AUTOMATED TESTS FOR ERRORS IN COMPUTER SYSTEM 'ENVIRONMENT'

Oleksii Vasylenko, Oleksandr Kuzomin

Abstract: *This paper describes technology of automated testing for complex computer systems like 'smart house', etc. The basic idea to abstract from natural environment to cyber analogue.*

Existing predicates contained in the semantic network to the new analysis may serve as benchmarks. Thus, the proposed recognition model of natural language objects can be faster and more efficiently than the existing ones.

Keywords: *knowledge acquisition, SAP, DoD, Automated Testing, knowledge processing, automated knowledge management system, Internet, Intranet, logical testing, Smart testing, Complex Computer System (CCS), Selenium –Bamboo.*

Introduction

Recently, one often hears about creating smart homes and automation of everyday life. Many of today tend to make "smart" every detail of daily life. But few of us think about what a great information systems can make mistakes. And the more the system - the more expensive the cost of failure or malfunction. In order to monitor the performance of the whole agglomeration area connections of information systems need a comprehensive and versatile solution [Kuzomin and Vasylenko, 2014].

Imagine the future. One 'smart house' is connected to another one. Each 'smart house' – independent smart complex computer system. We have the network, which based on other, smaller networks. Lets call them 'basic element' or element level 1. The next level of this system is communication of these elements from level 1. I will call it "elements level 2". Finally, we have correlation of elements level 2, which create special "environment".

On the other hand, everybody knows what is it the 'state of Emergency'. Where is it possible to happen? Nature, Factories, etc. What is it? The hurricane, earthquake, floods, landslides, etc. When they going to happen? Everytime. No one know date and time. Everybody afraid and tries to make the prognosis for the next Emergency.

We live in 21st Century so why we think only about the common Environment? Why we postpose the one which is going to stay the most important for ages now. I will call it 'Cyber Environment'.

Computers, networks, complicated systems – everything is part of this new Environment. Lets take a look here from the point of Earth (Natural) Environment investigation. We will find all things common. Why we don’t investigate the emergencies which are in this ‘CyberInvironment’?

I purpose the Test Solution, based on Atlassian Bamboo, running by Selenium server. Selenium is still actively supported (mostly in maintenance mode) and provides some features that may not be available in Selenium 2 for a while, including support for several languages (Java, Javas Script, Ruby, PHP, Python, Perl and C#) and support for almost every browser out there.

First, we will describe how the components of Selenium RC operate and the role each plays in running your test scripts. e-mail for questions: ichbinerste@gmail.com

The essence of work

Task formulation:

1) Develop intellectual language to express, statistical tracking the errors, data mining, based on CCX automated logical test results .
2) Develop model of complex result/dependence matrix (model: test – error tracked - reason – feature request)
3) Develop general model of Emergency Forecasting (EF) for basic CCS
4) Develop logical intellectual language for automated decision making based EF for basic CCS
5) Develop complex decision making matrix for velocity & quality increasing for EF and trouble shooting in CCS
6) Create the feature request system logical language based on EF

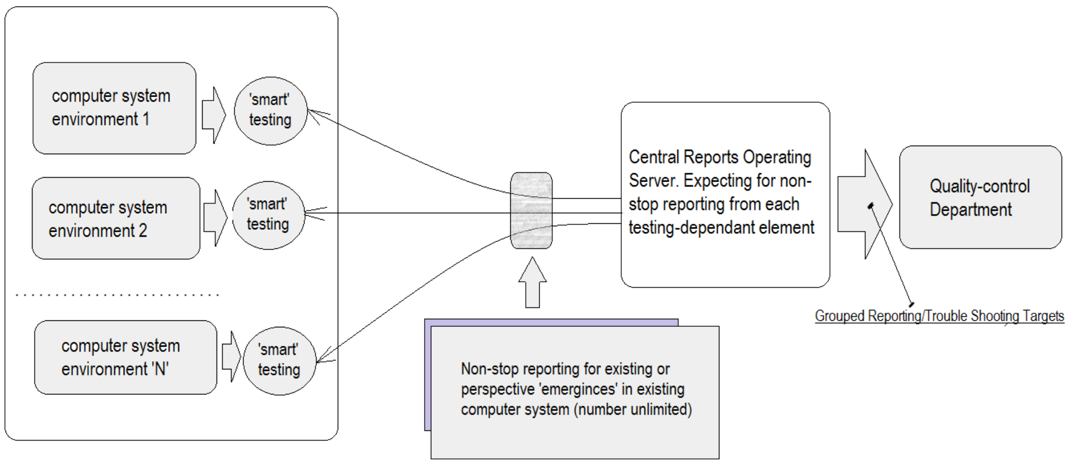
Scientific innovation: The practical novelty is in the development of the intelligent decision-making language based on EF for CCS. Basic principle is completely new and based on systematical automated finding and tracking the errors (Emergences in CCS environment), creating the tracking and data mining model to find and keep logical depended data for Emergence practice in CCS environment. This data will be a row input for automated EF and Decision Making for Emergency prevention in CCS.

Goals and objectives of my work: The aim is to develop a model of knowledge extraction by running ‘smart’ tests in huge computer systems. The aim is to create system with automated testing and reporting, trouble shooting and prognosis stages set up. The main aim for project is to make life easier by giving the hard and boring job for machine and results for us – people (Fig.1)).

There are basic stages of my work and several ‘pre-aims’ of the whole project:

1) Create test logic level 1. (deep investigation for test-ready fields of computer system environment)
2) Create test logic level 2. (look for test-connection ready stages)

3) Create test logic level 3. (whole self-dependant smart test architecture network projecting)
4) Final test logic stage. Automated smart-unitTest builds creating. Creating whole smart testing network for each possible error.
5) Coding and realising the 'road-map' for the smart unitTest build.
6) Creating the test-analysis algorithm and automatisation for the whole project


computer system environment 1
'smart' testing
computer system environment 2
'smart' testing
computer system environment 'N'
'smart' testing
Central Reports Operating Server. Expecting for non-stop reporting from each testing-dependant element
Quality-control Department
Grouped Reporting/Trouble Shooting Targets
Non-stop reporting for existing or perspective 'emergences' in existing computer system (number unlimited)

Concept Scheme

Figure 1 – Concept Scheme

Sphere of application:

1. Information intelligence. Data mining for emergency predicting environment. 'Smart' prognosis for computer systems for life support, etc.
2. Bamboo is the central management server which schedules and coordinates all work.
3. Bamboo itself has interfaces and plugins for lots of types of work.

Generating Idec load

In order to build up load in a structured, guided ways from SAP there are a few ideas of what can be done. My initial hopes, were to push IDocs from a load generator to the message broker. This would be the easiest way in which to control the flow of data towards the broker and thus the easiest way to make sure we are fully in control of how busy the broker is. Alas, when talking to the guys behind the broker interfaces it turned out that this method would not work for the setup used. The only way the broker would actually do something with the IDocs was if it could pull them from the SAP RFC port, pushing to the broker would not work, since the RFC receiving end of the broker is not listening, it is pulling.

Alternatively sending data off into the message queue would fill up the MQ, but not help with getting the messages pushed through the Broker, again, due to the specific setup of the Enterprise Service Bus which contains the broker interfaces.

So alternatives needed to be found. One obvious alternative is setup a transaction in SAP which generates a boat-load of IDocs and sends the to the RFC port in one big bulk. This would generate a spike, such as shown in this image, rather than a guided load. In other words, this is not what we want for this test either. It might be a useful step for during a spike test, however the first tests to be completed are the normal, expected load tests.

The search for altneratives continued. At my customer, not a lot of automation tools were available, especially not for SAP systems. One tool however has been purchased a while ago and apparently is actively used: Win Shuttle

Win shuttle seems to be able to generate the required load, based on Excel input, the main issue with Win shuttle however, was the lack of available licenses. There are two licenses available and both are single use licenses. This meant I would have to find a way to hijack one of the PC's it was installed on, script my way through it and run the tests in a very time boxed manner. In other words, not really a solution to the problem. The resulting load ramp up was a linear ramp up of IDocs being generated and sent to the SAP RFC port, where they were picked up by the Message Broker and subsequently transformed and sent back to the Locus system, where the load turned out to be quite on the high side.

All in all this was a fun excercise in automating SAP to do something it is absolutely not meant to do with a tool not built nor designed to do what it did.

Definition of Done

In the majority of the DoD's I have seen, one of the items is something referring to "tests automated". The thing I have thus far not seen however, is the team adding as much value to the automation code as they do to the production code. Quite a lot of DoD's refer to certain coding standards, however these standards often seem to not apply to functional test automation. Isn't your functional automation code also just code? If so, why then should this not be covered in code reviews, be written according to some useful guidelines and standards and hopefully use a framework to make the code sustainable (Fig.2)?

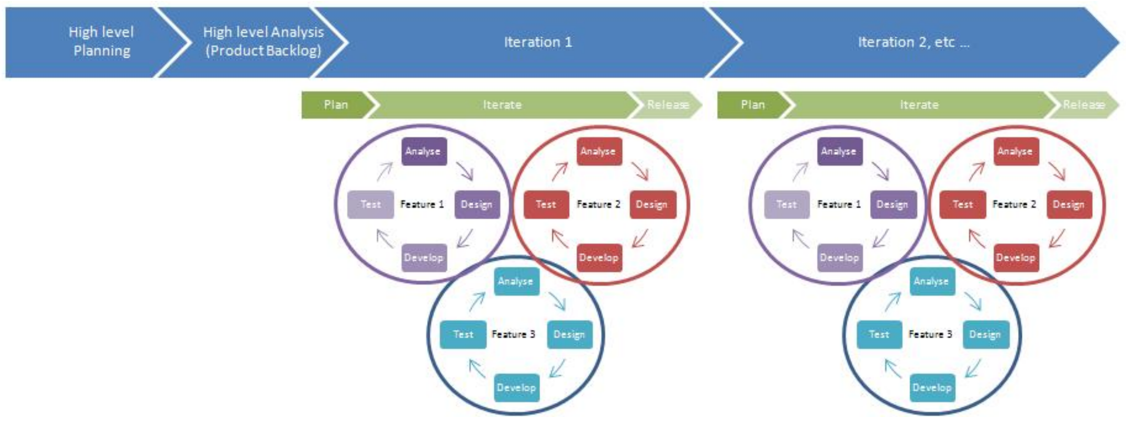


Figure 2 – Agile Development Cycle diagram for logical testing

There aren't really any distinct stages during the development. Instead, each feature is taken from start to finish within an iteration, with the software being released at the end of each iteration, or if appropriate even during an iteration.

An iteration is simply a fixed, short period of time that the team chooses to work within. Typically for agile teams, an iteration is between 1 week and 30 days. Strictly speaking, the Scrum agile development methodology advocates 30 days, but I've encountered very few teams that actually do this. 2 or 3 weeks seems to be more common. The Extreme Programming agile methodology advocates 1 week. This is very short and in my experience requires quite a lot of maturity in the team and its processes to achieve, because getting to a stable release every week can be difficult.

Either way, the principles are the same. The idea is to stick to short, fixed-length iterations and complete all stages of the development cycle for each feature in turn within an iteration. Basically, idea is to SWOPE teams of people by teams of hardware. The same cycle, the same responsibility stages & reporting.

Scientific novelty: The model extraction knowledge by using the automated smart unit testing environment is pretty much new. The main logic for this – make the technical support stage much more easier than it is now. Smart test will be able to run all scheduled time, give the report and make each part of the system tested by time, make it ready and non-forgiven for the possible mistakes, that are able to destroy the whole component or system at all. Aim for this project is automation for system support part. As huger computer system is becoming – as harder and more expensive support level becoming as well. That is why we have this big challenge to develop new strategic for keeping system's support 'eyes' clear and ready to react.

Practical value: Bamboo is a continuous integration (CI) server that can be used to automate the release management for a software application, creating a continuous delivery pipeline.

CI is a software development methodology in which a build, unit tests and integration tests are performed, or triggered, whenever code is committed to the repository, to ensure that new changes integrate well into the existing code base. Integration builds provide early 'fail fast' feedback on the quality of new changes.

Release management describes the steps that are typically performed to release a software application, including building and functional testing, tagging releases, assigning versions, and deploying and activating the new version in production.

Selenium Usage: Selenium Server receives Selenium commands from your test program, interprets them, and reports back to your program the results of running those tests.

The RC server bundles Selenium Core and automatically injects it into the browser. This occurs when your test program opens the browser (using a client library API function). Selenium-Core is a JavaScript program, actually a set of JavaScript functions which interprets and executes Selenium commands using the browser's built-in JavaScript interpreter.

The Server receives the Selenium commands from your test program using simple HTTP GET/POST requests. This means you can use any programming language that can send HTTP requests to automate Selenium tests on the browser.

Client Libraries

The client libraries provide the programming support that allows you to run Selenium commands from a program of your own design. There is a different client library for each supported language. A Selenium client library provides a programming interface (API), i.e., a set of functions, which run Selenium commands from your own program. Within each interface, there is a programming function that supports each Selenium command.

The client library takes a Selenium command and passes it to the Selenium Server for processing a specific action or test against the application under test (AUT). The client library also receives the result of that command and passes it back to your program. Your program can receive the result and store it into a program variable and report it as a success or failure, or possibly take corrective action if it was an unexpected error.

So to create a test program, you simply write a program that runs a set of Selenium commands using a client library API. And, optionally, if you already have a Selenium test script created in the Selenium-IDE, you can generate the Selenium RC code. The Selenium-IDE can translate (using its Export menu item) its Selenium commands into a client-driver's API function calls. See the Selenium-IDE chapter for specifics on exporting RC code from Selenium-IDE.

Logical Testing Organizing:

1. The client/driver establishes a connection with the selenium-RC server.
2. Selenium RC server launches a browser (or reuses an old one) with a URL that injects Selenium-Core's JavaScript into the browser-loaded web page.
3. The client-driver passes a Selenium command to the server.
4. The Server interprets the command and then triggers the corresponding JavaScript execution to execute that command within the browser. Selenium-Core instructs the browser to act on that first instruction, typically opening a page of the AUT.
5. The browser receives the open request and asks for the website's content from the Selenium RC server (set as the HTTP proxy for the browser to use).
6. Selenium RC server communicates with the Web server asking for the page and once it receives it, it sends the page to the browser masking the origin to look like the page comes from the same server as Selenium-Core (this allows Selenium-Core to comply with the Same Origin Policy).
7. The browser receives the web page and renders it in the frame/window reserved for it.

Many applications switch from using HTTP to HTTPS when they need to send encrypted information such as passwords or credit card information. This is common with many of today's web applications. Selenium RC supports this.

To ensure the HTTPS site is genuine, the browser will need a security certificate. Otherwise, when the browser accesses the AUT using HTTPS, it will assume that application is not 'trusted'. When this occurs the browser displays security popups, and these popups cannot be closed using Selenium RC.

When dealing with HTTPS in a Selenium RC test, you must use a run mode that supports this and handles the security certificate for you. You specify the run mode when your test program initializes Selenium. In Selenium RC 1.0 beta 2 and later use *firefox or *iexplore for the run mode. In earlier versions, including Selenium RC 1.0 beta 1, use *chrome or *iehta, for the run mode. Using these run modes, you will not need to install any special security certificates; Selenium RC will handle it for you.

In version 1.0 the run modes *firefox or *iexplore are recommended. However, there are additional run modes of *iexploreproxy and *firefoxproxy. These are provided for backwards compatibility only, and should not be used unless required by legacy test programs. Their use will present limitations with security certificate handling and with the running of multiple windows if your application opens additional browser windows.

In earlier versions of Selenium RC, *chrome or *iehta were the run modes that supported HTTPS and the handling of security popups. These were considered â€˜experimental modes although they became quite stable and many people used them. If you are using Selenium 1.0 you do not need, and should not use, these older run modes (Fig.2).

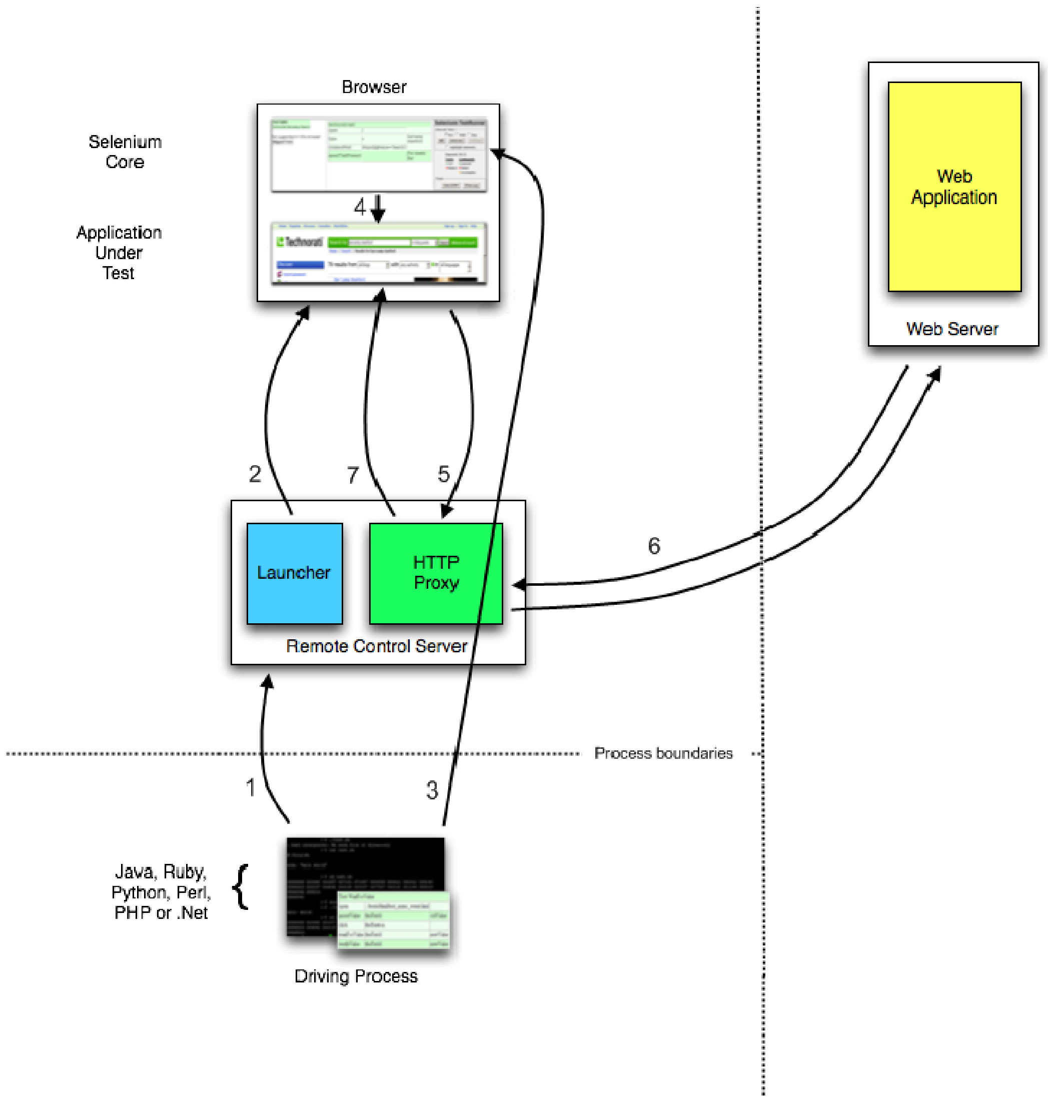


Figure 2 - Security popups

Work Flow organization

Bamboo uses the concept of a 'plan' with 'jobs' and 'tasks' to configure and order the actions in the workflow.

Project Has one, or more, plans.
Provides reporting (using the wallboard, for example) across all plans in the project.
Provides links to other applications.

Plan Has a single stage, by default, but can be used to group jobs into multiple stages.
Processes a series of one or more stages that are run sequentially using the same repository.
Specifies the default repository.
Specifies how the build is triggered, and the triggering dependencies between the plan and other plans in the project.
Specifies notifications of build results.
Specifies who has permission to view and configure the plan and its jobs.
Provides for the definition of plan variables.

Stage Has a single job, by default, but can be used to group multiple jobs.
Processes its jobs in parallel, on multiple agents (where available).
Must successfully complete all its jobs before the next stage in the plan can be processed.
May produce artifacts that can be made available for use by a subsequent stage.

Job Processes a series of one or more tasks that are run sequentially on the same agent.
Controls the order in which tasks are performed.
Collects the requirements of individual tasks in the job, so that these requirements can be matched with agent capabilities.
Defines the artifacts that the build will produce.
Can only use artifacts produced in a previous stage.
Specifies any labels with which the build result or build artifacts will be tagged.

Task Is a small discrete unit of work, such as source code checkout, executing a Maven goal, running a script, or parsing test results.
Is run sequentially within a job on a Bamboo working directory.

The proposed method: After carefully considering all the existing methods, capabilities and operating time, We can interpretive our hike to understand the meaning of natural language by computer.

The concepts can replace each other on the principle of consistency of meaning. It is above all that are essential to accelerate the process of text recognition as well - reducing the load on the knowledge base. (Kuzemin A., thesis work). That is – replaceability of the concepts significantly reduces the volume of the knowledge base by reducing the number of own concepts as its components. Create the new theorem replace ability of the concepts.

Theorem 1. If the concept Po_2 inherits the concept Po_1 – $\mathrm{G}(\mathrm{Po}_2, \mathrm{Po}_1)$ in a certain category of concepts C, the notion of Po_2 can substitute for a mikrosituations concept Po_1 without losing the semantic load and informative of this mikrosituation [Kuzomin and Vasylenko, 2010].

Proof. In accordance with the structure and strategy categories identify the concept to identify the concept Po_2 must affirmatively respond to the decision rules that relevant concepts $\mathrm{Po}_1, \mathrm{Po}_{k_1}, \ldots, \mathrm{Po}_{k_n}, \mathrm{Po}_2$ This means that identified the problematic concept as we have passed these decision rules, including p_1 which refers to the notion Po_1 . Hence, the notion Po_2 may be perceived as a concept Po_1, possessing its characteristic features.

It is possible to obtain the final simplified formula

To begin to define a mathematical model for this area, that is, a preliminary algorithm of the program domain.

Thus, we set the field, which we consider to find items that need to be put in the text as a priority. Items that are behind them before the end of the line, will be the ones most desired predicates that become the nodes of a semantic network specification.

So, try to express this simple mathematical formula:

Consider the above described concept $t_{i_m j}$ - the word in a sentence t_j determinants in appliance has a_i and τ_i:

a_{i_m}, τ_{i_m} when m=1 (definite value) <=> a_{i_1}, τ_{i_1} - given, = const

(The above mentioned node predicates, which is searched, that is - nodes ontology similarity Product RCO).

If the nodes of ontologies are not specified clearly (for thematic units characterized by peculiar and persistent concept), then to search for meaning in a sentence erants, first highlight the main predicate, then place it in a network node.

Consider the 1-st case with known predicates:

Suppose there is $t_{i_m j}$, that

$$(a_{i_1}, \tau_{i_1} \in t_{i_m j}, \{t_{i_m j_1}, t_{i_m j_2}, t_{i_m j_3}, \ldots, t_{i_m j_n}\}) \in t_j \Rightarrow$$

$$\Rightarrow \quad [\Pi_n] = \{t_{i_m j_2}, \ldots\ldots t_{i_m j_n}\},$$

Where Π_n - predicate found meeting the criteria a_{i_1} and τ_{i_1}

Consider Case 2, where the proposal is a set of elements without an explicit predicate. In this case, I have proposed for the allocation of nodal predicates analyze the proposal for the parts of speech, then find the subject and the predicate method for counting the frequency of occurrence of noun and a verb. Mathematically, it is possible to express this:

We have a set of words

$$\{t_{i_m j_1}, t_{i_m j_2}, t_{i_m j_3}, \ldots, t_{i_m j_n}\} \in t_j,$$

Let each of them has an additional parameter *α* - the frequency for each element, as well as the parameter responsible for the definition of the speech clearly designed combinations of endings – p (look application "A", list of terminals), where (1…n) ∈ p, we have a structure:

$$\{ t_{i_m j_{p\,(1\ldots n)}}{}^{\alpha},\ t_{i_m j_{p\,(2\ldots n)}}{}^{\alpha},\ t_{i_m j_{p\,(3\ldots n)}}{}^{\alpha}, \ldots, t_{i_m j_{p(n)}}{}^{\alpha}\} \in t_j,$$

Moreover, if found by the predicate coincides with the already existing sites - it is not analyzed in the future, and put in its place. If the predicate is unique - it is analyzed as part of speech and related items - similar to the ongoing analysis of combinations.

After receiving the new value Π_n, determine its meaning - it is entered in the knowledge base, which compares to the value of existing concepts $\mathrm{Po}_{1\ldots n}$. Further processing of meaning is on a similar principle, but with reference to a specific predicate.

Conclusion

In this work the method of analysis of natural language objects, which accelerates the work with large volumes of the analyzed verbal text. By itself, the semantic network is a self-learning, as accumulating predicates, new to them in their meaning. Existing predicates contained in the semantic network to the new analysis may serve as benchmarks. Thus, the proposed recognition model of natural language objects can be faster and more efficiently than the existing ones.

Bibliography

[Kuzomin and Vasylenko, 2014] Obespechenie bezopasnosti ispolzovaniia baz dannyh v usloviiah chrezvychainyh situazii.// Kuzomin, O.Ya., Vasylenko, O., International Journal "Information Technologies Knowledge", Vol. 8, Num. 2. 2014. pp. 173-187.

[Kuzomin and Vasylenko, 2010] Analiz estestvenno iazykovyh obiektov I predstavlenie znanii // Kuzomin, O.Ya., Vasylenko, O. / Vostochno-Evropeiskii zhurnal peredovyh technologii, Vol. 6/2(48). 2010. PP. 60-64.

Authors' Information

Oleksii Vasylenko *– Aspirant of Kharkiv National University of Radioelectronics; Kharkiv, Ukraine;*

e-mail*: ichbierste@gmail.com tel.: +380 63 841 66 23*

Major Fields of Scientific Research*: General theoretical information research, Knowledge Discovery and Engineering, Business Informatics.*

Prof. Dr.-hab. Oleksandr Kuzomin *– Informatics chair of Kharkiv National University of Radio Electronics; Kharkiv, Ukraine Ukraine;*

e-mail*: kuzy@daad-alumni.de tel.: +38(057)7021515*

Major Fields of Scientific Research*: General theoretical information research, Decision Making, Emergency Prevention, Data Mining, Business Informatics.*