Towards a set of Measures for Evaluating Software Agent Autonomy

Fernando Alonso, José L. Fuertes, Löic Martínez
Facultad de Informática
Universidad Politécnica de Madrid
28660 – Boadilla del Monte, Madrid (Spain)
e-mail: {falonso, jfuertes, loic}@fi.upm.es

Héctor Soza
Facultad de Ingeniería y Ciencias Geológicas
Universidad Católica del Norte
Avda. Angamos 0610, Antofagasta, Chile
mail: hsoza@ucn.cl

***Abstract*—Agent-oriented software is an established research field. For this reason, it is important to develop comprehensive measures of excellence to evaluate this software. No set of measures defining the overall quality of an agent has been developed to date. Some attempts at evaluating agent quality have addressed certain agent features, like the development process. We believe that agent quality can be determined as a function of well-defined characteristics. Evaluated using appropriate measures, these characteristics will assure an agent's reliability and correct functionality. This paper deals with an important agent feature, namely, autonomy. Autonomy is considered to be the agent's ability to operate independently, without the need for human guidance or the intervention of external elements. The article proposes a set of measures used to evaluate the autonomy of a agent and presents a case study analysing the behaviour of these measures.**

Keywords: Autonomy, Measures, Quality, Software agents

I. INTRODUCTION

A number of software development paradigms — procedural software, object-oriented software, agent-oriented software, etc. — have been developed throughout computer science history. Each development paradigm was designed to more efficiently produce software depending on the application type. However, software quality, that is, how able the software is to satisfy user needs, is the goal of any development. Several quality measures have been designed so far. In the case of the procedural and object-oriented paradigms, these measures have resulted in quality models [1], [2]. In 2001, the ISO and IEC established an international standard quality model [3]. This model decomposes overall software product quality into characteristics, sub-characteristics (attributes) and associated measures.

Research has been conducted on adapting some measures of procedural and object-oriented software to evaluate agent-oriented software quality. This initiative is based on the fact that these concepts have some characteristics in common with the agent paradigm, like its procedural programming approach, encapsulation, information hiding, etc. [4], [5], [6]. Few studies, though, have set out to develop measures exclusively targeting agent-oriented software [6], [7], and none have determined a quality model considering specific characteristics associated with the development and application of an agent.

The work presented here is part of a line of research aiming to evaluate the overall quality of an embedded agent considering its interactions with the user and other agents, to determine the efficiency and quality of its application. Research focuses on analysing the characteristics defining a agent. We presented early results measuring the social ability characteristic of a software agent elsewhere [8].

This paper presents a set of measures for evaluating agent autonomy, considering different attributes associated with this characteristic. Agent autonomy means the agent's ability to operate on its own, without the need for any human guidance or the intervention of external elements, and to control its own actions and internal states [4], [7].

The paper is structured as follows. The next section presents some research on measures related to agent autonomy. In section 3 we discuss software agent autonomy and its attributes. Section 4 suggests measures for evaluating agent autonomy attributes. Section 5 summarizes the process of calculating autonomy and its application to a case study. The last section includes some concluding remarks and discusses future research.

II. RELATED WORK

Although there are several studies on agent autonomy, we have found little relevant research on quality measures related to agent autonomy.

Barber and Martin authored one of the early papers on the issue of measuring agent autonomy [9]. They presented a complete framework for interpreting agent autonomy and delivering an autonomy representation for quantitatively assessing an agent's degree of autonomy. They state that overall agent autonomy could be measured as the mean or sum of the autonomies for all the pursued goals. They expressly state that such an evaluation is beyond the scope of their study.

Dumke, Koeppe and Wille set out a set of measures considering product, process and resources to evaluate the performance of agents and bring an empirical criterion into the evaluation [4]. In this study, the measure of autonomy of the agent's design considers measuring agent size and complexity.

Cernuzzi and Rossi proposed a framework for evaluating the agent-oriented analysis and design modelling methods [10]. The proposal takes into consideration qualitative evaluation criteria employing quantitative methods. They evaluated autonomy considering whether or not the

978-0-7695-3933-1/09 $26.00 © 2009 IEEE
DOI 10.1109/MICAI.2009.15

modelling technique checks if agents have control over their internal state and their own behavior.

In a project report, Shin outlined the results of adapting some product measurements from the procedural and object-oriented paradigms to agent-oriented software [6]. Shin compared objects and agents, and developed a program to evaluate the measures applied to an example. He suggested that agent complexity could be viewed as a way of measuring agent autonomy.

Huber authored another significant paper on measuring agent autonomy [11]. This work stressed the social aspects of agents and described autonomy as a measure of an agent's social integrity and social dependence. The measure of social dependence is a function of tasks imposed upon the agent by a superior agent, tasks accepted from peers, tasks contracted out to peers (and dependent upon completion) and tasks imposed upon inferior agents (and dependent upon completion). To compute an overall autonomy value for an agent, Huber combines the social integrity autonomy value and the social dependency autonomy value.

Huebscher and McCann presented a survey about autonomic computing [12]. They showed that this discipline has evolved to create self-managing software systems in a bid to overcome the complexities and inability to effectively maintain current and emerging systems. They proposed the main properties of self-management of an autonomic computing system: self-configuration (configures itself according to high-level goals), self-optimization (optimizes its use of resources), self-healing (detects and diagnoses problems) and self-protection (protects itself from malicious attacks but also from end users who inadvertently make software changes). They do not propose measures to evaluate these properties.

Generally, none of the above studies provides specific quality measures of software agent autonomy that could be used to get a global quality measure of the software agent. This is the main focus of this research.

III. SOFTWARE AGENT CHARACTERISTICS: AUTONOMY AND ITS ATTRIBUTES

In conformance with other key studies of software quality [13], [14], the quality of a agent can be analysed on the basis of its characteristics, sub-characteristics (or attributes) and attribute measures.

It is now an acknowledged fact in the agent-oriented software field that an agent must have the following basic characteristics: social ability, autonomy, proactivity, adaptability, intelligence and mobility (if the agent is mobile) [4], [7], [8]. Agent quality will then be determined by the set of quality attributes for each of the above characteristics, and these attributes can be evaluated by a set of measures.

Agent autonomy is a characteristic that is interpreted as freedom from external intervention, oversight, or control [15]. Autonomous agents are agents that "are able to work on behalf of their user without the need for any external guidance" [16]. This type of definition fits the concept of autonomy in domains that involve an agent interacting with other agents, as seen in this research.

From our experience with agents [8] and the literature survey of the field [4], [6], we propose that agent autonomy should consider three key attributes: self-control, functional independence and evolution capability. They are:

Self-Control: The ability of self-control is identified by the level of control that the agent has over its own state and behaviour [4]. For an agent to operate effectively, its self-control has to be effective and fast. The more complex its state is the less self-control it will have. This implies that the agent's internal state, which accommodates the agent's beliefs, goals and plans, should have a simple structure and be of a reasonable size to be operated on [6]. Good self-control depends on the complexity of the agent's internal state (evaluated as a function of structural complexity and internal state size) and of its behaviour complexity.

Functional Independence: Agent autonomy is a function of its structural and functional dependence [6], [16]. Functional dependence is related to executive tasks requiring an action that the agent has to perform on behalf of either the user it represents or other agents. A good level of functional independence will indicate that the agent does not have to perform many executive tasks.

Evolution Capability: The evolution of a agent refers to the capability of the agent to adapt to meet new requirements [17] and take the necessary actions to self-adjust to new goals [18]. An autonomous agent must be able to learn to adapt its state to attain new goals. This means that the agent's knowledge of how to modify its state must be permanently updated. Therefore, a good evolution capability depends on its state update capacity, and the frequency of state update [6].

IV. MEASURES FOR THE ATTRIBUTES OF AUTONOMY

Before introducing the measures for each attribute defining agent autonomy that we will use to evaluate this characteristic, we will discuss some general points related to the measures that we are introducing.

A. Considerations on Measurement

The measures used in this research are dynamic measures (i.e. measure the characteristics of the software during execution) and static measures (i.e. examine the source code to measure the characteristics of the software) [19].

To gather valid results in a software product evaluation using dynamic measures, this evaluation should be conducted in a controlled environment [20]. We will call this environment the benchmark. This assures that the evaluated measures are repeatable and comparable. This benchmark shall precisely specify the conditions in which the system under evaluation should be run for each dynamic measure.

We define each measure by a formula. The results of the measures are normalized in the interval [0, 1] (where 0 means a poor result for the measure and 1 means a good result). To normalize the values of the resulting measures, we use the functions shown in Figure 1. Some of these functions were successfully used for social ability software agent characteristic [8].

The constants k, k_1 and k_2 are parameters of these functions. They represent measure evaluation turning points. Taking the values of each measure, our proposal, to assess it, is to set a value interval in which k, k_1 and k_2 can be defined in terms of the evaluated measure. This provides for an adequate normalization of the measurement data.

We recommend that the software engineer can configure the value of the parameters to fine tune the formula performance for each particular case, considering a values interval for each parameter according to each measure.

The functions depend on the argument x and function (c) also depends on the argument p. The values of these arguments are defined by each measure.

Function (a) indicates that the value of the measure is constant at *1* (optimum measure value) until x reaches a value k (k indicates the point at which the value of the measure should no longer be considered optimum). So, as x grows, the value of the measure gently descends to zero, describing an exponential curve.

Function (b) indicates that the measure grows describing a parabola, as x increases up to a value defined by the parameter k_1. At this point, the measure remains unchanged at the maximum value *1* as long as x is between the parameters k_1 and k_2. Then its value starts to descend gently down to zero, describing an exponential curve.

Function (c) indicates that the measure descends constantly as it progresses, until it reaches the value *0* at the point given by argument p.

B. Definitions of Autonomy Attribute Measures

In this section we present the measures for the agent attributes of self-control, functional independence and evolution capability. These attributes define the agent's autonomy. Some of these measures are based on research into the question of autonomy within the agent paradigm, others were extracted from other paradigms and adapted to agent technology, and others are new measures proposed here.

1. Self-Control

The self-control attribute can be measured using the following measures:

Structural Complexity (*SC*): State structural complexity is determined by the quantity and complexity of the pointers or references that the agent uses.

Let n be the number of pointers and references existing in the agent's internal state and let CP_i be the complexity of the i^{th} pointer or reference.

Then *CP* is defined as equation (1):

$$CP = \sum_{i=1}^{n} CP_i \cdot \tag{1}$$

The *SC* measure follows function (a) in Figure 1, where x is *CP*. The complexity of CP_i is evaluated by counting the embedding level of the structures referenced by the i^{th} pointer or reference.

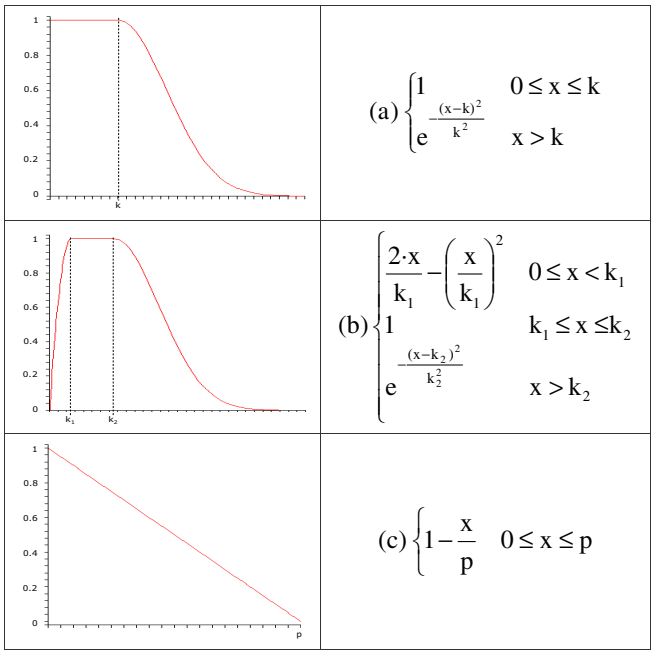

Figure 1. Formula types used in the measures.

The measure is considered optimum when the value of *CP* is less than k. On the other hand, if *CP* is greater than k, the value of the measure decreases rapidly. This is because an increased number and complexity of the structures referenced by pointers or references may take longer to execute resulting in a reduction in the agent's autonomy.

Internal State Size (*ISS*): The internal state size is determined by the size and number of the variables that the agent needs to define its internal state. *ISS* measures the agent's variable size. This is an adaptation of a measure described in [6]. Let us define n ($n > 0$) as the total number of variables, and let VB_i be the bytes size of the memory needed to represent the i^{th} variable or agent pointer (if the pointer measures the memory size of the referenced structure), then *MD* is defined as equation (2):

$$MD = \sum_{i=1}^{n} VB_i \cdot \tag{2}$$

The *ISS* measure follows function (b) in Figure 1, where x is *MD*. If the value of *MD* is less than k_1, the agent will have less self-control because it will not have all the information required for self-control and to attain its goals. *ISS* reaches its optimum value for values of *MD* from k_1 to k_2, and then drops rapidly as *MD* increases. This decline is due to the growing complexity of the agent's internal state, leading it to have to handle too much information to be able to control itself.

Behaviour Complexity (*BC*): This measure evaluates the complexity of the services that the agent offers (only applies to agents that offer services). A service offered by an agent implies performing a series of actions, such as the operations to be executed by the agent to carry out the offered service. The complexity of these services differs

depending on the paradigm implementing the agent (object-oriented, knowledge-based system, etc.). Therefore, agent services complexity will be a function of the paradigm implementing the agent [21]. Let us define n ($n > 0$) as the total number of services and let CS_i be the complexity of the i^{th} service, considering the measure defined for the paradigm used as the complexity measure, then CS is defined as equation (3):

$$CS = \sum_{i=1}^{n} CS_i \cdot \quad (3)$$

The BC measure follows function (a) in Figure 1, where x is CS. Good measures of agent complexity are presented in [22], [23] and [24]. The value of BC is considered to be optimum if the value of CS is low, less than k for the system. For values of CS greater than k, the value of the BC measure starts to decrease. This is because the increased complexity of implemented services could possibly affect agent autonomy as it could increase the time and effort required to execute its services [21].

2. *Functional Independence*

The functional independence attribute can be measured using the following measure:

Executive Messages Ratio (*EMR*): It measures the influence on the agent of the ratio of executive messages (requiring an action) received from the user that the agent represents or other agents (to which it is obliged to respond) to all the received messages (considering the communication actions). Above all, it takes into account FIPA Request messages [25]. Let MR be the total number of messages received and let ME be the number of executive messages received by the agent during execution. This measure is only applicable when the agent receives messages ($MR > 0$). EMR, which follows function (c) in Figure 1, is defined by equation (4):

$$EMR = 1 - \frac{ME}{MR} \cdot \quad (4)$$

If the value for EMR is high, the agent's autonomy is high because, as it receives few executive messages, it has to execute fewer actions. This affects its autonomy. Having to respond to a high number of executive messages (a low EMR value) can penalize the agent's functional independence.

3. *Evolution Capability*

The evolution capability attribute can be measured using the following measures:

State Update Capacity (*SUC*): This static measure is useful for evaluating the software agent's capability to update its state. The agent's state is defined by a set of variables that are dependent on different event occurrences, where the event would change the variable value, and therefore the agent state [6]. This is an adaptation of a measure described in [6]. Let us define n ($n > 0$) as the number of all executable statements, let m be the number of variables and let S_{ij} be 1 if the i^{th} statement updates the j^{th} variable, and 0 otherwise. Let AS be the mean value of variables updated by agent statements, defined as equation (5):

$$AS = \sum_{i=1}^{n} \sum_{j=1}^{m} S_{ij} \cdot \quad (5)$$

The SUC measure follows function (b) in Figure 1, where x is AS. As the value of AS increases up to the value of k_1, the value of SUC also grows rapidly because each variable is dependent on a growing number of statements with changeable values. This influences the agent's internal state and, therefore, its evolution capability [6]. The measure reaches the optimum value between k_1 and k_2. Finally, when the value of AS goes above k_2, the value of SUC decreases because the agent's knowledge update process now involves so many variables that it is unable to evolve properly.

Frequency of State Update (*FSU*): This measure is useful for evaluating the impact of the state update frequency during the execution of the variable defining the agent state. Depending on what the knowledge is used for, this frequency of change could have a big impact on agent predictability and behaviour [6]. This is an adaptation of a measure described in [6]. Let us define n ($n > 0$) as the number of all executable statements, let m be the number of variables and let us define VC_{ij} as 1 if the i^{th} statement accesses and modifies the j^{th} variable during the execution of the benchmark. Then, we define FV (the frequency of the change of variables inside the agent) by equation (6):

$$FV = \sum_{i=1}^{n} \sum_{j=1}^{m} VC_{ij} \cdot \quad (6)$$

The FSU measure follows function (b) in Figure 1, where x is FV. As long as FV is less than k_1, the value of FSU increases because the agent's knowledge update frequency, and hence its evolution capability, increases up to the optimum value when FV is between the values of k_1 and k_2. Above this second value (k_2), its evolution capability starts to drop because the knowledge update frequency can become so high as to prevent it from being able to take the appropriate actions to evolve. The values of parameters k_1 and k_2 depend on agent programming.

V. CASE STUDY

To evaluate the studied measures we have used a multi-agent system with six agent types: three buyers and three sellers [26]. It is an intelligent agent marketplace which includes different kinds of buyer and seller agents that cooperate and compete to process sales transactions for their owners. Additionally, a facilitator agent was developed to act as a marketplace manager. We have used this system to evaluate the functional quality of the buyer and seller agents' social ability [8] and autonomy within the system.

The agents are basic, intermediate and advanced buyers and sellers. They have the same negotiation capacities, but they differ as to how sophisticated the techniques used to implement their negotiation strategies are, ranging from simple, hard-coded logic to forward-chaining rule inference. The seller agents send messages reporting the articles that they have to sell, and the buyers respond stating their willingness to buy and what they offer for the article. The seller agents respond by accepting or rejecting the offer, and,

when they receive this message, the buyer agents return a confirmation message.

The autonomy study focuses on the six system agents. Figure 2 shows the values of the measures taken during the evaluation (the box-and-whisker plots in Figures 2 and 3 compare the minimums, 25th percentiles, medians, 75th percentiles and maximums of the measured values).

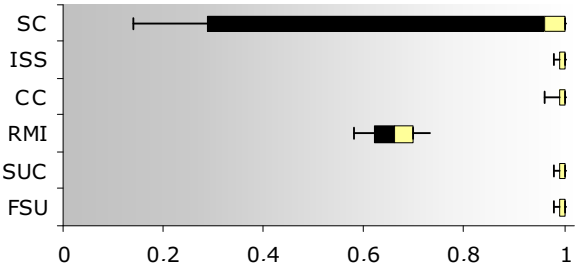

Figure 2. Box-and-whisker plot of the autonomy attribute measures

Figure 2 shows that, unlike the *SC* and *EMR* measures, the *ISS*, *BC*, *SUC* and *FSU* measures have very high values. *SC* has a very low minimum value. It inherits this value from the basic agents, as a great many, very complex pointers and references are used for these agents, whereas the intermediate and advanced agents have less complex pointers. On these grounds, the median is high and the percentiles have a wide range of values. The *EMR* measure has scored intermediate values, as the ratio of the number of executive messages over the total number of messages received is similar for all three agent types, and its values are close to the mean. The results for the other measures applied to all three agents are excellent, highlighting that the properties of these attributes are good for these agents and have a positive impact on their autonomy.

Figure 3 shows that basic agents have a very high median, as the intermediate value between the results of agent structural complexity (low value) and the other measures considered (high value) leads to a dispersion of the percentile values and a high maximum value. The measures for the other agents are higher thanks to which their results are less dispersed. From this we conclude that the agents are more autonomous.

Figure 4 shows the values of the measures for the attributes of the autonomy characteristic aggregated using the arithmetic mean of their values. We find that the EMR and SC measures are lower because the basic system agents have a lower associated value and the other measures are close to the maximum possible value. From these results we conclude that the system agents have a very high evolution capability, high self-control and above average functional independence.

Figure 5 shows that evolution capability is very high for all agents. This is followed by self-control. Self-control is lower for the basic agents because their state change is low. Finally, functional independence is lower for all agents, as the *EMR* measure, which depends on the ratio of executive messages received over all messages received, is average for the agents of this system.

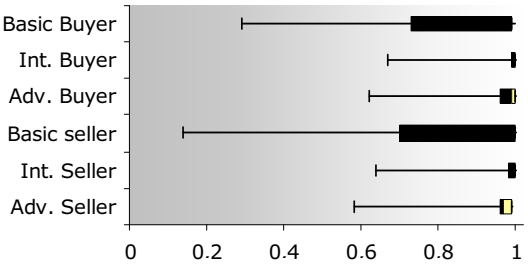

Figure 3. Box-and-whisker plot for system agents

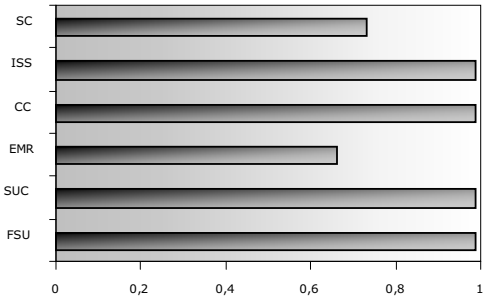

Figure 4. System average by autonomy attribute measures

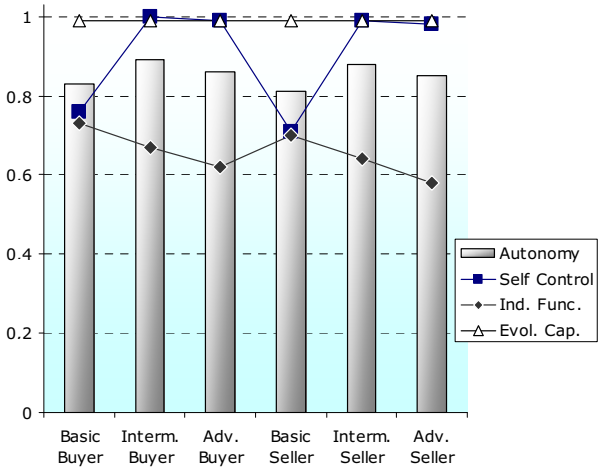

Figure 5. Values of autonomy and its attributes

Because all agents have an adequate state update capacity, the evolution capability attribute scores high for all agents, as *SUC* (state update capacity) and *FSU* (frequency of state update) have high values in all cases. This can be attributed to agent programming. Thanks to these measure values, this is the highest-scoring, and therefore the most important, attribute for this system. According to these results, mean autonomy for this system is *85%* (aggregated using the arithmetic mean of their values), all agents scoring above *80%*, where the autonomy of the basic agents is slightly lower than for the others.

VI. CONCLUSIONS AND FUTURE WORK

This paper is part of ongoing research aimed at defining the global quality of software agents. To define this overall quality, we first identified relevant software agent characteristics from the literature: social ability, autonomy, proactivity, reactivity, adaptability, intelligence and mobility. For each characteristic, we plan to define a set of attributes and for each attribute, a set of measures.

We addressed the social ability characteristic in previous work [8]. In this paper, we present a first approximation to a set of measures for the autonomy characteristic. This characteristic is divided into three attributes: self-control, functional independence and evolution capability. We also provide a total of six measures for the attributes. We have applied these measures to a typical case study to evaluate the applicability of the proposed measures and the relevance of the identified attributes. This case study is an intelligent agent marketplace with three types of seller agents (basic, intermediate and advanced), three types of buyer agents and one facilitator agent. The autonomy of the designed agents is high (85%). Two attributes have very good values: evolution capability (99%) and self-control (91%), whereas the values for functional independence are much lower (although greater than 60%).

Our future work pursues the ultimate goal of evaluating the global quality of software agents. First, we will have to evaluate the remaining characteristics: proactivity, adaptability, intelligence and mobility. Then we will define an aggregation method for computing the global quality of a software agent, given the results of all the measures of all the attributes of the agent's characteristics. This method will have to deal with the diversity of agent types and multi-agent systems. For this reason, it has to provide ways to adapt the computation process to different situations. What we are actually doing is developing a quality evaluation model for software agents. This model considers agent types, agent characteristics, attributes and measures.

REFERENCES

[1] J. A. McCall, "An Introduction to Software Quality Metrics", Software Quality Management, J. D. Cooper and M. J. Fisher, (eds.) Petrocelli Books, New York, NY, 1979, pp. 127–142

[2] J. L. Fuertes, "Modelo de Calidad para el Software Orientado a Objetos", Doctoral Thesis. School of Computing, Technical University of Madrid, Madrid, Spain, 2003.

[3] ISO/IEC. 2001. Software engineering- Product quality- Part 1: Quality Model. International Standard ISO/IEC 9126-1:2001.

[4] R. Dumke, R. Koeppe, C. Wille, "Software Agent Measurement and Self-Measuring Agent-Based Systems", Preprint No. 11, Fakultät für Informatik, Otto-von-Guericke-Universität, Magdeburg, 2000.

[5] B. Far, T. Wanyama, "Metrics for Agent-Based Software Development", Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering CCECE 2003, pp. 1297—1300.

[6] K. Shin, "Software Agents Metrics. A Preliminary Study & Development of a Metric Analyzer", Project Report No. H98010, Dept. Computer Science, School of Computing, National University of Singapore, 2003/2004.

[7] C. Wille, R. Dumke, S. Stojanov, "Quality Assurance in Agent-Based Systems Current State and Open Problems", Preprint No. 4, Fakultät für Informatik, Universität Magdeburg, 2002.

[8] F. Alonso, J. L. Fuertes, L. Martínez, H.Soza,"Measuring the Social Ability of Software Agents", Proceedings of the Sixth International Conference on Software Engineering Research, Management and Applications, Prague, Czech Republic, 2008, SERA 2008, pp. 3-10.

[9] K. S. Barber, C. E. Martin, "Agent Autonomy: Specification, Measurement, and Dynamic Adjustment", Proceedings of the Autonomy Control Software Workshop at Autonomous Agents, Seattle, WA, Agents'99,1999, pp. 8–15.

[10] L. Cernuzzi, G. Rossi, "On the Evaluation of Agent Oriented Methodologies", Proceedings of the International Conference on Object Oriented Programming, Systems, Languages and Applications - Workshop on Agent-Oriented Methodologies OOPSLA 02, Seattle, WA,. 2002, pp. 2 –30.

[11] M. J. Huber, "Agent Autonomy: Social Integrity and Social Independence", Proceedings of the International Conference on Information Technology ITNG'07, Las Vegas, Nevada. IEEE Computer Society, Los Alamitos, CA, 2007, pp. 282--290.

[12] M.C. Huebscher, J. A. McCann, "A survey of Autonomic Computing-Degrees, Models, and Applications", ACM Computing Surveys, Vol. 40, No. 3, Article 7, 2008, pp. 1–25.

[13] S. Covey, "The Seven Habits of Highly Effective People", Free Press, 15th anniversary edition, 2004.

[14] K. Dautenhahn, C. L. Nehaniv, "The Agent-Based Perspective on Imitation. In: Imitation in Animals and Artefacts" C. L. Nehaniv and K. Dautenhahn, (eds.) The MIT Press, Cambridge, MA, 2002, pp. 1--40.

[15] R. Beale, A. Wood, "Agent-based Interaction. In: Proceedings of People and Computers", IX: Proceedings of HCI'94, Glasgow, UK, 1994, pp. 239–245.

[16] M. Wooldridge, "An Introduction to Multiagent Systems", John Wiley & Sons, Inc., 2002.

[17] J. Murdock, "Model-Based Reflection for Agent Evolution", Technical Report GIT-CC-00-34, Doctoral Thesis. Georgia Institute of Technology, Atlanta, 2000.

[18] B. Friedman, H. Nissenbaum, "Software Agents and User Autonomy". Proceedings of the First International Conference on Autonomous Agents, 1997, pp. 466—469.

[19] G. M. Barnes,B. R. Swim, " Inheriting Software Metrics", J. Object-Orient. Prog. 6, 7, 1993, 27—34.

[20] ISO/IEC. 2004. Software engineering- Product quality- Part 4: Quality in use metrics. International Standard ISO/IEC TR 9126-4:2004.

[21] S. R. Chidamber, C. F. Kemerer, "A Metrics Suite for Object Oriented Design", IEEE T. Software Eng. 20, 6, 1994, pp. 476—493.

[22] L. Etzkorn, J. Bansiya, C. Davis, "Design and Code Complexity Metrics for OO Classes", J. Object-Orient. Prog. 12, 1, 1999, pp. 35—40.

[23] T. J. McCabe, "A Complexity Measure", IEEE T. Software Eng. SE-2, 4 Dec. 1976, pp. 308—320.

[24] D. Tran-Cao, G. Levesque, A. Abran, "Measuring Software Functional Size: Towards an Effective Measurement of Complexity", Proceedings of the International Conference on Software Maintenance ICSM'02, Montreal, Canada, 2002, pp.11-17.

[25] Foundation for Intelligent Physical Agents. FIPA Communicative Act Library Specification, Document Number SC00037J, Geneva, Switzerland, 2002.

[26] J. P. Bigus,J. Bigus,"Constructing Intelligent Agents using Java" 2nd ed. John Wiley & Sons, Inc., New York, 2001.