of $M_{pot}$) from total 59 methods (from the set of $M_{max}$). The methods and the attacks are investigated on 27 objects (from the set of $O_{pot}$) from total 42 objects (from the set of $O_{max}$).

During description of the model, the following terms will be used:

Risk – when we speak of risk we will have in mind the risk of achieving a lower value of $K^{IS}$ of the object when applying a method of compression as means of protection from different attacks;

Profit – the profit of application of the method of compression on an object is connected with the achievement of higher value of $K^{IS}$.

## 5.2. Choosing of evaluation criteria

It is necessary to systematize the available information to realize this model. For that purpose a matrix $B_{(q,p)}$ is built, which includes the most efficient methods for one object (which are the different alternatives for decision-maker) and the attacks, which can access to these objects processed by these methods (which are the set of characteristics). The vector of numerical values for characteristics, which is assigned for each element, is the co-efficient of information security ($K_{INF}$). The matrix $B_{(q,p)}$ is built for each object from the set of *potential* number of objects.

The best variant for decision-maker can be determined with the help of the matrix and different methods for the game theory [14] and multi-criteria evaluation methods [27]. This variant includes the method of compression with the lowest risk with respect to the co-efficient of information security of the object, which is chosen by the decision-maker in connection to investigating attacks.

Under lowest risk alternative we assume the compression method, which best satisfies the execution of the target of the model, namely to achieve the best information security of the object towards all attacks through application of a compression method.

For the finding of alternative with lowest risk are used the following criteria from game theory:

- Maximum of the mathematical expectation for the profit ($E_a$);

- Minimum of the mathematical expectation for the risk (*$E_r$*);
- Criteria of Laplace for the profit (*$L_a$*);
- Criteria of Laplace fir the risk (*$L_r$*);
- Criteria of Wald (*WA*);
- Criteria of Savage (*SA*);
- Criteria of pessimism-optimism (*H*);
- Criteria of pessimism-optimism of risk (*F*).

The methods of compression can be sorted according level of preference with the help of the following two methods for multi-criteria evaluation (multi-criteria evaluation methods):

- Method of the linear combination of formal criteria (*$S_j$*);
- Method of maximum guaranteed result (*$t_j$*).

Both methods are based on the same model.

## Maximum of the mathematical expectation for the profit (Ea)

Maximum of the mathematical expectation for the profit (*$E_a$*) can be determined as (Formula 5) [13]:

$$E_a = \max_j \overline{b_j} \tag{5}$$

where $\overline{b_j} = \sum_{i=1}^{p} \lambda_i b_{ji}$ , *p* is the number of attacks, *$b_{ji}$* is a component of the matrix *$B_{(q,p)}$*, $\lambda_i$ is the weight co-efficient and it represents the possibility of one in preliminary chosen attack to get access to one object from preliminary chosen set of objects. The vector components $\vec{\lambda} = \left(\lambda_1, \lambda_2, ..., \lambda_i, ..., \lambda_p\right)$ are real non-negative numbers and represents weight for decision-making. The following limitation $\sum_{i=1}^{p} \lambda_i = 1$ has to be observed.

## Minimum of the mathematical expectation for the risk (Er)

Minimum of the mathematical expectation for the risk (*$E_r$*) can be determined as (Formula 6):

$$E_r = \min_j \overline{r_j} \tag{6}$$

where:

$$\overline{r_j} = \sum_{i=1}^{p} \lambda_i r_{ji},$$

$$r_{ji} = \beta_i - b_{ji},\ \beta_i = \max_j b_{ji}.$$

By analogy a variant of these both methods for multi-criteria evaluation can be examined when the values of weighted co-efficient are equal. For more detailed analysis can be assumed that the weighted co-efficient ($\lambda$) cannot always be known. In that case can be made the assumption that all values of $\lambda$ are equal (Laplace principle).

## Criteria of Laplace for the profit (La)

Criteria of Laplace [18] for the profit (*L<sub>a</sub>*) can be determined as (Formula 7):

$$L_a = \max_j \overline{b_j}^L \tag{7}$$

where $\overline{b_j}^L$ is the average value of the profit in cases when the weights co-efficient are equal: $\overline{b_j}^L = \sum_{i=1}^{p} \lambda_i^L b_{ji} = p^{-1} \sum_{i=1}^{p} b_{ji}$, where *p* is the number of the situations (attacks).

## Criteria of Laplace fir the risk (Lr)

Criteria of Laplace fir the risk ($L_r$) can be determined as (Formula 8):

$$L_r = \min_j \overline{r_j}^L \tag{8}$$

where $\overline{r_j}^L$ is the average value of the risk in cases when the weights co-efficient are equal: $\overline{r_j}^L = \sum_{i=1}^{p} \lambda_i^L r_{ji} = p^{-1} \sum_{i=1}^{p} r_{ji}$, where *p* is the number of the situations (attacks).

## Criteria of Wald (WA)

This is criterion of pessimism. Its calculation is necessary because the aim of the model is to select this variant which is with maximum profit and minimum risk. The Wald criterion (*WA*) is called maximin (criterion of pessimism) [21] and it selects for optimal this strategy which responds to (Formula 9):

$$WA = \max_j \alpha_j \tag{9}$$

where:

$$\alpha_j = \min_i b_{ji} .$$

## Criteria of Savage (SA)

This criterion also works with the risk and it is criterion of pessimism too. Criteria of Savage (*SA*) [15] selects for optimal this strategy, where the risk value is minimal in cases of the more unfavorable situation. The optimal strategy can be fined as (Formula 10):

$$SA = \min_j \gamma_j \tag{10}$$

where:

$$\gamma_j = \max_i r_{ji}.$$

## Criteria of pessimism-optimism (H)

Under existing circumstances a pessimistic position of the decision making person can be chosen in other cases – optimistic. The criterion uses the matrix $B_{(q,p)}$. The number $\theta$, which is a measure for the pessimism of the decision making person. The number can be between 0 and 1. When $\theta$=1, then we have situation of extreme pessimism. When $\theta$=0, this criterion become criterion of extreme optimism. Criterion of Hurwicz (*H*) [29], [32] recommends selecting this alternative, which (Formula 11):

$$H = \max_j h_j \tag{11}$$

where:

$$h_j = \theta\alpha_j + (1-\theta)\max_j b_{ji}.$$

The criterion is calculated in the model when $\theta$=0, $\theta$=0,5 and $\theta$=1.

## Criteria of pessimism-optimism of risk (F)

Criteria of pessimism-optimism of risk (*F*) [24] can be defined as (Formula 12):

$$F = \min_j f_j \tag{12}$$

where $f_j = \theta \max_i r_{ji} + (1-\theta)\min_i r_{ji}$.

The criterion is calculated in the model when $\theta$=0, $\theta$=0,5 and $\theta$=1.

The next stage of the model is related with the estimation of the decision. Therefore two main methods for multi-criteria evaluation are used: Method of the linear combination of formal criteria and Method of maximum guaranteed result. Both methods use the matrix ($B_{(q,p)}$), whose elements are normalized in matrix $C_{(q,p)}$ (Formula 13) [1].

$$c_{ji} = \frac{b_{ji}}{\beta_i} = \frac{b_{ji}}{\max_j b_{ji}} \tag{13}$$

Method of the linear combination of formal criteria (Sj)

The matrix with normalized values $C_{(q,p)}$ is used [7]. It uses the vector $\vec{\lambda} = \left(\lambda_1, \lambda_2, ..., \lambda_i, ..., \lambda_p\right)$ as an input parameter.

The calculating is:

(1) For each alternative (methods' group) is assigned the number $S_j$ where (Formula 14):

$$S_j = \sum_{i=1}^{p} \lambda_i c_{ji} \tag{14}$$

(2) The alternatives are sorted in ascending order by the number $S_j$, i.e. on the first place is the alternative with the maximum value of $S_j$, if there are several such an alternatives, their order in the list is arbitrary. Alternatives with lower values of $S_j$, follow, etc.

By analogy a variant of this both method can be examined when the values of weighted co-efficient (λ) are equal (Laplace). In this case (Formula 15):

$$S_j^L = \sum_{i=1}^{p} \lambda_i^L c_{ji} = p^{-1} \sum_{i=1}^{p} c_{ji} \tag{15}$$

Method of maximum guaranteed result (tj)

The matrix with normalized values $C_{(q,p)}$ [7], [34] is used. It uses the vector $\vec{\lambda} = \left(\lambda_1, \lambda_2, ..., \lambda_i, ..., \lambda_p\right)$ as an input parameter too, for which the following limitation $\sum_{i=1}^{p} \lambda_i = 1$ has to be observed for.

The calculating is:

(1) For each alternative (method of compression) is assigned the number $t_j$ where (Formula 16):

$$t_j = \min_i \left(\lambda_i c_{ji}\right) = \min\left(\lambda_1.c_{j1}, \lambda_2.c_{j2}, ..., \lambda_i.c_{ji}, ..., \lambda_p.c_{qp}\right) \tag{16}$$

(2) The alternatives are sorted in ascending order by the number $t_j$.

In case when the values of weighted co-efficient (λ) are equal (Laplace) (Formula 17):

$$t_j^L = \min_i \left(\lambda_i^L c_{ji}\right) \tag{17}$$

## 5.3. Procedure of choosing an alternative

These methods for multi-criteria evaluation are applied for each matrix (i.e. for each objects' group). Thus, for each object we can choose an alternative (method) we give preference to, with respect to the co-efficient of information security to all attacks.

## 6. Experiments Proving the Reliability of the Approach

The methods used for conducting of the experiments include the use of corresponding hardware and software instruments, which is connected with matrixes, matrix transformations, methods for evaluation of the risk, multi-criteria valuation, multi-criteria choice and other.

For the software realization is used the program system for scientific studies of the company "*The MathWorks*" which includes the *family фамилията Matlab® and Simulink®* – widely used software for analytical transformations, numerical calculations and graphical presentation of obtained results.

The used for the experiments apparatuses includes two server configurations each with two work stations. The first server configuration (in conjunction with two work stations) is used for investigation of "attacking behavior". The other server configuration (in conjunction with two work stations) is used for investigation of "protecting behavior". Each couple work stations are used accordingly for "managing station" and "standard station".

## 7. Assessments and Conclusion

### 7.1. With respect to the sets of attacks, methods and objects:

The selected number of maximum attacks (89 numbers), methods (59 numbers) and objects (42 numbers) is enough for determination of the sets of *potential* number of attacks, methods and objects.

After conducted expert evaluations and experiments can be summarized that:

- from total 3738 relations attack—objects, *expert* evaluated 2231 relations, *experimental* verified are 2861 relations and 2188 *possible* relations attack—object are formed;
- from total 2478 relations method—objects, *expert* evaluated 588 relations, *experimental* verified are 845 relations and 585 *possible* relations method—object are formed;
- from total 5251 relations attack—method, *expert* evaluated 3540 relations, *experimental* verified are 3835 relations and 3540 *possible* relations method—object are formed.

From total 220542 *maximum* triple relations, 33811 *real* triple relations attack—method--object are determined.

The number of attacks, methods and objects from the set of *potential* number of attacks, methods and objects, which take a part in the investigations are as follows: $A_{pot}$=60 numbers, $M_{pot}$=53 numbers and $O_{pot}$=30 numbers.

1) The chosen methodology for analyzing the relations attack-method-object by means of matrix transformations is effective and operative, and it contains the necessary potential for new deep analyses in this and another related areas.
2) The results give a possibility of specific planning of safety procedures and safety policies for the different computers, systems and networks configurations. Conditions are created for precise planning of economic expenses, connected with a specific safety policy with a specific configuration of computer, system and network.

### 7.2. With respect to the information security of object:

1) The selected for the investigation parameters TIME and SIZE are enough to investigate the information security not only of objects, but also of computer systems and networks when standard and not corporate (government) requirements are used.
2) The evaluation with respect to the chosen objects, which will be processed by methods of compression, is positive and suppositions don't influence on the obtained results. The evaluation with respect to the

chosen methods of compression is also positive and the conducted experiments can be generalized for other methods of compression, which don't take part in the investigation.

3) Based on the conducted experiments, we can make the conclusion that reducing the object's size after compression leads to increasing the time required by an attack to get an access to an object.

With respect to $K^{IS}$ the best results are shown with objects from the group of **data file objects**, processed with a method of compression belonging to the group of *dictionary* methods of compression. The worst results are shown with objects from the group of **binary file objects** and **graphic file objects**, processed with a method of compression belonging to the group of *statistical* methods of compression.

Fro total 53 numbers methods from the set of *potential* numbers of methods, 20 achive the highest values of $K^{IS}$ of the object. They are from the group of: *dictionary* methods of compression, *image* mathods of compression and *audio* methods of compression.

### 7.3. With respect to the procedure for methods of compression with lowest risk

1) The main task of risk management is risk optimization, i.e. to find the moment where the risk and attaining higher level of information security when methods of compression are applied on objects are compensate each other.
2) Independently from the made expenses, the risk assessment shows that the application of methods of compression to a great extent heightens the information security of objects under attacks.
3) The conducted experiments shows that the lowest risk with respect to the information security is obtained for objects from the groups **scientific file objects** and **data file objects**, exposed to attacks when methods of compression from the group of *dictionary* methods of compression are applied.

### 7.4. With respect to the experiments:

The program systems for scientific investigations *"The MathWorks"* can be successfully applied when determining the sets of real relations between attacks, methods and objects, obtained by using matrix transformations and reducing by stages the sets of *maximum* relations. Thus, from total 220542 *maximum* relations attacks—methods—objects are determined 33811 *real* relations, which are investigated with respect to the information security of objects.

With respect to the number of the elements of the sets of *expert* relations attack—object, methods—object and attack—method, can be concluded that they are accordingly 2231 from total 3738, 588 from total 2478 and 3540 from total 5251.

## Bibliography

Barley, M., Kasabov, N., *Intelligent Agents And Multi-Agent Systems: 7th Pacific Rim International Workshop on Multi-Agent Systems*, Springer (2005), ISBN 3540253408, p. 154

Butner, S, Ghodoussi, M., Transforming a Surgical Robot for Human Telesurgery, IEEE Trans. on Robotics and Automation, vol. 19, iss. 5, Oct. 2003, pp. 818 – 824

Cokus, M., Winkowski, D., XML Sizing and Compression Study For Military Wireless Data, Proceedings of XML Conference & Exposition 2002, Baltimore Convention Center, Baltimore, MD, USA, December 8-13, 2002

Damm, W., Josko, B., Pnueli, A., Votintseva, A., A Discrete-Time UML Semantics for Concurrency and Communication in Safety-Critical Applications, Science of Computer Programming, Vol. 55, 1-3/2005

David Dittrich. The DoS Project's "trinoo" Distributed Denial of Service Attack Tool (1999) (http://staff.washington.edu/dittrich/misc/trinoo.analysis)

Denning, D., A lattice model of secure information flow, Communications of the ACM, v. 19 n. 5, May 1976, pp. 236-243

Ferrari, E., Thuraisingham, B., Web And Information Security, IRM Press (2006), ISBN: 1591405890

FILExt – The File Extension Source (http://www.filext.com/)

Galotti, K., Making Decisions That Matter: How People Face Important Life Choices, Lawrence Erlbaum Associates (2002), ISBN 080583396X, p. 58

Gilbert, J., Brodersen, R., A lossless 2-D image compression technique for synthetic discrete-tone images, Data Compression Conference, 1998. DCC apos; 98. Proceedings Volume , Issue , 30 Mar-1 Apr 1998, pp. 359–368

Haffner, P., LeCun, Y., Bottou, L., Howard, P., Vincent, P., Riemers, B., Color documents on the Web with DjVu, Proc. IEEE Int. Conf. Image Processing, Kobe, Japan, Oct. 1999

Hasenauer, H., Sustainable Forest Management: Growth Models for Europe, Springer (2006), pp. 267-269

Iddan, G., Meron G, Glukhovsky A, Swain P, Wireless Capsule Endoscopy, Nature, vol. 405, 25 May 2000

Istepanian, R., Pattichis, C., Laxminarayan, S., M-Health: Emerging Mobile Health Systems, Springer (2006), ISBN 0387265589, p. 282

Karlin, S., Mathematical Methods and Theory in Games, Programming, and Economics, Courier Dover Publications (2003), ISBN 0486495272, pp. 179-182

Keeping, E., Introduction to Statistical Inference, Courier Dover Publications (1995), ISBN 0486685020, pp. 151-173

Kerzner, H., Project Management: a systems approach to planning, scheduling, and controlling, John Wiley and Sons (2003), ISBN 0471225770, p. 659

Klein, D., Foiling the Cracker: A Survey of, and Improvements to, Password Security, In UNIX Security Workshop II, August 1990

Koohang A., Harman, K., Learning Objects and Instructional Design, Informing Science (2006), ISBN-13: 978-8392233770, p. 180

Kundisch, D., New Strategies for Financial Services Firms: The Life-Cycle-Solution Approach, Springer (2003), ISBN 379080066X, pp. 148-149

Marco de Vivo, Gabriela O. de Vivo, Germinal Isern. Internet Security Attacks at the Basic Levels. ACM SIGOPS Operating Systems Review, 32(2):4–15, (1998)

Martin, T., Hsiao, M., Ha, D., Krishnaswami, J., Denial-of-Service Attacks on Battery-powered Mobile Computers, Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04), (2004), ISBN 0769520901, pp. 309-318

Nielsen, T., Zhang, N., Symbolic and Quantitative Approaches to Reasoning With Uncertainty, 7th European conference ECSQARU 2003 Aalborg, Denmark, July 2003 Proceedings, Springer (2003), ISBN 3540404945, pp. 3-4

Radhamani, G., Rao, R., Web Services Security and E-business, Global (2007), ISBN-13: 978-1599041681, p. 115, p. 25

Reilly, E., Concise Encyclopedia of Computer Science, John Wiley and Sons (2004), ISBN 0470090952, p. 388

Ricci, P., Environmental and Health Risk Assessment and Management: Principles and Practices, Springer (2006), ISBN 1402037759, p. 54

Rodgers, C., Threats to TCP/IP Network Security. (2001)

Salomon, D., Data Compression: The Complete reference, Springer (2004), ISBN 0387406972, p. 868

Sandblom, C.-L., Eiselt, H., Decision Analysis, Location Models, and Scheduling Problems, Springer (2004), ISBN 3540403388, pp. 19-150

Sayood, K., Lossless Compression Handbook, Elsevier (2003), ISBN 0126208611, pp. 229-234, pp. 273-273, pp. 301-310

Schniederjans, A., Information Technology Investment: Decision-making Methodology, World Scientific (2004), ISBN 9812386955, p. 244, p. 249

Shaw, W., Cybersecurity for SCADA Systems, PennWell Corp. (2006), ISBN-13: 978-1593700683, p. 194

Straffin, P., Game Theory and Strategy, The Mathematical Association of America (1996), ISBN 0883856379, pp. 7-22, pp. 56-62

Vigna, G., Jonsson, E., Kruegel, C., Recent Advances in Intrusion Detection: 6th International Symposium, RAID 2003, Pittsburgh, PA, USA, September 2003, Proceedings, Springer (2003), ISBN 3540408789, p. 146

Vinze, A., Chen, H., Raghu, T., Zeng, D., Ramesh, R., National Security, Elsevier (2007), ISBN 0444519963, p. 69

Попчев, И., Метев, Б., Христов, Ч., Маркова, Л. Диалогова система за многокритериална оценка и избор, Печатница на Издателството на БАН (1985), 3-5 стр., 8-12 стр.

Стенли, Т., Компресиране на данни, Интерфейс България, (1998)

http://ncs.nlcv.bas.bg/index_en.htm

http://www.sptimes.com/Hackers/history.hacking.html

## Authors' Information

*Dimitrina Polimirova, PhD, Research Associate, National Laboratory of Computer Virology, Bulgarian Academy of Sciences, Part-time lecturer at New Bulgarian University, Phone: +359-2-9733398, E-mail: polimira@nlcv.bas.bg*

*Eugene Nickolov, Prof., DSc, PhD, Eng, National Laboratory of Computer Virology, Bulgarian Academy of Sciences, Regular lecturer at New Bulgarian University, Phone: +359-2-9733398, E-mail: eugene@nlcv.bas.bg*

# OBJECT LEVEL RUN-TIME COHESION MEASUREMENT

Varun Gupta, Jitender Kumar Chhabra

*Abstract: Most of the object-oriented cohesion metrics proposed in the literature are static in nature and are defined at the class level. In this paper, new dynamic cohesion metrics are proposed which provide scope of cohesion measurement up to the object level and take into account important and widely used object-oriented features such as inheritance, polymorphism and dynamic binding during measurement. The proposed dynamic measures are computed at run-time, which take into consideration the actual interactions taking place among members of a class. The proposed measures are evaluated using a theoretical framework to prove their usefulness. A dynamic analyzer tool is presented which can be used to perform dynamic analysis of Java applications for the purpose of collecting run-time data for the computation of the proposed metrics. Further, a case study is conducted using a Java program to demonstrate the computation process for the proposed dynamic cohesion measures.*

*Keywords: Cohesion; Software metrics; Static metrics; Dynamic metrics; Aspect oriented programming; Object-oriented software.*

*ACM Classification Keywords: D.2.8 [Software Engineering]: Metrics; D.2.3 [Software Engineering]: Coding Tools and Techniques - Object-oriented programming*

## Introduction

High quality software, among many other principles, should obey the principle of high cohesion. Stevens et al. [1], who first introduced cohesion in the context of structured development techniques, define cohesion as a measure of the degree to which the elements of a module belong together. In a highly cohesive module, all elements are related to each other for performing a single function. The higher the cohesion of a module, the easier the module is to develop, maintain, and reuse, and the less fault-prone it is [2], [3], [4]. The principle of high cohesion has been transferred to object-oriented software by Coad and Yourdon [5, 6] and research in this field has lead to a large number of cohesion measures for object-oriented systems being defined [7-21]. However, most of the cohesion metrics proposed in the literature for measurement of cohesion are static in nature and static cohesion metrics may be insufficient in evaluating the dynamic behavior of an application at runtime, as its behavior will be influenced by the execution environment as well as the complexity of the source code. Object-oriented features such as polymorphism, dynamic binding, inheritance and common presence of unused code in commercial software, cause the static metrics to be inaccurate, as they do not precisely reflect the run-time situation of the software [22]. Moreover, the complex dynamic behavior of many real-time applications motivates us to focus on dynamic cohesion metrics in place of static cohesion metrics. Dynamic cohesion metrics are obtained from the execution traces of the code or from the executable models. Till date, only a few dynamic metrics have been proposed for the measurement of cohesion. Gupta et al. [23] have proposed program execution based module cohesion metrics based on the dynamic slicing of the program. However, these metrics only deal with procedure-oriented program. The run-time cohesion metrics proposed by Mitchell et al. [24, 25] are just dynamic equivalent of the existing cohesion metrics such as LCOM given by Chidamber and Kemerer [7].

The remainder of the paper is organized as follows. Section 2 discusses advantages of dynamic cohesion measures over static cohesion measures and Section 3 contains the definitions of the proposed dynamic cohesion measures. In Section 4, the proposed measures are validated theoretically and Section 5 provides a dynamic analyzer tool for computation of the proposed measures. In Section 6, a case study is conducted to demonstrate the process of computation of the proposed measures and finally, Section 7 concludes the work.

## Advantages of Dynamic Cohesion Metrics

Static cohesion metrics are obviously simpler to collect because there is no need to run the software. Moreover, to obtain dynamic cohesion metrics, code or simulation model of the software system is needed, which is available very late in the development life cycle. Static cohesion metrics are widely used due to the fact that they are easier to obtain, especially at the early stages of software development. But, the potential benefits of dynamic cohesion metrics collected by executing the program outweigh the complexity and cost of measuring them. The ability to static cohesion metrics to measure the quality attributes of a software system is less apparent, as the static cohesion metrics are evaluated only by means of static inspection of the software artifact. Since, it is the actual runtime behavior of the system that determines its quality, not the potential characteristics implied by the static analysis of the software system and dynamic cohesion metrics are computed based on the data collected during actual execution of the system, and thus directly reflect the quality attributes (performance, change-proneness, error rates etc.) of the software in its operational mode. Moreover, static cohesion metrics deal with the structural aspects of a software system, whereas dynamic cohesion metrics also deal with the behavioral aspects of the system. Moreover, static cohesion metrics are somewhat constrained in their ability to deal with inheritance, polymorphism and dynamic binding issues since the run-time types at field access and method invocation sites are not known, whereas dynamic cohesion metrics are capable to deal with such issues.

## Dynamic Cohesion Metrics

The mapping level of dynamic cohesion measurement can be either object or class. Object-level dynamic cohesion quantifies the extent of dependencies between the members of an object at run-time. As object is an instance of a class created at runtime, class-level dynamic cohesion aggregates the object-level cohesion values of all instances of a class. As an object or a class consists of two types of elements i.e. attributes and methods and there are mainly two kinds of dependencies among elements of an object or a class: (i) dependency between attributes and methods, and (ii) dependency between a pair of methods. First types of dependency exists due to read and write types of interactions present between methods and attributes i.e. when a method reads or writes the value of an attribute. Second type of dependency takes place due to the presence of call type of interactions between methods i.e. when a method calls other method of the class or object. However, not all the methods of a class contribute to its cohesion [21]. There exist some special methods such as constructor, destructor, access methods and delegation methods intrinsically accessing only some of the attributes in the class [17], [18], [21]. A constructor is a type of method that initializes essential attributes of the class and a destructor is a type of method that may only de-initialize crucial attributes of the class. An access method is a method that only reads or writes a particular attribute of the class. A delegation method is a method that only delegates a message to another object, especially to an attribute in the class, thus, generally have only one interaction with one attribute. These special methods may not essentially access all of the attributes. It has been widely accepted by a number of authors that these methods have no influence on the cohesion of a class [4], [17], [18], [21]. Thus, these methods need to be excluded in the measurement of cohesion of a class. The cohesion measurement of an object or class should consider only two types of elements: normal methods (except special methods) and attributes and two types of dependencies between elements i.e. access relations between normal methods and attributes and call relations between pairs of normal methods. Thus, the dynamic cohesion of an object or class should be measured from the two aspects.

## Dynamic Access Cohesion

Dynamic access cohesion exists between methods and attributes of an object when a method of an object reads or writes an attribute of the same object during execution of the program. This type of dynamic cohesion for an object o is defined as the ratio of actual number of distinct dependence relations between all methods and all attributes to the maximum possible number of dependence relations of this type between them (i.e. $n \times m$). In case, if either number of methods or number of attributes are zero for an object then this type of cohesion would be nil for that object. This kind of dynamic cohesion for an object is defined as follows: -

$$DC_{ACC}(o) = \begin{cases} 0 & \text{n=0 or m=0} \\ \dfrac{\sum_{i=1}^{n}\sum_{j=1}^{m} Dep_R(m_i, a_j)}{n \times m} & \text{n} \neq 0 \text{ and m} \neq 0 \end{cases}$$

Where $Dep_R(m_i, a_j)$ is the access dependency present at run-time between a method $m_i$ and an attribute $a_j$ of an object o. Also, n is the total number of methods of object o and m is the total number of attributes of object o.

## Dynamic Call Cohesion

Dynamic call cohesion exists between a pair of methods of an object when a method $m_i$ calls other method $m_j$ of the object during program execution. This kind of dynamic cohesion of an object o is defined as the ratio of actual count of distinct dependence relations between all ordered pairs of methods to the maximum possible number of relations of this type between them (i.e. $n \times (n-1)$). In case, if number of methods of an object is zero then this type of cohesion is also zero for that object and if a single method exists for an object, then this type of cohesion is maximum i.e. 1 for that object. This form of dynamic cohesion for an object o is defined as follows: -

$$DC_{CALL}(o) = \begin{cases} 0 & \text{n} = 0 \\ \dfrac{\sum_{i=1}^{n}\sum_{j=1 \wedge j \neq i}^{n} Dep_R(m_i, m_j)}{n \times (n-1)} & \text{n} \neq 0 \\ 1 & \text{n} = 1 \end{cases}$$

Where $Dep_R(m_i, m_j)$ is the call dependency present at run-time between methods $m_i$ and $m_j$ of an object o. Also, n is the total number of methods of object o.

## Weight-age of Different Types of Cohesion

There are two types of dynamic cohesion for an object as defined above. All these types of cohesion have got different weight-ages due to the different types of dependence relations attached with them. The weight-ages of these types of cohesion having defined after taking into consideration their relative ordering as well as expert developers' opinions and are given in Table 1. The weights to these types of cohesion are assigned as per the intensity of the relation attached with them. First, the dynamic cohesion due to access dependency between methods and attributes ($DC_{ACC}$) is more significant than dynamic cohesion due to call dependency between methods ($DC_{CALL}$) due to the fact that most of the cohesion measures are defined in terms of degree of interactions among methods and attributes [7], [11], [8], [13], [9], and [10]. Thus, $DC_{ACC}$ has got more weight-age than $DC_{CALL}$.

Table 1 Weight-age of different types of cohesion

| Dynamic Cohesion Type | Weight-age |
|---|---|
| Dynamic access cohesion ($DC_{ACC}$) | 2 |
| Dynamic call cohesion ($DC_{CALL}$) | 1 |

## Object Level and Class Level Cohesion Measures

Object level dynamic cohesion for an object is defined as the weighted summation of two types of cohesions defined above. The Dynamic cohesion for an object o is defined as:

$$ODC(o) = \frac{w_1 * DC_{ACC}(o) + w_2 * DC_{CALL}(o)}{w_1 + w_2}$$

Where, $w_1$=2 and $w_2$=1

Class level Dynamic Cohesion for a class is defined as the average of the values of Object level Dynamic Cohesion for all objects of a class created at run-time i.e.

$$CDC(c) = \frac{\sum_{i=1}^{k} ODC(o_i)}{k}$$

Where *k* is the number objects of class created at run-time.

## Theoretical Validation

The purpose of this section is to validate the proposed measures theoretically by using the four properties given by Briand et al. [26]. The four cohesion properties defined by Briand et al. characterize cohesion in a reasonably intuitive and rigorous manner. A well-defined cohesion measure should have the following four properties. These properties provide a guideline to develop a good cohesion measure.

Property 1 (Non-negativity and Normalization). Normalization of a cohesion measure makes it possible to carry out meaningful comparisons between the cohesion values of classes or objects having different number of elements, since they all belong to the same interval [6]. As per the definitions of the above-defined measures, the cohesion of an object or a class c lies within a specified range i.e. $ODC(o) \in [0,1]$ and $CDC(c) \in [0,1]$. Thus, Property 1 holds for the proposed cohesion measures.

Property 2 (Null value). This property states that if there is no dependency among the members of an object or class, then the cohesion of that object or class should be null. As per the definitions of the proposed measures, if there is no dependency relation between the elements of an object at run-time, then the values of the two types of cohesions for an object o i.e. $DC_{ACC}$(o) and $DC_{CALL}$(o) will certainly be zero and as a result dynamic cohesion of an object o, ODC(o) will also be zero as ODC(o) is the weighted summation of the above measures only. Moreover, the cohesion of a class c will also be null if cohesion values of all objects of the class are null. Thus, the proposed measures satisfy Property 2.

Property 3 (Monotonicity). This property requires that by addition of dependency relationships among elements of an object or a class should not decrease its cohesion.

Let object, o = <$E^R$, $R^R$>, where $R^R$ represents the set of relations among set of elements, $E^R$ of an object o at run-time. Let a relationship is added to o to form a new object o′ = <$E^R$, $R^{R'}$>, which is identical to o except that $R^R \subset R^{R'}$. Then, as per the above given definitions of the measures, dynamic cohesion value of new object will only increase or will remain the same but will never decrease.

For objects, o = <$E^R$, $R^{R'}$> and o′ = <$E^R$, $R^{R'}$>, if $R^R \subset R^{R'}$ then ODC(o) $\leq$ ODC(o'). Similarly, the property holds at class level also. Thus, the proposed measures satisfy this property as well.

Property 4 (Merging of objects or classes). This property states that the cohesion of an object or a class obtained by putting together two unrelated objects or classes is not greater than the maximum cohesion of the two original objects or classes. If two unrelated objects $o_i$ and $o_j$ are merged to form a new object $o_k$ then the cohesion of $o_k$ is no larger than the maximum cohesion of $o_i$ and $o_j$ or if two unrelated classes $c_i$ and $c_j$ are merged to form a new class $c_k$, then cohesion of $c_k$ is no larger than the maximum cohesion of $c_1$ and $c_2$.

For, $o_i$ = <$E_i^R$, $R_i^{R'}$> and $o_j$ = <$E_j^R$, $R_j^R$> where $R_i^R \cap R_j^R = \phi$.

and $c_i$ = <$E_i$, $R_i$> and $c_j$ = <$E_j$, $R_j$> where $R_i \cap R_j = \phi$

Since, two unrelated objects or classes have been combined to form a new object or class; there is a proportionate increase in number of dependency relations as well as in number of elements. As per the definition of cohesion measures which measure cohesion in terms of ratio of actual number of dependence relations existing at run-time divided by the maximum possible number of relations among elements. There is no net increase in the value of cohesion measure since numerator values as well as denominator values have increased together. Thus, the cohesion value of the combined object or class cannot be more than the maximum of the two unrelated objects or classes i.e.

$Max\{ODC(o_i), ODC(o_j)\} \geq ODC(o_k)$ and $Max\{CDC(c_i), CDC(c_j)\} \geq CDC(c_k)$

Hence, the proposed cohesion measures satisfy property 4 also.

## Dynamic Analyzer for the Proposed Measures

We used aspect-oriented programming (AOP) approach for dynamic analysis of object-oriented programs for the purpose of computation of the proposed measures, as AOP is an efficient technique for dynamic analysis without any side effects [27]. We used AspectJ [28] to develop a dynamic analyzer tool dynamic analysis of Java applications for computation of the proposed measures. We have written an aspect using Aspectj for dynamic analysis of target Java programs and this aspect is an independent programming unit and can be merged with the target Java programs without altering the behavior of the target programs. Figure 1 presents the key features of the dynamic analyzer tool implemented using AspectJ.

```
public aspect DynamicCohesionAnalyser{

//Pointcuts defined
pointcut traceMethods() : (execution (* *.*(..)) ) …
pointcut traceAttribs() : ( (get(* *) || set(* *) ) …
pointcut traceAccess() : ( (get(* *) || set(* *) )  && withincode(* *.*(..)) …
pointcut traceCall () : call(* *.*(..))  …

// Advices defined for capturing pointcuts
before(): traceMethods(){
Signature sig=thisJoinPointStaticPart.getSignature();
…
 }

after(): traceAttribs(){
Signature sig=thisJoinPointStaticPart.getSignature();
…
}
after(): traceAccess() {
…
}
   before(): traceCall (){
…
}

// methods storing data collected at run-time into files
void writeToFile_traceCall(Signature sig)
{
…
}

void writeToFile_traceCall (Signature sig)
{
…
}
} //aspect
```

Figure 1 Main features of the Dynamic Analyzer Tool

## Case Study

In this section, a case study is carried out to demonstrate the process of computation of the proposed dynamic cohesion measures using a program written in Java [29] shown in Figure 2. This program consists of a class ArrayQueue [30]. This class consists of four attributes and seven normal methods.

```
public class ArrayQueue {
private Object [ ] theArray;
private int  currentSize;
private int front;
private int back;
 public ArrayQueue( )    {
 theArray = new Object[10];
        makeEmpty( );    }
 public boolean isEmpty( )    {
        return currentSize == 0;    }
public void makeEmpty( )    {
        currentSize = 0;
        front = 0;
        back = -1;    }
public Object dequeue( )    {
        if( isEmpty( ) )
            throw new UnderflowException( "ArrayQueue dequeue" );
        currentSize--;
       Object returnValue = theArray[ front ];
        front = increment( front );
        return returnValue;    }
   public Object getFront( )    {
        if( isEmpty( ) )
            return theArray[ front ];    }
 public void enqueue( Object x )    {
        if( currentSize == theArray.length )
            doubleQueue( );
        back = increment( back );
        theArray[ back ] = x;
        currentSize++;    }
 private int increment( int x )    {
        if( ++x == theArray.length )
            x = 0;
        return x;    }
 void doubleQueue( )    {
        Object [ ] newArray;
        newArray = new Object[ theArray.length * 2 ];
        for( int i = 0; i < currentSize; i++, front = increment( front ) )
            newArray[ i ] = theArray[ front ];
        theArray = newArray;
        front = 0;
        back = currentSize - 1;    }
public static void main(String str[]) {
ArrayQueue q1=new ArrayQueue(5);
…
…
}//main method
} //ArrayQueue Class
```

Figure 2 Java program [29]

On the execution of the above program along with the dynamic analyser tool, values of different cohesion measures obtained are as follows: -

$DC_{ACC}$ (q1) =0.61

$DC_{CALL}$ (q1) = 0.07

Thus, dynamic cohesion values for object q1 and class ArrayQueue are calculated as follows: -

ODC (q1) = (2*0.61+1*0.07)/3= 0.84

CDC (ArrayQueue) =0.84

## Conclusion

This paper proposes new well-defined dynamic cohesion measures which satisfy the four cohesion properties defined by Briand et al. and in comparison with the existing cohesion measures, the proposed measures have the following advantages: -

- The proposed dynamic cohesion measures are more accurate as they are defined at run-time and take into consideration the actual interactions taking place rather than the potential interactions which may or may not happen as is the case with static cohesion metrics.
- The proposed cohesion metrics take inheritance and polymorphism into consideration as the actual targets of polymorphic invocations can only be determined at run-time due to the presence of inherited members of a class.
- The scope of measurement of the proposed dynamic cohesion metrics can be specific to a single object. Whereas, other existing cohesion metrics are able to measure cohesion up to the class level only.

In future work, we plan to conduct some empirical study and compare these new dynamic cohesion measures with the existing static and dynamic cohesion measures to prove that the proposed measures are better indicators of dynamic cohesion in comparison to the existing metrics.

## Bibliography

[1] W. Stevens, G. Myers, and L. Constantine, Structured design, *IBM Systems J.* 13(2) (1974) 115–139.

[2] D.N. Card, V.E. Church, and W.W. Agresti, An empirical study of software design practices, IEEE Transactions on Software Engineering 12(2) (1986) 264-271.

[3] L. Briand, S. Morasca, and V. Basili, Defining and validating high-level design metrics, Technical Report, University of Maryland, CS-TR 3301, 1994.

[4] L.C. Briand, J.W. Daly,and J. Wust, A unified framework for cohesion measurement in object-oriented systems, Empirical Software Engineering 3(1) (1998) 65–117.

[5] P. Coad and E. Yourdon, Object-Oriented Analysis (Prentice Hall, 1991).

[6] P. Coad and E. Yourdon, Object-Oriented Design (Prentice Hall, 1991).

[7] S.R. Chidamber and C.F. Kemerer, A metrics suite for object-oriented design, IEEE Transactions on Software Engineering 20(6) (1994) 476–493.

[8] J. Bieman and B. Kang, Cohesion and reuse in an object-oriented system, In: Proceedings of the ACM Symp. Software Reusability (SSR'95), (1995) 259–262. Reprinted in ACM SIGSOFT Software Engineering Notes (1995).

[9] L.M. Ott, J.M. Bieman and B.K. Kang, Developing measures of class cohesion for object-oriented software, In: Proceedings of the 7th Annual Oregon Workshop on Software Metrics, (Oregon, Portland, 1995).

[10] L.M. Ott and J.M. Bieman, Program slices as an abstraction for cohesion measurement, Journal of Information and Software Technology 40(11–12) (1998) 691–699.

[11] M. Hitz and B. Montazeri, Measuring coupling and cohesion in object oriented systems, In: Proceedings of the International Symposium on Applied Corporate Computing, (Monterrey, Mexico, 1995) 25–27.

[12] Y.S. Lee and B.S. Liang, Measuring the coupling and cohesion of an object-oriented program based on information flow, In: Proceedings of the International Conference on Software Quality, (Maribor, Slovenia, 1995) 81–90.

[13] B. Henderson-Sellers, Software Metrics, (Prentice Hall, Hemel Hempstaed, UK, 1996).

[14] S. Moser, V.B. Misic, Measuring class coupling and cohesion: a formal meta- model approach, In: Proceedings of the Asia Pacific Software Engineering Conference and International Computer Science Conference, (IEEE Computer Society Press, Hong Kong, 1997) 31–40.

[15] S. Counsell, E. Mendes, S. Swift, A. Tucker, Evaluation of an object-oriented cohesion metric through Hamming distances. Tech. Rep. BBKCS-02-10, Birkbeck College, University of London, UK, 2002.

[16] J. Bansiya, L.H. Etzkorn, C.G. Davis, W. Li, A class cohesion metric for object oriented designs, Journal of Object-oriented Programming 11(8) (1999) 47–52.

[17] H.S. Chae and Y.R. Kwon, A cohesion measure for classes in object-oriented systems, In: Proceedings of the Fifth International Software Metric Symposium (METRICS'98), (Bethesda, MD, USA, IEEE Computer Society Press, 1998) 158–166.

[18] H.S. Chae, Y.R. Kwon, D.H. Bae, A cohesion measure for object oriented classes, *Software Practice and Experience* 30(12) (2000) 1405–1431.

[19] Z. Chen, Y. Zhou, B. Xu, J. Zhao, H. Yang, A novel approach to measuring class cohesion based on dependence analysis, In: *Proceedings of the International Conference on Software Maintenance*, (IEEE Computer Society Press, Montreal, Canada, 2002) 377–384.

[20] Y. Zhou, B. Xu, J. Zhao and H. Yang, ICBMC: an improved cohesion measure for classes, In: *Proceedings of the International Conference on Software Maintenance*, (IEEE Computer Society Press, Montreal, Canada, 2002) 44–53.

[21] J. Wang, Y. Zhou, L. Wen, Y. Chen, H. Lu and B. Xu, DMC: a more precise cohesion measure for classes, *Information and Software Technology* 47(3) (2005) 167-180.

[22] A. Mitchell, J.F. Power, An Empirical Investigation into the Dimensions of Run-Time Coupling in Java Programs, In: *Proceedings of the 3rd Conference on the Principles and Practice of Programming in Java*, (Las Vegas, Nevada, 2004) 9–14.

[23] N. Gupta and P. Rao, Program execution based module cohesion measurement, In: *Proceedings of the 16th International Conference on Automated on Software Engineering (ASE '01)*, (San Diego, USA, 2001).

[24] A. Mitchell and J.F. Power, Run-Time Cohesion Metrics for the Analysis of Java Programs, *Technical Report Series no. NUIM-CS-TR-2003-08, National University of Ireland*, (Maynooth, Co. Kildare, Ireland, 2003).

[25] A. Mitchell and J.F. Power, Run-Time Cohesion Metrics: An Empirical Investigation, *In: Proceedings of the SERP*, 2006.

[26] L.C. Briand, S. Morasca, and V.R. Basili, Property-based software engineering measurement, *IEEE Transactions on Software Engineering* 22(1) (1996) 68–85.

[27] V. Gupta and J.K. Chhabra, Measurement of dynamic metrics using dynamic analysis of programs, *In: Proceedings of the WSEAS International Conference on Applied Computing Conference*, Istanbul, Turkey, (2008) 81-86.

[28] AspectJ, Available at http://www.eclipse.org/aspectj (accessed 05 Dec 2009).

[29] P. Naughton and H. Schildt, Java 2: The Complete Reference, 3rd revised edition (McGraw-Hill, 1999).

[30] ArrayQueue implementation in Java, Available at http://www.java-tips.org (accessed 15 Dec 2009).

## Authors' Information

***Varun Gupta*** *– Researcher; Department of Computer Engineering, National Institute of Technology, Kurukshetra, Kurukshetra-136119 India; e-mail: varun3dec@yahoo.com*

*Major Fields of Scientific Research: Software Engineering, Object Oriented Design & Development, Aspect Oriented Programming.*

***Jitender Kumar Chhabra*** *– Astt. Professor; Department of Computer Engineering, National Institute of Technology, Kurukshetra, Kurukshetra-136119 India;*

*e-mail: jitenderchhabra@rediffmail.com*

*Major Fields of Scientific Research: Software Engineering, Database System, Data Structure, Procedural and Object-Oriented Programming.*

# PROGRAMMING OF AGENT-BASED SYSTEMS

Dmitry Cheremisinov, Liudmila Cheremisinova

*Abstract: The purpose of the paper is to explore the possibility of applying the language PRALU, proposed for description of parallel logical control algorithms and rooted in the Petri net formalism for design and modeling real-time multi-agent systems. It is demonstrated with a known example of English auction on how to specify an agent interaction protocol using considered means. A methodology of programming agents in multi-agent system is proposed; it is based on the description of its protocol on the language PRALU of parallel algorithms of logic control. The methodology consists in splitting agent programs into two parts: the block of synchronization and the functional block.*

*Keywords: multi-agent system, interaction protocol, BDI agent, parallel control algorithm.*

*ACM Classification Keywords: I.2.11 [Computer Applications]; Distributed Artificial Intelligence, Multiagent systems; D.3.3 [Programming Languages]: Language Constructs and Features – Control structures, Concurrent programming structures*

## Introduction

In recent years one of the most rapidly growing areas of interest for distributed computing is that based on the concept of agents and multiagent systems (MAS). The first MAS applications have appeared in the mid-1980s. Now they are becoming one of the most important topics in distributed and autonomous decentralized systems. There are increasing attempts to use agent technologies in variety of domains, ranging from manufacturing to process control, air traffic control and information management. Programming technology based on the use of interacting agents is considered the most promising tool of modern programming.

MAS is a computational system which consists of a number of agents that operate together in order to perform a set of tasks or to achieve a set of goals [Burmeister, 1992, Jennings, 2001, Lesser, 1999, Subrahmanian, 2000]. There exists variety of definitions of the notion of agent. The most of researchers adhere to the definition of M. Wooldridge and N.R. Jennings: agent is a hardware or software-based computer system that enjoys the following properties:

– *autonomy*: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal states;

– *social ability*: agents interact with other agents (and possibly humans) via some kind of *agent-communication language*;

– *reactivity*: agents perceive their environment, and respond in a timely fashion to changes that occur in it;

– *pro-activeness*: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative.

MAS is usually specified as a concurrent system that consists of autonomous, reactive and internally-motivated agents acting in a decentralized environment. One key reason of the growth of interest in MAS is that the idea of an agent as an autonomous system, capable of interacting with other agents, is a naturally appealing one for software designers. Reactive agents do not have representations of their external environment and act using a stimulus response type of behavior, they respond to the present state of the environment. Robots are examples of artificial agents. ARCHON [Burmeister, 1992] is the most well-known commercial system constructed on the basis of the conception of agents and designed for controlling the process of manufacturing articles.

The history of the development of the agent theory began with the problem of modeling the properties of living systems and dates back to the work of W. Pitts and W. MacCulloch on formal neurons, John von Neumann on self-reproducing automata, A.N. Kolmogorov on the theory of complexity, H. von Forster and Ilya Prigogine on the theory of self-organization, William Ashby on models of homeostasis, W.G. Walter on reactive robots, and John Holland on genetic algorithms.

Designing and building agent systems is difficult. They have all the problems associated with building traditional distributed, concurrent systems and have the additional difficulties that arise from having flexible and sophisticated interactions between autonomous problem-solving components. The big question then becomes how effective MASs can be designed and implemented.

The core concept of multi-agent systems is interaction; it is the foundation for cooperative or competitive behavior among several autonomous agents. Agent interactions are established through exchanging information in the form of messages that specify the desired performatives of interacting agents. Formalization of the concept of interaction as a method of transmitting messages from one dispatcher to several recipients through a transmission environment derives from Claude Shannon’s work on communication theory [Shannon, 1948]. The subsequent formalization of this concept employed the theory of speech acts [Searle, 1986]. In this theory communication between agents is considered as a form of behavior, since particular types of sentences of natural language are of the nature of actions (called speech acts) and presume a “rational effect.” Speech acts form the basis of the communication languages used by agents (e.g., the KQML and FIPA-ACL languages [ARPA, 1993, Finin, 1997, FIPA, 2002]); these languages also define the sets of permissible speech acts and semantics associated with these sets.

A set of interrelated messages forms conversations in which agents play different roles as a functions of their individual or common goals. Conversations in multiagent systems are based on interaction protocols that define all possible flows of conversations. Agent system can operate if agents have a common understanding of the possible types of messages, then they must know which messages they can expect in a particular situation and what they may do when they got some message. So messages exchanged between agents in some multi-agent system need to follow some standard patterns which are described in agent interaction protocol. Protocols play a central role in agent communication; they specify the sequences in which messages should be arranged in agent’s interactions.

At this time, there are two major technical obstacles to the widespread adoption of multiagent technology: 1) the lack of systematic methodologies enabling designers to clearly specify and structure their applications as MASs and 2) the lack of widely available MAS toolkits. Flexible sets of tools are needed that enable designers to specify an agent’s problem-solving behavior and to specify agents interaction, and then visualize and debug the behavior of the agents and the entire MAS. That is because designing agent systems is difficult enough: they have all the problems associated with building traditional distributed, concurrent systems and have the additional difficulties that arise from having sophisticated interactions between autonomous problem-solving components.

The other difficulty of existing methodologies of MAS designing consists in follows. In the majority of MAS models autonomous behavior of agents is described in terms of belief, desires and intentions (BDI) [Burmeister, 1992], and their communications are specified in terms of protocols which have no a direct relation with the first formalism. The behavior of agents is described in terms of formalisms of a high level abstraction (such as temporal logic), but the communication is specified in the concepts close to realization. The independent behavior of agents in the majority of models of multi-agent systems is described by means of formalisms of high level abstractness, but the communication is specified by the concepts close to realization. The difference of levels of the description does not allow model communications between agents at the level in which their independent autonomous behavior is described. This problem arises because of absence of agent models that unify all aspects of local behavior and the communications (BDI-behavior and communication), and will be suitable for automated implementation. The main reason for the absence of such a unifying model is that there is no general conceptual basis that combines all the abstractions related to agents.

Representation languages from the theory of agents are used to overcome these methodological difficulties. Such languages are based on some formalism and a programming system that together specify semantics of the language used in programming agents. Though ever newer agent programming languages have been proposed in the literature, only some of them are entirely intelligible from the semantic point of view.

This paper explores the possibility of applying existing formal theories of description of distributed and concurrent systems to interaction protocols for real-time multi-agent systems. In particular it is shown how the language PRALU [Zakrevskij, 1996, Zakrevskij, 1999, Zakrevskij, 2001] proposed for description of parallel logical control algorithms and rooted in the Petri net formalism, can be used to describe agent interaction protocols. The described approach can be used for modeling complex, concurrent conversations between agents in a multi-agent system. It can be used to define protocols for complex conversations composed of a great number of simpler conversations. With the language PRALU it is possible to express graphically concurrent characteristics of a conversation, to capture the state of a complex conversation during runtime, and to reuse described conversation structure for processing multiple concurrent messages. It is demonstrated with a known example of English auction [FIPA, 2002] on how to specify an agent interaction protocol using considered means. Finally, using PRALU language we can verify some key behavioral properties of our protocol description that is facilitated by the use of existing software for the language PRALU [Cheremisinov, 1986, Cheremisinova, 2002, Zakrevskij, 1999].

Then, we introduce a methodology of programming agents in MAS; it is based on the language PRALU. When more complicated, multilayered and concurrent conversations take place among groups of agents, PRALU approach that we use appears to offer advantages over the colored Petri net techniques that are proven to provide the most powerful mechanism for specifying interaction protocols up until now [Bai, 2004]. Further, PRALU as the agent-oriented programming language has such an advantage that its semantics is based on logic formalism, this language allows a simple realization. We show how the behavior of MAS can be simulated entirely in the language PRALU. The offered methodology is based on two-block architecture of organization of the description and realization of MAS that consists of the block of synchronization and the functional block. The first one coordinates performance of parallel processes of agent program, that is, it controls the agent behavior. The second part operates data and carries out calculations connected with complex information structures. The distinctive feature of the methodology is that it allows separating development of the synchronizing part of agent programs from the functional one.

## An Agent-Oriented Model

The application domains of MASs are getting more and more complex. Firstly, many current application domains of MASs require agents to work in changing environment (or world) that acts on or is acted on by the system. A closed system is one that has no environment; it is completely self-contained in contrast to an open (uncertain) system, which interacts with its environment. Any real system is open. The MAS must decide what to do and develop a strategy in order to achieve its assigned goals, within open environment. For this, the MAS must have a representation or a model of the environment within which it evolves. The environment is composed of situations. A situation is the complete state of the world at an instant of time.

The application of multi-agent systems to real-time environments can provide new solutions to very complex and restrictive systems such as real-time systems. A suitable method for real-time multi-agent system development must take into account the intrinsic characteristics of systems of this type. As a rule they are distributed, concurrent systems with adaptive and intelligent behavior. For agent-based systems to operate effectively, they must understand messages that have a common ontology underlying them. Understanding messages that refer to ontology can require a considerable amount of reasoning on the part of the agents, and this can affect system performance.

Taking the view of agents as practical systems the predominant approach to specifying agents has involved treating them as intentional systems that may be understood by attributing to them mental states such as beliefs, desires, and intentions [Wooldridge, 1995]. Following this idea, a number of approaches for formally specifying agents have been developed, which are capable of representing the following aspects of an agent-based system:

– the beliefs that agents have – the information they have about their environment, which may be incomplete or incorrect;

– the goals that agents will try to achieve;

– the actions that agents perform and the effects of these actions;

– the ongoing interaction that agents have – how agents interact with each other and their environment over time.

BDI agents are systems that are situated in a changing environment, receive continuous perceptual input, and take actions to affect their environment, all based on their internal mental state. In practical terms, beliefs represent the information an agent has about the state of its environment. It is updated appropriately after each action. The desires denote the objectives to be accomplished, including what priorities are associated with the various objectives. Intentions reflect the actions that must be fulfilled to achieve the goal (the rules to be fired).

Multi-agent conversations are built upon two components:

– agent communication language;

– interaction protocol.

There are a number of agent communication languages, such as Knowledge Query and Manipulation Language (KQML) [Finin, 1997] and Agent Communication Language (ACL), proposed by the Foundation for Intelligent Physical Agents (FIPA) [FIPA, 2002], and others designed for special purposes and that are like mentioned ones. These agent communication languages specify a domain specific vocabulary (ontology) and the individual messages that can be exchanged between agents.

Interacting agents should comply with an interaction protocol in order to engage permissible sequences of message exchange. When agent sends a message it can expect a response to be among a set of messages indicated by the accepted protocol. The interaction protocol can be assigned by the designer of the multi-agent system otherwise an agent needs to indicate the protocol that it wants to follow before it starts to interact with other members of the system.

## Agent Interaction Protocols

Interaction protocols [FIPA, 2002] specify the sequences in which messages should be arranged in agent interactions. Protocol constrains number of sequences of allowed messages for each agent at any stage during a communicative interaction, i.e. it describes some standard pattern messages exchanged between agents need to follow. By the very nature of protocols as public conventions, it is desirable to use a formal language to represent them. When agents are involved in interactions where no concurrency is allowed, most conversation protocols are traditionally specified as deterministic finite automata (DFA) [Pitt, 1999] of which there are numerous examples in the literature. DFA consists of a set of states, an input alphabet and a transition function, which maps every pair of states and input to the next state. In the context of interaction protocols, the transitions specify the communicative actions to be used by the various agents involved in a conversation. A protocol based on such a DFA representation determines a class of well-formed conversations. Conversations that are defined in this way have a fixed structure that can be laid down using some kind of graphical representation. However, they are not enough expressive to model complex interactions, especially those with some degree of concurrency.

Protocols can be represented as well in a variety of other ways. The simplest is a message flow diagram, as used by FIPA [FIPA, 2002]. More complex protocols will be better represented using a UML sequence (Unified Modeling Language) [Booch, 1999] and AUML (Agent UML) [Bauer, 2001, Odell, 2001, Winikoff, 2005], interaction diagram, statechart [Harel, 1998] and Colored Petri Net (CPN) [Bai, 2004, Jensen, 1992].

UML is one of the currently most popular graphical design languages that is de facto standard for the description of software systems. AUML extends UML sequence diagrams to support the specification of interactions between agents. The advantage of AUML is its visual representation as well as statechart [Harel, 1998], but unfortunately, AUML suffers from two issues. Firstly, the notation itself is not formally and precisely defined; and secondly, tool support for AUML is largely non-existent [Cost, 1999]. AUML diagrams only offer a semi-formal specification of interactions. This weakness may generate several problems. Indeed, the lack of formal semantics in AUML, such as in UML, can lead to several incoherences in the description of a MAS's behavior. It is difficult, especially in the case of complex MAS, to detect this kind of defects [Poutakidis, 2002].

A CPN model of a system describes the states, which the system may be in, and the transitions between these states. CPNs provide an appropriate mathematical formalism for the description, construction and analysis of distributed and concurrent systems. CPNs can express a great range of interactions in graphical representations and well-defined semantics, and allow formal analysis and transformations [Bai, 2004, Murata, 1989]. By using CPNs, an agent interaction protocol can be modeled as a net of components, which carry the protocol structure. Using CPNs to model agent interaction protocol, the states of an agent interaction are represented by CPN places. Each place has an associated type determining the kind of data that the place may contain. Data exchanges between agents are represented by tokens, and the colors of tokens indicate the data value of the tokens. The interaction policies of a protocol are carried by CPN transitions and their associated arcs. A transition is enabled if all of its input places have tokens, and the colors of these tokens can satisfy constraints that are specified on the arcs. A transition can be fired, which means the actions of this transition can occur, when this transition is enabled. When a transition occurs, it consumes all the input tokens as computing parameters, conducts conversation policy and adds new tokens into all of its output places. After a transition occurs, the state (marking) of a protocol has been changed and a protocol will be in terminal state when there is no enabled or fired transition.

There are a number of works using Petri Nets or CPNs to model agent interaction protocols [Cost, 1999, Poutakidis, 2002], there have been also some works on the investigation of flexibility, robustness and extensibility of protocols [Hutchison, 2002]. Today, only Petri Net models are considered to be one of the best ways to model agent interaction protocols. However the notion of an agent executing an action with Petri Net is not explicit in the notation [Paurobally, 2002]. A different PN can be assigned to each agent role, raising questions about how the entire protocol is inferred and the reachability and consistency of shared places. If a single Petri net is partitioned for each role, this leads to a complex diagram where a partition is required for each agent identified. Furthermore, alternative actions and states either agree or reject but not both, cannot be expressed in standard Petri nets.

It is necessary for any protocol itself to be correct and verifiable. If it is not correct then the agents that follow it may perform contradictory and unexpected actions leading to possible breakdown of the interaction. The central problem of the verification of interactions that take place in open (not being cooperative) systems is the problem of conformance inspection between behavior of agents and interaction protocol. That is the protocol must be understandable by all agents of the system and they must behave according to this protocol. Thus a language used to represent MAS protocol must be clear enough to allow means for protocol verification. We consider that a language for developing protocols is needed which can ideally meet the following requirements:

1) provide a graphical representation for ready perception of structure by MAS designer;

2) have an unambiguous formal specification with clear semantics for verification;

3) be close to an executable language for implementation purposes;

4) for relative tractability, maintain a propositional form for a formal language;

5) provide well-defined program logic for ensuring complete protocols and validating the properties of a protocol;

6) exhibit enough expressiveness for agent interactions and nested interactions.

Keeping in mind complex systems characterized by complex interaction, asynchronism and concurrency we propose to use for the purposes of their description a special language PRALU [Zakrevskij, 1996, Zakrevskij, 1999, Zakrevskij, 2001] satisfying all mentioned requirements. It has its background in the Petri net theory (expanded nets of free choice – EFC–nets investigated by Hack [Hack, 1972]) but possesses special means for keeping track of the current states of the conversation, receiving messages and initiating responses. The language combines properties of "cause-effect" models with Petri nets. It is intended for a wide application in engineering practice and is well suited for representation of the interactions involved in concurrent system; synchronization among them and then it is simple enough for understanding. The language PRALU supports hierarchical description of the algorithms that is especially important in the case of complex systems. At last a powerful software has been developed that provides correctness verifying, simulation, hardware and software implementation of PRALU-descriptions [Cheremisinov, 1986, Cheremisinova, 2002]. The review of obtained results in this field one can find in [Cheremisinova, 2002].

## PRALU Language

Any algorithm in PRALU consists of sequences of operations to be executed in some pre-determined order. Two basic operations are used in PRALU: acting "$\to A$" and waiting "$- p$" operations. The first one changes the state of the object under control, whereas the second one is passive waiting for some event without affecting anything. In simple case $A$ and $p$ are conjunctive terms, so acting and waiting operations can be interpreted as waiting for event $p = 1$ and producing event $A = 1$. But $A$ can be understood too as a formulae defining operations to be performed and $p$ as a predicate defining condition to be verified. For example, acting and waiting operations could be specified by the expressions such as

$-(a > b + c)$ and $\to (a := b + c)$.

The sequences consisting of action and waiting operations are considered to be linear algorithms. For instance, the following expression means: wait for $p$ and execute $A$, execute $B$, then wait for $q$ and execute $C$:

$-p \to A \to B - q \to C$.

In general, a control algorithm can be presented as an unordered set of chains $\alpha_j$ in the form

$\mu_j : - p_j L_j \to \nu_j$ ,

where $L_j$ is a linear algorithm, $\mu_j$ and $\nu_j$ denote the initial and the terminal chain labels being some subsets of integers from the set $M = \{1, 2, ..., m\}$: $\mu_j, \nu_j \subset M$ and the expression "$\to \nu_j$" presents the transition operation: to the chains with labels from $\nu_j$.

Chains can be fulfilled both serially and in parallel. The order in which they should be fulfilled is determined by the variable starting set $N_t \subseteq M$ (its initial value $N_0 = \{1\}$ as a rule): a chain $\alpha_j = \mu_j : - p_j L_j \to \nu_j$ (that was passive) is activated if $\mu_j \subseteq N_t$ and $p_j = 1$. After executing the operations of the algorithm $L_j$, $N_t$ gets a new value $N_{t+1} = (N_t \setminus \mu_j) \cup \nu_j$. The algorithm can finish when some terminal value of $N$ is reached (one-element as a rule), at which time all chains became passive. But the algorithms can also be cyclic; they are widely used when describing production processes.

When the conditions $\mu_j \subseteq N_t$ ($|\mu_j| > 1$) and $p_j = 1$ are satisfied for several chains simultaneously these chains will be fulfilled concurrently. On the contrary chains with the same initial labels are alternative (only one of them can be fulfilled at a time), they are united in a sentence with the same label as will be shown below.

Thus PRALU allows concurrent and alternative branching, as well as merging concurrent and converging alternative branches. These possibilities are illustrated with the following examples of simple fragments [Zakrevskij, 2000]:

| Concurrent branching | Merging concurrent branching | Alternative branching | Converging alternative branching |
|---|---|---|---|
| 1: ...$\to$2.3 | 2: ...$\to$4 | 1: – $a$...$\to$2 | 2: ...$\to$4 |
| 2: ... | 3: ...$\to$5 | – $\overline{a}$ ...$\to$3 | 3: ...$\to$4 |
| 3: .. | 4.5: ... | | 4: ... |

There exist in PRALU two syntactic constraints on chains that restrict concurrent and alternative brunching. If some chains are united in the same sentence (they have the same initial labels) they should have orthogonal predicates in the waiting operations opening the chains:

$(i \neq j)$ & $(\mu_i \cap \mu_j \neq \varnothing) \to (p_i$ & $p_j = 0)$.

The other constraint is similar to the corresponding condition specific for extended nets of free choice (Hack [Hack, 1972]):

$(i \neq j)$ & $(\mu_i \cap \mu_j \neq \varnothing) \to (\mu_i = \mu_j)$.

PRALU language has some more useful properties that can be usefull for description of complex interaction protocols.

1. PRALU algorithms can be expressed both in graphical and symbolic forms.

2. PRALU language permits hierarchical description of protocols. The two-terminal algorithms (having the only initial and the only terminal chain labels) may be used as blocks (invoked as complex acting operations) in hierarchical algorithms.

3. In PRALU there exist some additional interesting operations that can be useful when describing interaction protocol. Among those are suppression operations that provide response on special events that can take place outside or within control systems. Suppression operations

"$\to *$", "$\to * \gamma$", "$\to ' \gamma$" (where $\gamma \subseteq M$)

interrupt the execution of all concurrently executed algorithm chains (in the case of "$\to *$"), only those ones mentioned in $\gamma$ (in the case of "$\to * \gamma$") or are not mentioned in $\gamma$ (in the case of the operation "$\to ' \gamma$"). These operations break the normal algorithm flow.

4. In addition to logical variables, arithmetic variables are also used in the PRALU language. In particular, among arithmetic operations ought to be mentioned the following operations are introduced:

– timeout operation "– $n$" – delay for $n$ unit times;

– counting operations that count event occurrences:

“($x = n$)” – assignment of a natural value $n$ to a multivalued variable $x$;

“($x$ +)” and “($x$ –)” – assignment of unit positive and unit negative increment in value;

“–($x = n$)” – awaiting the start of an event: the value of $x$ is equal to $n$.

## Representing Agent Interaction Protocols in PRALU

As an example of an interaction protocol let consider English auction [FIPA, 2002]. The auctioneer seeks to find the market price of a good by initially proposing a price below that of the supposed market value and then gradually raising the price. Each time the price is announced, the auctioneer waits to see if any buyers will signal their willingness to pay the proposed price. As soon as one buyer indicates that it will accept the price, the auctioneer issues a new call for bids with an incremented price and continues until no buyers are prepared to pay the proposed price. If the last price that was accepted by a buyer exceeds the auctioneer's (privately known) reservation price, the good is sold to that buyer for the agreed price. If the last accepted price is less than the reservation price, the good is not sold.

In the case of the auction there are participants of two types: the Auctioneer and Buyers. So, we have two kinds of interaction protocols – those of Auctioneer and of Buyers. So, we have two kinds of interaction protocols – those of Auctioneer and of Buyers. The last participants are peer and should be described with identical interaction protocols.

Interaction protocol as a whole can be represented in PRALU as three complex acting operations – blocks that are exchanging with values of logical variables, only such variables are mentioned in them. Each block has some sets of inputs and outputs that are enumerated in brackets following the block name (the other variables of a block are its internal). Initialization of a block algorithm is depicted by the fragment such as “→∗Buyer”. The operation Buyer exists in as many copies as the number of participants of the auction, the copies differ in their indexes only.

The modeling of the process of auction begins with the execution of “Main_process” triggering event that initiates the interaction protocol execution. Here the processes Auctioneer and Buyer$_n$s are executed concurrently. For the sake of simplicity we limit the number of buyers to two. The process Auctioneer starts with sending the first message (start_auction) that is waited by others participants to continue communication.

Below PRALU description of the auction interaction protocol is shown. Here we show the only block Buyer$_n$, but for real application (intending to simulate the process of auction, for example) we should have as many proper copies as it has been used (in our case – two). Here through “′var1” the inversion of the Boolean variable var1 is denoted.

```
Main_process ()
1: → 2.3.4
2: →∗Auctioneer → 5
3: →∗Buyer_1 → 6
4: →∗Buyer_2 → 7
5.6.7: → .
Buyer_n (start_auction, price_proposed, end_auction / accept_price_n, not_understand)
1: – start_auction → 2
2: – price_proposed →∗Decide ( / decision_accept, decision_reject) → 3
```

– end_auction $\to$ .

3: – decision_accept $\to$ accept_price$_n$ $\to$ 4

– decision_reject $\to$ ′accept_price$_n$ $\to$ 4

– not_understand $\to$ 4

4: – timeout $\to$ 2

Auctioneer (accept_price$_1$, accept_price$_2$, not_understand / start_auction, price_proposed, end_auction)

1: $\to$ start_auction $\to$ 2

2: $\to$ ′accept_price$_1$.′accept_price$_1$.′not_understand $\to$∗Price_propose (/price_proposed) $\to$ 3

3: – not_understand $\to$ 2

– accept_ price$_1$ $\to$ 4

– accept_ price$_2$ $\to$ 4

–′not_understand.′accept_ price$_1$.′accept_ price$_2$ $\to$ end_auction $\to$
∗Is_reservation_price_exceeded ( / is_exceeded) $\to$ 6

4: $\to$′price_proposed.′win$_1$.′win$_2$ $\to$ 5

5: – accept_ price$_1$ $\to$win1 $\to$ 2

– accept_ price$_2$ $\to$win2 $\to$ 2

6: – is_exceeded $\to$good_sold $\to$ 7

– ′is_exceeded $\to$′good_sold $\to$ 7

7: $\to$ .

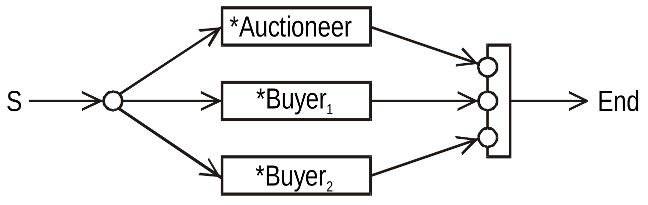
Fig. 1 depicts graphical schemes of three mentioned PRALU-blocks: Main_process, Auctioneer and Buyer$_n$.

It is assumed that all unformalized operations are referred to as acting operations that set values of logical variables assigned to them. For example, Buyer's operation "Decide" decides for accepting or rejecting the announced price. Depending on adopted decision, it outputs true value of logical variable "decision_accept" or of logical variable "decision_reject". In a similar, Auctioneers operation "Price_propose" proposes an initial price or increments the charged price outputting true value of logical variable "price_proposed"; the operation "Is_reservation_price_exceeded" verifies if the price accepted by a buyer exceeds the auctioneer's reservation price outputting true or false value of logical variable "is_exceeded".

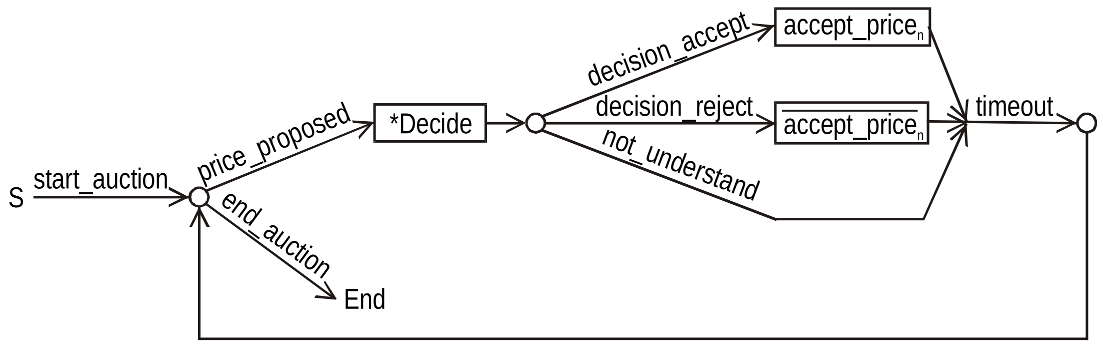
The operation "–timeout" (where "timeout" is an integer number) means waiting for "timeout" unit times before doing something followed it. The operation "$\to$." is interpreted as the transition to an end of the process described by the block. When the processes of Auctioneer and all of Buyers reach their end in the Main_process the transition to its finish is executed.

## BDI Architecture for the Language PRALU

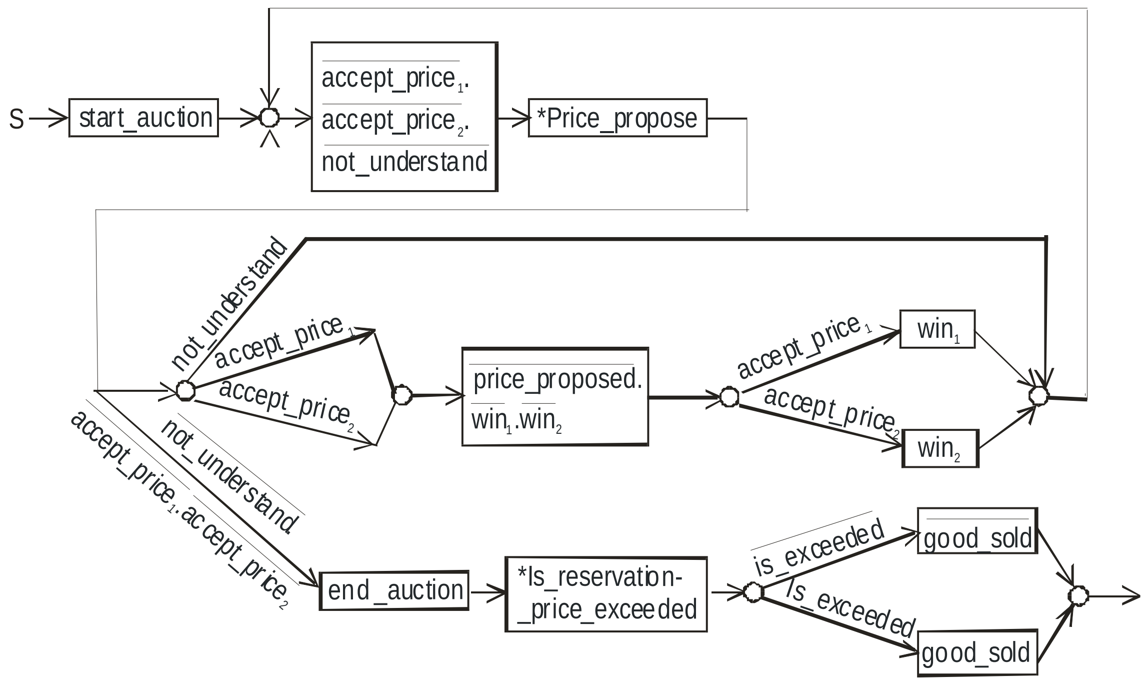
In practical programming, an agent is a well-formed entity incorporated in a computer system designed to perform flexible, independent actions for the purpose of attaining specified goals. Agents differ from ordinary software by the complexity of the interaction and communication scenarios. Any physical multiagent system is open, i.e., the agents function in a varying environment, which affects the agents and varies as a consequence of their operations. The system must arrive at decisions so as to attain the objectives assigned to it, and for this purpose it must possess a model of the environment in which its behavior evolves.



Main_process ()



Buyer$_n$ (start_auction, price_proposed, end_auction / accept_price$_n$, not_understand)



Auctioneer (accept_price$_1$, accept_price$_2$, not_understand / start_auction, price_proposed, end_auction)

Fig. 1. English auction interaction protocol in PRALU

Over the last decade, the specification and application of BDI agents (belief, desire, intention) [Burmeister, 1992] have received a great deal of attention. Within the BDI architecture agents are associated with beliefs (typically about the environment and other agents), desires or goals to achieve, and intentions or plans to act upon to achieve its desires. These three components completely set a state of the "mind" of the agent.

In programming terms, beliefs represent knowledge (information) a BDI agent has about the state of the environment; that is updated after each action. The desires denote the objectives to be accomplished, taking into account their priorities. Intentions reflect the actions that should be fulfilled to achieve the goal (the rules to be fired). Interaction protocols reduce search space of possible decisions, owing to limited range of answers to possible messages for the given situation. Ontology [Gruber, 1993] as a formal specification of terms of a subject domain and relations between them is almost equivalent to the notion of semantics of programming language.

In our case ontology will define the terms of BDI architecture in terms of the language PRALU. It is possible to consider, that BDI agent consists of the sets of beliefs *B*, plans *P*, situations *E*, actions *A* and intentions *I*. When the agent notes a change in its environment, it decides, that there an event takes place representing some situation from *E*. Registration of the event consists in changing a state of the agents “mind”: a choice of some belief from *B*. According to it and the desire (defined by some plan from *P*) the agent intends to execute some intention representing some sequence of actions from *A* to achieve the goal chosen from *I*. Thus, planned actions are defined by the chosen plan from *P*. After they were carried out, the current situation of the environment is changed.

In traditional parallel programming languages the basic concepts are data and calculation control. Data are represented by values of variables, and the control is specified by a set of processes which transform local memory states defining variable values. The concept of the intellectual agent introduces new ideas of data manipulation in parallel programming. Agent’s states are set by more complex information structures of predicates logic of the first order or modal logic. Calculations as transformations of memory states are controlled by protocols specifying communications between agents.

The program of a BDI agent is a set of plans defining actions by means of which the agent should reach a goal of its functioning. The plan consists of head labels, a body and tail labels. The body of the plan is a sequence of actions, by means of which the agent should reach a goal of its functioning, and conditions which the agent should check up. The head and tail labels of the plan symbolize intentions. Critical concept for the behavior of the agent is the concept of active intention. The plan will be carried out only when all intentions from its head are active. After executing the plan all intentions from the head become inactive and intentions from the tail quite the contrary become active. The current set of active intentions always is not empty, the body of a plan and a set of tail intentions can be empty. Such a model of BDI agent can be easily described on PRALU: a plan has a direct analogy with a chain $\alpha_j$ (in the form $\mu_j : - p_j L_j \to \nu_j$, $L_j$: “$-p \to A -q \to B -r \to C$...”). The action leading to some goal can be described by acting operation “$\to g$” (done *g*) that may be executed, if some belief has appeared correct. Check of such condition is described by waiting operation “– a” (happens a). An agent carries out the action following waiting operation only after the moment when “belief *a*” becomes true.

## Methodology of Programming Agents on PRALU

We propose the natural methodology of designing agent program based on splitting the program into two parts: synchronization and functional blocks (Fig. 2). The first block coordinates performance of parallel processes of the agent program, that is, it controls the agent behavior. The synchronization block should be described on PRALU language. The functional block operates data and carries out calculations connected with complex information structures (formulas of predicate or modal logic). This block is realized in programming language.

Such a splitting is carried out at the level of the project statement of MAS. The functional part is presented by predicates that describe memory states of the agent program (or/and external environment) or prescribe performance of some actions. In PRALU-description the appropriate logical variable is introduced for each predicate, setting true value to this variable starts the process of the predicate calculation. At the stage of verification of logical consistency it is better to interpret predicates as independent logical variables.

For example, the predicate *Decide* is used in the description of the behavior of agent Buyer. In the synchronization block in Fig. 3 it is represented by the Boolean variable *Decide*. In the functional block, the meaning of a predicate is indicated by pseudo-code, which shows the use of the variable *Decide* and rules for computing two other Boolean variables, *decision_accept* and *decision_reject*.

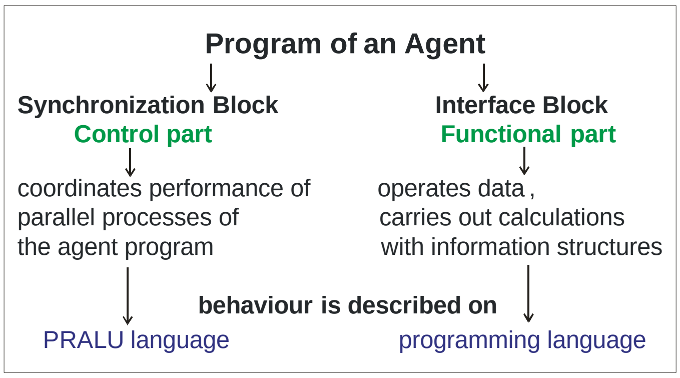

Program of an Agent
Synchronization Block
Control part
coordinates performance of parallel processes of the agent program
Interface Block
Functional part
operates data, carries out calculations with information structures
behaviour is described on
PRALU language
programming language

Figure 2. Splitting an agent program

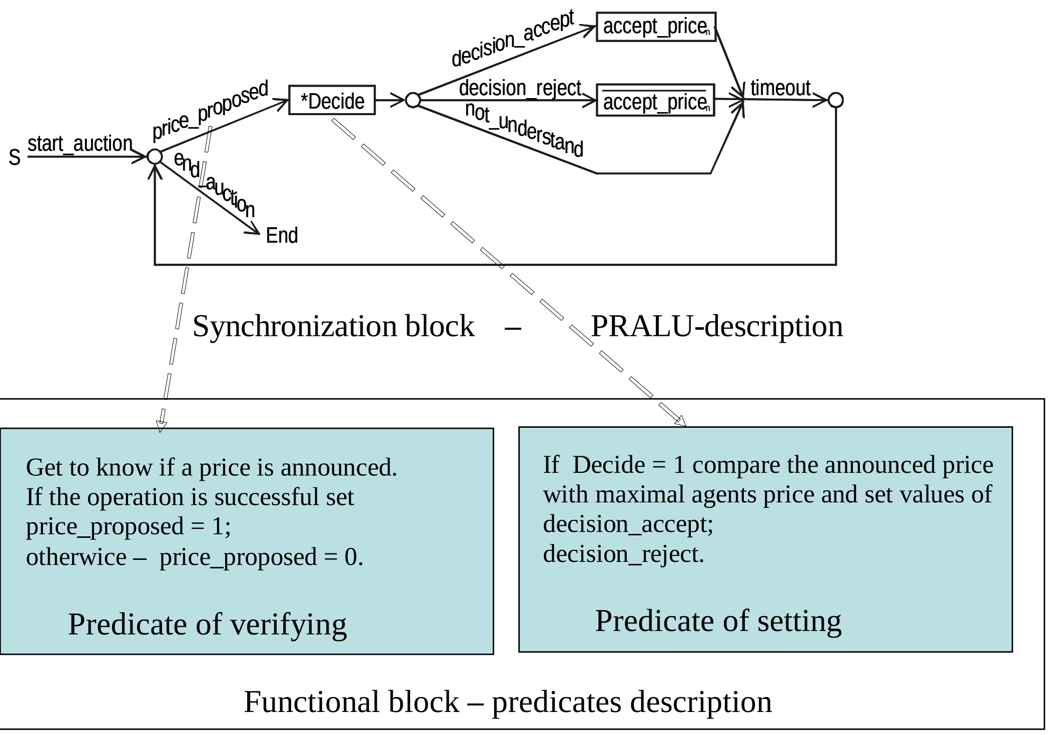

S
start_auction
price_proposed
*Decide
end_auction
End
decision_accept
decision_reject
not_understand
accept_price
accept_price
timeout
Synchronization block – PRALU-description
Get to know if a price is announced.
If the operation is successful set
price_proposed = 1;
otherwice – price_proposed = 0.
Predicate of verifying
If Decide = 1 compare the announced price with maximal agents price and set values of
decision_accept;
decision_reject.
Predicate of setting
Functional block – predicates description

Figure 3. The program of the agent Buyer in English auction

Such an approach allows separating development of the synchronization part of the PRALU-description from the functional one. When designing MAS, the implementation of the functional part can be delayed concentrating designer's attention on working out the synchronization block implementing interaction protocol. This may significantly simplify the process of verifying the logical consistency of the behaviors of agents.

## Program Implementation of PRALU-Descriptions

Programming in PRALU, the designer of the agent specifies sets of its belief, plans and intentions. The process of designing an agent program consists of the following stages.

1. Splitting at functional level the specification of MAS to be designed into synchronization and functional parts.

2. Development and analysis of PRALU-description of the synchronization part of MAS, being based on informal specification of its interaction protocol.

3. Verification of logic consistency of the MAS behavior by checking correctness and simulation of PRALU-description. Elimination of mistakes found out.

4. Program implementation of the functional part of MAS.

5. Development of computer program implementing behavior of MAS: translating PRALU-description of MAS on programming language and binding it with the programs of the functional part.

6. Testing of the generated programs.

The use of the PRALU language as a tool for representing the specifications of agents as well as the executability of these specifications makes the process of designing a multiagent system a structured process. All formal tasks may be implemented in the language and software tools for the implementation of these tasks are available. Moreover, there is the powerful theory supported with software tools for verification, simulation, hardware and software implementation of the descriptions of algorithm entity incorporated in a computer system in the PRALU language [Cheremisinov, 1986, Cheremisinova, 2002, Zakrevskij, 1999].

In the process of program implementation of PRALU-description it is translated on the intermediate language applied in the simulation program [Cheremisinov, 1986] too. The program that is generated by the PRALU compiler consists of two interacting blocks – 1) calculation of responses, and 2) control of sensors and servo mechanisms of the system. The control structure of the response computation program consist in an infinite cycle involving input of signals from the agent's sensors that constitute its beliefs into the internal memory, computation of the agent's responses from the values of these signals, and output of signals to the agent's servo mechanisms. All communication related to control and to data between the response computation block, sensor control block, and the servo mechanisms are input into the representation of the PRALU description implemented in the intermediate language.

To implement a parallel algorithm on the only processor [Cheremisinov, 1986], it is necessary to order the operations of the intermediate language by means of the PRALU intermediate language planner, which has its own properties and methods. The properties of the planner consist in the presence of two queues, a queue of waiting and a queue of ready branches. The methods of a program planner consist in triggering, halting, and pausing subprocesses. In the planner's initial state the queue of ready branches contains the first operation of the algorithm while the queue of waiting branches is empty. The program automatically orders the sequence of execution of operations that are in the process of being executed and there is no need for preliminary planning.

In each cycle the planner extracts the next branch from the queue of ready branches, executes a corresponding chain of algorithm in PRALU, and places all the branches that must be executed following the given branch into the waiting queue. If the planner discovers that the queue of ready branches is empty, (1) it transfers the contents of the queue of waiting branches to the queue of ready branches, rendering the queue of waiting branches empty; and (2) executes information exchange with the environment. Such a sequence of execution of chains of an algorithm in PRALU guarantees that operations of the initial algorithm that may be executed in parallel are all of the same length. Thus, the program implementation of PRALU possesses the semantics of measurable time that satisfy a rendezvous condition [Cheremisinov, 2008].

In the two-block model of the program, the predicates of the functional part are considered as a “supplementary tool” of sensors and servo mechanisms of the agent that generates logical signals or is triggered by a signal. If the results of the execution of computational operations must be taken into account in the control process, the supplementary tool will generate logical signals expressly for this goal. In the program that controls the sensors and servo mechanisms, this supplementary tool is described in the form of expressions in the ordinary programming language by the designer of the agent’s program.

This strategy of programming of agents is especially simple where the Forth programming language is used as a platform [Moore, 1974]. In this case it is only necessary to define a single word of Forth for each operator of an intermediate language.

The principle of implementation of the Forth language, i.e., threaded code, may be used for programming of agents in a C (or C++) language platform. The threaded code, a strategy used in the Forth language, consists in representation of a program in which nearly all elements are denoted by procedure calls (some procedures presuppose that data are located behind them in the threaded code). A stack is used to implement procedure calls; to distinguish it from a data stack, such a stack is referred to as a return stack and is used for storage of procedure data. Queues of ready and waiting branches of the intermediate language are implemented by means of return and data stacks, respectively.

The transformation of a representation of an algorithm in PRALU into a C program is executed by a single-pass text translator that converts the operators of the intermediate language into procedure calls in C language syntax. The efficiency of a program constructed in this way may be estimated using as an example the implementation of one of the most difficult operations of the intermediate language, the pause operation. The job executed by this operation consists in storage of the address of the next procedure call in the data stack and a transfer to execution of a procedure the address of which is extracted from the return stack. The C compiler constructs a fragment of code consisting of only two machine instructions for execution of these operations.

When programming agents on the base of a C language platform, procedures defined by predicates should be written in C using the descriptions like in Fig. 3 as a model.

Most of the well-known systems of logical programming of agents use a computation model from the Prolog language [Endriss, 2004]. Prolog is based on first-order predicate logic and the objects which the program manipulates are symbols without any meaningful interpretation. Execution of a Prolog program involves a process called unification and consists in scanning a database of facts and selecting values that satisfy some query. Unification is based on syntactic identity. In order to find a set of solutions, Prolog traverses a search tree, tests a set of variants, most of which do not occur in a solution, and, if unsuccessful, returns to a previous state and tests a different branch. For complex problems this search process requires great expenditure of memory and time, which is one reason for the computational inefficiency of Prolog.

In comparing the proposed PRALU-based logical programming system with other systems, for example, Prolog-based systems [Endriss, 2004], the improved computational efficiency of PRALU-based systems is apparent; in fact, the computational efficiency of PRALU-based systems is comparable with that of C language programs.

Software tools for automatic development, debugging, hardware and software implementation have been developed for the PRALU language [Cheremisinov, 1986, Cheremisinov1, 1986, Cheremisinova, 2002]. In particular, a modeling program may be used to display the behavior of new protocols written in PRALU. Display of behavior consists in exhibiting the control states and states of the variables of an algorithm in PRALU. The control states are exhibited by distinguishing operations in the text of the algorithm that are being executed at a given time (display of activity points), as it is done in most debuggers.

## Conclusion

This paper has addressed the need for formalized and more expressive logical and graphical methodologies for specifying (and then validating) interaction protocols in multi-agent systems. Towards this, it was proposed to use the formal language PRALU intended for the representation of complex interactions involved in concurrent system, being in need of synchronization among these interactions. It was demonstrated as well how PRALU algorithms could be used for the specification of multi-agent interaction protocols by the example of English auction. In favor of using the language PRALU is the existence of a great deal of methods and software developed for simulation of PRALU algorithms as well as for their hardware and software implementation.

The methodology is suggested that ensures structuring the process of MAS designing on the base of PRALU language by means of separating calculation part of MAS specification from its control part. That allows the designer to concentrate on more complex stage of MAS designing – its interaction protocol. It is shown that language PRALU is very suitable for specifying interaction protocols of MAS and for implementation of suggested methodology. The obtained results are directed towards the usage when designing parallel/distributed technical systems too.

## Bibliography

[ARPA, 1993] Specification of the KQML Agent-Communication Language, ARPA Knowledge Sharing Initiative, External Interfaces Working Group, July 1993.

[Bai, 2004] Quan Bai, Minjie Zhang Khin, Than Win. A Coloured Petri Net Based Approach for Multi-agent Interactions. In: The 2nd International Conference on Autonomous Robots and Agents, Palmerston North, New Zealand, 13 – 15 December 2004.

[Bauer, 2001] B. Bauer, J.P. Müller, J. Odell. Agent UML: A Formalism for Specifying Multiagent Interaction. In: Agent-Oriented Software Engineering, Ed. P. Ciancarini and M. Wooldridge, Springer-Verlag, Berlin, 2001.

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

[Burmeister, 1992] B. Burmeister, K. Sundermeyer. Cooperative problem-solving guided by intensions and perception. In: Demazean Decentralized A.I. 3. Ed. E. Werner and Y. Amsterdam, The Netherlands, North Holland, 1992.

[Cheremisinov, 1986] D.I. Cheremisinov. Implementation of parallel algorithms on a microprocessor. In: Programming, 1986, No 1 (in Russian).

[Cheremisinov1, 1986] D.I. Cheremisinov. LyaPAS-M Programming System Based Package for the Development of Microprocessor Programs. In: Upravlyayushchie sistemy i mashiny, 1986, No 1 (in Russian).

[Cheremisinov, 2008] D. Cheremisinov, L. Cheremisinova. Formalization of interaction events in Multi-agent Systems. In: Information Technologies & Knowledge (IJ ITK), 2008, Vol. 2, No. 2.

[Cheremisinova, 2002] L.D. Cheremisinova. Realization of parallel algorithms for logical control. Institute of Engineering Cybernetics of NAS of Belarus, Minsk, 2002 (in Russian).

[Cost, 1999] R. Cost. Modeling Agent Conversations with Coloured Petri Nets. In: Working Notes of the Workshop on Specifying and Implementing Conversation Policies, Seattle,Washington, 1999.

[Endriss, 2004] U. Endriss, N. Maude, F. Sadri, F. Toni. Logic-Based Agent Communication Protocols. In: Advances in Agent Communication, Ed. F. Dignum, Laboratory Notes in Artificial Intelligence, 2004, Vol. 2922.

[Finin, 1997] F. Finin, Y. Labrou, J. Mayfield. KQML as an agent communication language. In: Software Agents, Ed. J.M. Bradshaw, MIT Press, 1997.

[FIPA, 2002] Foundation for Intelligent Physical Agents (FIPA), Communicative Ac Library Specification, 2002; http://www.fipa.org/specs/fipa00037.

[Gruber, 1993] T.R. Gruber. A Translation Approach to Portable Ontologies. In: Knowledge Acquisition, 1993, Vol. 5, No 2.

[Hack, 1972] M. Hack. Analysis of production schemata by Petri nets. Project MAC-94, Cambridge, 1972.

[Harel, 1998] D. Harel, M. Politi. Modeling reactive systems with statecharts. McGraw-Hill, 1998.

[Hutchison, 2002] J. Hutchison, M. Winikoff. Flexibility and Robustness in Agent Interaction Protocols. In: Proceedings of the 1st International Workshop on Challenges in Open Agent Systems, Bologna, Italy, 2002.

[Jennings, 2001] N. R. Jennings. An Agent-Based Approach for Building Complex Software Systems. In: Communications of the ACM, 2001, Vol. 44, No 4.

[Jensen, 1992] K. Jensen. Colored Petri Nets – Basic Concepts. In: Analysis Methods and Practical Use, Springer-Verlag, Berlin, 1992, Vol. 1.

[Lesser, 1999] V. Lesser. Cooperative Multiagent Systems: A Personal View of the State of the Art. In: IEEE Trans. on Knowledge and Data Engineering, 1999, Vol. 11, No 1.

[Moore, 1974] C.H. Moore. FORTH: A New Way to Program a Mini-computer. In: Astro. Astrophys. Suppl., 1974, Vol. 5.

[Murata, 1989] T. Murata. Petri Nets: Properties, Analysis and Applications. In: Proceedings of the IEEE, 1989, Vol. 77, No 4.

[Odell, 2001] J.J. Odell, H.V.D. Parunak, B. Bauer. Representing Agent Interaction Protocols in UML. In: Agent-Oriented Software Engineering, Ed. P. Ciancarini and M. Wooldridge, Springer-Verlag, Berlin, 2001.

[Paurobally, 2002] S. Paurobally, J. Cunningham. Achieving Common Interaction Protocols in Open Agent Environments. In: Proceedings of 1st International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS'02), Bologna, Italy, July 2002.

[Pitt, 1999] J. Pitt, A. Mamdani. Protocol-based Semantics for an Agent Communication Language. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-1999), Morgan Kaufmann Publishers Inc., San Francisco, USA, 1999.

[Poutakidis, 2002] D. Poutakidis, L. Padgham, and M. Winikoff. Debugging Multi-Agent Systems Using Design Artefacts: the Case of Interaction Protocols. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi Agent Systems, Bologna, Italy, 2002.

[Searle, 1986] J.R. Searle. What is a Speech Act. In.: Novoe v zarubezhnoy lingvistike, 1986, issue 17 (in Russian).

[Shannon, 1948] C.E. Shannon. A Mathematical Theory of Communication. In: Bell Syst. Tech. J., 1948, Vol. 27.

[Subrahmanian, 2000] V.S. Subrahmanian, P. Bonatti, J. Dix et al. Heterogeneous Agent Systems. MIT Press, 2000.

[Winikoff, 2005] M. Winikoff. Towards Making Agent UML Practical: A Textual Notation and a Tool. In: Proceedings of the Fifth International Conference on Quality Software, September 19–20 2005,.

[Wooldridge, 1995] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. In: The Knowledge Engineering Review, 1995, 10(2).

[Zakrevskij, 1996] A.D. Zakrevskij. Parallel logical control algorithms: verification and hardware implementation. In: Computer Science Journal of Moldova, 1996, Vol. 4, No 1.

[Zakrevskij, 1999] A.D. Zakrevskij. Parallel algorithms for logical control. Minsk, Institute of Engineering Cybernetics of NAS of Belarus, 1999 (in Russian).

[Zakrevskij, 2000] A. Zakrevskij, V. Sklyarov. The Specification and Design of Parallel Logical Control Devices. In: Proceedings of PDPTA'2000, June, Las Vegas, USA, 2000.

[Zakrevskij, 2001] A. Zakrevskij, L. Zakrevski. Algorithms for logical control: their description, verification and hardware implementation. In: Proceedings of PDPTA'01, Las Vegas, 2001, Vol. 2.

## Authors' Information

***Dmitry Cheremisinov, Liudmila Cheremisinova*** *– The United Institute of Informatics Problems of National Academy of Sciences of Belarus, Surganov str., 6, Minsk, 220012, Belarus, Tel.: (10-375-17) 284-20-76, e-mail: cher@newman.bas-net.by, cld@newman.bas-net.by*

# SOME APPROACHES FOR SOFTWARE SYSTEMS ANALYSES AND ITS UPGRADE PREDICTION

Igor Karelin

*Abstract: This paper proposes and discusses a new type of tool for Linux based software systems analysis, testing and optimization as well as the new approach which is based on this tool and will help to define the best moment for executing of effective and smart software upgrade in automatic mode on active and online system. The best moment means one when the software upgrade will cause the minimal services losses.*

*The presented tool is called CAP (Characterization of Availability and Performance) and provides engineers with an instrument for systems performance tuning, profiling and issues investigation as well as stress and load testing. The described system operates with over 150 Linux based system parameters to optimize over 50 performance characteristics. The paper discusses the CAP tool's architecture, multi-parametric analysis algorithms, and application areas. Furthermore, the paper presents possible future work to improve the tool and extend it to cover additional system parameters and characteristics.*

*The prediction of the software upgrade (SU) best moment mentioned above is supposed to be made on basis of performance and availability statistics got from the CAP tool.*

*Keywords: Software Systems Analyses, Linux Servers, Telecommunication System, Performance, Availability, Serviceability, Software Upgrade Prediction.*

*ACM Classification Keywords: C.4 Computer Systems Organization - Performance of Systems - Reliability, availability, and serviceability.*

## Introduction

The telecommunications market is one of the fastest growing industries where performance and availability demands are critical due to the nature of real-time communications tasks with requirement of serving thousands of subscribers simultaneously with defined quality of service. Before Y2000, telecommunications infrastructure providers were solving performance and availability problems by providing proprietary hardware and software solutions that were very expensive and in many cases posed a lock-in with specific vendors. In the current business environment, many players have come to the market with variety of cost-effective telecommunication technologies including packed data technologies such as VoIP, creating server-competitive conditions for traditional providers of wireless types of voice communications. To be effective in this new business environment, the vendors and carriers are looking for ways to decrease development and maintenance costs, and decrease time to market for their solutions.

Since 2000, we have witnessed the creation of several industry bodies and forums such as the Service Availability Forum, Communications Platforms Trade Association, Linux Foundation Carrier Grade Linux Initiative, PCI Industrial Computer Manufacturers Group, SCOPE Alliance, and many others. Those industry forums are working on defining common approaches and standards that are intended to address fundamental problems and make available a modular approach for telecommunication solutions, where systems are built using well defined hardware specifications, standards, and Open Source APIs and libraries for their middleware and applications [Haddad, 2006] ("Technology Trends" and "The .org player" chapters).

The Linux operating system has become the de facto standard operating system for the majority of telecommunication systems. The Carrier Grade Linux initiative at the Linux Foundation addresses telecommunication system requirements, which include availability and performance [Lehrbaum, 2002].

The performance of Linux servers running in mission critical environments such as telecommunication networks becomes a critical attribute. Its importance is growing due to incorporated high availability approaches, especially for servers requiring five and six nines availability. With the growing number of requirements that Linux servers must meet in areas of performance, security, reliability and serviceability, it is becoming a difficult task to optimize all the architecture layers and parameters to meet the user needs. Linux servers also require different approaches to optimization to meet specific constraints of their operating environment, such as traffic type and intensity, types of calculations, memory, CPU and IO use.

There are many examples of Linux-based, carrier-grade platforms used for a variety of telecommunication server nodes. Depending on the place and functionality of the particular server node in the telecommunication network infrastructure, there can be different types of loads and different types of performance bottlenecks. Many articles and other materials are devoted to questions like "how will Linux-based systems handle performance critical tasks?" In spite of the availability of carrier-class solutions, the question is still important for systems serving a large amount of simultaneous requests, e.g. WEB Servers [Haddad, 2001] as Telecommunication specific systems.

Telecommunication systems such as wireless/mobile networks have complicated infrastructures implemented, where each particular subsystem solves its specific problem. Depending on the problem, the critical systems' resource could be different. For example, Dynamic Memory Allocation could become a bottleneck for Billing Gateway, Fraud Control Center (FCC), and Data Monitoring (DMO) [Haggander, Lundberg, 2000] even in SMP architecture environment. Another example is WLAN-to-WLAN handover in UMTS networks where TCP connection reestablishment involves multiple boxes including HLR, DHCP servers and Gateways, and takes significant time (10–20 sec.) which is absolutely unacceptable for VoIP applications [Korhonen, 2004]. A similar story occurred with TCP over CDMA2000 Networks, where a bottleneck was found in the buffer and queue sizes of a BSC box [Mattar et al., 2007]. The list of the examples can be endless.

If we consider how the above examples differ, we would find out that in most cases performance issues appear to be quite difficult to deal with, and usually require rework and redesign of the whole system, which may obviously be very expensive. The performance improvement by itself is quite a well known task that is being solved by the different approaches including the Clustering and the Distributed Dynamic Load Balancing (DDLB) methods [Nehra et al., 2007]; this can take into account load of each particular node (CPU) and links throughput. However, a new question may arise: "Well. We know the load will be even and dynamically re-distributed, but what is the maximum system performance we can expect?" Here we are talking not about performance problems, but about performance characterization of the system. In many cases, people working on the new system development and having performance requirements agree on using prototyping techniques. That is a straightforward but still difficult way, especially for telecommunication systems where the load varies by types, geographic location, time of the day, etc. Prototyping requires creation of an adequate but inexpensive model which is problematic in described conditions.

The author of this paper is working in telecommunication software development area and hence tends to mostly consider problems that he faces and solves in his day-to-day work. It was already said that performance issues and characterization are within the area of interest for a Linux-based system developer. Characterization of performance is about inexpensive modeling of the specific solution with the purpose of predicting future system performance.

What are the other performance-related questions that may be interesting when working in telecommunications? It isn't just by chance I placed the word Performance close to Availability; both are essential characteristics of a modern telecommunication system. If we think for a moment about the methods of achieving of some standard level of availability (let's say the five- or six- nines that are currently common industry standards), we will see that it is all about redundancy, reservation, and recovery. Besides specific requirements to the hardware, those methods require significant software overhead functionality. That means that in addition to system primary functions, it should provide algorithms for monitoring failure events and providing appropriate recovery actions. These algorithms are obviously resource-consuming and therefore impact overall system performance, so another problem to consider is a reasonable tradeoff between availability and productivity.

Let us consider some more problems related to telecommunication systems performance and availability characterization as well as its overall analysis that are not as fundamental as those described above, but which are still important.

Performance profiling. The goal of performance profiling is to verify that performance requirements have been achieved. Response times, throughput, and other time-sensitive system characteristics should be measured and evaluated. The performance profiling is applicable for release-to-release testing.

Load testing. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload. The load testing subjects the system to varying workloads to evaluate the performance behaviors and ability of the system to function properly under these different workloads. The load testing also could be applicable on design stage of a project to choose the best system architecture and ensure that requirements will be achieved under real/similar system workloads [Ong et al., 2005], [Shende et al., 2007].

Stress testing. The goal of stress testing is to find performance issues and errors due to low resources or competition for resources. Stress testing can also be used to identify the peak workload that the system can handle.

Performance issue investigation. Any type of performance testing in common with serious result analysis could be applicable here. Also in some cases, snapshot gathering of system characteristics and/or profiling could be very useful.

Performance Tuning. The goal of performance tuning is to find optimal OS and Platform/Application settings, process affinity, and schedule policy for load balancing with the target of having the best compromise between performance and availability. The multi-objective optimization algorithm can greatly reduce the quantity of tested input parameter combinations.

The problem of issues investigation deserves special attention. As it was already said modern telecommunication systems have extremely difficult architecture – they are distributed in relation to either hardware or software. The growth of a system leads to the increase of data level required for comprehensive systems state analysis and supervision and life cycle description. The majority of this information is represented as log files – text files consisting of time-stamped status and error messages detailing the operational history of a given piece of software.

As we can see system state and lifecycle are described by the huge amount of jumbled data produced and distributed by multiple software units. These data represent the behavior of each unit on a long time scale. The problem is that during system analysis (failure investigation for ex.) the search for information is time-consuming. Maintenance engineer should manually filter and sort data from all the log files to assess system state and its behavior. The situation can be more complicated in case of log files allocation in different network nodes (multiboard systems). Fortunately the software tool called Log Merger has already been developed and presented by the author and helps the raw data to be automatically collected, analyzed, reordered and filtered according to engineer's needs in each particular case [Karelin et al., 2008].

This long introduction was intended to explain why the author started to look at performance and availability characterization problems, their applications to Linux-based, carrier-grade servers. Further along in this paper, we will consider existing approaches and tools and share some more approaches that were successfully used by the author in his work.

## Overview of the Existing Methods for Characterization of Performance and Availability

A number of tools and different approaches for performance characterization exist and are available for Linux systems. These tools and approaches target different problems and use different techniques for extracting system data to be analyzed as well, and support different ways to represent the results of the analysis. For the simplest cases of investigating performance issues, the standard Linux tools can be used by anyone. For example, the GNU profiler gprof provides basic information about pieces of code that are consuming more time to be executed, and which subprograms are being run more frequently than others. Such information offers understanding where small improvements and enhancements can give significant benefits in performance. The corresponding tool kprof gives an opportunity to analyze graphical representation gprof outputs in form of call trees, e.g. comprehensive information about the system can be received from /proc (a reflection of the system in memory). Furthermore, for dealing with performance issues, a variety of standard debugging tools such as instrumentation profilers (oprofile which is a system-wide profiler), debuggers kdb and kgdb, allowing kernel debugging up to source code level as well as probes crash dumps and many others are described in details in popular Linux books [Best, 2005]. These tools are available and provide a lot of information. At the same time a lot of work is required to filter out useful information and to analyze it. The next reasonable step that many people working on performance measurement and tuning attempt to do is to create an integrated and preferably automated solution which incorporates in it the best features of the available standalone tools.

Such tools set of benchmarks and frameworks have appeared as the well known package lmbench, which is actually a set of utilities for measurement of such characteristics as memory bandwidth, context switching, file system, process creating, signal handling latency, etc. It was initially proposed and used as a universal performance benchmarking tool for Unix-based systems. There were several projects intended to develop new micro benchmark tools on the basis of lmbench in order to improve measurement precision and applicability for low-latency events by using high-resolution timers and internal loops with measurement of the average length of events calculated through a period of time, such as Hbench-OS package [Haddad, 2006]. It is noticeable that besides widely used performance benchmarks, there are examples of availability benchmarks that are specifically intended to evaluate a system from the high availability and maintainability point of view by simulating failure situations over a certain amount of time and gathering corresponding metrics [Haggander, Lundberg, 2000].

Frameworks to run and analyze the benchmarks were the next logical step to customize this time-consuming process of performance characterization. Usually a framework is an automated tool providing additional customization, automation, representation, and analysis means on top of one or several sets of benchmarks. It makes process of benchmarking easier, including automated decision making about the appropriate amount of cycles needed to get trustworthy results [Wright et al., 2005].

Therefore, we can see that there are a number of tools and approaches one may want to consider and use to characterize a Linux-based system in terms of performance. Making the choice we always keep in mind the main purpose of the performance characterization. Usually people pursue getting these characteristics in order to prove or reject the assumption that a particular system will be able to handle some specific load. So if you are working on a prototype of a Linux-based server for use as a wireless base site controller that should handle e.g. one thousand voice and two thousand data calls, would you be happy to know from the benchmarks that your system is able to handle e.g. fifty thousand TCP connections? The answer isn’t trivial in this case. To make sure we have to prepare a highly realistic simulated environment and run the test with the required number of voice and data calls. It is not easy, even if the system is already implemented, because you will have to create or simulate an external environment that is able to provide an adequate type and amount of load and which behaves similarly to a live wireless infrastructure environment. In case you are in the design phase of your system it is just impossible. You will need to build your conclusion on the basis of a simplified system model. Fortunately, there is another approach—to model the load, not the system. Looking at the architecture we can assume what a specific number of voice and data calls will entail in the system in terms of TCP connections, memory, timers, and other resources required. Having this kind of information, we can use benchmarks for the identified resources and make the conclusion after running and analyzing these benchmarks on the target HW/SW platform, without the necessity of implementing the application and/or environment. This approach is called workload characterization [Avritzer et al., 2002].

Looking back to the Introduction section, we see that all the target questions of Performance and Availability characterization are covered by the tools we have briefly looked through above. At the same time there is no single universal tool that is able to address all these questions. Further in the paper we are introducing the Characterization of Performance and Availability (CAP) tool that combines the essential advantages of all the approaches considered in this chapter and provides a convenient framework to perform comprehensive Linux-based platforms characterization for multiple purposes.

## CAP Architecture

### 1. Experimental Approach.

Anyone who is trying to learn about the configuration of Linux servers running in mission-critical environments and running complex applications systems will have to address the following challenges:

- An optimal configuration, suitable for any state of environmental workload, does not exist;
- Systems are sophisticated: Distributed, Multiprocessor, Multithreaded;
- Hundreds or even thousands of configuration parameters can be changed;
- Parameters can be poorly documented, so the result of a change for a group of parameters or even single parameter can be totally unpredictable.

Based on the above described conditions, an analytical approach is scarcely applicable, because a system model is not clear. An empirical approach could be more applicable to find optimal configuration of a system, but only experimental evaluation can be used to validate the correctness of optimal configuration on a real system. The heart of CAP is the concept of the experimentation. A single experiment consists of the following parts:

- Input parameters: let us call them Xs. Input parameters are all what you want to set up on a target system. Typical examples here are Linux kernel variables, loader settings, and any system or application settings.
- Output parameters: let us call them Ys. Output parameters are all that you want to measure or gather on a target system: CPU and Memory utilization, any message throughput and latency, system services bandwidth, and more. Sources for Ys could be: /proc file system, loaders output, profiling data, and any other system and application output.

- Experiment scenarios: An experiment scenario is a description of actions which should be executed on target hosts.

Typical experiment scenario follows a sequence of action: setup Xs that can't be applied on-the-fly (including execution required actions to apply such Xs like restart node or processes, down and up network interfaces, etc.), then setup Xs that can be applied on-the-fly and loader's Xs, start loaders, next setup Xs like: schedule policy, priority, CPU binding etc., finally collect Ys such as CPU/Memory usage, stop loaders, and overall statistics.

Every scenario file may use preprocessor directives and S-Language statements. S-Language is a script language which is introduced specifically for the project. Both preprocessor and S-Language are described in more detail following. One of the important parts of the scenario executor is a dynamic table of variables. Variable is a pair-variable name and variable value. There are two sources of the variables in the dynamic table:

- Xs (Input variables). They are coming from an experiment.
- Ys (Collected variables). They are coming from remote hosts.

In the case of input variables, the names of the variables are provided by the XML-formatted single experiment file. In the case of collected variables, the names of the variables are provided by scripts or other executables on the target hosts' side. Whenever the same executable could be run on many different hosts, a namespace mechanism is introduced for the variable names. A host identifier is used as a namespace of the variable name.

## 2. Overview of CAP Architecture

The simplified CAP architecture is presented in the Figure 1.

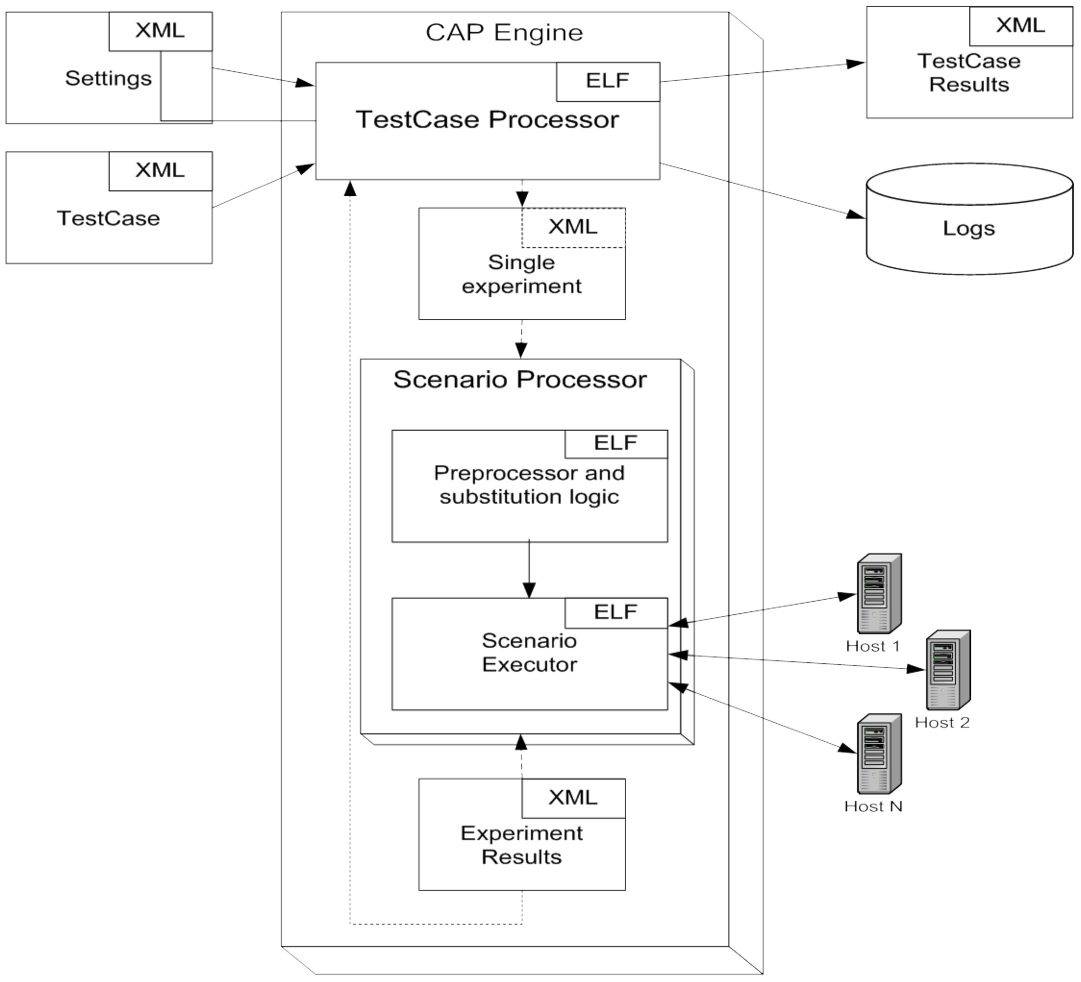


Figure 1. CAP architecture

A test case consists of a set of experiments. Each experiment is essentially a set of Xs that should be used while executing a scenario. Set of Xs within one Test Case boundaries is constant and only values of these Xs are variable. Each experiment is unambiguously linked to a scenario. A scenario resides in a separate file or in a group of files. The CAP engine overall logic is as follows:

- Takes a test case file;
- For each experiment in the test case, performs the steps below;
- Selects the corresponding scenario file;
- Executes all the scenario instructions using settings for fine tuning the execution logic;
- Saves the results into a separate result file;
- Saves log files where the execution details are stored.

Test Cases. The basic unit of test execution is an experiment. A single experiment holds a list of variables, and each variable has a unique name. Many experiments form a test case. The purpose of varying Xs' values depends on a testing goal. Those Xs' names are used in scenarios to be substituted with the values for a certain experiment.

Scenarios. A scenario is a description of actions which should be executed on target hosts in a sequence or in parallel in order to set up Xs' values in accordance with Test Case/Experiments and gather the values of Ys. The solution introduces a special language for writing scenarios. The language simplifies description of actions that should be executed in parallel on many hosts, data collection, variable values, substitution, etc.

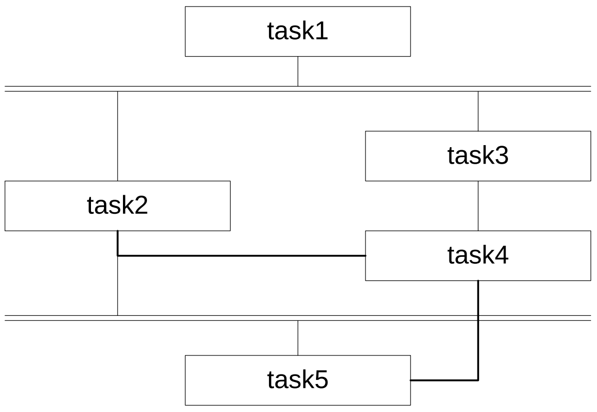
Results. The results files are similar to Test Case files. However, they contain set of Ys coming from target hosts and from input experiment variables (Xs).

The scenario processor consists of two stages, as depicted in the figure above. At the bottom line there is a scenario executor which deals with a single scenario file. From the scenario executor's point of view, a scenario is a single file; however, it is a nice feature to be able to group scenario fragments into separate files. To support this feature the preprocessor and substitution logic is introduced in the first stage. The standard C programming language preprocessor is used at this stage, so anything which is supported by the preprocessor can be used in a scenario file. Here is a brief description of the C preprocessor features which is not a complete one and is given here for reference purposes only:

- Files inclusion;
- Macro substitutions;
- Conditional logic.

Summarizing, the complete sequence of actions is as follows: The single experiment from the Test Case is applied to the scenario file. It assumes macro substitutions of the experiment values (Xs), file inclusions, etc. The scenario executor follows instructions from the scenario file. While executing the scenario some variables (Ys) are collected from target hosts. At the end of the scenario execution, two files are generated: a log file and a results file. The log file contains the report on what was executed and when, on which host, as well as the return codes of the commands. The results file contains a set of collected variables (Xs and Ys).

The main purpose of the introduced S-Language is to simplify a textual description of the action sequences which are being executed consecutively and/or in parallel on many hosts. Figure 2 shows an example of a task execution sequence.



(a) Block diagram

```
TITLE ‘‘Scenario example’’
#include "TestHosts.incl"
#include "CommonDef.incl"
/* Task1 */
WAIT ssh://USER:PASSWD @ {X_TestHost} "set_kernel_tun.sh SEM \
{X_IPC_SEM_KernelSemmni} {X_IPC_SEM_KernelSemopm}"
PARALLEL
{
/* Task2 */
COLLECT @ X_TestHost "sem_loader -d {X_Duration} \
-r {X_IPC_SEM_NumPVOps} -t {X_IPC_SEM_LoaderNumThreads}"
SERIAL
{
/* Task3&4 */
COLLECT @ {X_TestHost} "get_overall_CPUusage.pl -d
{X_Duration}"
COLLECT [exist(Y_Memory_Free)] @ {X_TestHost} \
"get_overall_Memoryusage.pl -d X_Duration"
}
}
/* Task5 */
NOWAIT [IPC_iteration >1] @ {X_TestHost} ‘‘cleanup_timestamps.sh’’
```

(b) S-Language Code

Figure 2. Scenario example

ExecCommand is a basic statement of the S-Language. It instructs the scenario executor to execute a command on a target host. The non-mandatory Condition element specifies the condition when the command is to be executed. There are five supported command modifiers: RAWCOLLECT, COLLECT, WAIT, NOWAIT, and IGNORE. The At Clause part specifies on which host the command should be executed. The At Clause is followed by a string literal, which is the command to be executed. Substitutions are allowed in the string literal.

## 3. CAP Agents

We are going to refer to all software objects located on a target system as CAP agents. The server side of CAP does not contain any Performance and Availability specifics, but it is just intended to support any type of complex testing and test environment. Everybody can use CAP itself to implement their own scenario and target agents in order to solve their own specific problem related to the system testing and monitoring.

CAP agents, which are parts of the CAP tool, are the following:

- Linux service loaders;
- Xs adjusting scripts;
- Ys gathering scripts.

Firstly let us consider loaders as one of the most interesting part of CAP agents. Every loader receives a command line argument which provides the number of time slices a base interval (usually one second) is going to be divided into. For example: <loader> --rate 20 means that a one-second interval will be divided into 20 slices. At the very beginning of each time slice, a loader calls a function which performs a required functionality/load. The functionality depends on a loader type. For example, the file system loader performs a set of file operations, while the shared memory loader performs a set of shared memory operations, and so on. If the required functionality has been executed before the end of the given time slice, a loader just sleeps until the end of the slice. If the functionality takes longer than a time slice, the loader increments the corresponding statistic's counter and proceeds. There are several common parameters for loaders.

Input:

- The first one is a number of threads/processes. The main thread of each loader's responsibility is to create the specified number of threads/processes and wait until they are finished. Each created thread performs the loader-specific operations with the specified rate.
- The second common thing is the total loader working time. This specifies when a loader should stop performing operations.

- Loaders support a parameter which provides the number of operations per one “call of F() functionality.” For example, a signal loader takes an argument of how many signals should be sent per time slice. This parameter, together with the number of threads, rate, and number of objects to work with (like number of message queues), gives the actual load.
- Besides that, each loader accepts specific parameters (shared memory block size in kilobytes, message size, and signal number to send, and so on).

Output:

- Number of fails due to the rate requirement not being met.
- Statistics—figures which are specific for a loader (messages successfully sent, operations successfully performed, etc.)

The loaders implementation described above allows not only the identification of Linux service breakpoints, but also—with help of fine rate/load control—the discovery of the service behavior at different system loads and settings. The following loaders are available as part of CAP tool:

- IPC loaders: Shared memory loader; Semaphore loader; Message queues loader; Timer loader.
- CPU loaders: CPU loader; Signal loader; Process loader.
- IP loaders: TCP loader; UDP loader.
- FS loader (File & Storage);
- Memory loader.

One more CAP agent is PPA (Precise Process Accounting). PPA is a kernel patch that has been contributed by Motorola. PPA enhances the Linux kernel to accurately measure user/system/interrupt time both per-task and system wide (all stats per CPU). It measures time by explicitly time-stamping in the kernel and gathers vital system stats such as system calls, context switches, scheduling latency, and additional ones. More information on PPA is available from the PPA SourceForge web site: http://sourceforge.net/projects/ppacc/.

The Ys gathering is used in the SU best moment definition and will be discussed further. Now we will show how to use CAP.

## CAP in Use

Let us assume that you already have the CAP tool and you have decided to use it for your particular task. First of all, you will have to prepare and plan your experiments:

- Identify list of input parameters (Xs) that you would like to set up on the target. That could be kernel parameters, a loader setting like operational rate, number of processes/threads, CPU binding,etc.
- Identify list of output parameters (Ys) that you would like to measure during an experiment: everything you want to learn about the system when it is under a given load.

If we are talking about Linux systems, you are lucky then, because you can find in the CAP toolset all the necessary components for the CAP agent that have been already implemented: set scripts for Xs, get scripts for Ys, and predefined scenarios for Linux’s every service. If you are not using Linux, you can easily implement your own scripts, scenarios, and loaders. When you have identified all the parameters that you want to set up and measure, you can move on to plan the experiments to run.

### Performance Profiling

- Set up predefined/standard configuration for the kernel and system services.

- Setup loaders to generate the workload as stated in your requirements.
- Perform experiments.
- Check whether the measured data shows that requirements are met.

### Load Testing

- Set up predefined/standard configuration for the kernel and system services.
- Use a long experiment duration.
- Mix the workload for all available services.
- Vary workloads.
- Vary the number of threads and instances for the platform component.
- Analyze system behavior.
- Check that Ys are in valid boundaries.

### Stress Testing

- Use a high workload.
- Operate by the loader with the target to exhaust system resources like CPU, memory, disk space, etc.
- Analyze system behavior.
- Check that all experiments are done and Ys are in valid boundaries.

### Performance tuning

- Plan your experiments from a workload perspective with the target to simulate a real load on the system.
- Vary Linux settings, process affinity, schedule police, number of threads/instances, etc.
- Analyze the results in order to identify the optimal configuration for your system. Actually we believe a multi-objective optimization can be very useful for that. This approach is described in more detail later on.

### System Modeling

- Take a look at your design. Count all the system objects that will be required from an OS perspective, like the number of queues, TCP/UDP link, timers, semaphores, shared memory segment, files, etc.
- Examine your requirements in order to extrapolate this on every OS service workload.
- Prepare test case(s).
- Analyze the obtained results to understand whether your hardware can withstand your design.

Another kind of system modeling is described in the following chapter and uses the Ys gathering functionality of the CAP tool.

## Software Upgrade Prediction Concept

As it was stated in the introduction the described tool is expected to be used by the author for the best moment of software upgrade definition. For a start I will explain why it is important to know the best moment for SU.

One of the most important requirement for software systems which work in 24/7 mode is the possibility of online SU. The simple duplication of the system might seem to be the possible way to meet this requirement. Nowadays it is rather widespread solution – all the components of the system are set in pairs – one active and one standby. If the active element is lost the standby one takes over all the functionalities.  However this solution does not take into account the time on setting all the connections between the standby and other active elements of the system as well as can be extremely expensive. The introduced method is aimed not on the upgrade of the system without services losses but on the SU best moment (from the services losses point of view) prediction – note that the SU term does not surely mean the whole system upgrade – it could be some components upgrade.

The principal scheme of the method is showed in the Figure 3.

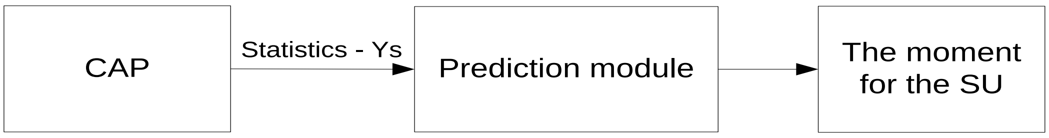


Figure 3. Principle scheme of the SU moment definition

The prediction module will perform following actions:

- Process the statistic information and get probability distribution functions for all characteristics (density functions) $-P_i(Y_i), i \in 1...n$, n is the number of characteristics $Y_i$. The methods here could be, for example:
  - Testing of statistical hypothesis
  - Approximation by known distribution functions
  - Monte Carlo method
- Perform a correlation analysis to identify dependencies - influence coefficients calculation. The methods here could be, for example:
  - Elementary methods (parallel comparison of samples etc.)
  - Complex methods (dispersion analysis methods, theory of correlation and regression methods, multivariate analysis methods etc.)
- The third and the most labour-intensive step of the prediction module development is the statistic simulation which task is to get the functional model which will describe the real system behavior (in terms of system modules load). The book “Statistic simulation” by Ermakov S.M. will be helpful here [Ermakov, 1976]. Object identification, which is considered in [Черноруцкий, 2004], could be an alternative to the complex method of the statistic simulation and in case of known type of functional model operator will turn into the parametric identification which could be simply implemented.
- The final step of the prediction module is the parameter optimization of the model got in the previous step. The goal of this optimization is to determine the moment of time when the whole system, or the module being updated, have the least load in terms of services usage. The teaching aid mentioned above, [Черноруцкий, 2004], fully describes this task solution.

Let us go into details of the third and the fourth steps. As it was said the goal of the third step is to get the operator F which will describe the real system behavior in terms of services availability and its components loads. Using the parametric identification the operator F will get the vector $Y$ of n stochastic parameters (from Statistic Module), one dependent parameter t (time) - and will return the vector $L$ of m services availabilities

$$F(Y,t) = L, \qquad Y = \begin{Bmatrix} y_1 \\ ... \\ y_n \end{Bmatrix}, \; L = \begin{Bmatrix} l_1 \\ ... \\ l_m \end{Bmatrix}$$

The time (t) is the dependent parameter from the possibility to vary point of view – the task is to determine the best moment [Черноруцкий, 2004].

This operator F will be obtained by minimization of misalignment function

$$\psi(Y,t,P)=\sum_{i=1}^{m}\ \left[L_i(Y,t)-L_{iM}(Y,t,P)\ \right]^2;\quad \left(\sum_{j=1}^{N}\psi(Y_j,t_j,P)\rightarrow \min\right)\Rightarrow P$$

Here $L_i$ is the output of the real system, $L_{iM}$ is the model output and $P$ is the vector of unknown model parameters.

The problem here could become the interference coefficients calculation of the characteristics for the every system component. These coefficients are quite important for the specified task resolution because the connections between different parts of the system often define the services availability (presence). Here we mean systems components interconnection which is the essential part of every distributed system. These problem will be solved by the means of statistical simulation which takes into account the influence of different components on each other [Ермаков и др., 1976].

The fourth step further will be the parametric optimization implementation for the $F$ operator – identification of the moment of time when the $F$ operator will possess the minimal value.

$$\left(F(Y,t)\rightarrow \min\right)\Rightarrow t_{\min}$$

This task will be solved with the help of mathematical programming. It is very important to note that the CAP tool starting and the needed statistics gathering will be done periodically and with high frequency which will enable the prediction module to recreate the system model approximating it to the real system.

## Future Plans for Further Approaches Development

In its current version, the CAP tool allows us to reach the goals we set for ourselves, and has a lot of potential opportunities for further improvements. We realize a number of areas where we can improve this solution to make it more and more applicable for a variety of performance- and availability-related testing. Through real applications of the CAP tool to existing telecommunication platforms, we realized that this approach from the experimental perspective could be very fruitful. However, we noticed that results may vary depending on the overall understanding and intuition of the person performing the planning of the experiments. If a person using CAP does not spend enough time to investigate the nature of the system, she/he may need to spend several cycles of experiments-planning before she/he identifies right interval of system parameters — i.e., where to search for valuable statistical results. This is still valuable, but requires the boring sorting of a significant amount of statistical information. In reality, there may be more than one hundred tunable parameters. Some of them will not have any impact on certain system characteristics; others will certainly have impact. It is not always easy to predict it just from common perspective. An even more difficult task is to imagine and predict the whole complexity of the inter-relationships of parameters. Automation of this process seems to be reasonable here and there is a math theory devoted to this task that we believe could be successfully applied. We are talking about multi-parametric optimization. It is a well described area of mathematics, but many constraints are applied to make this theory applicable for discrete and non-linear dependencies (which are true for most of dependencies we can meet in system tunable parameters area). We are currently looking for numeric approaches for these kinds of multiparametric optimizations—e.g., NOMAD (Nonlinear Optimization for Mixed variables and Derivatives) [Audet, Orbany, 2004], GANSO (Global And Non-Smooth Optimization) [CIAO, 2005], or ParEGO Hybrid algorithm [Knowles , 2005]. A direct search for the optimal parameters combination would take too much time (many years) to sort out all possible combinations, even with fewer than one hundred parameters. Using math theory methods, we will cut the number of experiments down to a reasonable number and shorten the test cycle length in the case of using CAP for load and stress testing purposes. Our discussion in the previous paragraph is about performance tuning, but it is not the only task that we perform using the CAP tool. Another important thing where the CAP tool is very valuable is the investigation of performance and availability issues. Currently, we perform analysis of the results received manually through the use of packages such as MiniTAB, which in many case is time consuming. Our plan is to incorporate statistical analysis methods in the CAP tool in order to allow it to generate statistical analysis reports and to perform results visualization automatically by using GNU GSL or similar packages for data analysis, and such packages as GNUPLOT, LabPlot, or Root for visualization [Galassi et al., 2006] [Brun, Rademakers, 1996].

## Conclusion

In this paper, we have provided an overview of specific performance and availability challenges encountered in Linux servers running on telecommunication networks, and we demonstrated a strong correlation between these challenges and the current trend from Linux vendors to focus on improving the performance and availability of the Linux kernel.

We have briefly described the existing means to address basic performance and availability problem areas, and presented the reason why in each particular case the set of tools used should be different, as well as mentioned that in general the procedure of performance and availability characterization is very time- and resource consuming.

We presented on the need to have a common integrated approach, i.e., the CAP tool. We discussed the tool architecture and used examples that significantly simplify and unify procedures of performance and availability characterization and may be used in any target problem areas starting from Linux platform parameters tuning, and finishing with load/stress testing and system behavior modeling.

After all we presented a new concept of the SU best time prediction on the basis of statistics provided by the CAP tool. This new method can be advantageous for the SU either of existing systems which require 24/7 operational mode or of the systems which are being developed and the decision about their architecture should be made – now there is no need to secure ourselves by duplicating the system – we can just predict the best moment to execute the SU. It is important to note that such approach might not meet the requirement like five or six nines when the systems downtime should be approximately 5 min per year.

All in all the developed tool from one side provides an ability to perform an overall testing and optimization of the system and from the other side – to gather various statistics on the systems components load which are used by the author to predict the best moment of SU.

## Bibliography

[Haddad, 2006] Ibrahim Haddad. Linux and Open Source in Telecommunications. Linux Journal, September 2006. http://www.linuxjournal.com/article/9128.

[Lehrbaum, 2002] Rick Lehrbaum. Embedded Linux Targets Telecom Infrastructure. LINUX Journal, May 2002. http://www.linuxjournal.com/article/5850.

[Haddad, 2001] Ibrahim Haddad. Open-Source Web Servers: Performance on a Carrier-Class Linux Platform. Linux Journal, November 2001. http://www.linuxjournal.com/article/4752.

[Haggander, Lundberg, 2000] Daniel Haggander and Lars Lundberg. Attacking the Dynamic Memory Problem for SMPs. In the 13th International Conference on Parallel and Distributed Computing System (PDCS'2000). University of

Karlskrona/Ronneby Dept. of Computer Science, 2000.
http://www.ide.hk-r.se/~dha/pdcs2.ps.

[Korhonen, 2004] Jouni Korhonen. Performance Implications of the Multi Layer Mobility in a Wireless Operator Networks, 2004. http://www.cs.helsinki.fi/u/kraatika/Courses/Berkeley04/KorhonenPaper.pdf.

[Mattar et al., 2007] Karim Mattar, Ashwin Sridharan, Hui Zang, Ibrahim Matta, and Azer Bestavros. TCP over CDMA2000 Networks : A Cross-Layer Measurement Study. In Proceedings of Passive and Active Measurement Conference (PAM 2007). Louvain-la-neuve, Belgium, 2007.
http://ipmon.sprint.com/publications/uploads/1xRTT_active.pdf.

[Nehra et al., 2007] Neeraj Nehra, R.B. Patel, and V.K. Bhat. A Framework for Distributed Dynamic Load Balancing in Heterogeneous Cluster. Journal of Computer Science v3, 2007.
http://www.scipub.org/fulltext/jcs/jcs3114-24.pdf.

[Ong et al., 2005] Hong Ong, Rajagopal Subramaniyan, and R. Scott Studham. Performance Implications of the Multi Layer Mobility in a Wireless Operator Networks, 2005. Performance Modeling and Application Profiling Workshop, SDSC, http://xcr.cenit.latech.edu/wlc/papers/openwlc_sd_2005.pdf.

[Shende et al., 2007] Sameer Shende, Allen D. Malony, and Alan Morris. Workload Characterization using the TAU Performance System, 2007.
http://www.cs.uoregon.edu/research/paracomp/papers/talks/para06/para06b.pdf.

[Karelin et al., 2008] Karelin I., Gavrilova T., Lyubimov B. “A Log tool for software systems analyses” // “Information Technologies and Knowledge”, Volume 2/2008, Number 4, p. 65
http://www.foibg.com/ibs_isc/ibs-04/IBS-04-p09.pdf

[Best, 2005] Steve Best. LinuxR Debugging and Performance Tuning: Tips and Techniques. Prentice Hall PTR, 2005. ISBN 0-13-149247-0.

[Haddad, 2006] Ibrahim Haddad. Linux and Open Source in Telecommunications. Linux Journal, September 2006. http://www.linuxjournal.com/article/9128.

[Haggander, Lundberg, 2000] Daniel Haggander and Lars Lundberg. Attacking the Dynamic Memory Problem for SMPs. In the 13th International Conference on Parallel and Distributed Computing System (PDCS’2000). University of Karlskrona/Ronneby Dept. of Computer Science, 2000.
http://www.ide.hk-r.se/~dha/pdcs2.ps.

[Wright et al., 2005] Charles P. Wright, Nikolai Joukov, Devaki Kulkarni, Yevgeniy Miretskiy, and Erez Zadok. Towards Availability and Maintainability Benchmarks: A Case Study of Software RAID Systems. In proceedings of the 2005 Annual USENIX Technical Conference, FREENIX Track, 2005.
http://www.am-utils.org/docs/apv2/apv2.htm.

[Avritzer et al., 2002] Alberto Avritzer, Joe Kondek, Danielle Liu, and Elaine J. Weyuker. Software Performance Testing Based on Workload Characterization. In Proceedings of the 3rd international workshop on Software and performance, 2002. http://delivery.acm.org/10.1145/590000/584373/p17-avritzer.pdf.

[Ермаков и др., 1976] Ермаков С. М., Михайлов Г. А. «Курс статистического моделирования» // Главная редакция физико-математической литературы изд-ва «Наука», М., 1976

[Черноруцкий, 2004] Черноруцкий И. Г. «Методы оптимизации в теории управления» // Учебное пособие, изд. Питер, Санкт-Петербург, 2004.

[Audet, Orbany, 2004] Charles Audet and Dominique Orbany. Finding optimal algorithmic parameters using a mesh adaptive direct search, 2004.http://www.optimization-online.org/DB_HTML/2004/12/1007.html.

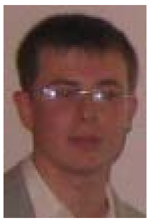
[CIAO, 2005] CIAO. GANSO. Global And Non-Smooth Optimization. School of Information Technology and Mathematical Sciences, University of Ballarat, 2005. Version 1.0 User Manual. http://www.ganso.com.au/ganso.pdf.

[Knowles , 2005] Joshua Knowles. ParEGO: A Hybrid Algorithm with On-line Landscape Approximation for Expensive Multiobjective Optimization Problems. In Proceedings of IEEE Transactions on Evolutionary Computation, Vol. 10, No. 1, 2005. http://ieeexplore.ieee.org/iel5/4235/33420/01583627.pdf .

[Galassi et al., 2006] Mark Galassi, Jim Davies, James Theiler, Brian Gough, Gerard Jungman, Michael Booth, and Fabrice Rossi. GNU Scientific Library. Free Software Foundation, Inc., 2006. Reference Manual. Edition 1.8, for GSL Version 1.8. http://sscc.northwestern.edu/docs/gsl-ref.pdf.

[Brun, Rademakers, 1996] Rene Brun and Fons Rademakers. ROOT – An Object Oriented Data Analysis Framework. In Proceedings of AIHENP conference in Laussanne, 1996.
http://root.cern.ch/root/Publications.htm

## Authors' Information



*Igor Karelin – Bachelor of information technologies. Student of Saint Petersburg State Polytechnic University (master's degree program), Software Engineer, Motorola Software Group; e-mail: ikarus47@mail.ru*

*Major Fields of Scientific Research: telecommunication systems analysis, statistical modeling, parameter optimization.*

# USING RANDOMIZED ALGORITHMS FOR SOLVING DISCRETE ILL-POSED PROBLEMS

Elena G. Revunova, Dmitri A. Rachkovskij

*Abstract: In this paper, we develop an approach for improving the accuracy and speed of the solution of discrete ill-posed problems using the pseudo-inverse method. It is based on a random projection of the initial coefficient matrix. First, a brief introduction is given to least squares and discrete ill-posed problems, approximate matrix decompositions with randomized algorithms, and randomized least squares approximations. Then, we describe the techniques used in this paper to solve discrete ill-posed inverse problems, including the standard Tikhonov regularization and pseudo-inverse after a random projection. The bias-variance decomposition of solution errors is provided for different solution techniques and it is shown experimentally that the error has a minimum at some value of the varied smaller dimension of the random projection matrix. Taking two well-known test examples of the discrete ill-posed problems of Carasso and Baart, we obtain experimental results of their solution. The comparison shows that the minimal error of the pseudo-inverse solution after a random projection is close to the error of the standard Tikhonov regularization, but the former solution is obtained faster, especially at the larger noise values.*

*Keywords: discrete ill-posed problems, pseudo-inverse, regularization, random projection, bias, variance*

*ACM Classification Keywords: I.5.4 Signal processing, I.6 SIMULATION AND MODELING (G.3), G.1.9 Integral Equations*

## Introduction

Many practical applications require solving a discrete problem in the form:

$$\mathbf{Ax} \approx \mathrm{b}. \tag{1}$$

Here the matrix A and the vector b are known, the vector **x** must be evaluated. In case when

- the singular values of A decay gradually to zero,

- the ratio between the largest and the smallest nonzero singular values is large,

the problem is known as discrete ill-posed problem [Hansen, 1998].

Approximate solutions of discrete ill-posed problems as the least squares problem using standard methods of numeric linear algebra such as LU, Cholesky, QR factorizations are unstable. It means that small perturbations in the input data lead to large perturbations in the solution.

Ill-posed problems abound in many areas of science and engineering. Typical examples of discrete ill-posed problems arise from discretization of continuous ill-posed problems, such as Fredholm integral equations of the first kind. Important problems of spectrometry [Zabulonov, Korostil, & Revunova, 2006], gravimetry [Bulakh, 2006], magnitometry [Strakhov, 2008], electrical prospecting [Khmelevskii & Bondarenko, 1999], etc. have the properties of discrete ill-posed problems.

To overcome the instability and to improve the accuracy of solutions to discrete ill-posed problems, regularization methods have been proposed [Hansen, 1998; Tikhonov & Arsenin, 1977; Morozov, 1984; Engl, Hanke, & Neubaer, 2000]. Regularization imposes some constraints on the desired solution that stabilizes the problem and leads to meaningful and stable solution. For example, Tikhonov regularization [Tikhonov & Arsenin, 1977; Hansen, 1998] penalizes solutions with large $l_2$-norms.

The drawbacks inherent in the methods of solving discrete ill-posed problems based on Tikhonov regularization include their high computational complexity and the difficulty of selecting the proper regularization parameter (penalty weight) which influences the solution stability. Therefore, alternative approaches are required for solving discrete ill-posed problems that would have the accuracy comparable to Tikhonov regularization at lower computational costs.

We develop such an approach using the ideas of our previous work on distributed representations and random projections [Misuno, Rachkovskij, & Slipchenko, 2005; Revunova & Rachkovskij, 2009]. Recently, researchers working in the area of numeric linear algebra applied similar ideas to get fast randomized algorithms for the least squares problem, matrix factorization, principal component analysis, etc. It is therefore of interest to study those techniques and apply them to discrete ill-posed inverse problems.

This paper is structured as follows. In the first three sections, we provide a brief survey of linear least squares and discrete ill-posed problems, approximate matrix decompositions with randomized algorithms, and randomized least squares approximations. Then, we describe the approach we use to solve discrete ill-posed problems by several methods, including those employing random projections. We provide the bias-variance decomposition of solution error and show experimentally that it has a minimum at some value of the varied smaller dimension of the random projection matrix.. Taking two well-known test examples of discrete ill-posed problems of Carasso and Baart, we obtain experimental results of their solution using both Tikhonov regularization and pseudo-inverse after a random projection. The comparison shows that the minimal error of the pseudo-inverse solution after a random projection is at the level of the standard Tikhonov regularization, but the solution is obtained faster, especially at the larger noise values. The final section provides conclusions and directions of future work.

## Linear least squares and discrete ill-posed problems

Many applications in mathematics, physics, data analysis, etc. require finding an approximate solution to a system of linear equations that has no exact solution. For example, (1) with $\mathrm{A}\in\Re^{m\times n}$ for $m>n$ or $m<n$ or $\min(m,n) \neq \mathrm{rank}(\mathrm{A})$ generally does not have x such that Ax = b. The method of least squares selects x that minimizes the sum of squares of the elements of the residual vector by solving the optimization problem

$$\mathbf{x'} = \mathrm{argmin}_{\mathbf{x}} \, ||\mathrm{Ax} - \mathrm{b}||_2. \tag{2}$$

It is well-known that the minimum-length vector among those satisfying (2) may be computed based on the Moore-Penrose pseudo-inverse of the matrix A [Demmel, 1997]:

$$\mathbf{x'} = \mathrm{A}^{+}\mathrm{b}. \tag{3}$$

In particular, the traditional solution can be obtained by taking the derivative of $||\mathrm{Ax} - \mathrm{b}||^2 = (\mathrm{Ax} - \mathrm{b})^{\mathrm{T}} (\mathrm{Ax} - \mathrm{b})$ with respect to x and setting it equal to zero. This gives the system of the so-called normal equations

$$\mathrm{A}^{\mathrm{T}}\mathrm{Ax'} = \mathrm{A}^{\mathrm{T}}\mathrm{b} \tag{4}$$

that should be solved to provide the minimizing vector x'. This requires the residual vector (Ax'−b) to be orthogonal to the column space of A, i.e., $(\mathrm{Ax'} - \mathrm{b})^{\mathrm{T}} \mathrm{A} = 0$. If the matrix $\mathrm{A}^{\mathrm{T}}\mathrm{A}$ has full numerical rank and is well-conditioned, (4) can be solved using the Cholesky decomposition $\mathrm{A}^{\mathrm{T}}\mathrm{A} = \mathrm{R}^{\mathrm{T}}\mathrm{R}$, giving $\mathrm{R}^{\mathrm{T}}\mathrm{Rx'} = \mathrm{A}^{\mathrm{T}}\,\mathrm{b}$, where R is an upper triangular matrix.

In case A is rank-deficient or ill-conditioned, solution by a QR decomposition can be used. Here A=QR is obtained, where Q is an orthogonal matrix and R is an upper triangular matrix. Substituting it to (4), one obtains the system $Rx' = Q^T\mathbf{b}$ that can be immediately solved sequentially, by the backward substitution, since R is triangular.

The Singular Value Decomposition (SVD) can also be used. SVD represents A as

$$A = USV^T, \tag{5}$$

where U is the matrix of left singular vectors with orthonormal columns, and V is the matrix of right singular vectors with orthonormal columns, S is the diagonal matrix of singular values.

The Moore-Penrose generalized inverse, or pseudo-inverse, of A may be expressed in terms of SVD as

$$A^+ = VS^{-1}U^T. \tag{6}$$

Then the solution **x'** is obtained by (3).

Known techniques find a solution to (2) in $O(mn^2)$ time. The SVD method is the most computationally intensive among those mentioned above, but it can be useful if A is very ill-conditioned. Let us consider this in more detail. Denote the *i*-th singular value of A as $\sigma_i(A)$, $\sigma_1(A) \geq \sigma_2(A) \geq ... \geq 0$, and the maximum and minimum singular value of A as $\sigma_{max}(A) = \sigma_1(A)$ and $\sigma_{min}(A)$. The condition number of A is $\mathrm{cond}(A) = \sigma_{max}(A)/\sigma_{min}(A)$. When cond(A) is large, the matrix A is ill-conditioned, implying that the solution is potentially unstable and inaccurate. An indication of instability is the fact that small changes in the vector **b** (e.g. due to the noise) cause large changes in the solution x', and the solution error is typically large, especially when the noise level increases. Actually, for large cond(A) the inverse of singular values in $S^{-1}$ become very large, and so the values of x' components, due to (3) and (6).

If there is a sharp gap in the spectrum of singular values, and the singular values after the gap are very small, they may be considered as noisy ones and are eliminated in some implementations by thresholding to overcome the instability. However, this does not work for discrete ill-posed problems, since, as mentioned in the Introduction, they does not have the gap in their singular values, and so their numerical rank is ill-determined and the choice of an appropriate numerical rank is not easy.

The classical method of solving discrete ill-posed problems is Tikhonov regularization [Tikhonov & Arsenin, 1977; Hansen, 1998]. The standard form of Tikhonov regularization problem is formulated as follows

$$\mathbf{x}_{reg} = \mathrm{argmin}_{\mathbf{x}} \ (||\mathbf{Ax}-\mathrm{b}||_2+\lambda||\mathbf{x}||_2), \tag{7}$$

where $\lambda$ is the regularization parameter.

Solution of (7) can be obtained by the method of filtered SVD [Hansen, 1998]:

$$\mathbf{x}_{reg} = V \ \mathrm{diag} \ (f_i / \sigma_i) \ U^T \ \mathrm{y}, \tag{8}$$

where $f_i = \sigma^2_i / (\sigma^2_i + \lambda^2)$ are filter factors.

One problem here is using the SVD of A, since it is a computationally expensive decomposition method. Another problem, that may appear even more important, is selecting the proper regularization parameter $\lambda$.

A number of methods for selecting the regularization parameter have been proposed [Engl, Hanke, & Neubaer, 2000]. The L-curve method [Hansen & O'Leary, 1993] makes a plot of $||x_{reg}||_2$ vs $||Ax_{reg} - \mathbf{b}||_2$ for all valid regularization parameters. For discrete ill-posed problems the L-curve, when plotted in log-log scale, often has a characteristic L-shape appearance (hence its name). A distinct corner separates the vertical and the horizontal parts of the curve. The regularization parameter not far from the corner is selected as optimal.

The discrepancy principle [Morozov, 1984] chooses the regularization parameters such that the residual norm for the regularized solution satisfies $||Ax_{reg} – b||_2 = ||e||_2$, where e in the norm of perturbation of the right-hand side, and therefore requires an estimation of $||e||_2$.

The generalized cross-validation [Wahba, 1990] method is based on the idea that an arbitrary element $b_i$ of the right-hand side b can be predicted by the corresponding regularized solution, and the choice of regularization parameter should be independent of an orthogonal transformation of b. This leads to choosing the regularization parameter that minimizes $||Ax_{reg} – b||^2 / D^2$, where $D$ is a squared effective number of degrees of freedom (which is not necessarily an integer) that can be calculated as $D = m – \Sigma_i f_i$. Here the errors of b are considered as uncorrelated zero-mean random variables with a common variance, i.e., white noise.

However, those methods do not always produce stable results. So, at the wrong values of the regularization parameter the error of solution may be substantial.

Now, let us consider some recent randomized approaches whose ideas, in our opinion, may be useful for solving discrete ill-posed problems. Though randomization is aimed at faster computations, we are also interested in its employing for stabilizing the solution of discrete ill-posed problems.

## Approximate matrix decompositions with randomized algorithms

As follows from the previous sections, matrix decompositions play a major role in solutions of least squares problems and are used in many numerical computer science applications. Let us briefly consider recent approaches to approximate matrix decompositions with randomized algorithms (for an extended review, see [Halko, Tropp & Martinsson, 2009]).

The main motivations of research in this area are as follows. Today, data sets represented by matrices are very large, so that classical algorithms are not always well suited for their processing. Also, traditional algorithms were designed to produce highly accurate matrix decompositions, but it seems that this is not always justified, since the data in large matrices are often inherently imprecise. Other motivations include overcoming the data transfer bottleneck that may dominate the computational costs by getting algorithms with fewer passes over the data, and exploiting the power of novel processor architectures optimized for specific vector operations.

It should be noted that randomized algorithms are often more accurate and robust than deterministic ones, as their error bounds show. And, though their results are probabilistic, the probability of failure may be set by parameter choice to be negligible (e.g., less than $10^{–15}$) while preserving the computational power advantages.

Randomization may be accomplished by random sampling or random projection or their combination. Sampling is considered here as getting some random subset of matrix, such as individual entries (components, cells) or columns and rows, according to some probability distribution. Projecting is a random linear mapping (embedding), usually implemented using multiplication by a random matrix whose components are generated according to some probability distribution. Both sampling and projecting reduce the dimensionality of the initial task and so decrease the computational complexity of subsequent processing.

A low-rank approximation of a given matrix is expressed as

$$A=BC,\ A\in\Re^{m\times n},\ B\in\Re^{m\times k},\ C\in\Re^{k\times n}. \tag{9}$$

The inner dimension $k$ is sometimes called the numerical rank of the matrix.

The fixed-precision approximation problem is formulated as finding A' so that

$$\min||A – A'||_2 < \varepsilon, \tag{10}$$

whereas the fixed-rank approximation problem is finding A' given its target rank *k* so that

$$\min_{\mathrm{rank}(A')\leq k} ||A - A'||_2. \qquad (11)$$

The task of a low-rank approximation to a given matrix can be efficiently computed using randomization by the following two stages.

Stage I (the randomization stage) is to construct a low-dimensional randomized "reduced representation" of the input matrix A that captures its action. This includes the following steps.

Step 1. Make a randomized procedure to get a reduced representation of A.

This procedure is represented as an operator (matrix) X. The random matrix X is carefully constructed, often as a product of several matrices, to ensure the required computational complexity and accuracy of the final approximation.

Step 2. Apply X to A by forming the matrix product Y = AX.

Here Y contains the obtained reduced representation of A. Explicit dimensionality and/or rank of Y is less than that of A, thus ensuring a lower computational complexity of the second stage.

Stage II (the deterministic stage) is to postprocess the result of Stage I to compute a final approximation, typically with well-established deterministic methods from numerical linear algebra, such as a standard factorization (QR, SVD, etc.). This step may require another look at the matrix A.

In some procedures those stages may be distinguished not so clearly, but the main idea is preserved: use some random transformation of A to accelerate computing of the final result.

In the matrix approximation framework, Stage I generally dominates the cost of Stage II. Within Stage I, the computational bottleneck is usually the matrix-matrix product Y = AX in Step 2. However, for some specific constructs of X and/or computational architectures this multiplication can be implemented very efficiently. Anyway, its cost is usually much lower that the cost of A factorization by the traditional methods – otherwise, randomization should not be used.

As an example, let us consider the task of computing an approximate SVD of $A\in\Re^{m\times n}$ with a target numerical rank *k* by the Randomized PCA Algorithm, as considered in [Halko, Tropp & Martinsson, 2009] after [Liberty et al., 2007; Sarlos, 2006].

The randomizing Stage I looks as follows:

I.1. As a randomized procedure, generate a projector matrix $G\in\Re^{n\times(k+p)}$. The elements of G are the realizations of a Gaussian random variable with zero mean and unit variance; *p* is usually a small integer. (12)

I.2. Apply G to A to obtain Y = AG.

The classical deterministic stage (Stage 2) includes the following steps:

II.1 Construct a matrix Q whose columns form an orthonormal basis for the range of Y: Y=QR.

II.2 Form B = QA.

II.3 Compute an SVD of the small matrix: B = U' S V. (13)

II.4 Set U = QU'.

In order to reduce the error bound of this approximation, this technique were elaborated in [Rokhlin, Szlam, & Tygert, 2009]. There, $p=k$, and I.2 is changed to $Y=(AA^T)^q AG$, where $q$ are the steps of a power iteration, $q = 1, 2$ is usually sufficient in practice.

The randomized PCA algorithm we have just considered is an instantiation of the **dimension reduction** (or **projective**) **approach** to matrix approximation. The rationale behind it is that since the rows of a low-rank matrix are linearly dependent, they can be embedded into a low-dimensional space preserving their geometric properties. A random linear map G provides an efficient way to perform this embedding: collecting random samples from the column space of the matrix is equivalent to reducing the dimension of the rows. Adaptation to A is not required to obtain the sampling distribution. Taking samples requires substantial computations, but can be optimized. Then, the samples are orthogonalized as preparation for constructing various matrix approximations.

The work on randomized matrix approximations has been substantially influenced by the field of random embeddings. [Johnson & Lindenstrauss, 1984] showed that the pairwise Euclidean distances between the points in the input space are approximately preserved when the points are mapped to a Euclidean space of a much smaller dimension using a random Gaussian projector. This and subsequent work suggested that some computational problems of a geometric nature may be solved in a more computationally efficient manner by first translating them into a lower-dimensional space. This was used in some applications that require a fast estimation of vector similarity. [Papadimitriou et al., 2000] were the first who applied this approach to linear algebra in context of "latent semantic indexing" based on SVD.

Some researchers studied the issues of simplifying the mapping itself and the cost of its applying. [Achlioptas, 2003] demonstrated that discrete random matrices with $\{-1,0,1\}$ components perform nearly as well as a Gaussian matrix. [Li, Hastie, & Church, 2006] considered very sparse (and therefore potentially fast) discrete random matrices, however, as we know, so far their work has not been applied to numerical linear algebra. On the other hand, the fast Johnson-Lindenstrauss transform [Ailon & Chazelle, 2006] which combines the speed of the Fast Fourier Transform with the embedding properties of a Gaussian matrix (further developed in [Ailon & Liberty, 2008; Liberty, Ailon, & Singer, 2008]) has got substantial interest in the area under consideration. In particular, [Sarlos, 2006] applied these techniques to matrix approximation, which has led to some of the fastest algorithms available today [Liberty et al., 2007; Woolfe et al., 2008].

[Woolfe et al., 2008] propose a fast randomized algorithm for the approximation of matrices by constructing and applying a structured random matrix $R\in\Re^{l\times m}$ ($l > k$) to each column of A, Y=RA. Here R is obtained with the subsampled random Fourier transform. The structure of R allows one to apply it to an arbitrary $m$-vector rapidly, at a cost proportional to $m\log(l)$. So, the resulting procedure can construct a rank-$k$ approximation from the entries of A at a cost proportional to $mn\log(k)+l^2(m+n)$.

The accuracy bound of the algorithm guarantees that the spectral norm $||A-A'||$ is of the same order as $(\max\{m,n\})^{1/2}\sigma_{k+1}$ of A. Note, that in order to compute a similarly accurate rank-$k$ approximation, the standard approach to approximate SVD by computing the full SVD and truncating it requires $O(mn \min\{m;n\})$ floating-point operations. A more efficient deterministic scheme using a partial QR factorization and factor postprocessing takes $O(kmn)$ operations. So, the classical decomposition algorithms require more operations than the present one [Woolfe et al, 2008]. In practice, the algorithm runs faster than the classical algorithms, even when $k$ is quite small or large.

Another approach to randomized matrix approximation may be considered as **the sampling approach**. Its simplest manifestation is matrix sparcification that randomly preserves a fraction of non-zero entries of A with the aim to reduce storage or/and to accelerate matrix computations. Another related instance of this approach stems from the methods that build a matrix approximation from a submatrix and computed coefficient matrices.

Approximations using a subset of columns are known as the interpolative decomposition; those using a subset of rows and a subset of columns are known as the CUR decomposition.

The interpolative decomposition is based on the idea that a small set of columns describes most of the action of a numerically low-rank matrix [Ruston, 1964]. A number of works develop randomized algorithms for this class of matrix approximations. These methods first compute a sampling probability for each column using their leverage scores that reflect the relative importance of the columns to the action of the matrix. Columns are then selected randomly according to this distribution, and post processed to provide the final matrix approximation. Such a column sampling can also be viewed as an adaptive form of dimension reduction. Realization of this method by [Drineas, Kannan, & Mahoney, 2006] showed that, given a target rank $k$ and a parameter $\varepsilon>0$, it produces the approximation matrix B for which

$$||A - B||_F \leq || A - A_{(k)} ||_F + \varepsilon \, ||A||_F, \tag{14}$$

where $||\cdot||_F$ denotes the Frobenius norm, $A_{(k)}$ is a best rank-$k$ approximation of A. The same column sampling method also yields spectral-norm error bounds [Rudelson & Vershynin, 2007].

[Deshpande et al., 2006; Deshpande & Vempala, 2006] and [Har-Peled, 2006] demonstrated that the error in the column sampling approach can be improved by an iterative $k$-pass algorithm for which

$$||A - B||_F \leq (1+\varepsilon) \, || A - A_{(k)} ||_F. \tag{15}$$

Subsequent work [Boutsidis, Mahoney, & Drineas, 2008; Boutsidis, Drineas, & Mahoney, 2009] showed that post processing the sampled columns with a rank-revealing QR algorithm can reduce the number of output columns required while improving the classical existential error bound [Ruston, 1964].

Drineas et al. have developed randomized techniques for computing CUR decompositions, which express $A \approx CUR$, where C and R are small column and row submatrices of A, and U is a small linkage matrix. These methods identify columns (rows) that approximate the range (corange) of the matrix; the linkage matrix is then computed by solving a small least-squares problem. [Drineas, Kannan, & Mahoney, 2006a] developed a randomized CUR algorithm with controlled absolute error, and [Drineas, Mahoney, & Muthukrishnan, 2008] developed a relative error algorithm.

## Randomized Least Squares Approximations

In this section we consider recent work aimed at a rapid solution of the least squares approximation problem. Though the main ideas were considered in the previous section, in this area they are immediately applied to this very problem, and the error bounds and complexity are investigated directly for this case.

As before, randomization is viewed as preconditioning the input matrix A, and now also the target vector b with a carefully-constructed data-independent random matrix X (often constructed from several other matrices). Thus, the least squares approximation problem (2) is replaced with

$$\mathbf{x'} = \operatorname{argmin}_{\mathbf{x}} ||Xb - \mathbf{XAx}||_2. \tag{16}$$

Solution to (16) can be explicitly computed using some traditional deterministic algorithm, e.g., SVD, to compute the generalized inverse

$$x' = (XA)^{+} Xb. \tag{17}$$

Alternatively, standard iterative methods could be used, such as the Conjugate Gradient Normal Residual method. It can produce an $\varepsilon$-approximation to the optimal solution of (16) in $O(\text{cond}(XA)\, rn \log(1/\varepsilon))$ time, where cond(XA) is the condition number of XA, and $r$ is the number of rows of XA.

The idea to apply randomization to the least squares problem using sampling was proposed in [Drineas, Mahoney, & Muthukrishnan, 2006]. [Sarlos, 2006] pioneered applying a random projection matrix to this problem. The error bounds and speed for both approaches were improved in [Drineas et al., 2007] for the over constrained ($m>>n$) least squares approximation problems.

Their random sampling algorithm starts by preprocessing the matrix A and the right hand side vector b with a randomized Hadamard transform H. It then constructs a smaller problem by uniformly randomly sampling a small number $r$ of constraints from the preprocessed problem. So, here X = SH, where S is a matrix that represents the sampling operation. Then, this algorithm solves the least squares problem on just those sampled constraints.

In a similar manner, the second algorithm also initially transforms the input A and b by H. Then it multiplies the result by a sparse projection matrix T of $O(n/\varepsilon) \times m$. So, here X = TH. Finally, the second algorithm solves the least squares problem on just those $O(n/\varepsilon)$ coordinates to obtain $n$-vector x'.

In both cases, the solution to the smaller problem provides a relative-error $\varepsilon$-approximation to the exact solution, while making no assumptions on the input data. And if $m$ is sufficiently larger than $n$, the approximate solution can be computed in $o(mn^2)$ time. Traditional methods (of Gauss, etc.) find a solution in $O(mn^2)$ time.

In [Rokhlin & Tygert, 2008] similar techniques were used to analyze theoretically and evaluate empirically random-projection based algorithms for over determined least-squares problems. X implements the subsampled randomized Fourier transform. Their procedure computes an $n$-vector x such that x minimizes the Euclidean norm Ax−b to a relative precision ε. The algorithm typically requires $O((\log(n) + \log(1/\varepsilon))mn + n^3)$ floating-point operations, whereas the earlier algorithms involve costs proportional to $1/\varepsilon$. This cost is less than $O(mn^2)$ required by the classical schemes based on QR-decompositions or bidiagonalization. Moreover, the methods extend to underdetermined ($m<n$) least-squares [Tygert, 2009].

## Solving discrete ill-posed problems using random projections

Now let us consider discrete ill-posed problem in the form:

$$\mathbf{\Phi x} = \mathbf{y}. \tag{18}$$

Here we have the matrix $\mathbf{\Phi} \in \Re^{N\times N}$ with singular values $\sigma_i$ gradually decaying zero and large condition number. The vector $\mathbf{y} \in \Re^N$ is distorted by the additive noise $\boldsymbol{\varepsilon} \in \Re^N$: $\mathbf{y}=\mathrm{y}_0+\boldsymbol{\varepsilon}$.

Let us use the randomized algorithms (described in the survey sections above) not only to accelerate, but also to stabilize the solution x' of the ill-posed problem, as follows. Multiply both sides of (18) by the matrix $\mathbf{\Omega} \in \Re^{K\times N}$, $K \le N$, whose elements are realizations of a normal random variable with zero mean and unit variance. The number of columns N of matrix $\Omega$ is determined by the dimension of the matrix $\mathbf{\Phi}$, the number of rows K is a priori unknown since the numerical rank of $\mathbf{\Phi}$ is ill-determined and the required numerical rank of approximation is unknown. We obtain

$$\mathbf{\Omega\Phi}\ \mathbf{x} = \mathbf{\Omega y}, \text{ where } \mathbf{\Omega\Phi} \in \Re^{K\times N}, \mathbf{\Omega y} \in \Re^K. \tag{19}$$

Then the least-squares problem is

$$\mathbf{x}_{\mathrm{Pr}} = \mathrm{argmin}_{\mathbf{x}}\ ||\mathbf{\Omega\Phi}\ \mathbf{x} - \mathbf{\Omega y}||_2. \tag{20}$$

Signal reconstruction based on pseudo-inverse is obtained as

$$\mathbf{x}_{pinPr} = (\mathbf{\Omega\Phi})^{+}\,\mathbf{\Omega y}. \tag{21}$$

Signal reconstruction by Tikhonov regularization is obtained as

$$\mathbf{x}_{regPr} = \text{argmin}_{\mathbf{x}}\,(\,||\mathbf{\Omega\Phi x}-\mathbf{\Omega y}||_2+\lambda||\mathbf{x}||_2\,). \tag{22}$$

The accuracy of solving the inverse problem will be evaluated using the error $d$ of recovery of the true signal $\mathbf{x}0$:

$$e = ||\mathbf{x}0 - \mathbf{x'}||_2 = ||e||_2, \tag{23}$$

where x' is the reconstructed signal vector, e is the solution error vector.

The error vector e is represented [Vogel, 2002; Goldenshluger & Pereverzev, 2000] as the sum of bias and variance. Let us calculate them for our task as follows. Denote P the operator that transforms y to x': x' = Py. Then, taking into account that $y = y_0 + \boldsymbol{\varepsilon}$ and $y_0 = \mathbf{\Phi x}0$, the expression for x' can be represented as:

$$\mathbf{x'} = P\,(y_0 + \boldsymbol{\varepsilon}) = P\,y_0 + P\,\boldsymbol{\varepsilon} = P\,\mathbf{\Phi x}0 + P\,\boldsymbol{\varepsilon}. \tag{24}$$

Using the expression for x', we obtain the expression for e:

$$e = \mathbf{x'} - x0 = P\,\mathbf{\Phi x}0 - x0 + P\,\boldsymbol{\varepsilon} = (P\,\mathbf{\Phi} - I)x0 + P\,\boldsymbol{\varepsilon}. \tag{25}$$

Thus,

$$e = e_1 + e_2, \text{ где } e_1, e_2 \in \Re^N,\ e_1 = (P\,\mathbf{\Phi} - I)x0,\ e_2 = P\,\boldsymbol{\varepsilon}. \tag{26}$$

$e_1$ is called bias, $e_2$ is called variance [Vogel, 2002; Goldenshluger & Pereverzev, 2000].

To obtain the solution **without projection**, we use the following methods.

The pseudo-inverse solution based on SVD is actually computed as:

$$\mathbf{x}_{pin} = \mathbf{\Phi}^{+}\,y,\ P_{pin} = \mathbf{\Phi}^{+} = V\,\text{diag}\,(\varphi_i / \sigma_i)\,U^{T},\ \text{iff } \sigma_i > \textit{tresh}\ \varphi_i = 1,\ \text{otherwise } \varphi_i = 0.$$
$$\textit{tresh} = \textit{max}(K,N)\ \text{eps}(\textit{max}(\sigma_i)), \tag{27}$$

where U, V, S are obtained by SVD of $\mathbf{\Phi} = USV^{T}$; $\sigma_i$ = diag S are singular values, the elements of a diagonal matrix S; floating-point relative accuracy eps($z$) is the positive distance from abs($z$) to the next larger in magnitude floating point number of the same precision as z.

Here the bias-variance decomposition is

$$e_{1pin} = (P_{pin}\,\mathbf{\Phi} - I)x0,\ e_{2pin} = P_{pin}\,\boldsymbol{\varepsilon}. \tag{28}$$

For the solution based on Tikhonov regularization (22)

$$P_{reg} = V\,\text{diag}\,(f_i / \sigma_i)\,U^{T}. \tag{29}$$

$$e_{reg} = \mathbf{x}_{reg} - x0 = P_{reg}(\mathbf{\Phi x}0 + \boldsymbol{\varepsilon}) - x0 = (P_{reg}\,\mathbf{\Phi} - I)\,x0 + P_{reg}\,\boldsymbol{\varepsilon}; \tag{30}$$

$$e_{1reg} = (P_{reg}\,\mathbf{\Phi} - I)\,x0,\ e_{2reg} = P_{reg}\,\boldsymbol{\varepsilon}. \tag{31}$$

**After the projection**, the bias-variance decomposition of the solution error vector takes the following forms.

By analogy to (25), we can write:

$$e_{Pr} = \mathbf{x}_{Pr} - x0 = P_{Pr}\,\mathbf{\Omega}\,y - x0 = P_{Pr}\,\mathbf{\Omega}\,(\mathbf{\Phi x}0 + \boldsymbol{\varepsilon}) - x0 = (P_{Pr}\,\mathbf{\Omega\Phi} - I)\,x0 + P_{Pr}\,\mathbf{\Omega}\boldsymbol{\varepsilon}, \tag{32}$$

where $\mathbf{x}_{Pr}$ is the solution after projection, $P_{Pr}$ is the operator that transforms $\mathbf{\Omega}$y to $\mathbf{x}_{Pr}$: $\mathbf{x}_{Pr} = P_{Pr}\,\mathbf{\Omega y}$.

For the pseudo-inverse solution after the projection:

$$P_{pinPr} = \mathbf{C}\,\text{diag}\,(\,\varphi_i / s_i)\,B^{T}, \tag{33}$$

where B, C, $\Sigma$ are the SVD result for the matrix $\mathbf{\Omega\Phi} = B\,\Sigma\,C^T$, $s_i = \text{diag}\,\Sigma$ are singular values, the elements of the diagonal matrix $\Sigma$, $s_i$ are thresholded by $\varphi_i$ as in (27). The error terms are

$$e_{1pinPr} = (P_{pinPr}\,\mathbf{\Omega\Phi} - I)x0,\ e_{2pinPr} = P_{pinPr}\,\mathbf{\Omega\varepsilon};\ e_{1pinPr}+e_{2pinPr} = e_{pinPr}, \tag{34}$$

where $e_{pinPr}$ is the solution error vector for the pseudo-inverse using the random projection.

For Tikhonov regularization after the projection:

$$P_{regPr} = C\,\text{diag}(\varphi_i / s_i)\,B^T, \tag{35}$$

where $\varphi_i = s^2_i /( s^2_i + \lambda^2)$ are filter multipliers.

$$e_{1regPr}= (P_{regPr}\,\mathbf{\Omega\Phi} - I)x0,\ e_{2regPr}= P_{regPr}\,\mathbf{\Omega\varepsilon};\ e_{1regPr}+e_{2regPr} = e_{regPr}, \tag{36}$$

where $e_{regPr}$ is the solution error vector for Tikhonov regularization using the random projection.

For the solution based on the method of approximate matrix factorization (AMF) (12)-(13).

$$P_{amfPr} = (\mathbf{\Phi'})^+ = V\,\text{diag}(\varphi_i / z_i)\,U^T, \tag{37}$$

where $\mathbf{\Phi'}$ is the result of $\mathbf{\Phi}$ approximation after projection using $\mathbf{\Omega} \in \Re^{N\times(K+2)}$, V is the matrix obtained by (13.II.3), U is obtained by (13.II.4), the singular values $z_i$ are thresholded by $\varphi_i$ as in (27).

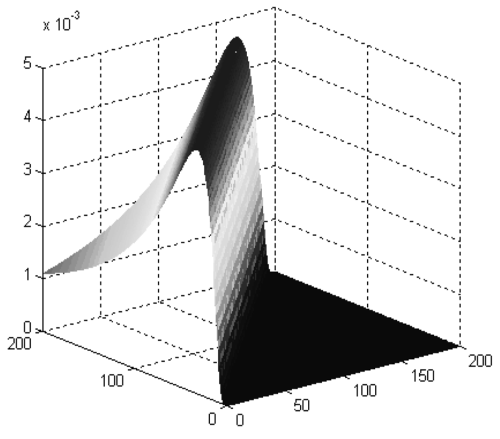


Fig.1. The matrix $\mathbf{\Phi}$ of the Carasso problem.

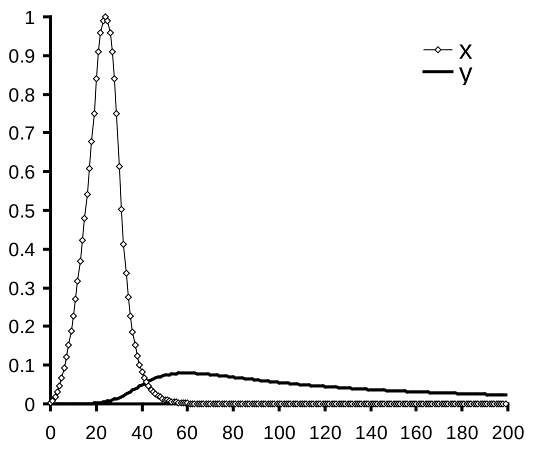


Fig. 2. The signal vector **x** and the right hand side y of the Carasso problem.

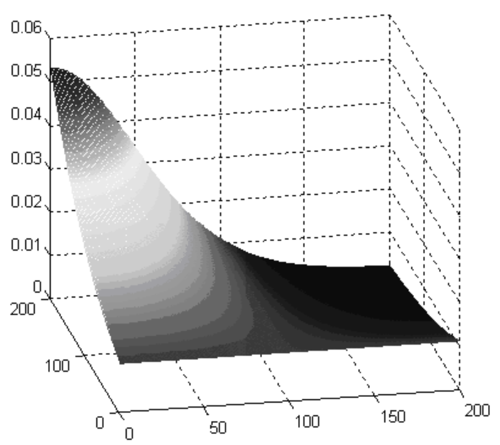


Fig. 3. The matrix $\mathbf{\Phi}$ of the Baart problem.

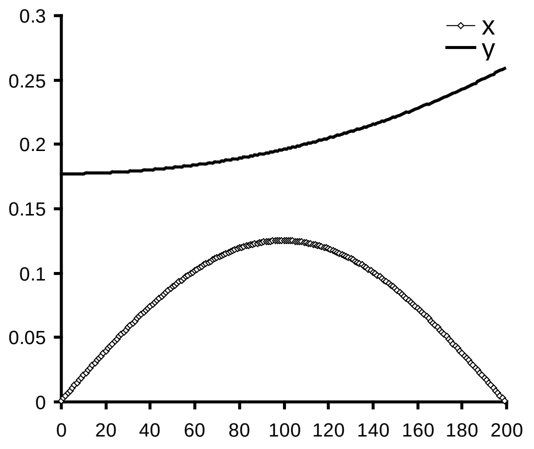


Fig.4. The signal vector **x** and the right hand side y of the Baart problem.

The error terms are

$$e_{1amfPr}=(P_{amfPr} - I)x0,\ e_{2mfPr}= P_{amfPr}\ \varepsilon;\ e_{1amfPr} + e_{2amfPr} = e_{amfPr}, \qquad (38)$$

where $e_{amfPr}$ is the solution error vector for the approximate matrix factorization using the random projection (12)-(13).

## Experiments

Let us conduct an experimental investigation using two well-known discrete ill-posed problem.

The problem of Carasso [Carasso, 1982] seeks to reconstruct the time profile of a heat source by monitoring the temperature at a fixed distance away. The inverse heat equation here is a Volterra integral equation of the first kind with [0,1] as integration interval. The kernel is

$$K(s,t) = k(s-t),\ k(t) = t^{3/2} / 2\pi^{1/2} \exp( - 1/4t). \qquad (39)$$

The integral equation is discretized by the quadrature method using simple collocation and the midpoint rule with N points (e.g., [Carasso, 1982]). An exact solution $x_0$ is constructed, and then $y_0$ is produced as $y_0=\mathbf{\Phi x}_0$.

The problem of Baart [Baart, 1982] is discretization of an artificial Fredholm integral equation of the first kind with kernel K and right-hand side h given by

$$K(s,t) = \exp(s \cos(t)),\ h(s) = 2 \sin(s)/s;\ \ s\in[0,\pi/2],\ t\in[0,\pi]. \qquad (40)$$

Discretization is done by the Galerkin method with orthonormal basis functions. The known solution is also represented as a vector $x_0$. In this problem, $\mathbf{\Phi x}_0$ is not exactly equal to $y_0$.

In both problems, the matrix $\mathbf{\Phi}$ has dimensionality of 200×200. The right-hand side y0 is distorted by an additive noise with the uniform distribution and various amplitudes. The random projection matrix $\mathbf{\Omega}\in\Re^{K\times N}$, $N$=200, $K\leq N$ is the Gaussian random matrix with the entries of zero mean and unit variance.

Let us consider how the random projection changes the singular value spectrum. In Fig. 5, the singular values of $\mathbf{\Phi}$, $\mathbf{\Omega\Phi}$ ($K$=30), $\mathbf{\Phi'}$ ($K$=30) are shown for the Carasso problem. Evidently, the numerical rank of $\mathbf{\Phi}$ is ill-determined, whereas the rank of $\mathbf{\Omega\Phi}$ and $\mathbf{\Phi'}$ is equal to 30.

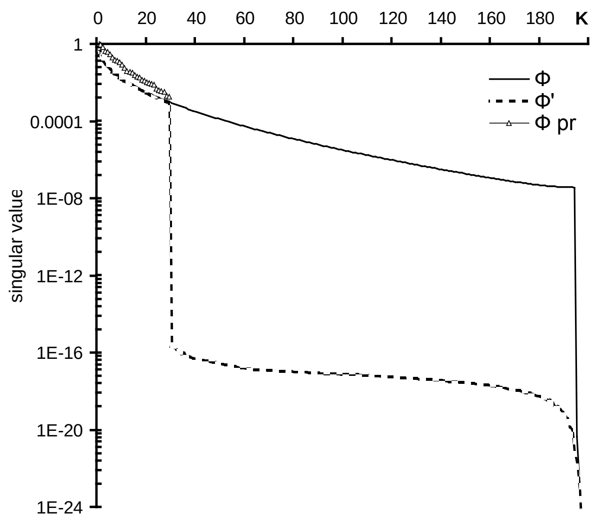


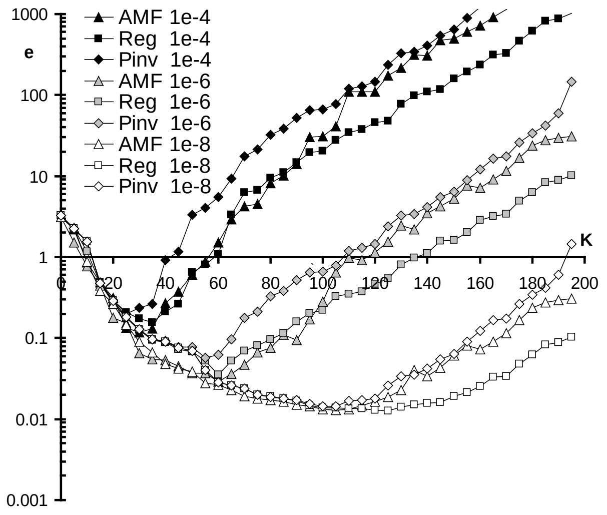
Fig. 5. The singular values of the Carasso problem

for $\mathbf{\Phi}$ (Φ), $\mathbf{\Omega\Phi}_{K=30}$ (Φ pr), $\mathbf{\Phi'}_{K=30}$ (Φ').

Fig. 6 shows the dependence of the signal reconstruction error *e* on the row number *K* of the projector matrix **Ω**(*K*×*N*) the Carasso problem. The plots are provided for three solutions that use the projected matrix **ΩΦ**: the pseudo-inverse (20)-(21), the Tikhonov regularization (22) with the selection of λ using the discrepancy principle [Morozov, 1984], and the pseudo-inverse of the approximate matrix **Φ'** obtained by (12)-(13).

Three noise levels are used. For all three solution techniques at some value of *K* the minimum of the error *d* can be observed for all noise levels. When the noise level increases, the minimum position shifts towards the smaller values of *K* and the error value *d* at the minimum increases.

Fig.7 shows the dependences of the two error terms on K: the norm of the bias error term e1 = ||e1||2 and variance e2 = ||e2||2 (37)-(38) obtained for the solution by the approximate matrix factorization (12)-(13). With the increasing K, ||$\mathbf{e}_1$|| decreases and ||$\mathbf{e}_2$|| increases, so that ||$\mathbf{e}_1$+$\mathbf{e}_2$|| has a minimum. When the noise level increases, position of the error minimum shifts towards the lower values of K. This occurs because the dependence of ||$\mathbf{e}_1$|| on K is practically the same for all levels of noise, whereas ||$\mathbf{e}_2$|| increases with the increasing noise level.



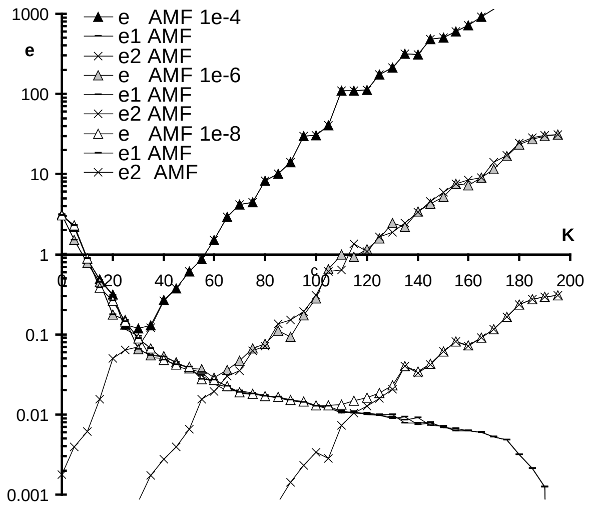


Fig. 6. Dependence of the signal reconstruction error e for the Carasso problem on the row number K of the projection matrix at the noise levels 10–8, 10–6, 10–4. Pinv is pseudo-inverse, Reg is the Tikhonov regularization with the selection of λ using the discrepancy principle, AMF is the approximate matrix factorization.

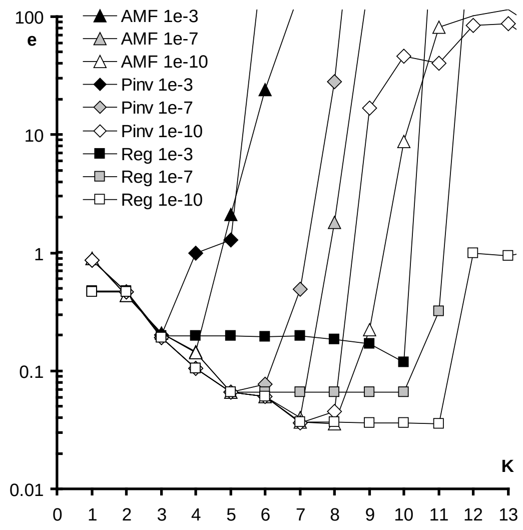
Fig. 7. The dependences of bias *e*$_1$, variance *e*$_2$, and the total error on *K* at the noise levels 10–8, 10–6, 10–4 for the solution of the Carasso problem by the approximate matrix factorization AMF.

For the Baart problem, the dependences of the signal reconstruction error e (Fig. 8) and the bias e1 and variance e2 terms (Fig. 9) are similar to those of the Carasso problem. Here, bias decreases not so profoundly, probably because for this problem $\mathbf{\Phi x}_0$ is not exactly equal to $\mathbf{y}_0$.

Table 1 shows the error results. For the Carasso problem, for the techniques without projection, the error *e* is given, and for the techniques with projection, the minimum value of errors obtained at some *K* is given. Without projection, the standard pseudo-inverse provides a somewhat acceptable error only at the lowest noise level and cannot be used at the larger noise levels. The error levels for Tikhonov regularization are comparable for all three noise levels, and are at the level the pseudo-inverse error at 10–8.

With projection, the error values at the minimum are comparable and small for all techniques. The pseudo-inverse with projection becomes stable and shows the error values at the level of Tikhonov regularization with projection. The error values are much smaller than those for the Tikhonov regularization without projection.

For the Baart problem, the minimal errors after projection for all techniques are at the level of the Tikhonov regularization without projection, however the error is somewhat smaller for the approximate matrix factorization technique.



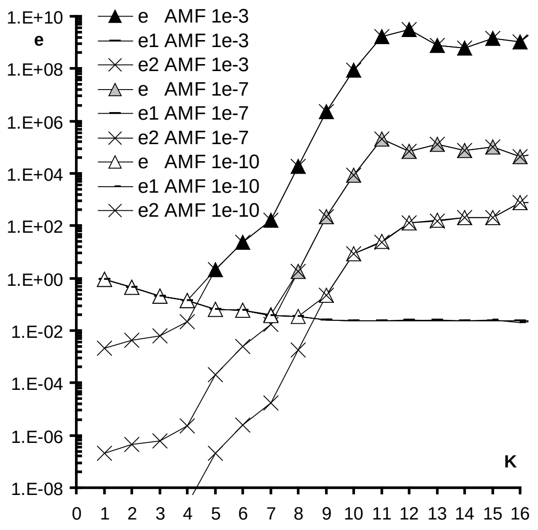


Fig. 8 Dependence of the signal reconstruction error *e* for the Baart problem on the row number *K* of the projection matrix at the noise levels $10^{-10}$, $10^{-7}$, $10^{-3}$. Notations are as in Fig.6.

Fig. 9 Dependences of the bias e1 and variance e2 error terms for the Baart problem on *K* for the solution based on the approximate matrix factorization AMF.

Table 1. The solution errors for the Carasso and the Baart problems. $e_{\text{pin}}$ and $e_{\text{reg}}$ are the errors of the pseudo-inverse and the Tikhonov regularization solutions without projection, $e_{\text{pinPr}}$ and $e_{\text{regPr}}$ are those with projection, $e_{\text{amfPr}}$ is the solution error obtained using the approximate matrix factorization with projection.

| The Carasso problem | | | | | | The Baart problem | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Noise level | Error, solutions w/o projection | | Min error, solutions w/ projection | | | Noise level | Error, solutions w/o projection | | Min error, solutions w/ projection | | |
| | $e_{\text{pin}}$ | $e_{\text{reg}}$ | $e_{\text{pinPr}}$ | $e_{\text{regPr}}$ | $e_{\text{amf Pr}}$ | | $e_{\text{pin}}$ | $e_{\text{reg}}$ | $e_{\text{pinPr}}$ | $e_{\text{regPr}}$ | $e_{\text{amf Pr}}$ |
| $10^{-8}$ | 0.308 | 0.308 | 0.0136 | 0.013 | 0.013 | $10^{-10}$ | 231 | 0.037 | 0.036 | 0.036 | 0.036 |
| $10^{-6}$ | 41.7 | 0.870 | 0.05 | 0.036 | 0.029 | $10^{-7}$ | $2\times10^{5}$ | 0.065 | 0.066 | 0.065 | 0.04 |
| $10^{-4}$ | 3000 | 0.66 | 0.169 | 0.179 | 0.110 | $10^{-3}$ | $8\times10^{9}$ | 0.225 | 0.2 | 0.12 | 0.15 |

Fig 10. shows the computation times of different techniques vs K. As expected, the solutions based on pseudo-inverse and Tikhonov regularization without the random projection have constant and rather large computation times compared to the solutions using the random projection at the smaller values of K. The computation times for the three methods with the projection are rather close, however the AMF time reaches the constant no-projection times earlier.

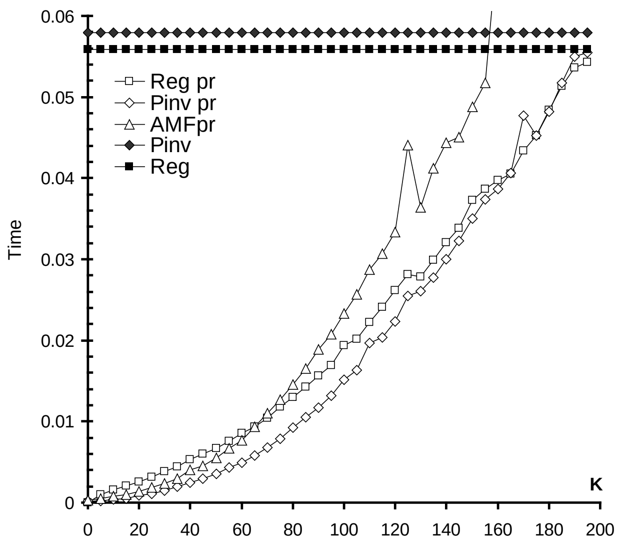


Fig 10. The computation times of studied techniques vs K. pr denote the times with the random projection.

## Conclusion

We conducted an experimental study of techniques for solving discrete ill-posed problems. The test discretized problems were those of Carasso (an inverse heat equation) and Baart (an artificial Fredholm integral equation of the first kind). We used the pseudo-inverse solution and the Tikhonov regularization with the selection of the regularization parameter $\lambda$ by the discrepancy principle, and the same methods, but using preliminary projection with a random Gaussian matrix ($K \times N$).

We provided the bias-variance decomposition of solution errors for different solution techniques. Investigation of the behavior of the bias-variance terms vs the number K of the projection matrix rows showed that the bias term decreases with increasing K and the variance term increases with increasing K, so that the total error has a minimum at some K. With increasing noise level, position of the error minimum shifts to the lower values of K and the error value at the minimum increases.

With the proper choice of *K*, the accuracy of the pseudo-inverse with projection is at the level of or exceeds the accuracy of the Tikhonov regularization without projection for the studied test problems. In most cases, projecting reduces the solution error for Tikhonov regularization, especially significantly for the cases with large solution error before the projection. The pseudo-inverse solution error without projecting is several orders of magnitude higher than the error of the other methods, so that the pseudo-inverse solution without projection is not suitable for the considered discrete ill-posed problems.

Thus, the study and application of techniques for solving discrete ill-posed problems based on pseudo-inverse with projection is a promising direction due to their stability, manifested in the smooth change of the signal recovery error with increasing noise, and also because of the lower computational costs. The latter is due to the reduction of the computational costs of the singular value decomposition for the resulting ($N$×$K$) matrix after projection, when $K$ constitutes a small fraction of $N$, compared to the complexity of singular value decomposition of the original ($N$×$N$) matrix. Particularly large gains are achieved at the higher noise levels because for them the optimal $K$ is small.

Directions of further research include techniques for a computationally efficient choice of the projection matrix dimensionality $K$ at which the solution error is close to the minimum for real situations, i.e. where the exact solution is unknown.

## Bibliography

[Achlioptas, 2003] D. Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. J. Comput. System Sci., 66(4): 671-687, 2003.

[Ailon & Chazelle, 2006] N. Ailon & B. Chazelle Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In STOC '06: Proc. 38th Ann. ACM Symp. Theory of Computing, pp. 557-563, 2006.

[Ailon & Liberty, 2008] N. Ailon & E. Liberty Fast dimension reduction using Rademacher series on dual BCH codes. In STOC '08: Proc. 40th Ann. ACM Symp. Theory of Computing, 2008

[Baart, 1982] M.L. Baart. The use of auto-correlation for pseudo-rank determination in noisy ill-conditioned linear least-squares problems. IMA J.Numer.Anal., 2: 214-247, 1982

[Boutsidis, Drineas, & Mahoney, 2009] C. Boutsidis, P. Drineas, & M.W. Mahoney. An improved approximation algorithm for the column subset selection problem. In STOC '09: Proc. 41st Ann. ACM Symp. Theory of Computing, 2009.

[Boutsidis, Mahoney, & Drineas, 2008] C. Boutsidis, M. W. Mahoney, & P. Drineas. Unsupervised feature selection for principal components analysis. In Proc. ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining (KDD), 2008.

[Bulakh, 2006] E.G. Bulakh. A new approximation approach to solving inverse problems of gravimetry in a class of three-dimensional contact surfaces. Dokl. NASU, 1:108–112, 2006 (in Russian).

[Carasso, 1982] A.S. Carasso Determining surface temperatures from interior observations. SIAM J.Appl.Math, 42:558-574, 1982.

[Demmel, 1997] J.W. Demmel. Applied Numerical Linear Algebra. SIAM, Philadelphia. 419p., 1997.

[Deshpande et al., 2006] A. Deshpande, L. Rademacher, S. Vempala, & G. Wang. Matrix approximation and projective clustering via volume sampling. In Proc. 17th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA), pp. 1117-1126, 2006.

[Deshpande & Vempala, 2006] A. Deshpande & S. Vempala Adaptive sampling and fast low-rank matrix approximation. In Approximation, randomization and combinatorial optimization, vol. 4110 of LNCS, Springer, Berlin, pp. 292-303, 2006.

[Drineas, Kannan, & Mahoney, 2006] P. Drineas, R.Kannan, & M. W. Mahoney (Fast Monte Carlo algorithms for matrices. II. Computing a low-rank approximation to a matrix. SIAM J. Comput., 36:158-183 (electronic), 2006.

[Drineas, Kannan, & Mahoney, 2006a] P. Drineas, R.Kannan, & M. W. Mahoney. Fast Monte Carlo algorithms for matrices. III. Computing a compressed approximate matrix decomposition. SIAM J. Comput., 36:184-206, 2006

[Drineas, Mahoney, &. Muthukrishnan, 2006] P. Drineas, M. W. Mahoney, & S. Muthukrishnan. Sampling algorithms for regression and applications. In Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1127–1136, 2006.

[Drineas, Mahoney, & Muthukrishnan, 2008] P.Drineas, M.W. Mahoney, & S. Muthukrishnan. Relative-error CUR matrix decompositions. SIAM J. Matrix Anal. Appl., 30:844-881, 2008.

[Drineas et al., 2007] P. Drineas, M.W. Mahoney, S. Muthukrishnan, & T. Sarlos. Faster least squares approximation. Tech. Rep. 0710.1435, 2007.

[Engl, Hanke, & Neubaer, 2000] H.W. Engl, M. Hanke, & A. Neubaer. Regularization of inverse problems. Kluwer Academic Publishers, Dordrecht, 321 p., 2000.

[Goldenshluger & Pereverzev, 2000] A. Goldenshluger & S.V. Pereverzev. Adaptive estimation of linear functionals in Hilbert scales from indirect white noise observations. Probab. Theory Relat. Fields, 118:169–186, 2000.

[Halko, Tropp & Martinsson, 2009] N. Halko, J.A. Tropp, P.G. Martinsson Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. In: ACM Report 2009-05, Caltech, 2009.

[Hansen, 1998] P.C. Hansen. Rank-deficient and discrete ill-posed problems. Numerical Aspects of Linear Inversion. SIAM, Philadelphia, 247 p., 1998.

[Hansen & O'Leary, 1993] P.C. Hansen & D.P. O'Leary, The use of L-curve in the regularization of discrete ill-posed problems, SIAM J.Sci.Comput. 14:1487-1503, 1993.

[Har-Peled, 2006] S. Har-Peled. Matrix approximation in linear time. Manuscript. Available at http://valis.cs.uiuc.edu/~sariel/research/papers/05/lrank/, 2006.

[Johnson & Lindenstrauss, 1984] W.B. Johnson & J. Lindenstrauss Extensions of Lipschitz mappings into a Hilbert space // Contemporary Mathematics, 26:189-206,1984.

[Khmelevskii & Bondarenko, 1999] V.K. Khmelevskii, V.M. Bondarenko. Electrical exploration prospecting. Nedra, 438 p., 1999 (in Russian).

[Li, Hastie, & Church, 2006] P. Li, T.J. Hastie, & K.W. Church Very sparse random projections. 12th ACM SIGKDD international conference on Knowledge discovery and data mining. Philadelphia, PA, USA: ACM Press, pp. 287-296, 2006.

[Liberty, Ailon, & Singer, 2008] E. Liberty, N. Ailon, & A. Singer. Dense fast random projections and lean Walsh transforms. In APPROX and RANDOM 2008, A. Goel et al., ed., in LNCS, Springer, Berlin, 5171:512-522, 2008.

[Liberty et al., 2007] E. Liberty, F.F. Woolfe, P.G. Martinsson, V. Rokhlin, & M. Tygert. Randomized algorithms for the low-rank approximation of matrices. Proc. Natl. Acad. Sci. USA, 104: 20167-20172, 2007.

[Misuno, Rachkovskij, & Slipchenko, 2005] I.S. Misuno, D.A. Rachkovskij, & S.V. Slipchenko. Vector and distributed representations reflecting semantic relatedness of words. Mathematical Machines and Systems, Issue 3:50–66, 2005.

[Morozov, 1984] V.A.Morozov, Methods for solving incorrectly posed problems, Springer Verlag, New York, 1984.

[Papadimitriou et al., 2000] C.H. Papadimitriou, P. Raghavan, H. Tamaki, & S. Vempala. Latent semantic indexing: A probabilistic analysis, J. Comput. System Sci., 61:217-235, 2000.

[Revunova & Rachkovskij, 2009] E.G. Revunova & D.A. Rachkovskij. Increasing the accuracy of solving the inverse problem using random projections. International Conference "Knowledge-Dialogue-Solution" (KDS-2), International Book Series "INFORMATION SCIENCE & COMPUTING", Sofia, Bulgaria: Institute FOI ITHEA, 15, pp. 93-98, 2009 (in Russian).

[Rokhlin, Szlam, & Tygert, 2009] V. Rokhlin, A. Szlam, & M. Tygert A randomized algorithm for principal component analysis. SIAM J. Matrix Anal. Appl., 31(3):1100-1124, 2009.

[Rokhlin & Tygert, 2008] V. Rokhlin & M. Tygert A fast randomized algorithm for overdetermined linear least-squares regression PNAS September, 105( 36):13212-13217, 2008.

[Rudelson & Vershynin, 2007] M. Rudelson & R. Vershynin. Sampling from large matrices: An approach through geometric functional analysis. J. ACM, 54(4): article 21 (electronic), 2007.

[Ruston, 1964] A. F. Ruston. Auerbach's theorem. Math. Proc. Cambridge Philos. Soc., 56:476-480, 1964.

[Sarlos, 2006] T. Sarlos. Improved approximation algorithms for large matrices via random projections. In Proc. 47th Ann. IEEE Symp. Foundations of Computer Science (FOCS), pp. 143-152, 2006. ,

[Strakhov, 2008]. V.N. Strakhov. The fundamental computational problem in gravimetry, magnetometry, geodesy, and geoinformatics, Izvestiya Physics of the Solid Earth, 44(2):142-157, 2008.

[Tikhonov & Arsenin, 1977] A.N. Tikhonov & V.Y Arsenin. Solution of ill-posed problems. V.H. Winston, Washington, DC., 1977.

[Tygert, 2009] M. Tygert. A fast algorithm for computing minimal-norm solutions to underdetermined systems of linear equations. arXiv:0905.4745, May 2009.

[Vogel, 2002] Vogel C.R. Computational methods for inverse problems. Philadelphia: SIAM. 183 p., 2002.

[Wahba, 1990] G. Wahba, Spline models for observation data, CBMS-NSF Regional Conference Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.

[Woolfe et al., 2008] F. Woolfe, E. Liberty, V. Rokhlin, & M. Tygert. A fast randomized algorithm for the approximation of matrices. Appl. Comp. Harmon. Anal., 25: 335-366, 2008.

[Zabulonov, Korostil, & Revunova, 2006] Yu.L. Zabulonov, Yu.M. Korostil, E.G. Revunova Optimization of inverse problem solution to obtain the distribution density function for surface contaminations. Modeling and information technologies, 39:77-83, 2006 (in Russian).

## Authors' Information

*Elena G. Revunova – PhD, Research Scientist, International Research and Training Center for Information Technologies and Systems, NAS and MES of Ukraine; Pr. Acad. Glushkova, 40, Kiev, 03680, Ukraine; e-mail: egrevunova (at) gmail.com*

*Major Fields of Scientific Research: Signal Processing*

*Dmitri A. Rachkovskij – PhD, DSc, Leading Research Scientist, International Research and Training Center for Information Technologies and Systems, NAS and MES of Ukraine; Pr. Acad. Glushkova, 40, Kiev, 03680, Ukraine; e-mail: dar (at) infrm.kiev.ua*

*Major Fields of Scientific Research: Computational and Artificial Intelligence*

# KIRLIAN IMAGE PREPROCESSING DIAGNOSTIC SYSTEM

Vitaly Vishnevskey, Vladimir Kalmykov, Tatyana Romanenko, Aleksandr Tugaenko

*Abstract: The information technology of Kirlian images preprocessing is developed for decision making support in the telemedicine diagnostic system. Particularly, the image preprocessing includes the selection of objects - fingers emissions in a general image. The algorithms and image processing examples are decrypted.*

*Keywords: information technology, Kirlian image, preprocessing.*

*ACM Classification Keywords: J.3 Life and medical sciences – Medical information systems*

## Introduction

The information technology, processing Kirlian images, are presented to make a decision in the telemedicine diagnostic system. The Kirlian image is discharge gas glow, registered on photo material, arising up near-by the surface of object in the electric field of high tension. Kirlianografy has become widespread in the world as a method of experimental researches. Most interest was caused by researches of biological objects, mainly human kirlianograms. The kirlianograms view depend on the state of human. For example, by the kirlianograms view of hands and feet fingers it is possible to judge about a general level and character of human physiological activity, to estimate the state of his separate systems and to watch after the different influences: preparations, therapies etc. [Pesotskaya,1]

The Kirlian effect is presently the unique instrumental method, allowing on physical and energy information levels to estimate the state of not only an organ or a system, but the entire organism on the whole in interrelation of separate parts with each other.

In a prospect this method is seen as a practical instrument on the table of any doctor. Verifiable and constantly updatable Kirlian images database is needed for development of diagnostics method, using the Kirlian effect. In order to create such database telemedicine information technology is developed.

## Telemedicine information technology

The telemedicine information technology apparatus consists of local and server parts, integrated via Internet. Local part includes of device for Kirlian emission registration, scanner. The special software is assigned to preprocess the images in the interactive mode to save the information in the local database and also to transmit and to save the information in the central database in the protected format. The central database is located in the server side, where information is saved being preprocessed in local parts. The general arrangement of the technology is represented at fig.1.

The chosen configuration allows operatively collecting and accumulating in one center information about patients which are far enough from each other. The large volume of preprocessed data creates the possibility to diagnose large quantity of patients by the several highly skilled specialists and, simultaneously, perfect a diagnostic method.

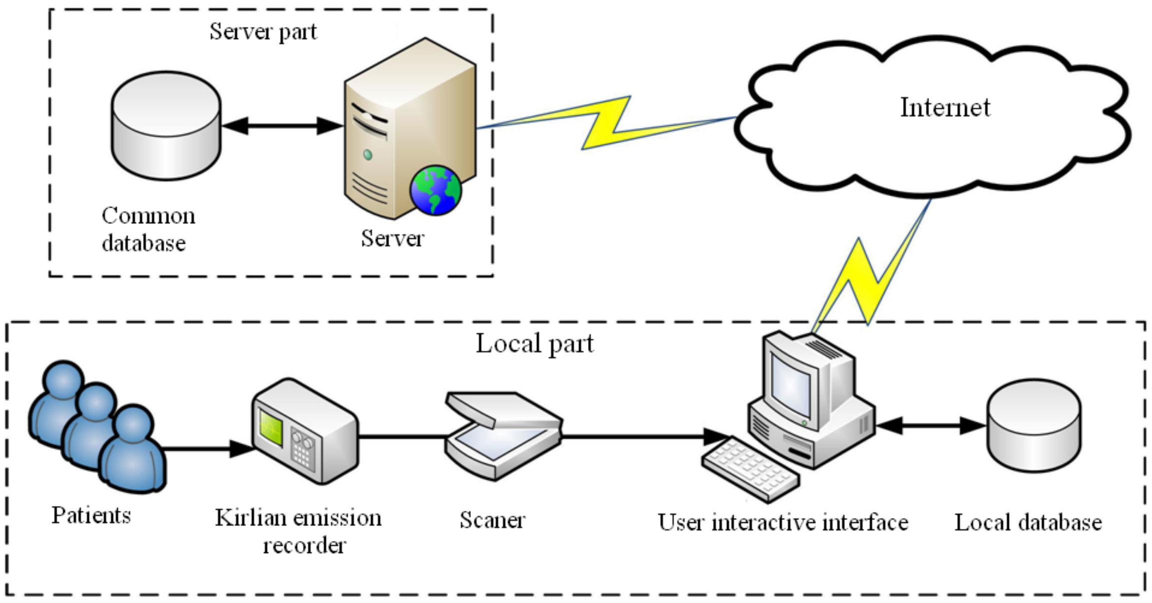


Fig.1.The telemedicine information technology for Kirlian diagnostics

## The preprocessing of Kirlian images in the diagnostic system

The Kirlian images are pictures, got on the special film, by the size of A4, on which emissions as objects are fixed from each of five fingers of both hands (fig.2). It should be noted that there are background variations, noises, which contrast and size are comparable with objects, instability of objects form and contrast. Although on diagnostic content at the present time these images must be considered as binary, however, even the task of such images binarization can not be considered as simple, all the more the tasks of further processing, in particular, the recognition of the objects to be used for diagnostics. The image of every finger emission has the appearance of dark halo, framing a light spot, which form is near to the ellipse. The halo width can be different, even for one image. Often a halo has a not continuous, but interrupted form, and also can consist of separate fragments. A light spot into a halo corresponds to the contact place of finger with film and his brightness corresponds the brightness of image background.

Presently for diagnostic aims standard application software is used, supplied together with a device, realizing the Kirlian method [Korotkov,2]. This device is appropriated for the successive process of emission image fixation separately on every finger (fig.3). The software permits to form the general Kirlianogram of patient using the emissions of ten fingers and to make a diagnosis. However in the successive receiption process of every finger image emission the patient state can substantially change, why a final diagnosis can be distorted. So a one moment Kirlianogram reception for all fingers is more preferable. The possibility appears to research the Kirlianogram sequences and their diagnostic possibilities in the case of the patient state rapid change.  To utilize the available software, it is necessary to segment such images, to select the image of every finger from a general image and turn him, to correspond to vertical direction of finger.

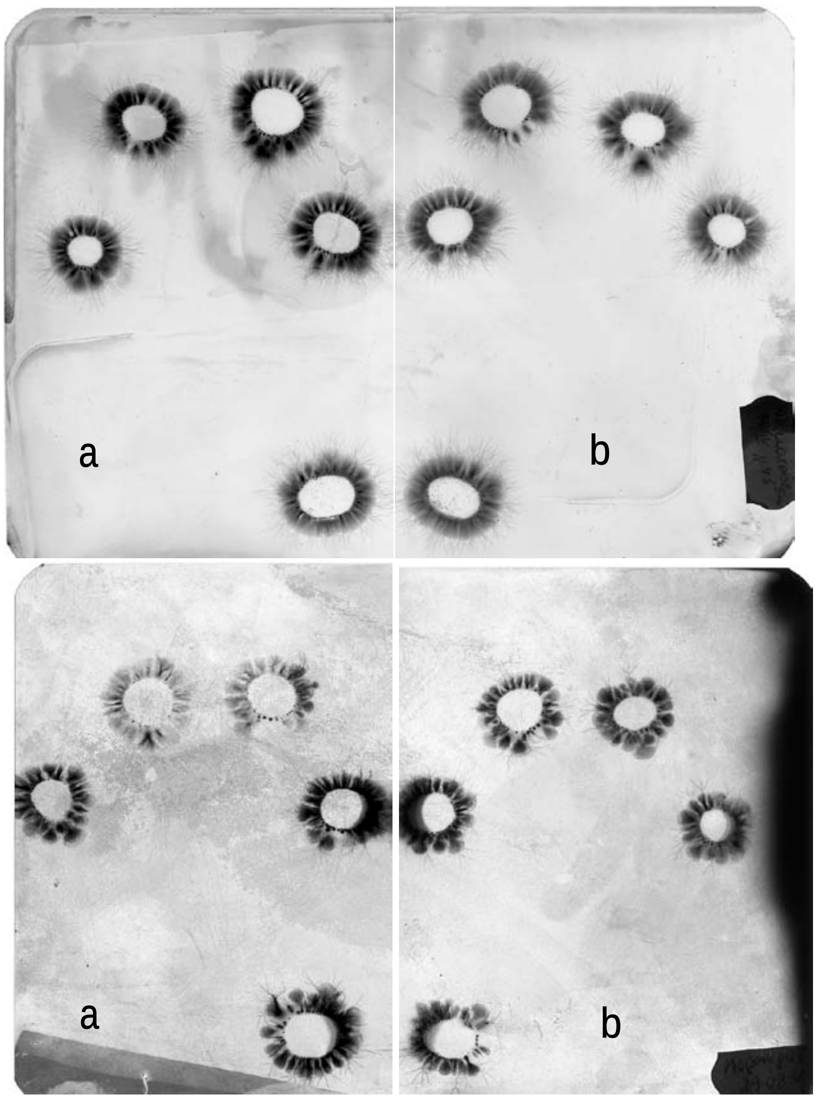


Fig. 2. Kirlian emission images examples:
a - left arm fingers, b - right arm fingers

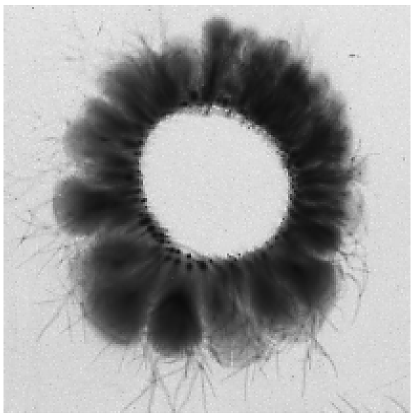


Fig.3. Kirlian emission image of one finger.

## Basic functions and algorithms of Kirlian images preprocessing

While Kirlian images preprocessing semi-automatic segmentation is executed on the emission images of every finger separately and corrected the every finger image orientation automatically. The image emission rotation is fulfilled round a conditional center for every finger image.

Software executes the followings functions:

- the emission file opening, view it for a preliminary estimation by an expert and possible adjustment in the interactive mode in order to delete noises;
- interactive determination of fingers emission location by operator pointing the conditional centers;
- appropriation of number for every fingers emission on their mutual location;
- determination of conditional palm center and of corners to be rotated the fingers emission images;
- determination of rectangles, including separate fingers emissions, taking into account the rotation corner;
- rotation of separate fingers images;
- forming the emission files for every finger;
- the results adjustment in interactive mode.

The most important tasks are the determination of accurate location finger emission, emission boundaries and conditional emission center.

On the basis of the general concepts about Kirlian images, it is expedient to approximate the internal finger emission contour as ellipse, and external emission boundary as circumference, to single out the emission of every finger as image object and to determinate its parameters. The parameters of ellipse are co-ordinates of its center; semi axes sizes and slope angle of large axe. The parameters of circumference, described round the emission of every finger, are a center and diameter. The ellipse and circumference centers coincide. A circumference must envelope all emission (dark halo), on possibility except peripheral striolas - «dendrites». It is assumed that a palm center is on the line segment middle, connecting the centers of the first and fifth fingers emission. The palm center is connected with the all emission centers to determinate the rotation corner of every finger in relation to a vertical line. We will consider that every finger emission image will be a square, described round a circumference that enveloped emission, and rotated on the slope angle of line connecting the center of palm with the center of this emission.

The initial information is a preliminary co-ordinates list of emission centers *of x1l*, *y1l*, *l* =1,5 - five fingers for every palm.

The output information is a parameters list of five circumferences, bounded emissions, parameters list of five ellipses, bounded the emission interior.

The optimization algorithms are used, realizing the gradient descent method to search the parameters of circumferences and ellipses.

For circumferences the optimization parameters are center co-ordinates *of* $x_ц, y_ц$ and radius magnitude *of* $r$. Objective function

$$c = \max_{x_ц, y_ц, r} S_{окр}\text{, при } \sum_{(x-x_ц)^2+(y-y_ц)^2<r^2} \nu(x,y) > \theta \cdot \Psi \qquad (1)$$

where $\theta \approx 0.9$ is some threshold, determined experimentally,

$\nu(x,y)$ is a optical density value of pixel with the co-ordinates *of x,y*;

$\Psi = \sum_{(i-x_ц)^2+(j-y_ц)^2<r_{\max}} \nu(i,j)$ – optical density accumulative value within the circle of maximal radius enveloped emission *of* $r_{max}$ with the center co-ordinates *of* $x_ц, y_ц$

*i,j* are integer co-ordinates of pixel.

For ellipses the optimization parameters are center co-ordinates of $x_ц, y_ц$, big *a* and small *b* semiaxes magnitude and rotation corner of $\alpha$. Objective function

$$c = \max_{x_ц, y_ц, a, b, \alpha} S_{эл}\text{, при } \sum_{\frac{(x-x_ц)^2}{a^2}+\frac{(y-y_ц)^2}{b^2}<1} \nu(x,y) < (1-\theta)\cdot\Psi \qquad (2)$$

The system functioning algorithm consists in the following (fig.4).

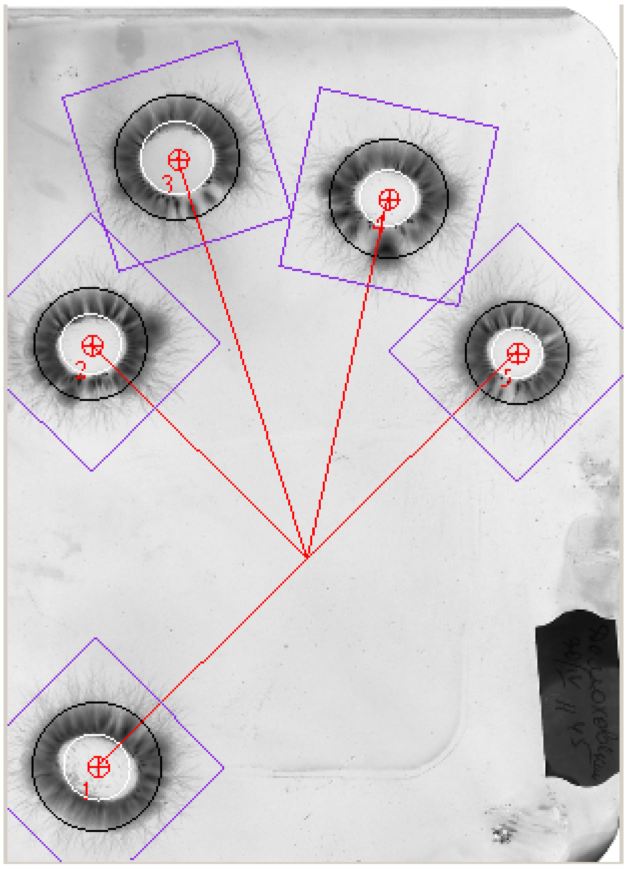


Fig.4 Kirlian image and selected emission images of separate fingers.

1. An operator specifies a cursor on the monitor screen exemplary place of center location for image of every finger. The center location can be indicated approximately, but necessarily must be within the bounds of light spot. The system stores the co-ordinates of the indicated centers.

2. On the indicated co-ordinates the parameters of ellipse and circumference are automatically determined for the image of every finger. If necessary an operator can correct the finger image center, the parameters of ellipse and circumference will be recalculated automatically.

3. The co-ordinates of palm center, values of images rotation corners and parameters of the described squares, boundary the images of separate fingers, are automatically determined.

4. The images of fingers are written down in separate raster files (fig.5.).

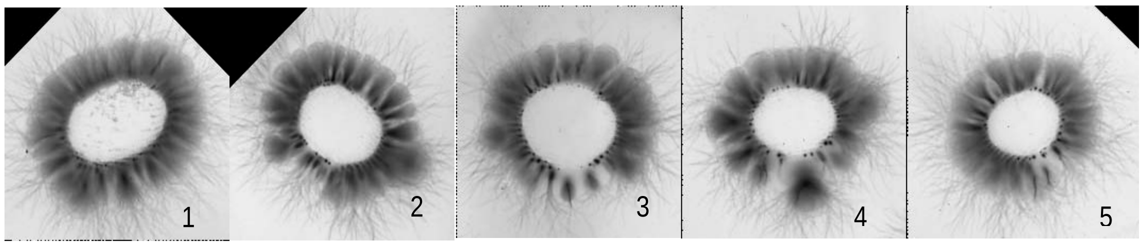


Fig. 5. Separate fingers emission images were selected from the general Kirlian image and formed as separate files.

The ellipse and circumference parameters determination algorithm for every finger consists of the following:

1. The background brightness in the place of every finger emission location is determined. As a background brightness a middle brightness in a square 10×10 pixels is assumed, the square center coincides with the center of finger image.

2. The ellipse parameters of maximal area is determined at the limitations (2), using the gradient descent method.

2.1. In a zero approaching the ellipse center co-ordinates is used, got in the interactive mode, the rotating corner is equal to the zero of degrees in relation to a vertical line.

2.2. Changing the values of parameters *of* $x_u$, $y_u$, $a$, $b$, $\alpha$, determine their values, maximizing the area of ellipse at the limitations (2).

3. The diameter of the enveloped circumference is determined. The center circumference co-ordinates coincide with the co-ordinates of ellipse center.

3.1. As a zero approaching the radius of circumference, equal to the greater semi axe of ellipse, is choose.

3.2. Changing the radius values of $r$, its value, maximizing the circumference area, is determined, using the limitations (1).

## Conclusion

1. The laboratory tests demonstrated the efficiency of the created technology to process the Kirlian images for support a decision in the diagnostic systems.

2. One moment Kirlian images reception allows to make a decision in the diagnostic systems at the rapid changes of the patient organism state, and also to research the possibility of using the Kirlianogram sequences in diagnostic aims.

## Bibliography

[1] Pesotskaya I.A., Kompaniets V.A. Modern Kirlian diagnostics - collected papers Kirlian effect. – Dnepropetrovsk, Dnepropetrovsk center NTI, 2008. p. 9-15.

[2] Korotkov K.G. Bases of GRV biotelectrography. - SPb, Izd-vo SPbGITMO, 2001. 360c.

## Authors' Information

*Vitaly Vishnevsky - head of division, senior researcher, candidate of engineering sciences, Institute of problems of mathematical machines and systems, prosp. akad. Glushkova 42, 03680, Kiev 187, Ukraine; e-mail: vit@immsp.kiev.ua*

*Major Fields of Scientific Research: Information technologies, Decision support systems*

*Vladimir Kalmykov - senior researcher, candidate of engineering sciences, Institute of problems of mathematical machines and systems, prosp. akad. Glushkova 42, 03680, Kiev 187, Ukraine; e-mail: vl.kalmykov@gmai.com*

*Major Fields of Scientific Research: Image processing, Visual information systems*

*Tatyana Romanenko - junior research worker, Institute of Mathematical Mashines and Systems, 42 Academician Glushkov St., 03680, Kiev 187, Ukraine;*

*e-mail: romanenko@immsp.kiev.ua*

*Major Fields of Scientific Research: Software technologies, Information technologies and image processing*

*Aleksandr Tugaenko - junior research worker, Institute of Mathematical Mashines and Systems, 42 Academician Glushkov St., 03680, Kiev 187, Ukraine;*

*e-mail: tugayenko@gmail.com*

*Major Fields of Scientific Research: Software technologies, Information technologies and data bases*

# TABLE OF CONTENTS OF IJ ITA, VOLUME 16, NUMBER 2