Software Metrics for Agent Technologies and Possible Educational Usage

MIRJANA IVANOVIĆ, University of Novi Sad
MARCIN PAPRZYCKI, Polish Academy of Sciences and Warsaw Management Academy
MARIA GANZHA, Polish Academy of Sciences and Warsaw University of Technology
COSTIN BADICA and AMELIA BADICA, University of Craiova

In a long history of inventing different methods for assessing software quality huge number of software metrics has been proposed. In this paper we will first give a brief overview of different approaches to measure software and all processes connected to software development. Particular attention further will be paid on metrics in agent technology and multiagent systems (MAS) that are based on "classical metrics" but introduce novel specific techniques in line with agent paradigm. In spite the fact that there is a number of metrics developed for software agents and MASs it is a little bit strange that in higher ICT education we do not have appropriate topics devoted to these very specific metrics. Such drawback should be resolved in some acceptable way. In this paper we try to propose some adequate ways and possibilities of improving current situation in ICT higher-education in agent paradigm and suggest several slight changes in existing curricula.

Categories and Subject Descriptors: **D.2 [SOFTWARE ENGINEERING]**; **D.2.8 [Metrics]; D.3 [PROGRAMMING LANGUAGES]; I.2.11 [Distributed Artificial Intelligence]:** *Multiagent systems;* **K.3 [COMPUTERS AND EDUCATION]**

General Terms: Measurement, Metrics, Education

Additional Key Words and Phrases: Agent-oriented metrics, Application of agent-oriented metrics in education.

1. INTRODUCTION

Software metrics are based on old discipline of measurement invented by different scientists. Based on this, overtime, essential principles to measure software and adequate connected activities are being proposed. First software metrics have appeared in the sixties. Among them, the most well-known is the Lines of Code (LOC) metrics. This and a lot of other metrics had been proposed to measure program quality through programmer's productivity. In long history of inventing different methods for assessing software quality in the literature, two important activities could be distinguished: the measurement and the software metrics.

Two fundamental definitions of the measurement were proposed by Norman Fenton [Alexandre 2002]. The first one states that: "Formally, we define measurement as a mapping from the empirical world to the formal, relational world. Consequently, a measure is the number or symbol assigned to an entity by this mapping in order to characterize an attribute". The second definition includes a numerical aspect: "Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules."

This work is partially supported by the Multilateral Agreement "Agent Technologies: Intelligent Agents and Smart Environments" between Serbia, Poland and Romania.

Author's address: Mirjana Ivanović, University of Novi Sad, Faculty of Sciences, Department of Mathematics and Informatics, Novi Sad, Serbia; e-mail: mira@dmi.uns.ac.rs; Marcin Paprzycki, Systems Research Institute Polish Academy of Sciences, Warsaw, Poland and Warsaw Management Academy, Warsaw, Poland; e-mail: paprzyck@ibspan.waw.pl; Maria Ganzha, Systems Research Institute Polish Academy of Sciences, Warsaw, Poland and Warsaw University of Technology, Warsaw, Poland; e-mail:, maria.ganzha@ibspan.waw.pl; Costin Badica, Department of Computer Science, University of Craiova, Craiova, Romania; e-mail: cbadica@software.ucv.ro; Amelia Badica, Faculty of Economics and Business Administration, University of Craiova, Craiova, Romania; e-mail: ameliabd@yahoo.com.

In: Z. Budimac, Z. Horváth, T. Kozsik (eds.): Proceedings of the SQAMIA 2016: 5th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications, Budapest, Hungary, 29.-31.08.2016. Also published online by CEUR Workshop Proceedings (CEUR-WS.org, ISSN 1613-0073)

The same author proposed the first definition of software metrics [Alexandre 2002]: "...software metrics is a collective term used to describe the very wide range of activities concerned with measurement in software engineering. These activities range from producing numbers that characterize properties of software code (these are the classic software 'metrics') through to models that help predict software resource requirement and software quality. The subject also includes the quantitative aspects of quality control and assurance - and this covers activities like recording and monitoring defects during development and testing."

Paul Goodman [Goodman 1993] proposed another definition of software metrics: "The continuous application of measurement-based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products".

Separately, measurements related to software efficiency are the core of, so-called, scientific computing. Here, the main question that is being answered can be stated as: "how efficient is given implementation of a given algorithm when being executed on a given computer" [Paprzycki and Stpiczynski 2006]. While it can be claimed, that this aspect of software metrics is already captured by the above definitions, what should be acknowledged is the fact that, during last 40+ years, scientific computing has developed its own metrics, well grounded in engineering practices.

During long period of developing and application of different software metrics, several domains of their general use are recognized:

- Metrics are essential to obtain objective reproducible measurements that can support software quality assurance, debugging, performance, estimating costs and management.
- Metrics are good instrument for discovering and predicting defects in code, but also for predicting project risks and success.
- Metrics are also oriented to quantifying several goals in project management: quality of software, cost and schedule of a software project, size/complexity of development.
- Metrics allow to select the most efficient approach to implementation of algorithms on complex large-scale computer systems.

Constant development, evaluation and application of different innovative software metrics are definitively necessary in order to obtain higher quality of software development and final product. Nevertheless the fact that majority of classical software metrics could be applied for software produced for specific areas, development of new metrics is still very dynamic area. As software is constantly getting more and more complex and adjusted to different very specific environments (distributed software, cloud computing, grid computing, web applications, ...), it is of essential importance to propose more adequate specific metrics. Majority of classical metrics also could be applied in agent technologies and multiagent systems, especially for measuring quality of source code, but specific behavior and functionalities of MAS need very subtle and particular metrics.

In this paper, we will present some existing approaches and propositions for software metrics in the area of agent technologies and multiagent systems. After that we will briefly discuss possibilities and need to include this topic in specific courses devoted to agent technologies and/or software testing in ICT study programs. Our main intention in this paper is just to open this important question as by our knowledge there are no studies that explore possibilities to include agent metrics in study programs.

As it is very delicate area and not we will avoid to precisely suggest topics, parts of courses, methodology of teaching and organization of exams but we will give general directions, suggestions and possible solutions.

The rest of paper is organized as follows. Section 2 gives an overview of classical software metrics that present bases for all other specific software types and their measurements and metrics. Section 3 brings different approaches and propositions of several authors to measure agents and multiagent systems. Section 4 considers several possibilities to introduce agent-oriented metrics in ICT study programs. Last section concludes the paper.

2. “CLASICAL” SOFTWARE METRICS

Software metrics could be described as process of measuring of specific properties of a segment of software or its specifications. As measurement and metrics are closely related in software engineering they are usually classified in: process measurement through process metrics, product measurement through product metrics, and project measurement through project metrics [Thakur 2016].

2.1 Process Metrics – Process metrics are connected to the quality and effectiveness of software process. They can determine different aspects like: effort required in the process, maturity of the process, effectiveness of defect removal during development, and so on. Some specific process metrics are: Number of errors discovered before the software delivered to users; Defect detected and reported during use of the software after delivery; Time spent to fix discovered errors; Human effort used for development of software; Conformity to schedule; Number of contract modifications; Relation between estimated and actual cost.

2.2 Product Metrics – Product metrics are oriented to the particular software products produced during the phases of software development. They can indicate if a product is developed strictly following the user requirements and involve measuring the properties of the software like the reliability, usability, functionality, efficacy, performance, reusability, portability, cost, size, complexity, and style metrics. These metrics are also known as quality metrics as they measure the complexity of the software design, size or documentation created. Product metrics additionally assess the internal product attributes like: (i) analysis, design, and code model; (ii) effectiveness of test cases; (iii) overall quality of the software under development. Some of important product metrics in the development process are mentioned below [Thakur 2016].

- ***Metrics for analysis model***: are connected to various aspects of the analysis model like system functionality, size.
- ***Metrics for design model***: are connected to the quality of design like architectural design metrics, component-level design metrics.
- ***Metrics for source code***: are connected to source code complexity, maintainability.
- ***Metrics for software testing***: are connected to the effectiveness of testing and are devoted to design of efficient and effective test cases.
- ***Metrics for maintenance***: are connected to assessment of stability of the software product.

2.3 Project Metrics –These metrics are connected to the project characteristics and its completion, and they could be of great help to project managers to assess their current projects. They enable: to track potential risks, adjust workflow, identify problem areas, and evaluate the project team's abilities. Some characteristic examples of these metrics are: number of software developers, productivity, cost and schedule.

Project measurements that include both product and process metrics are useful for project managers as they help in judging the status of projects so that the teams can react accordingly. Two groups of measures are recognized: *Direct and Indirect* [Jayanthi and Lilly Florence 2013].

- **Direct Measures**
  - Measured directly in terms of the observed attribute - length of source-code, duration of process, number of defects discovered.
  - Direct Measures (internal attributes) - cost, effort, LOC, speed, memory.
- **Indirect Measures**
  - Calculated from other direct and indirect measures.
  - Indirect Measures (external attributes) - functionality, quality, complexity, efficiency, reliability, maintainability.

Some authors also distinguish several other types of metrics mentioned below.

2.4 Requirements metrics [Costelo and Liu 1995] – are oriented towards different aspects of requirements and usually include: Size of requirements, Traceability, Volatility and Completeness.

2.5 Software package metrics [Kaur and Sharma 2015] – are connected to a variety of aspects of software packages:

- Number of classes and interfaces
- Afferent couplings – number of packages depending on the evaluated package
- Efferent couplings – number of outside packages the package under evaluation depends on
- Abstractness – number of abstract classes in the package
- Instability – represented as a ratio: efferent couplings/total couplings, in range 0-1.
- Distance from main sequence – represents how the package balances between stability and abstractness.
- Package dependency cycles – packages in the package hierarchy that depend on each other.

2.6 Software Maintenance Metrics [Lee and Chang 2013] – Some of very important metrics for the maintenance phase are: Fix backlog and backlog management index; Fix response time and fix responsiveness; Percent delinquent fixes and so on.

2.7 Resources metrics are also well known type of metrics, combined into several groups [Dumke et al. 2000]: personnel metrics that include skill metrics, communication metrics and productivity metrics; software metrics that include paradigm metrics, performance metrics and replacement metrics; hardware metrics that include reliability metrics, performance metrics and availability metrics.

2.8 Metrics related to the efficiency of implementation: as the area of large-scale computing has evolved "outside of mainstream software engineering" it is necessary to introduce and use these specific metrics as well. Metrics related to the efficiency of implementation among others include: raw speed, parallel speedup, parallel efficiency and so on [Paprzycki and Zalewski 1997], [Zalewski and Paprzycki 1997].

2.9 There are some metrics that are connected to customers like **Customer Problems Metric and Customer Satisfaction Metrics** [Lee and Chang2013] and they include: Percent of satisfied and completely satisfied customers; Percent of dissatisfied and completely dissatisfied customers; Percent of neutral customers.

In this section we tried to present very briefly essences of an important area of software engineering and software production. This area is in fact a new discipline oriented to software assessment, measurement, and constant development and application of wide range of software metrics. As different kinds of software systems and products are facilitate and support almost all human activities and everyday life, producing appropriate software is one of the most dynamic areas. As a consequence software has to be more and more reliable, safe and secure. Accordingly, for software engineering community, it is important and necessary to continue to develop more accurate ways of measuring very specific software products.

3. METRICS AND THEIR USAGE IN AGENT TECHNOLOGIES AND MULTIAGENT SYSTEMS

Agent technologies and Multiagent System (MAS) are specific software paradigm oriented towards building extensive distributed systems. This paradigm is based on several specific characteristics necessary for modeling, in natural way, very complex systems. Additionally these systems are deployed on variety of computing devices (within mobile ad hoc networks and satellite links) that are based on non-traditional communications ways. These systems are very specific and their development present new challenges for analyzing and describing system performance [Lass et al. 2009], [García-Magariño et al. 2010].

Regardless of popularity of this paradigm within the research community, majority of developed MASs are still a result of entirely theoretical research and there are very few examples of large scale systems applied in real environments, for more details, see [Ganzha and Jain 2013]. In spite this fact different approaches to applying measurement and more adjusted metrics in the area of agents and MAS have been developing recently. An older approach and description of essential characteristics of agents and multiagent systems and summary of metrics for agent-based systems is given in [Dumke et al. 2000]. In this paper we will be focusing to newest approaches in this area. Some of recent, particular approaches will be presented in the rest of the section.

3.1 METRICS IN AGENT TECHNOLOGIES I.

Some authors claim that measuring the complexity represent a good indicator to control the development of agent-based software and estimate the required effort to maintain it [Marir et al. 2014].

MAS are very complex software systems which abstract representation consists of agents. The agent is supported by an agent framework. Below the framework is the platform that executes on a host i.e. adequate computing device. All together they are incorporated, i.e. distributed in, and interact with the physical environment. According to that measurement can be performed at four layers in the model: agent, framework, platform, and environment/host layers [Lass et al. 2009].

In [Lass et al. 2009], authors recognize a variety of metrics that can be applied at each of the mentioned layers. These metrics may be divided into two groups, according to (i) their effectiveness (or performance), and (ii) types of data they represent.

Measures of Effectiveness – are connected to the ability of the system to complete assigned task in an environment.

Measures of Performance – describe not the quality of software solution but the quality of obtaining the solution. They represent quantitative measures of secondary performances like resource consumption (power consumed, communications range/time to successfully perform a certain task).

Data Classification – here metrics are divided into four categories depending on empirical determinations, mathematical transformations and statistical operations: nominal, ordinal, interval, and ratio. Below we will very briefly explain them, for more details see [Lass et al. 2009].

- ***Nominal*** measurements are labels assigned to data: equality of objects, number of cases, contingency correlation. This type of measure can determine equality of objects.
- ***Ordinal*** measurements are rankings or orderings of data described by “less than” or “greater than” property between data.
- ***Interval*** measurements are similar to ordinal measurements, but the differences between any two measures have meaning. In addition to the ordinal statistical operations, the mean, standard deviation, rank-order correlation and product-moment correlation are acceptable.
- ***Ratio*** measurements are similar to interval measurements except that there is an absolute zero point. Any statistical operation and multiplying by a constant are allowed.

3.2 METRICS IN AGENT TECHNOLOGIES II.

There are some approach based on the computational complexity of MAS [Dekhtyar et al. 2002], [Dziubiński et al. 2007] while others studied the complexity of MAS code i.e. software complexity [Dospisil 2003], [Klügl 2008].

Specific software complexity metrics are very important to evaluate developed MAS [Marir et al. 2014] and in this subsection we will concentrate on some of them.

The complexity of a software system is an important factor in measuring required effort to understand, verify and maintain the software. In MAS, the number of agents and structure of their interaction are the key factors that influence its complexity. Similarly the complexity of agents has a direct influence on the complexity of MAS.

At the agent-level, complexity can be analyzed according to two orthogonal criteria: the complexity of the agent’s structure and of its behaviors. On the other hand, complexity at the MAS level can be analyzed based on agents’ behavior and social interaction between agents in an environment. Some important metrics for measuring these components are proposed in [Marir et al. 2014] and will be very briefly present in the rest of this subsection. For more details see [Marir et al. 2014].

1. The structural complexity metrics - characteristic metrics in this group are enlisted below.

The Size of the Agent’s Structure - Different attributes and their combinations are used to specify the state of the agent. This metric is used to calculate the complexity of the agent’s structure.

The Agent's Structure Granularity - agent's structure is composed of attributes and some of them could be very complex. Accordingly it is necessary to assess the complex agent structure.

The Dynamicity of the Agent's Structure - apart from composed attributes, the ones of the agent can be of a container nature (for example attributes allow adding and removing variables like the list). Furthermore, the extensive dynamicity of the agent's structure causes instability and such dynamicity can be measured by identifying the structure update in two different moments.

2. The behavioral complexity metrics - using different metrics, also the behavioral complexity of the agent can be measured:

The Behavioral Size of the Agent - represents the indicator of the degree of agent's specialization and gives the number of behaviors ensured by the agent.

The Average Complexity of Behaviors - generally speaking, an agent with only one complex behavior can be more complex that an agent with several simple behaviors. So, it is possible to calculate the complexity of a behavior using well known cyclomatic metrics [McCabe 1976].

The Average Number of Scheduled Behaviors - an agent can launch several behaviors. So in the complex systems, the management of the scheduling different behaviors is not an easy task. In [Marir et al. 2014], authors proposed an indicator of the concurrency intra-agent. They calculate it as average number of behaviors scheduled in each moment.

3. The social structure complexity metrics - a set of agents that exist in an environment represent a MAS so the complexity of the social structure of MAS can be measured in different ways.

The Heterogeneity of Agents - this metric indicates the number of classes of agents, MAS composed of heterogeneity agents is more difficult than a homogeny MAS.

The Heterogeneity of the Environment's Objects - the existence of heterogeneous objects in the environment is in accordance to the complexity of the MAS.

The Size of the Agent's Population **-** in the MAS, the agents can be created and destroyed dynamically. The complexity of the MAS is increasing by increasing the number of agents.

4. The interactional complexity metrics - the interaction between agents is essential and it also can be measured by different metrics.

The Rate of Interaction's Code - this metric presents the rate of source code devoted to ensure the communication between agents. It gives only partial information about the collective behavior of agents as sent messages could be repeated.

The Average Number of Messages Exchanged per Agent - as previously mentioned metrics has drawbacks this metrics estimates the real number of exchanged messages. This metric is of dynamic nature and calculates required effort to understand the collective behavior of the MAS.

The Rate of the Environment's Accessibility – the agents can be interacting indirectly by manipulating the environment's objects. So, this static metrics shows the complexity of the MAS because of the existence of public attributes increases the agents coupling.

Proposed metrics can be also combined with, above mentioned, classical metrics (like the size in the *Lines of Code* metric), in order to provide more information about the complexity of agent-based software. Also it is possible to associate to each metric a weight, which reflects its importance. The appropriate weights are left to the appreciation of the users of the proposed metrics.

3.3 METRICS IN AGENT TECHNOLOGIES III.

In [Homayoun Far and Wanyama 2003], authors proposed several metrics based on the MAS complexity approached in objective and subjective way.

Subjective metrics - Subjective complexity is oriented to a human user who evaluates the complexity of the agent system. One proposition is to use a modified version of the Function Point method [Albrecht and Gaffney 1983] that accounts for algorithmic complexity. Important parameters involved in the model for each agent are: external inputs and outputs, external inquiries, external interfaces, internal data structures, algorithmic complexity and knowledge complexity factor. The overall complexity of the MAS is the mean of the adjusted MAS function point of its constituent agents.

Objective metrics - Objective complexity is based on complexity, seen as an internal property of the agent system. In this case, the cyclomatic complexity metrics can be used if the MAS system is nearly-decomposable. Complexity of the MAS is the sum of cyclomatic complexity of its constituent agents. For nearly-decomposability, the communicative cohesion metrics can be examined. The communicative cohesion metrics for an agent is defined in terms of the ratio of internal relationships (inter-actions) to the total number of relationships (i.e., sum of inter-actions and intra-actions).

3.4 METRICS IN AGENT TECHNOLOGIES IV.

In [Klügl 2008], the author distinguishes between overall system-level metrics that are relevant for the complete model, but also for the agent level metrics. Usually, agent metrics is oriented to measuring the population and the environmental complexity. However, interactions of agents are extremely interesting and are worth of being measured. Numerous metrics for system-level complexity are mentioned below [Klügl 2008].

- ***Number of agent types*** - is a measure for heterogeneity of the model i.e. it equals the number of agents (based on the number of classes).
- ***Number of resource types*** - is similar to the previous metrics, but for passive entities in the environment.
- ***Maximum number of agents*** - is metrics that represent the maximum number of agents concurrently present during a simulation run. However, there is a conceptual problem when the maximum number is only adopted at the beginning of the simulation.
- ***Minimum number of resources*** - is measure similar to the maximum number of agents and only actually used resources should be counted.
- ***Maximum delta of agent population*** - determines the variability of population numbers over a given interval of time. In fact it forms the rate of population change.
- ***Maximum delta of resource population*** - is the analogue to the previous metrics.
- ***Agent-resource relation -*** is the number of agents divided by the number of resources.
- ***Number of agent shapes*** - This is a measure for spatial complexity and represents different geometries that agents may possess.
- ***Number of resource shapes -*** is the similar to the previous metrics and represents number of different geometries that occur in the set of resources.
- ***Maximum resource status*** - Size Resources may be of differently complexity. This metrics counts the maximum number of status variables that resource may possess.
- ***Maximum resource parameter -*** This metrics computes the maximum number of parameters that influence the values of the status variables.

Abovementioned metrics are oriented to measuring population of agents and environmental complexity. Following group of metrics is oriented to measuring characteristics of individual agents.

- ***Architectural complexity -*** This is a measure of the agent architecture but indicators for it are not obvious. Authors of [Klügl 2008] proposed to rank the architectures into three categories based on their complexity and use it as a metrics: Behavior-describing architectures, Behavior-configuring architectures and Behavior-generating architectures.
- ***Action plasticity metric -*** Plasticity denotes the potential adaptivity of the behavior in reaction to the environmental influences. This metrics achieves full power when it is combined with additional measures concerning the behavioral plasticity and variability.
- ***Size of procedural knowledge*** - This metrics is also related to behavior plasticity. It is oriented to the size of the procedural knowledge that is available for the agent.
- ***Number of cognitive rules*** - This metrics is based on cognitive rules i.e. concept of sharing actions that affect the internal status or beliefs of an agent.

Above mentioned measures can be computed based on static model code. As interactions between agents are usually very dynamic, the values of metrics in the following group can be determined for agent-based simulations [Klügl 2008] during a simulation run.

- ***Sum of public information items*** - This measure is about the size of external interfaces and it represents the number of concurrently publicly accessible variables i.e. information items.
- ***Number of external accesses*** - This metrics is basically an abstraction from some message counting metrics. Here it is interesting to take care of how often external data is accessed by the agent in its behavior definition as addition to the number of available information units.
- ***Number of agent references (NAR) -*** This metrics represents the mean number of agents contained in one agents' internal model i.e. it addresses the coherence of the agent system. As this value may be varying over time, they can take value between NAR-mean and NAR-stdev but also minimum and maximum number of references as well as the time-related delta of these values.
- ***Number resource references*** - This metrics represent the number of references of an agent that it holds towards addressing resources.
- ***Number of mobility actions -*** This metrics represents the number of move actions per agent per time unit. This metric is only useful when there is an actual map where the agents may change their local position.

3.5 METRICS IN AGENT TECHNOLOGIES V.

Efficiency-oriented metrics are presented in [Chmiel et al. 2004], [Chmiel et al. 2005]. In these studies MASs were approached from the efficiency of implementation perspective. Here, among others, the following metrics have been experimentally evaluated – efficiency of: message exchanges, agent mobility, database access, agent creation and destruction, and combinations of these.

These metrics have been selected as they represent key characteristics of MAS that are actually being executed in a distributed environment.

The above mentioned metrics, in agent technologies cover a variety of relevant aspects. Although the authors proposed a wide range of appropriate metrics this area is still in constant development.

4. AGENT METRICS - POSSIBLE EDUCATIONAL USAGE

Empirical software engineering is one of the important directions in the software engineering approached as a discipline of knowledge. In this discipline, empirical methods are used to evaluate and develop wide range of tools and techniques. Recently, authors of [Birna van Riemsdijk 2012], proposed the use of empirical methods to advance the area of agent programming. Introducing systematic methodologies and qualitative measuring can establish multi-agent systems as a mature and recognized software engineering paradigm. Such methods could help in clear identification of advantages and application domains.

Furthermore, an essential question could be raised: is it necessary or could it be useful to introduce such specific topics in appropriate educational settings and ICT study programs? We can propose three possible ways of inclusion in educational processes specific empirical methods, measurements and metrics for agents and multi-agent systems.

- **Possibility 1** – Introduce brief, specific subtopic on quality measures and different metrics for MAS in regular, classical Software engineering courses. As it is usually very complex course that encompasses a lot of topics and practical work the idea would be just to give theoretical introduction of metrics for MAS and not to ask students for additional practical exercises.
- **Possibility 2** - One of interesting possibilities to apply quality measures and different metrics for MAS could address agent-oriented PhD theses. Usually PhD theses in area of agent technologies include implementation of a prototype or even a real world application. Some of them produce huge amount of code and high-quality software implementations. So application of different metrics in agent technologies and multiagent systems could be used for assessment of quality (or, at least, important characteristics) of these agent-oriented systems.

Some comparison between classical and MAS metrics would brought additional quality of theses.

- **Possibility 3** - Another interesting possibility to use quality measures and different metrics for MAS could be incorporation of several specific topics in some of existing courses within ICT study programs.

  Usually, within software engineering and/or ICT master study programs, there is a course devoted to software testing techniques. There are also some study programs that include agent technology course as an elective, or as a seminar course [Badica et al. 2014]. The one of motivations for this proposition comes also from positive experiences our colleagues from Poland have had in a "Software agents, agent systems and applications" course, offered for upper level undergraduate and first year MS students at the Warsaw University of Technology. They used specific experiments [Chmiel et al. 2004], [Chmiel et al. 2005] in order to design innovative homework exercises. During the course, students work in groups on semester-long projects. Their earlier experience indicated that students do not pay enough attention to the material delivered in-class, so they augmented it with homework exercises. Designing these activities, lecturers took into account the fact that, on the one hand, the key aspects of agent systems are messaging and mobility, on the other, students have only a limited amount of time, especially those who are close to graduating and work on their final projects. Therefore, they have designed two homework assignments. The first of them was similar to the "spamming test", while the other followed the "relay race" experiments. For each of them students had to implement a demostrator agent system (using JADE agent platform), perform series of experiments on multiple computers, and write a report on their findings. Lecturers came to quite interesting results. First, as expected, students have found that there is a direct relationship between the "quality of the code" and the efficiency of the developed system. For some of the students this was a real eye-opener, as execution time is rarely something that much attention is paid to. Second, they have found the JADE agent platform is quite robust. However, they have reported some issues when running it using wireless connections. Fourth, writing reports is an issue that is not paid enough attention to, during CS education and the proposed activities attempt at overcoming this shortcoming (see, also, [Paprzycki and Zalewski 1995]). Finally, it is definitively a valuable pedagogical addition to the agent systems course and leaves a room for introduction of some elements of agent systems measurements and application of some specific metrics.

  To conclude, for specific agent oriented courses (nevertheless mandatory or elective), it is essential to devote important part of the course to agent and MAS measurements and metrics, to compare and emphasize similarities and differences between them and classical metrics.

  Also in such courses is necessary to organize appropriate practical tasks and exercises. One possibility could be to give students a source code of MAS and ask them to perform different metrics and make comprehensive analysis and comparison of obtained results.

Recently agent technology becomes more and more important in realization of distributed, real-life very complex systems. So, students have to be familiar with such important technology and paradigm which they will probably use in their future jobs. So it is necessary to give them some at least basic insights in measuring software quality and application of appropriate metrics in agent systems. Courses devoted to agent technology or software testing represent good opportunity to introduce students with such specific and important topics.

5. CONCLUSION

Assessment and measurement of all phases, aspects, and activities of software development and final products is an old but still dynamic discipline and area of research. Nowadays there are a lot of developed and proposed metrics. Different types of software products with their specific characteristics require development and application of more appropriate metrics. It is evidently that there is even a number of metrics devoted to software agents and MASs.

Having in mind discussion presented in this papers it is a little bit strange that in study programs and in higher education generally we do not have appropriate topics of this, neither in Software engineering, Software testing, nor Agent-oriented courses. This is surely a mistake that should be considered and resolved and in the paper we try to propose some ways and possibilities of doing this.

ACKNOWLEDGMENT

This paper is a part of the Serbia-Romania-Poland collaboration within agreement on "Agent systems and applications."

REFERENCES

A. J. Albrecht and J. F. Gaffney. 1983. Software Function, Source Lines of Code and Development Effort Prediction: A Software Science Validation. *IEEE Trans. Software Engineering*, vol. 9, no. 6, pp. 639-648.

Simon Alexandre. 2002. Software Metrics: An Overview (Version 1.0). CETIC asbl - University of Namur, Software Quality Lab.

C. Badica, S. Ilie, M. Ivanović and D. Mitrović. 2014. Role of Agent Middleware in Teaching Distributed Network Application Development. *Advances in Intelligent Systems and Computing*, Volume 296/2014, Agent and Multi-Agent Systems: Technologies and Applications. ISBN: 978-3-319-07649-2 (Print) 978-3-319-07650-8 (Online). In *Proceedings of the 8th International KES Conference on Agents and Multi-agent Systems – Technologies and Applications*, Chania, Greece, 18-20. June, 2014, pp. 267-276. DOI: 10.1007/978-3-319-07650-8_27

M. Birna van Riemsdijk. 2012. Empirical Software Engineering for Agent Programming. *AGERE!* In: *Proceedings of the 2nd Edition on Programming Systems, Languages and Applications Based on Actors, Agents, and Decentralized Control Abstractions.* AGERE!'12. ACM, pp. 119–122.

K. Chmiel, D. Tomiak, M. Gawinecki, P. Kaczmarek, M. Szymczak and M. Paprzycki. 2004. Testing the Efficiency of JADE Agent Platform. In *Proceedings of the ISPDC 2004 Conference*, IEEE Computer Society Press, Los Alamitos, CA, pp. 49-57.

K. Chmiel, M. Gawinecki, P. Kaczmarek, M. Szymczak and M. Paprzycki. 2005. Efficiency of JADE Agent Platform, *Scientific Programming*, vol. 13, no. 2, pp. 159-172.

R. J. Costello and D. B. Liu. 1995. Metrics for requirements engineering. *Journal of Systems and Software*, vol. 29, pp. 39–63.

M. Dekhtyar, A. Dikovsky and M. Valiev. 2002. Complexity of Multi-agent Systems Behavior. *JELIA 2002, LNCS (LNAI),* vol. 2424, pp. 125–136.

J. Dospisil. 2003. Code Complexity Metrics for Mobile Agents Implemented with Aspect/JTM. *CEEMAS 2003, LNCS (LNAI),* vol. 2691, pp. 647–657.

R. R. Dumke, R. Koeppe and C. Wille. 2000. Software Agent Measurement and Self-Measuring Agent-Based Systems. *SMLab*. https://www.google.rs/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwir8JPYwo7OAhWE1SwKHYIED18QFggfMAA&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.156.3836%26rep%3Drep1%26type%3Dpdf&usg=AFQjCNHb8AYydoleabIt2cUigWLsKJjgCg

M. Dziubiński, R. Verbrugge and B. Dunin-Kęplicz. 2007. Complexity Issues in Multiagent Logics. *Fundamenta Informaticae* 75, 239–262

M. Ganzha and L. C. Jain. 2013. Multiagent Systems and Applications - Volume 1: Practice and Experience. *Intelligent Systems Reference Library 45*, Springer 2013. ISBN: 978-3-642-33322-4

I. García-Magariño, M. Cossentino and V. Seidita. 2010. A metrics suite for evaluating agent-oriented architectures. In *Proceedings of the 2010 ACM Symposium on Applied Computing SAC'2010*, pp. 912-919. DOI: http://dx.doi.org/10.1145/1774088.1774278

P. Goodman. 1993. Practical Implementation of Software Metrics. McGRAW HILL, New-York.

B. H. Far and T. Wanyama. 2003. Metrics for Agent-Based Software Development. *CCECE 2003 - CCGEI 2003*, Montréal, May 2003, pp. 1297-1300. DOI: http://dx.doi.org/10.1109/CCECE.2003.1226137

R. Jayanthi and M. Lilly Florence. 2013. A Study on Software Metrics and Phase based Defect Removal Pattern Technique for Project Management. *International Journal of Soft Computing and Engineering (IJSCE),* vol. 3, no. 4, pp. 151-155.

G. Kaur and D. Sharma. 2015. A Study on Robert C.Martin's Metrics for Packet Categorization Using Fuzzy Logic. *International Journal of Hybrid Information Technology*, vol. 8, no. 12, pp. 215-224. DOI: http://dx.doi.org/10.14257/ijhit.2015.8.12.15

F. Klügl. 2008. Measuring Complexity of Multi-Agent Simulations – An Attempt Using Metrics. *LADS 2007, LNCS (LNAI),* vol. 5118, pp. 123–138. DOI: http://dx.doi.org/ 10.1007/978-3-540-85058-8_8

R. N. Lass, E. A. Sultanik and W. C. Regli. 2009. Metrics for Multiagent Systems. *Performance Evaluation and Benchmarking of Intelligent Systems, Springer Science+Business Media, LLC 2009,* pp. 1- 19. DOI: 10.1007/978-1-4419-0492-8_1

M. C. Lee and T. Chang. 2013. Software Measurement and Software Metrics in Software Quality. *International Journal of Software Engineering and Its Applications,* vol. 7, no. 4, pp. 15-34.

T. Marir, F. Mokhati, H. Bouchelaghem-Seridi and Z. Tamrabet. 2014. Complexity Measurement of Multi-Agent Systems. *MATES 2014, LNAI 8732*, pp. 188–201.

T. J. McCabe. 1976. A Complexity Measure. *IEEE Transactions on Software Engineering*, vol. 2, no. 4, pp. 407-320.

M. Paprzycki and J. Zalewski. 1995. Should Computer Science Majors Know How to Write and Speak? *Journal of Computing in Small Colleges,* vol. 10, no. 5, pp. 142-151. http://ccscjournal.willmitchell.info/Vol10-94/No6b/Marcin%20Paprzycki.pdf

M. Paprzycki and J. Zalewski. 1997. Parallel Computing in Ada: An Overview and Critique. *Ada Letters*, vol. XVII, no. 2, pp. 55-62.

M. Paprzycki and P. Stpiczynski. 2006. A Brief Introduction to Parallel Computing. *Handbook of Parallel Computing and Statistics,* Taylor and Francis, Boca Raton, FL, pp. 3-41.

D. Thakur. 2016. Classification of Software Metrics in Software Engineering. http://ecomputernotes.com/software-engineering/classification-of-software-metrics, Accessed 1.7.2016.

J. Zalewski and M. Paprzycki. 1997. Ada in Distributed Systems: An Overview. *Ada Letters*, vol. XVII, no. 2, pp. 67-81.