VERSITA
Information Technology and Management Science
doi: 10.2478/v10313-012-0028-1
2012 /15

Approaches to the Construction of Behavioural Patterns of Information System Users

Pavels Osipovs[1], Arkady Borisov[2], [1-2]*Riga Technical University*

***Abstract* – The paper describes the methodology of constructing models of typical user behaviour in a distributed information system, which may operate with sensitive data. The model is designed for the detection of abnormal behaviour occurring during the invasion of an intruder in the system. Also, the paper deals with the general approach to the implementation of the model infrastructure and algorithms of the main modules. Moreover, two possible methods for implementing the model in the target system are described.**

***Keywords* – distributed information system, anomaly detection, user behaviour model, Python, e-Health, HL7, SOAP**

I. Introduction

This article describes the methodology of solving the problem of constructing a module of User Behaviour Model (UBM) of e-health information system [1]. The target system is an integration platform that combines a large number of distributed services, sources and consumers of health related information. There are security requirements, especially strict for the sensitive patient data. These requirements cannot be fully implemented within the capabilities of the standard security practices used. Experience shows that really relevant results, including scientific, can be obtained only in the process of working with important practical problems.

In the first section of the paper, we consider the general structure of the target system, the requirements for data security and standard methods provided by the system used in target system development technologies.

The second section, which is devoted to the opportunities in the field of security, provides the development and analysis of user behaviour patterns in real time. In the second section, several possible approaches are also considered, and the structure of the used solutions is briefly described.

The third section of the paper describes the features of the interaction process between the target e-health platform and the established system. The strengths and weaknesses of two approaches are evaluated in terms of the organization of this interaction. Additionally, the section considers the features of the Python programming language for this type of tasks.

II. Common Structure

The target system is an integration platform that combines, on the one hand, the set of data sources, on the other – a lot of public access interfaces. The general scheme of service interaction is shown in Fig. 1, which demonstrates that the structure is divided into four parts and only adjacent components interact directly:

- *Users of the system* – the authorized personnel directly working with these patients. It is they who are the target of modelling behaviour and abnormal activity detection.
- *Public WEB sites* – the Internet portals and services, which provide data and methods of access to the *integration platform* for the *users of the system*. Directly *users* do not interact with *integration platform*, only through the appropriate public WEB site.
- *Integration platform* – the internal system that unites all queries and transaction data streams.
- *Data sources* – the external (remote) databases and services that provide common API interfaces to work with their internal databases.

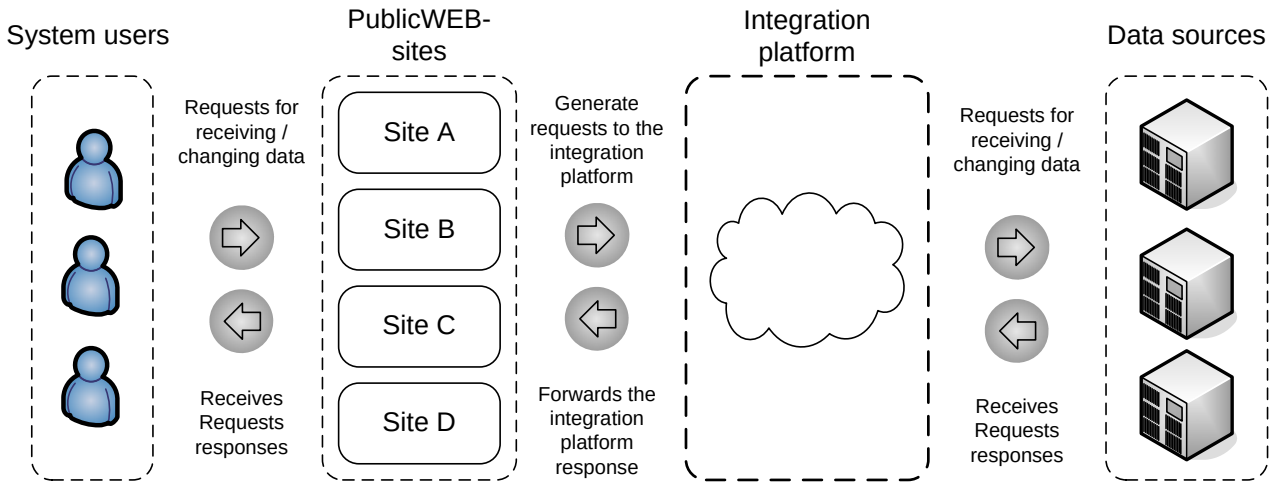

Fig. 1. Common service structure

Central *integration platform* integrates multiple data sources, on the one hand, and consumer data (doctors, laboratory technicians, patients), on the other. This approach allows us to create an effective framework that provides and controls all services and transaction data streams. The availability of public interfaces helps effectively isolate specifics of the internal implementation of each component of the system, as well as reducing the level of module interconnectivity that provides a corresponding increase in the overall flexibility of the system architecture.

III. Security System

Common structure of the security system is presented in Fig. 2. Centralized security module allows analyzing the entire transaction flow at one point. This makes possible to use the standard methods of security and also implement new ones.

Technically, the target system is implemented using the technology of WEB-services – SOAP [2]. This technology has the built-in methods to ensure security of transmitted data. Specification for these methods is called "WS-Security", and can be used in particular for adding specific new security modules.

A. WS-Security

In general, "WS-Security" is a protocol for inserting a security keys into headers of the SOAP messages. Information related to the protocol encryption, public keys, and other important attributes is added as a special text or binary fields into each message for messaging security support. In the case of the additional use of the HTTPS protocol to encrypt the channel, the probability of intercepting and decrypting the data (*Man in the Middle* [3]) is significantly reduced.

WS-Security provides the following procedures:

a) Identification;

b) Signs;

c) Encryption.

This triad refers to the major security problems and answers the following questions:

- Who is the person whom I authorize?
- Has the message been changed during the transmission process?
- Is it the message from the person I know?
- How do I hide things from which I want to see only certain parts?

Additional Security Methods

Additional protocol used for security context support is called *WS-SecureConversation* [4]. And *WS-Policy* [5] – the protocol of description of service additional requirements, opportunities and privileges.

Requirements for the Created UBM Module

A created UBM module provides a specialized service for system security support by building a model of personal behaviour for each user in the system. In the case of significant differences between the current behaviour and the typical behaviour of the currently logged in user, a common security system message is sent with the conclusion of the module and numerical assessment of the existence of abnormality.

System specification provides strict requirements on speed for each transaction processing: selected high threshold processing time of the 100 models that equals to 500 milliseconds. There is also a requirement to use typical server hardware without having to purchase the expensive high-performance equipment. Depending on the hardware processing power, speed of one model calculation may vary.

An important limitation in the software implementation is a

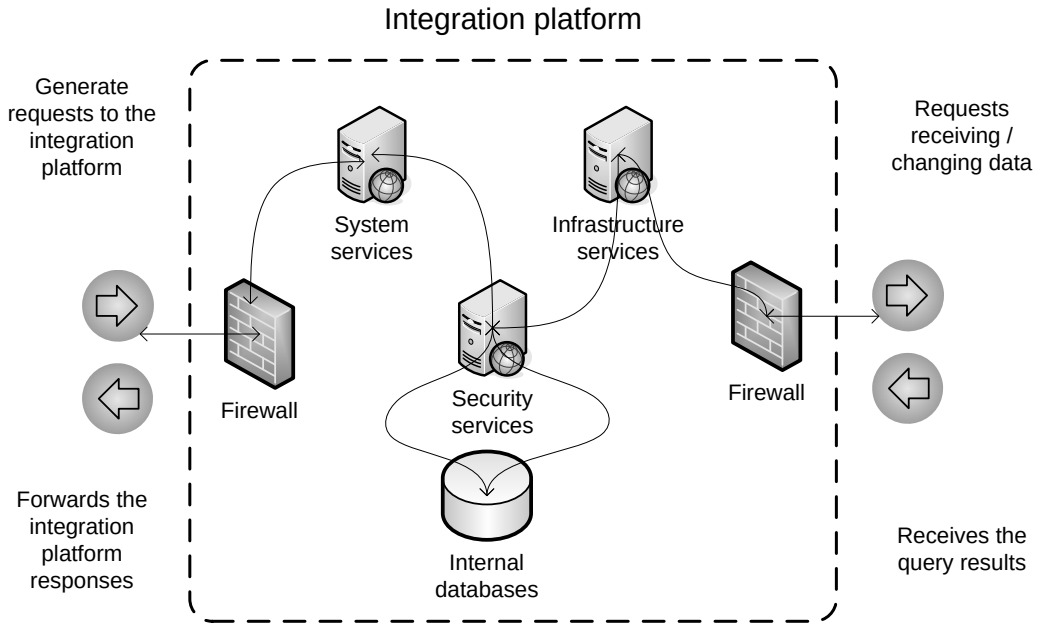

Fig. 2. The general structure of the security subsystem

strict requirement for UBM module self-sufficiency; it must have a minimum number of dependencies on other parts of the system.

The configuring of the (training) UBM should be automated as much as additional resources for interviewing users, inviting experts and similar activities are not provided.

IV. Creating Templates of the User Behaviour for Support of the System Safety Tasks

A. The Structure of the Created UBM

Capabilities of the method are in good agreement with the requirements of the target system within the existing constraints. More details of the approach can be found in [6]. An important part of research is focused on the possibility of an effective process for the receipt and processing of a large number of incoming transactions. Feasibility of this problem is described in [7], where the process and the result of testing different approaches are presented for the implementation of the method used. According to the results of the performed experiments, the "deferred" approach [9] has been chosen to implement module for transaction flow analysis.

B. Common Scheme of Interaction

According to the requirement to analyse each transaction in the system, a queue of the synchronous transactions has been implemented. For each request and response it is necessary to calculate the value of the presence of abnormality and consider it, when deciding whether to issue a permit for its service, i.e., the security system will not allow proceeding transaction if the service detects a suspicious user activity or other possible security threats.

C. UBM Implementation

Based on research performed in [7], the Python programming language has been chosen as a platform for implementing UBM; "Tornado" – as an asynchronous server working on the "deferred" ideology, and NoSQL [8] data store Redis that allows the effective management of the stored behaviour models.

We use the following abbreviations:

User role – the kind of user-defined goals of the system. In the case of e-health, the division may be related to personnel medical training.

Behaviour model – the complex data structure that provides a complete set of methods to calculate the value of the metric of the presence of anomalous user behaviour. Currently, *Behaviour model* can have one of two following types:

- *Model of the class* – it is used to represent a typical behaviour for a class of users, for example, when a new user is only added to the system, it has no history of its use, which is essential for the construction of his personal model, and in this case, he is assigned a *model of the class*, according to his role in the system.
- *Personal model* – it keeps the specific behaviour of a particular user in the system. In general, it can be constructed by continuously changing the original *model of the class* according to the results of each successful session of working with the system, or based on other rules.

MDB – the model database, which is a database of user models. Main tasks: *storage*, *retrieval*, and *delivery* of the requested model as efficiently as possible. NoSQL storage engine – "Redis" is currently used. Each model is identified by a unique key and presented as a serialized Python object.

SL – session logs of user activity per working session. Each log contains a vector of transaction identifications, performed by the user consistently over time in a single session. Session logs are used to create new model and to update the existing ones.

MG – the model graph, which is a graph of the Markov chain being part of each model. It holds the relationships, weights and the probabilities of transitions between states of the system that are specific to the current model. It is generally used as a convenient data structure for representing and operating information of such a type.

BM – the behaviour model, i.e., a model of user behaviour. It contains MG, additional metadata models and methods of operating with it. The main objectives are the following:

- The calculation of the value of the metric that describes the presence of anomaly in the received transaction;
- Updating the model based on the updated history of the user working with the system;
- Export / import models in various formats;
- Visualization of the target MG.

BMM – behaviour model manager, which is an object that contains the BM, manages the overall logic of its service. The main objectives are the following:

- to save / delete a model;
- to initialization the new model class for the user;
- to search for the return of class or personal models by request.

BMT – the behaviour model teacher, which is a component for model learning process supervision. It creates and updates the *model of the class*, also can update the personal model of each user. It provides methods to work both with the model and for the processing of the transaction log data. It contains BMM, which offers itself as a model, and interfaces to work with it.

MWS – the model window size, i.e., the number and size of the model window. It is the number of transaction IDs used to describe a single node in the MG graph. Each node in the graph is described as a vector of transaction IDs having size = WMS.

D. Typical Scenarios

UBM working algorithm contains several typical scenarios. Each of them gets the job done at a certain stage. Next, we will discuss the most important of them.

Creating a New *Model of Class*

The task of training a new *model of class* is controlled by the BMT component. The learning algorithm is described in detail in [6]. In general, the BMT takes as input a set of SL

sessions and builds a MG graph using certain rules, adding nodes, adjusting weights and transition probabilities.

Fig. 4 and Fig. 34 present the steps occurred in case of necessity to update or create a new *model of the class*. This current state is changed, based on the previous transactions having size of MWS.

Fig. 5 shows a general algorithm for graph of the model updating, when the receiving of each new ID from the training

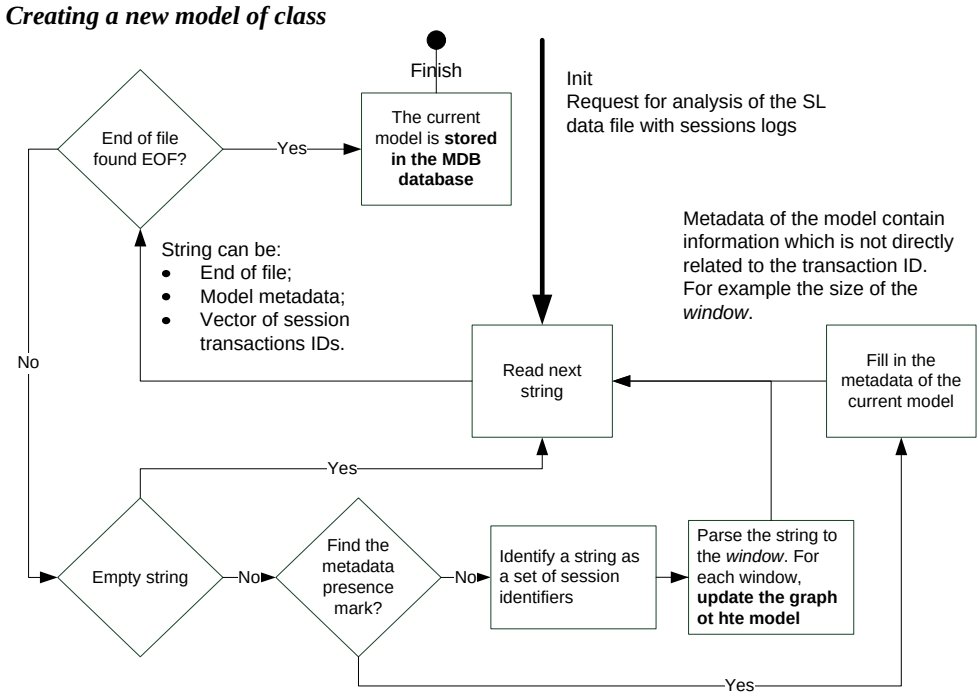

Fig. 3. The logic of creation of a *model of the class*

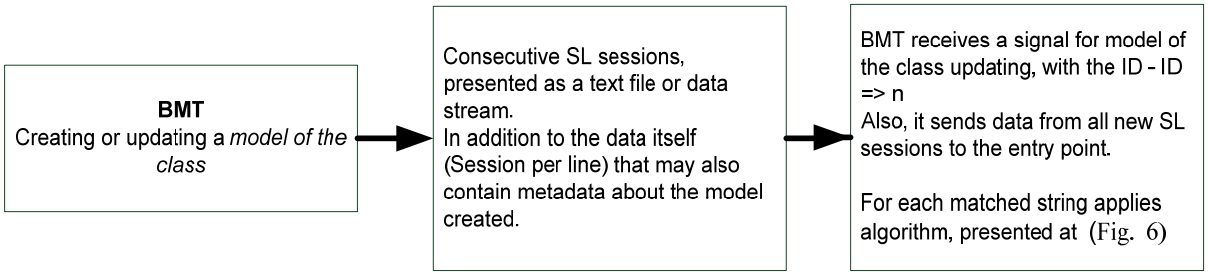

Fig. 4. The life cycle of a *model of the class*

procedure may take a long time, depending on the number and structure of the new sessions added to the model, and on the complexity of the existing model. In this process, the target model is updated based on the information on new typical patterns in behaviour of the class (group) of users.

Updating the Model

The algorithm of the model updates nodes in a MG, their weights and transition probabilities between them. The fact that the model has the Markov property [10] (lack of memory) allows us not to rebuild a whole graph after each update, and we can modify only a small number of nodes and arcs in the immediate vicinity of the point of the changes.

The update process is based on traversing of the vector of transaction identifiers for each user session *SL*. Each time the set of sessions *SL*.

Request for the Existing User Model

When interacting with a large number of models, there is a need to provide an efficient algorithm. At this moment, we use a simple approach, which is illustrated in **Error! Reference source not found.**.

Initialization of the User Model

Creation of a new personal user model is based on the fact that in the process of registration in the system, each user has to match one of the available typical roles. The first request copies a model of the class type, the corresponding role of the user, as the user's personal model. In the future, the resulting model will increasingly change, adaptively adjusting to the behaviour of each individual.

V. Specifics of Implementation

A. Interoperability, the Advantages and Disadvantages

Fig. 7 presents two possible ways for *UBM* connection to the target system:

I. Intercepting raw HL7 messages (short dashed line), analysis in real time, obtaining the necessary data, the calculation of the metric and the issuance of an absolute security subsystem.

Each option has specific advantages and disadvantages. Now we will discuss each of these approaches in detail.

Interception of HL7 messages

All request messages are sent using SOAP XML files that contain HL7 messages wrapped in the system specific envelopes. Namespaces are additionally used to avoid potential conflicts in the naming of the nodes from different

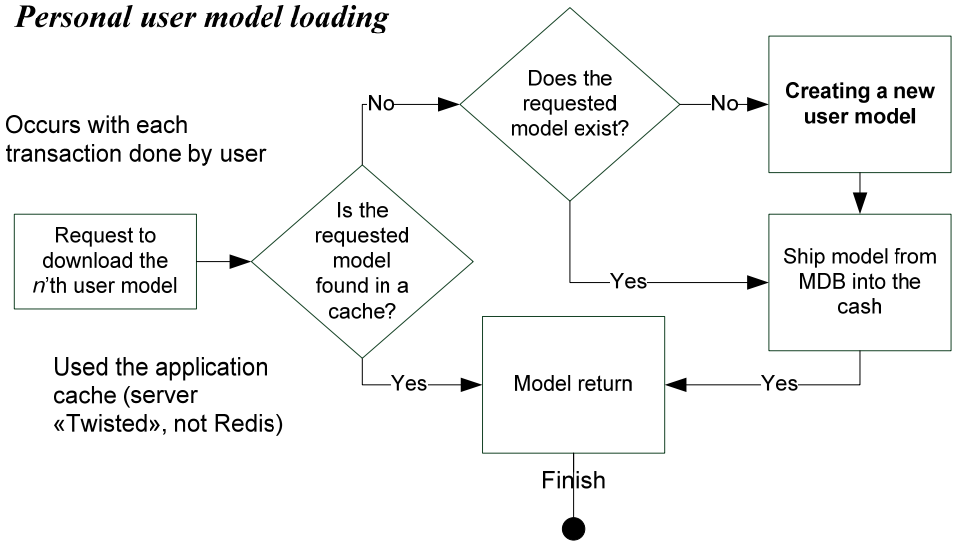

Fig. 5. Personal user model loading

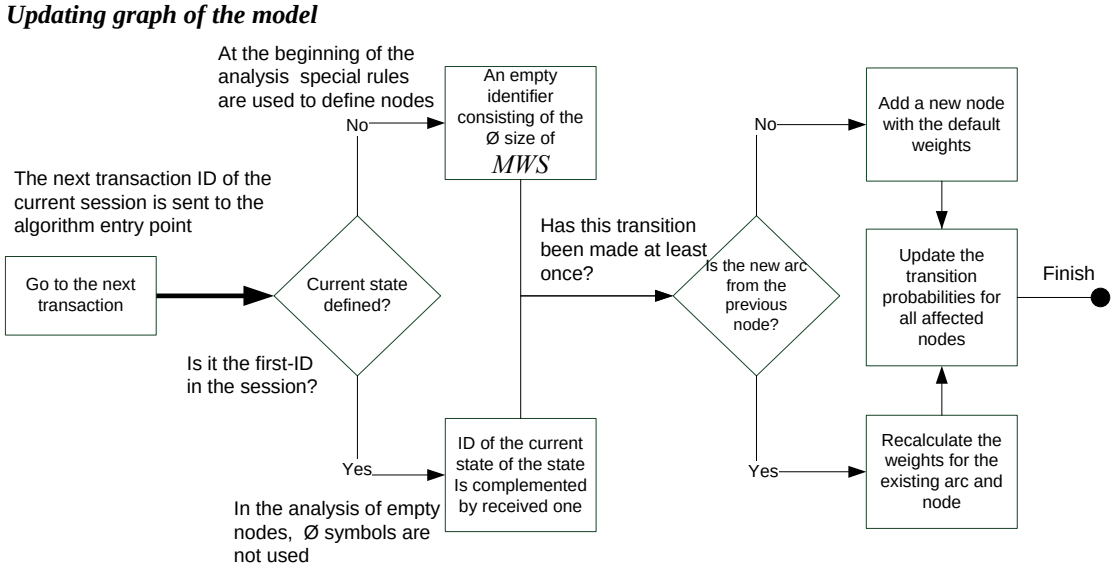

Fig. 6. Algorithm for the updating of a graph model

II. Using the already prepared data from MSSQL database (long dashed lines). Since each transaction is anyway parsed and stored in a database for further analysis and report generation, it is preferable not to disassemble it once more, and use queries to the database. It is important that the database maintained sufficient to analyse all the data.

parts of the transmitted document. The following namespaces are used: *soap*:, *saml*: and *hl7*:.

Answer messages will have a similar structure: HL7 data system is wrapped by SOAP envelopes and uses the same namespaces.

The process of interaction of the developed module by intercepting the flow of raw message is based on the system function – a listener for the "transaction" event, which captures and sends it for the analysis to the *UBM* module. The module processes the received message and calculates the metric value of the presence of anomalies in transaction, and then sends a response to the security subsystem. It, in turn, makes the decision on whether to allow this action based on its internal logic, but additionally (perhaps) with the conclusion

Also note that in order to provide additional security, only a certain group of processes and users has the right to work with the *audit* database.

Advantages of this approach:

- The data have been prepared for convenient use;
- It is more efficiently to work with the database than with the XML document;
- It is faster to work with the database than the XML document.

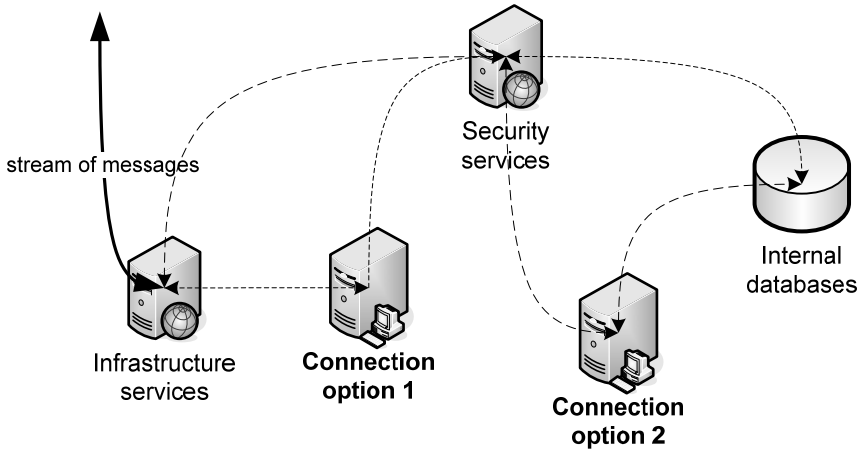

Fig. 7. Possible MMP connection points

of *UBM*.

Advantages of this approach:

- It allows performing the analysis in real time, as it is directly in the way of communication between the user and the system;
- It provides a complete set of data for the analysis from any package;
- It does not use intermediate layers between the data and the *UBM*.

Disadvantages:

- Since the flow of data is presented as a "raw" stream of XML messages, the analysis should also be done by the *UBM* that, in general, is not its responsibility area;
- Additionally, examining the XML is duplicating work, because it has already been decoded for filling the internal audit database.
- Accordingly, in case of message format changes it will require to do changes also in the message parser script.

Using the Prepared Data

As some of the requirements for the target system, the basic procedure involves the logging of data in all transactions made (*audit*). In the system, the database records values of certain fields from each transaction. This information is sufficient for the creation and use of *UBM*. The main advantage is the ability to avoid parsing raw of internal system messages by *UBM*, but getting them in an easy and structured way from the DBMS. In order to increase the efficiency of data access logs NOSQL approach is also used, offering easy access to the data from the internal services.

Disadvantages:

- There may be problems with the efficiency of interaction between Python and MSSQL server since it runs only on Windows, and the target system is currently implemented on the OS family of Linux.
- Additional software layer between the construction of model behaviour and data for it also requires extra resources.

Having performed the analysis of both approaches, the second approach has been selected: work with the already prepared data stored in *audit* database.

B. Specifics of Python Usage

The use of Python as the primary tool for the implementation of *UBM*, on the one hand, significantly increases the range of available technologies and methods, on the other hand, adds a certain set of constraints.

An important advantage is a large set of different modules and libraries that allow designing and implementing the system using the new and highly effective methods and approaches.

The main objectives have been as follows:

- Implementing a messaging server using the "deferred" approach.
- Implementing logic retention and retrieval models using the NOSQL approach.
- Software implementation of the methods for the creation, training and use of the model of the behaviour (MG, BM, BMM, BMT).

- Implementation of interaction with a remote database, which stores data transaction logs.
- Interaction with internal WSDL services.

In general, the use of Python as the primary tool has met the expectations. The time for the required functionality has decreased and support for the project has become less time-consuming in comparison with the use of an experimental model implemented using other programming language.

VI. Results and Conclusions

Implementation of such a system is a difficult and complex task that requires the careful planning and testing of the effectiveness of each of the key nodes. Requirements for the security systems and its functioning impose weight restrictions. The created system structure has shown good results under strict limitations of target system requirements.

VII. References

[1] Della Mea, Vincenzo. "*What is e-Health: The death of telemedicine?*". Journal of Medical Internet Research (Jmir.org) 3 (2): e22. doi:10.2196/jmir.3.2.e22. PMC 1761900. PMID 11720964. Retrieved 2012-04-15.

[2] Jack Koftikian; *Simple Object Access Protocol (SOAP)*; Technical University Hamburg-Harburg;

[3] Roi Saltzman; *Active Man in the Middle Attacks. A SECURITY ADVISORY*; A whitepaper from IBM Rational Application Security Group. February 27, 2009.

[4] WS-Secure specification description [Online]. Available: http://en.wikipedia.org/wiki/WS-SecureConversation, [Accessed: Sept. 01, 2012]

[5] Web Services Policy Framework *(WS-Policy);* [Online]. Available: http://specs.xmlsoap.org/ws/2004/09/policy/ws-policy.pdf, [Accessed: Sept. 01, 2012]

[6] P. Osipovs, A. Borisovs; *Abnormal action detection based on Markov models*; Automatic Control and Computer Sciences; Volume 41 / 2007 - Volume 45 / 2011; ISSN 0146-4116 (Print) 1558-108X (Online); May 05, 2011.

[7] P. Osipovs, A. Borisovs; *Using the Deferred Approach in Scientific Applications*; Scientific Journal of Riga Technical University, Series 5, Computer Science, Vol. 49, Information Technology and Management Science, pp. 139-144, 2011.

[8] Prof. Walter Kriha; Lecture: „*Selected Topics on Software-Technology Ultra-Large Scale Sites*"; University Hochschule der Medien, Stuttgart, 2010. [Online]. Available: www.christof-strauch.de/nosqldbs.pdf [Accessed: Sept. 23, 2012]

[9] Samek M. (2008), *Event-Driven Programming for Embedded Systems*, Newnes 2008. ISBN-10: 0750687061; ISBN-13: 978-0750687065.

[10] Markov, A. A. (1954). *Theory of Algorithms*. [Translated by Jacques J. Schorr-Kon and PST staff] Imprint Moscow, Academy of Sciences of the USSR, 1954 [Jerusalem, Israel Program for Scientific Translations, 1961; available from Office of Technical Services, United States Department of Commerce] Added t.p. in Russian Translation of Works of the Mathematical Institute, Academy of Sciences of the USSR, v. 42. Original title: Teoriya algorifmov. [QA248.M2943 Dartmouth College library. U.S. Dept. of Commerce, Office of Technical Services, number OTS 60-51085.]

Arkady Borisov holds a degree of Doctor of Technical Sciences in Control of Technical Systems and a habilitation degree in Computer Science.
He is Professor of Computer Science at the Faculty of Computer Science and Information Technology, Riga Technical University (Latvia). His research interests include fuzzy sets, fuzzy logic and computational intelligence. He has 220 publications in the fields of computer science and information technology.
He has supervised a number of national research grants and participated in the European research project ECLIPS. E-mail: arkadijs.borisovs@cs.rtu.lv.

Pavel Osipov Mg.Sc.Comp., Doctoral Student of the Institute of Information Technology, Riga Technical University. He received his Master Degree from Transport and Telecommunication Institute, Latvia.
His research interests include web data mining, machine learning and knowledge extraction. Research activities are mainly focused on different aspects of user behaviour modelling. A new area of interests is related to the exploration of application of Python programming language to all steps of scientific research. E-mail:pavels.osipovs@gmail.com

Pāvels Osipovs, Arkādijs Borisovs. Pieeja informācijas sistēmu lietotāju uzvedības modeļiem
Rakstā aprakstīta metodoloģija anomālas lietotāju uzvedības noteikšanas moduļa izveidei un tā ieviešanai sadalītā informācijas sistēmā. Šāda moduļa izveides iemesls bija nepieciešamība atklāt tādas ielaušanās, kad leģitīma lietotāja kontu izmanto cits cilvēks. Mērķa sistēmas svarīga īpatnība ir tās sadalītais raksturs, kā arī datu bāzē eksistējoši sensitīvi dati. Ir aprakstītas mērķa sistēmā izmantojamas tipveida pieejas, kas izmantotas vispārējās drošības nodrošināšanai, kā arī galvenās izmantojamās metodes un protokoli, un prasības pret papildus specifiskiem drošības moduļiem. Ir parādītas stingras prasības, kuras uzstāda mērķa sistēma katras lietotāja transakcijas apstrādes ātrumam. Tā kā drošības sistēma analizē katru lietotāja pieprasījumu reālā laika režīmā, laiks tās slēdziena izdošanai tiek pieskaitīts kopējam pieprasījuma apkalpošanas laikam. Parādīta izstrādātā risinājuma moduļveida struktūra. Katrai moduļa sastāvdaļai sniegts tās uzdevumu un iespēju apraksts. Aprakstīti divi moduļa pamatdarbības veidi: *apmācība* un *analīze*. Apmācības režīmā modulis atjauno vienu vai vairākus modeļus, balstoties uz atjauninātiem lietotāju sistēmas izmantošanas statistikas datiem. Katram no apmācības procesa posmiem ir sniegtas izmantoto algoritmu blokshēmas. Analīzes režīmā modelis uzrāda slēdzienu par anomālijas esamību katrai no lietotāja transakcijām. Slēdziens tiek atgriezts kā metrikas vērtība - ar frakcionētu skaitli starp 0 un 1, kur 1 ir pazīme visanomālākajai uzvedībai. Katram no transakcijas analīzes procesa posmiem tāpat ir sniegta izmantojamo algoritmu blokshēma. Ir aprakstīti divi iespējamie varianti moduļa iebūvēšanai mērķa sistēmas infrastruktūrā, katram ir uzrādītas priekšrocībās un trūkumi. Pirmais variants saņem neapstrādātu XML paziņojumu plūsmu, bet otrs izmanto jau apstrādātu atribūtu iekšējo bāzi, kas tiek izmantota audita veikšanai. Pieejas izvēle ir argumentēta. Noslēgumā īsumā aprakstītas Python programmēšanas valodas kā pamata platformas moduļu izstrādei, pielietošanas īpatnības.

Павел Осипов, Аркадий Борисов. Подходы шаблонов поведения пользователей информационных систем
Статья описывает методологию создания модуля обнаружения аномального поведения пользователей и внедрения его в распределённую информационную систему. Причиной создания такого модуля послужила потребность обнаружения такого типа вторжений, когда учётной записью легитимного пользователя пользуется другой человек. Важными особенностями целевой системы является её распределённый характер, а также наличие в базе данных сенситивных данных. Описаны типовые используемые в целевой системе подходы к обеспечению общей безопасности, основные используемые методы и протоколы, а также требования, предъявляемые к дополнительным специфическим модулям безопасности. Показаны жёсткие требования, предъявляемые целевой системой к скорости обработки каждой транзакции пользователя. Так как система безопасности анализирует каждый запрос пользователя в режиме реального времени, то время на выдачу её заключения добавляется к общему времени обслуживания запроса. Показана модульная структура разработанного решения. Для каждого составляющего модуля дано описание его задач и возможностей. Описаны два основных режима работы модуля: *обучение* и *анализ*. В режиме обучения модуль обновляет одну или несколько моделей на основе обновившихся данных статистики использования пользователями системы. Для каждого из этапов процесса обучения приведены блок-схемы использованных алгоритмов. В режиме анализа модель даёт заключение о наличии аномальности для каждой транзакции пользователя. Заключение возвращается в виде значения метрики - дробного числа в диапазоне от 0 до 1, где 1 это признак наиболее аномального поведения. Для каждого из этапов процесса анализа транзакции также приведены блок-схемы использованных алгоритмов. Описаны два возможных варианта встраивания модуля в инфраструктуру целевой системы, для каждого приведены достоинства и недостатки. Первый вариант получает необработанный поток XML сообщений, второй использует внутреннюю базу уже обработанных атрибутов, используемую для проведения аудита. Обоснован выбор использованного подхода. В заключении кратко описаны особенности применения языка программирования Python как основной платформы для создания модуля.