Multi-Agent Information Fusion: Methodology, Architecture and Software Tool for Learning of Object and Situation Assessment

V. Gorodetsky
SPIIRAS, 39, 14-th
Liniya, St.Petersburg,
199178, Russia
gor@iias.spb.su

O. Karsaev
SPIIRAS, 39, 14-th
Liniya, St.Petersburg,
199178, Russia
ok@iias.spb.su

I. Kotenko
SPIIRAS, 39, 14-th
Liniya, St.Petersburg,
199178, Russia
ivkote@iias.spb.su

V. Samoilov
SPIIRAS, 39, 14-th
Liniya, St.Petersburg,
199178, Russia
samovl@iias.spb.su

Abstract - *The paper presents the developed design methodology, architecture, technology, and software tool supporting the design, implementation and deployment of multi-agent systems for object and situation assessment. The above issues are demonstrated through multi-agent anomaly detection system operating on the basis of multiple heterogeneous data sources including streams of temporal data.*

Key words: Multi-agent system, information fusion, on-line update of situation assessment, software tool.

1 Introduction

JDL model considers hierarchy of tasks associated with situation awareness problem that concerns with in-depth comprehension, prediction, and management of what is going on within a system and/or environment of interest. Situational awareness is *situation-centric* problem, whose most important subtasks are *Object Assessment* (OA) referred to as Data Fusion and *Situation Assessment* (SA) referred to as Information Fusion. Both above subtasks are now of great concern and the subjects of intensive research.

Situation is understood as a complex system constituted of a set of semi -autonomous objects ("*situation objects*") having certain goals and operating to achieve a common goal. A "situation object" can be either physical (e.g., group of aircrafts involved in a mission), or "abstraction" (e.g., a component of software in which the traces of attacks are manifested). Any object and situation are characterized by "*state*" taking values from a finite set of labels. Both tasks, OA and SA, are classification tasks aiming to map labels to the current states of objects and situation respectively based on data perceived by multiple sensors.

Most of previous and current research deals with OA, whereas SA task is paid a noticeably less attention. This paper is devoted to both OA and SA tasks. Respectively, Sec. 2 states that the distinctions between OA and SA tasks can be understood while analyzing the distinctions between *data models* specifying states of situation objects and situation states and discuss these distinctions. This discussion is important due to the fact that it makes it possible to understand in what aspects the existing methodology of OA has to be extended to meet the SA peculiarities. The key procedure of this methodology is distributed learning of distributed classification. Its description and also engineering and implementation issues of both OA and SA systems are presented in sec.3. Sec. 4 describes the developed generic multi-agent architecture of a software system provided with learning capabilities destined for solving of both OA and SA tasks. Sec. 5 outlines the developed technology and software tool supporting design and implementation of multi-agent OA and SA systems. Hereinafter the last system is called multi-agent information fusion systems (IF MAS).

The proposed methodology, architecture, technology and software tool were validated based on a case study that is anomaly detection in computer network, which is an application of great concern. Experimental results demonstrating and evaluating proposed IF MAS learning methodology and also advocating in favor of the proposed technology and supporting software tool are described in sec. 6. Conclusion outlines the paper novel results and future work.

2 Object and Situation Assessment

In general case difference between notions of "*object*" and "*situation*" is rather vague. For example, if we consider a *multi-level* data processing within a situational awareness problem, an object of an intermediate data processing level can play the role of "situation" with regard to its adjacent lower level. To understand the difference between SA and OA tasks, it is necessary to compare the *data models* used for specifications of objects and situation states that, in turn, determine difference between OA and SA tasks.

A situation state is constituted by states of situation objects and meaningful relationships among them. The existence of *relationships* between objects, whose values affect on the class of situation state, is the *first* essential *distinctive feature* of SA, since objects are normally specified only in terms of features. These relationships can have diverse nature: spatial, temporal, ordering, etc., and importance of relationships is highly application–dependent. Temporal relationships normally reflect events occurring with particular objects. Examples of such events are changing state of an object or an event occurring with a subset of situation objects (for example, completion of an aircraft refueling, location of a subset of situation objects in a predefined region, etc.). Other relationships among objects can concern their spatial configuration. In some cases dynamic relationships are of the most importance ("objects are approaching to each other"). The existence of such relationships leads to a considerable peculiarity of the data model specifying a situation state. For example, absence of information about an object can lead to missing values of certain relationships what leads to the situation state data model containing *missing* values.

The second important *peculiarity* of the situation data model is that information about situation objects is collected by different means and at different times. Situation is a notion of dynamic nature and that is why input information can have different time stamps. As a rule, objects possess different dynamics and therefore components of the collected information possess different *life time*. Combining such information is a theoretical problem that is researched not well.

The third peculiarity of the data model specifying situation state is that input from different sources and different objects are not synchronized. Accordingly, instants of transitions of objects in new states can not coincide with the instants of SA update. This issue results in the fact that input data of SA *system* are represented *as asynchronous data streams* and at the time of SA update different components of data structure specifying situation state are marked by different time stamps. This can lead to the information ageing at the SA update times. In the simplified model, this case can be reduced to the tasks of SA training and update based on data with missing values.

Additionally, certain information specifying situation state can be not collected or lost. For example, airborne observation system can fail due to meteorological factors or masking. This makes the SA task especially hard, because this factor increases the number of missing values.

The main conclusion entailed from the above analysis is that data formal model specifying the state of a situation snapshot is represented in terms of asynchronous data streams with missing values. In contrast, OA task deals mainly with a "more static" data. Thus, it is necessary to consider the SA problem statement in which situation state data model is represented as a number of data streams containing missing values. The last aspect makes the SA task much more complex compared with OA task.

3 Methodology of Information Fusion and Information Fusion Learning

Information Fusion (IF) systems produce decisions based on input data distributed over many sources, measured in various scales, containing missing values, etc. These factors considerably influence on IF methodology. As a rule, knowledge used for IF is obtained via distributed data mining that is one of the most sophisticated task of IF methodology and technology. Experience shows that both tasks, learning of IF and IF procedure itself, are closely interconnected and must be considered in common at all steps of IF systems design. Let us outline the developed methodology of IF design, indicating how and in which order the design solutions are made.

1. Basic principle of data and information fusion. This principle determines how to allocate data and information processing functions to data source-based level and meta–level. There exist several approaches to information fusion [1]. In the developed methodology we use two-level information fusion architecture in which source-based mechanisms produce local classifications of situation states and then these decisions are combined at meta–level. Such a model is advantageous in many respects, in particular, (1) it considerably decreases communication overhead; (2) it is applicable in the cases if data of particular sources are heterogeneous (to the upper level only the local decisions are forwarded, and they are either binary or categorical); (3) there exist effective and efficient algorithms for combining such decisions in upper level, and (4) it preserves the source data privacy.

2. Structure of data and information fusion in SA system. This structure is called hereinafter *Information Fusion meta-model*, *IF meta–model*. It comprises three types of structures that are (1) source-based decision making structures (practically, they correspond to particular situation objects assessment tasks), (2) meta-model of decision combining, and (3) *classification tree*. Let us describe these structures and their compositions within IF meta–model.

It is supposed that *source-based decision making structure ("source-based decision making tree")* can contain one or several classifiers *(base classifiers)*. Such classifiers can produce decisions on the basis of the same or different attributes of source. They can be trained through the same or different datasets. In this respect, the developed methodology admits to use a variety of approaches. In the simplest case if dimensionality of data source attribute vector is small and attributes are more or less homogeneous (e.g. they are measured either in numerical or in discrete scales) then it can be reasonable to use single base classifier, whose decision is forwarded to meta–level. In this case source-based decision making structure consists of single base classifier. In more complicate cases, it could be reasonable to use several base classifiers producing classifications based on different subsets of attributes, trained via use of different training datasets, etc. The developed methodology and supporting software tool admit such possibilities. The decisions of these classifiers are forwarded to the meta–level for combining with decisions produced by base classifiers of other sources. An alternative is to combine them locally and forward the result to the meta–level.

Meta-model of decision combining is formed by the set of base classifiers of data sources, meta–classifier(s) and structure given over the aforementioned classifiers. At the meta–level, the system combines decisions received from source-based classifiers. An example of meta-model of decision combining is given in Fig.1 (lower part).

Meta-model of classification ("*classification tree"*) is a component of *IF meta–model* that is used to reduce multiple classification task to a number of binary ("*pair wise*") classifications. The nodes of the *classification tree* that are not leaves are called *meta-classes*. An example of

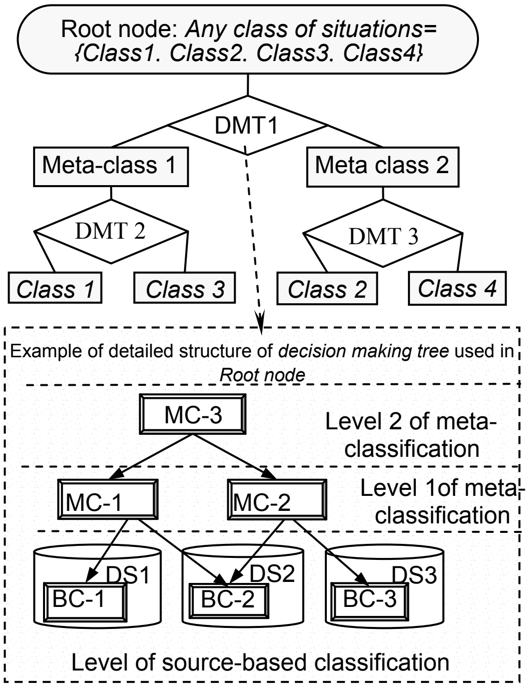

Root node: *Any class of situations= {Class1. Class2. Class3. Class4}*
DMT1
Meta-class 1
Meta class 2
DMT 2
DMT 3
Class 1
Class 3
Class 2
Class 4
Example of detailed structure of *decision making tree* used in *Root node*
MC-3
Level 2 of meta-classification
MC-1
MC-2
Level 1of meta-classification
DS1
DS2
DS3
BC-1
BC-2
BC-3
Level of source-based classification

Figure 1. Example of IF meta–model composing classification and decision making trees.
"*DMT*" – *Decision making tree*; "*DS*"-*Data source*; "*BC*"– *Base classifier*; "*MC*"-*meta–classifier*

a classification tree of SA is given in Fig.1 (upper part). In this tree, the root node corresponds to the *meta–class* containing all possible situation states. Classification in this node discriminates situation states of classes 1 and 3 from those of classes 2 and 4, i.e. it discriminates instances of meta–class 1 from those of meta–class 2. The above three structures compose what is called *"IF meta–model"* as follows. Each node of *classification tree* is mapped a particular classification task. It is solved according to a structure that is composed of *decision making trees* of sources and meta-model of decision combining. Such a *structure* can be different for different nodes of *classification tree* (Fig.1).

3. Structure of IF system distributed knowledge base. Knowledge base (KB) of IF system is distributed and consists of ontology and KBs of particular classifiers. Let us note that the structure of KB is definitely determined by the structure *IF meta–model*. IF system KB is *distributed* over hosts according to locations of its particular classifiers. Other peculiarity of KB is that its distributed components can have *heterogeneous representation structures* that leads to the necessity to resolve several specific problems. The *first* of them is to provide IF system components with a *shared thesaurus* needed for monosemantic understanding of the shared notions. This problem arises in case of privacy of local data when specifications of data of different sources can be independently developed by several experts and the latter can denote different notions by the same identifiers and vice versa. Next problem called *non-coherency of data measurement scales* comes out of the fact that the attributes of different sources can be overlapping and the same attribute can be measured differently in different sources. Nevertheless, they must be used equally in all IF procedures. That is why it is necessary to provide distributed data with *consistent measurements*. The third problem is called "*entity instance identification problem*" [1] and is entailed by the fact that complete specification of an input of SA system is composed of the fragments. Therefore, a mechanism to identify these data fragments representing certain instance of a situation state is needed.

In the developed methodology the above problems are effectively resolved through ontology-centered distributed knowledge representation. The focus of this approach is to explicitly represent all the notions of IF KB and meaningful relationships among them in terms of a top-level component of KB shared by all agents. This component of KB is called *shared ontology*. It provides consistent use and unique interpretations of terminology by all entities of IF system including consistent understanding of messages they exchange with.

4. Particular techniques used for learning of IF system distributed KB. The developed methodology (and software tool supporting it) uses so far three different techniques for base and meta-classifiers training. They are destined for extraction of production and association rules from databases. Let us only indicate these techniques.

Visual Analytical Mining is a technique destined for extraction of production rules and/or decision trees from datasets containing numerical and/or linear ordered attributes [2]. It is capable to extract production rules specified in terms of the first order logic fragment without quantifiers. *GK2* is a technique for extraction of production rules from data measured in discrete scales [3]. This technique is conceptually close to the well known AQ technique [4], but uses different algorithm for extraction of *maximally general rules*. It is also used for extraction rules from datasets with missing values. Also *FP-grows* algorithm for association rule mining is used.

The developed methodology admits to use two methods for learning of *decision combining*. They are (1) meta-learning algorithms based on *stacked generalization* and (2) Meta-learning algorithms based on *classifiers' competence evaluation*. The idea of the first one is to use the results of classifications (records of labels of classes) produced by base classifiers over a training dataset for training and testing of meta-classifier. The second, *competence-based*, method uses a procedure called "*referee*" associated with each classifier to assess its competence with regard to particular data instance. To provide the referee with such an ability, a learning procedure is used.

5. Methodology of allocation of training and testing datasets to classifiers. In case of IF learning, this task possesses a number of peculiarities that are not pertained to usual training and testing. *The first peculiarity* is that in many cases several base classifiers may be required even if a single data source is used as input. Therefore, it is necessary to use special means to split training and testing dataset of a source to allocate the samples of training and testing data to different base classifiers. This aspect is especially critical if the size of learning datasets is limited. In this case, an increase of decision making accuracy can be achieved through use of different learning techniques and respectively, several classifiers, which decisions must be further combined. *The second peculiarity* is caused by the necessity to use at least two-level SA scheme what requires reserving a certain part of learning dataset to produce meta–data used for meta–level learning.

The above peculiarities lead to the conclusion that allocation of training and testing datasets to classifiers and also training and testing samples to both base– and meta–level classifiers are very interdependent tasks and therefore training and testing procedures must be coordinated in a certain way and managed accordingly. In the developed methodology such coordination is achieved by use of a number of predefined protocols.

4 Multi-agent Architecture

For implementation of the developed methodology of IF, multi-agent architecture is used. There are several reasons to prefer such architecture. Actually, multi-agent system (MAS) represents an advantageous paradigm for analysis, design and implementation of complex systems. It proposes powerful metaphors for distributed system analysis and design, techniques and technologies specifically destined for large scale intelligent systems.

IF systems definitely fall into the class of potential MAS applications. Indeed, each IF application is naturally distributed: data sources are distributed; data processing is performed in distributed manner, the systems and/or users using the results produced by an IF system are distributed. If data of different sources are private or classified (military, commercial, etc.) they are not available for a centralized processing. However, data holders can make these data available to agents processing private data without revealing their content. Additionally, as a rule, IF systems are of large scale and naturally decomposable and thus they take the most of advantages provided by MAS architecture. Recent research shows growing of popularity of multi-agent paradigm for IF systems where it assesses as very promising.

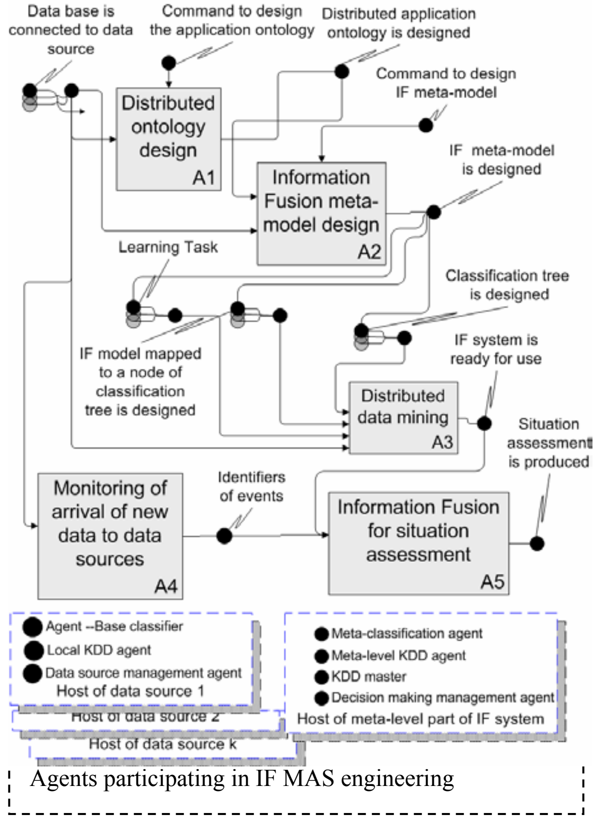

Data base is connected to data source
Command to design the application ontology
Distributed application ontology is designed
Command to design IF meta-model
Distributed ontology design
A1
Information Fusion meta-model design
A2
IF meta-model is designed
Learning Task
Classification tree is designed
IF model mapped to a node of classification tree is designed
IF system is ready for use
Distributed data mining
A3
Situation assessment is produced
Monitoring of arrival of new data to data sources
A4
Identifiers of events
Information Fusion for situation assessment
A5
Agent --Base classifier
Local KDD agent
Data source management agent
Host of data source 1
Host of data source 2
Host of data source k
Meta-classification agent
Meta-level KDD agent
KDD master
Decision making management agent
Host of meta-level part of IF system
Agents participating in IF MAS engineering

Figure 2. High level protocol of IF MAS engineering

The developed architecture [5] comprises two types of components. The components of the first type operate with the source-based data and situated at the same hosts as the respective sources, whereas he components of the second type operate with meta-information and can be situated in any host. Let us consider firstly the source-based components and their functionalities.

Data source managing agent

- Participates in the distributed design of the shared component of the application ontology;
- Collaborates with meta-level agents in management of training and testing of particular source-based classifiers and in forming meta-data sample;
- Supports gateway from ontology to databases through transformation of queries from the ontology language into, e.g., SQL language.

Data Source Knowledge Discovery (KDD) agent

It trains the source-based classification agents and assesses the resulting classifier's quality.

Classification agents of data source

They produce decisions based ob source data and are the subjects of training performed by KDD-agents.

Server of learning method comprises a multitude of classes implementing particular KDD methods (see sec. 3).

The meta-level components of IF MAS and their functions are as described below.

Meta-Learning agent ("KDD Master")

- Manages the distributed design of IF MAS application ontology;
- Computes meta-data sample;
- Manages design of meta-model of SA.

Meta-level KDD agent

- Trains and tests thef meta-level classification agent and assesses its quality.

Decision making management agent

- Coordinates operation of *Agent-classifier of meta-level* and *Meta-level KDD agent*.

5 Technology and Software Tool

The developed technology for IF MAS design is supported by two components of the software tool developed by authors. The first of them, *Multi-Agent System Development Kit* (MASDK) [6], is used for design and implementation of application–independent components of applied IF MAS, whereas the second one, *Information Fusion Design Toolkit* (IFDT), is destined for design of its application–dependent components [5]. Thus, MASDK supports design and implementation of such IF MAS components: agent classes and its instances, problem and application ontology and communication environment. It is also used for design and implementation of the basic structures of data and knowledge bases of agents and their behavioral components represented in terms of state machines [6]. MASDK software is responsible for deployment of the implemented IF MAS over given computer network.

IFDT software tool supports design operations aiming at specialization of certain application-oriented components of IF MAS. The subjects of specialization are shared ontology, structure (meta-model) of distributed decision making (the latter also determines the structure of IF learning and decision making), and training and testing techniques selected for use in learning procedures [5, 7, 8].

It is supposed that IF MAS is designed in distributed mode by several designers. Coordination of their activities is supported by a special set of protocols. High-level view of this technology and respective sub-protocols names used in subsequent design steps are presented in Fig.2 in terms of standard IDEF0 diagram. It comprises sub-protocols ordered as indicated below:

A1. Distributed ontology design.
A2. Decision making and IF meta-model design.
A3. Distributed data mining.
A4. Monitoring of new data input from sources.
A5. Information fusion for situation assessment (distributed decision making).

The diagram presents interaction, intermediate and final results and activities order. The most important protocols are A1, A2 and A3 [5, 8].

6 Experimental Results: Multi-agent Anomaly Detection Learning System

Several case studies were used for validation of the developed methodology, multi-agent architecture, technology and software tool for learning of OA and SA. Some of them (mostly pertained to OA) were described in [5, 8]. Below we describe a case study, which differs in several aspects from the previously published. Particularly, this case study uses more data sources; training and testing datasets used are largely heterogeneous, for example, they are represented as asynchronous data streams and include temporal data sequences to mine, the training and testing datasets contain up to 30% of missing values, etc. We talk about the task of learning of intrusion detection that is a task of great concern now.

Below we first describe the training and testing datasets and then present the results of testing of the multi-agent anomaly detection system developed and implemented by use of the methodology, technology an software tool described in this and previous sections. For comparison, two data sets containing no and 30% of missing values were used.

6.1 Training and testing data sets

We considered anomaly detection task. Along with the dataset of instances of the security status "*Normal*", the class "*Abnormal*" was considered. The dataset corresponding to the last case comprises the instances of *four types of attacks* against computer network categories: *Probing*, *Remote to local* (*R2L*), *Denial of service* (*DOS*) and *User to root* (*U2R*). The instances of attacks of each type included in the case study are *SYN-scan*, *FTP-crack*, *SYN flood*, and *PipeUpAdmin* respectively.

The following three data sources were used:

- network-based (input and output traffic),
- host-based , i.e. operating system (OS) log and
- application–based (FTP-server log).

Data of each source are represented by four *generic structures* resulted from raw data processing. They are identical for all the sources:

1. *Data streams of vectors of binary sequences specifying significant events of the respective level*. Practically, this data structure specifies discrete binary time series. Mining of such data at source-based level was performed by use of a specific technique developed by authors that is based on correlation and regression ideas[1]. The components of this vector in input traffic level are constituted by different parameters of packet headers. Features of operating system and application levels are constituted by the OS and FTP events numbers.

2. *Statistical attributes of connections (sessions of users) manifested in a data source*. As features of the traffic level we used, for example, length, status and total number of connection packets, etc. The set of features of OS and application levels includes the number of "hot indicators" (e.g., access to system directories, creation and execution of programs, etc.), number of failed logins, etc.

3. *Statistical attributes of traffic during the short time intervals*. The features of the traffic level include the numbers of connections and services of different types. The set of features of OS level includes selected processor queue length, number of threads in the computer, combined rate of file system read requests rate, at which packets are sent by the computer; percentage of elapsed time spent by processor to execute a non-Idle thread, etc. Features of application level are similar to ones corresponding to the above data structure.

4. *Statistical attributes of traffic (users' activity) for long time intervals*. The set of features of the traffic level includes the total amount of different types of connections, whereas the sets of features corresponding to the OS and application levels are the same as for the short time interval (see above).

Data structures of the traffic level were produced via processing of *tcpdump/windump* data. *TCPtrace* utility was used, as well as several ad-hoc programs developed by authors. The data structures of the OS level were generated via processing of operating system log *Security* (for *Windows 2000/XP*). The data structures of application level (*FTP-server*) were produced on the basis of d-hoc procedures processing the *FTP-server log*.

Meta-classification procedure peculiarities are entailed by the fact that anomaly detection system is a real-time one and its base classifiers make their decisions regarding the status of the same user activity in asynchronous mode. This entails certain peculiarities of both training and testing meta–data sets and meta-classification procedure.

Let us consider the question of how meta-data are computed and what new problems have to be resolved in these computations. While using meta–classification approach, the meta-data are composed of the decisions of the base classifiers, which decisions are combined in

[1] Its description is out of the scope of this paper.

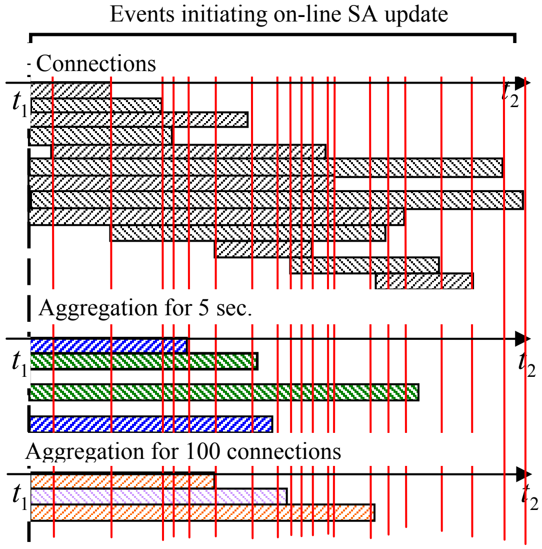

Events initiating on-line SA update
Connections
t_1
t_2
Aggregation for 5 sec.
t_1
t_2
Aggregation for 100 connections
t_1
t_2

Figure 3. Meta-data model used for on-line SA update. Vertical lines marks the SA update times.

meta–level by meta–classifier. Since anomaly detection system is a real-time one and outputs of classifiers are presented as flows of decisions, the *time of occurrence of certain events* can be used as an attribute needed for identification of the decisions that can be added to meta-data features. An event is understood as appearance of a new decision produced by base classifiers output decisions stream and this decision can be represented as a message sent from the source-based level to the meta-level with content <*Decision of base classifier X, Time*>.

Each base classifier produces its decision at the time when it receives all the needed data. Other situation takes place for meta-classifier. Fig.3 demonstrates asynchronous nature of the arrivals of decisions produced by base classifiers: different base classifiers produce their decisions at different times. This entails the main specificity of decision combining in the SA task. To combine decisions having different time stamps, we can act in two modes: (1) to wait when all base classifiers produce the decisions and then combine them, or (2) update the combined decision when a new decision arrives. According to the real life requirements update of SA must be done on-line that is at the time when a new portion of input information is received by meta-classifier. Exactly this approach to assessment of the situation state was used in the developed software prototype of the anomaly detection system based on the developed case study.

We used two strategies of meta–classification based on decision streams produced by base classifiers. In both of them meta-classification took into account the fact that some of previously produced decisions are already non relevant to the situation ("too archaic"). To take this fact into account, we introduced for the decisions of each base classifier so-called *life time* and if current time interval elapsed from the moment of its producing exceed this life time, then in the meta-data the value of this decision is reckoned as missing. Therefore, in this case meta-classification is performed based on data with missing values. In the second case we ignored ageing of the decisions of base classifiers and, thus, while performing meta-classification, we deal with complete vectors of decisions. It was done to compare the qualities of SA in both cases. Respectively, we describe below the results of two kinds of experiments.

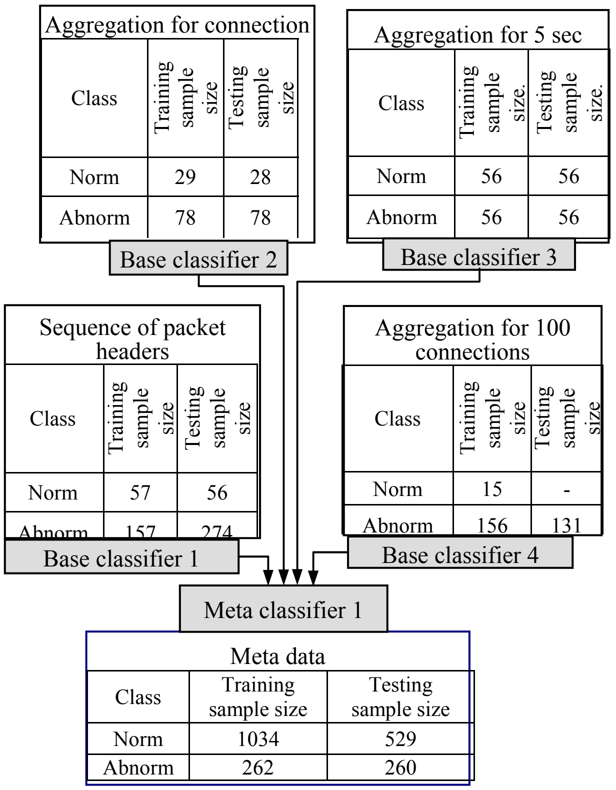
6.2 Experiments description

In the first type of experiments with the software prototype of anomaly detection system aiming at learning of anomaly detection and on-line SA update of the computer network security status we considered training and testing based on datasets without missing values. In this case the finite value of "life time" of input was ignored

The structure of the *Decision making tree* and properties of training and testing samples of traffic level are given in Fig.4. In the meta-level the decisions produced based on particular data sources were combined by use of *meta-classification approach.*

The results of the first group of experiments with the developed multi-agent anomaly detection system (learned IF MAS) is presented in Fig.5 by histograms of probability of correct decision making with regard to status of security situation and also for false positives (false alarms) and false negatives (missing of signals). The presented results illustrate qualitatively the satisfactory performance of the designed IF MAS and quality of its learning.

It is worthy to note that use of the developed methodology, technology and software tool presented in



Aggregation for connection

Class	Training sample size	Testing sample size
Norm	29	28
Abnorm	78	78

Base classifier 2

Aggregation for 5 sec

Class	Training sample size.	Testing sample size.
Norm	56	56
Abnorm	56	56

Base classifier 3

Sequence of packet headers

Class	Training sample size	Testing sample size
Norm	57	56
Abnorm	157	274

Base classifier 1

Aggregation for 100 connections

Class	Training sample size	Testing sample size
Norm	15	-
Abnorm	156	131

Base classifier 4

Meta classifier 1

Meta data

Class	Training sample size	Testing sample size
Norm	1034	529
Abnorm	262	260

Figure 4. Structure of decision making and structure of training and testing data at Network level

the paper made it possible to design and implement the above software prototype of anomaly detection system for less than 1 month effort of the qualified specialist.

6.3 Situation Assessment Issues

In the second group of experiments we dealt with the same application and used the same architecture of decision making and combining as in the first group of experiments. The difference was that the training and testing datasets contained 30% of missing values resulting from the fact that input data was considered as asynchronous data stream with finite life time of input data at that this life time was different for different data streams. Let us note that this case corresponds to a weakly explored scope from both learning and classification viewpoints.

General idea of the mining technique that was used in our experiments was developed by authors and published in [9]. Let us outline only its idea. Let training dataset contain missing values. If we assigned all the missing values in the training dataset in a way, we would be able to extract rules by use of an existing algorithm (e.g. algorithm presented in [3] or in [4]). Different assignments of the negative and positive examples would lead to different sets of extracted rules. Different assignments of the positive examples will also lead to different values of coverage factors [3] of the extracted rules. It was proved in [9] that for each positive example chosen as *seed* [3] there exist two special variants of the missing values assignments that correspond to extraction of two sets of *maximally general rules* and these sets can serve as *lower* and *upper* bounds for all sets of maximally general rules corresponding to all possible assignments of the given training dataset and given seed. These bound meet the following deducibility relations:

$$\boldsymbol{Z}^{+}_{low} \subseteq \boldsymbol{Z}^{+}_{*} \subseteq \boldsymbol{Z}^{+}_{upper} \quad ,$$

where $\boldsymbol{Z}^{+}_{low}$ – the *lower* bound of all the sets of maximally general rules; $\boldsymbol{Z}^{+}_{upper}$ –the *upper* bound of all the sets of *maximally general rules,* and $\boldsymbol{Z}^{+}_{*}$ – the set of maximally general rules for any arbitrary missing value assignments.

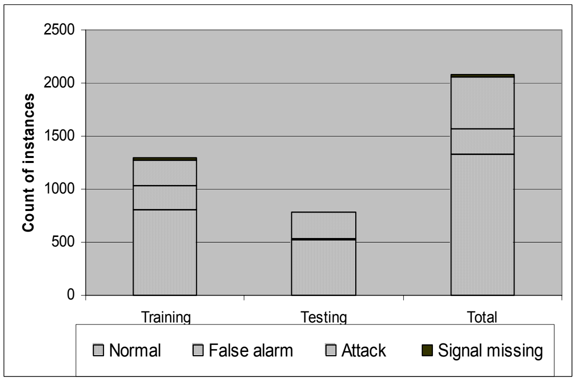


Figure 5. The testing results of anomaly detection based on traffic data: The probability of correct anomaly detection on testing data sample is equal to 0,98

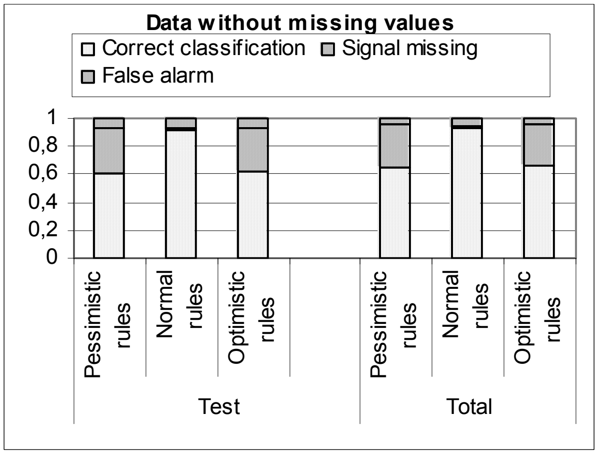


Figure 6. Results of testing of the rules extracted from dataset *without* and *with* missing values for "optimistic" and "pessimistic" case on complete testing dataset

Informally, $\boldsymbol{Z}^{+}_{upper}$ bound corresponds to the "*optimistic*" assignment of the missing values, whereas $\boldsymbol{Z}^{+}_{low}$ corresponds to the "*pessimistic*" one. The motivation why they can be interpreted as *optimistic* and *pessimistic* assignment can be found in the paper of authors [9]. An algorithm for their computation was proposed in [9].

Thus, the above idea and respective algorithm were used for learning and on-line update of the security status within the developed software prototype and in experiments based on use of training data with missing values. It is worthy to note that the training data model used in these experiments corresponds to the data model of a SA task but not OA (see sec. 2).

The results of testing of the resulting multi-agent anomaly detection system with the necessary explanations

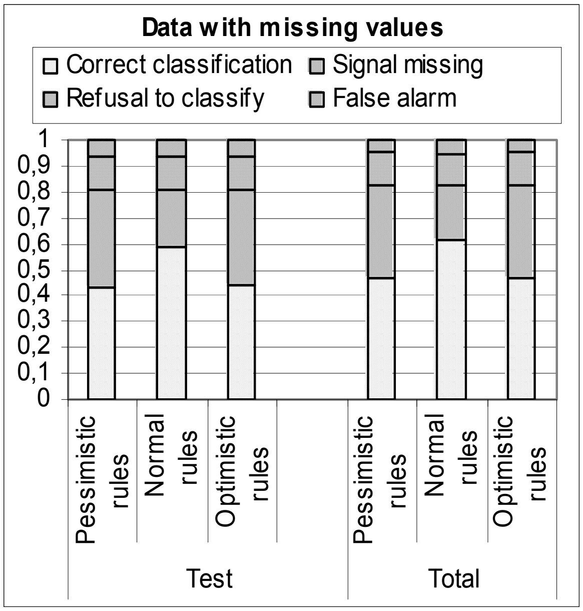


Figure 7. Results of testing of the rules extracted from dataset without and with missing values

are presented in Fig.6 and Fig.7. These results demonstrate that even without any assumptions concerning missing values, the developed IF MAS can be satisfactory learnt.

It should be noted here that in this experiment we used only the traffic level data sources.

7 Conclusions

The paper novel contribution is the developed and completely implemented methodology, technology, multi-agent architecture, and software tool for design, implementation and deployment of Information Fusion systems aiming to solve of the tasks of levels 1 and 2 of JDL model that are OA and SA tasks. The above methodology and software tool were validated via experimental exploration of the developed multi-agent anomaly detection system using multiple data sources. A novel result of the paper is also the proposed technique for on-line SA update if input is represented as asynchronous data streams of decisions produced by source-based classifiers. The results presented justify positive evaluation of the proposed methodology, architecture, technology and software tool as applied to both OA and SA tasks.

Future research will concern the further validation of the proposed methodology and software tool on various applications from SA scope for more real-life models of inputs.

Acknowledgements

We wish to thank European Office of Aerospace Research and Development of the USAF (Project 1993P) and Russian Foundation for Basic Research (grant # 04-01-00494) for support of this research.

References

[1] I.Goodman, R.Mahler, and H.Nguen. *Mathematics of Data Fusion.* Kluwer Academic Publishers, 1997.

[2] V.Gorodetski, V.Skormin, L.Popyack. Data mining technology for failure prognostics of avionics, *Int. J. "IEEE Transactions on Aerospace and Electronic Systems*.". 38 (2), 388-403, 2002.

[3] V.Gorodetski and O.Karsayev. Algorithm of rule extraction from learning data. *Proc. of 8th International Conference "Expert Systems & Artificial Intelligence"* (*EXPERSYS-96*), Paris, France, 133-138, 1996.

[4] R.S.Michalski. A Theory and methodology of inductive learning. *Machine Learning,* 1, Eds. J.G.Carbonel, R.S.Michalski and T.M.Mitchel, Tigoda, Palo Alto, 83-134, 1983.

[5] V.Gorodetsky, O.Karsaeyv, and V.Samoilov. Distributed learning of information fusion: A Multi-agent approach. *Proc. of the International Conference "Fusion 03"*, Australia, 318-325, 2003.

[6] V.Gorodetski, O.Karsaev, I.Kotenko, A.Khabalov. Software development kit for multi-agent systems design and implementation. *Lecture Notes in Artificial Intelligence,* Volume 2296, 121-130, 2002.

[7] V.Gorodetsky, O.Karsaeyv, and V.Samoilov. Multi-agent technology for distributed data mining and classification. *Proc. of the "IEEE Conference Intelligent Agent Technology" (IAT03),* Halifax, Canada, 438-441, 2003.

[8] V.Gorodetsky, O.Karsaeyv, and V.Samoilov. Software tool for agent-based distributed data mining. *Proc. of the IEEE Conference "Knowledge Intensive Multi-agent Systems" (KIMAS 03*), Boston, USA, 710-715, 2003.

[9] V.Gorodetski, O.Karsayev. Mining of data with missing values: A Lattice-based approach. Proc. of *International Workshop "Foundation of Data Mining and Discovery"*, Japan, 151-156 2002.