2010 Fifth International Conference on Software Engineering Advances

Measuring the Pro-Activity of Software Agents

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

Héctor Soza
Escuela de Ingeniería,
Universidad Católica del Norte
Larrondo 1281
Coquimbo, Chile
e-mail: hsoza@ucn.cl

***Abstract*—Despite having well-defined characteristics, software agents do not have a developed set of measures defining their quality. Attempts at evaluating software agent quality have focused on some agent aspects, like the development process, whereas others focusing on the agent as a software product have basically adopted measures associated with other software paradigms, like procedural and object-oriented concepts. Here we propose a set of measures for evaluating software agent pro-activity, the software agent's goal-driven behavioral ability to take the initiative and satisfy its goals.**

Keywords: Agents quality; pro-activity; software quality

I. INTRODUCTION

Some research has been conducted on adapting procedural and object-oriented measures to evaluate agent-oriented software [1], [2], [3]. However, few studies have focused on developing measures exclusively targeting agent-oriented software [3], [4]. In actual fact, there has been no investigation to set up a quality model considering the specific characteristics of a software agent.

The work presented here is part of a line of research focusing on evaluating the overall quality of a software agent as an autonomous entity interacting with other agents and users. Early results measuring software agent social ability and autonomy characteristics were presented in [5], and [6], respectively.

This paper presents a set of measures for evaluating a software agent's pro-activity, considering its associated attributes. Agent pro-activity means the agent's ability to exhibit goal-driven behavior by taking the initiative in order to achieve its goals [7].

The paper is structured as follows. Section 2 presents some related work on the development of agent software-related measures. In Section 3 we discuss agent pro-activity and its attributes. Section 4 suggests measures for agent pro-activity. Section 5 summarizes the process of calculating pro-activity and its application to a case study. The last section includes some concluding remarks and future research.

II. RELATED WORK

We have found very little relevant research on quality measures related to agent pro-activity. Of the few examples, Lin and Carley addressed the issue of proactive and reactive agent style from a theoretical perspective. They proposed evaluating the effect of agent style and found that it depends on how the agent is trained and the internal condition under which it operates [8].

Considering whether or not the agent dynamically assumes the different goals and whether it is possible to model goals, Cernuzzi and Rossi [9] evaluated software agent pro-activity using discrete values.

Shin measured agent pro-activity by evaluating its frequency of knowledge discovery [10]. This measure is the result of calculating the number of messages that the agent uses to discover knowledge.

So and Sonenberg [11] described an alternative approach of interpreting proactive behavior in software agency, closely related to the underpinning cognitive process. To demonstrate a practical use of their model, they outlined a meta-level control strategy that focuses agent attention on the key aspects of its environment at runtime.

To the best of our knowledge, there is no more literature related to measuring the characteristic of software agent pro-activity. This is the focus of this research.

III. PRO-ACTIVITY AND ITS ATTRIBUTES

Following on from other studies of software quality [1], [3], we divide quality into characteristics, subcharacteristics (or attributes) and measures of the attributes. As a result, the quality of the software agent will be evaluated by the set of measures of the attributes, associated with each of the characteristics.

It is well-known that the characteristics defining the behavior of the software agent are social ability, autonomy, pro-activity, reactivity, adaptability, intelligence and mobility [4], [5], [12], [13].

As regards the characteristic of software agent pro-activity, we propose, based on our experience with software agents [5], [6] and the existing research [7], [14], that it should be identified by three attributes:

- **Initiative** is the agent's ability to take an action with the aim of achieving its goals [13], [15]. Also, it is the ability to satisfy its goals by means of goal-driven behavior [7].
- **Interaction** is the agent's ability to interact with other agents, the user and its environment [14]. This is important when a group of agents interact with each other to solve a problem that is beyond the ability and knowledge of each individual agent [16].

978-0-7695-4144-0/10 $26.00 © 2010 IEEE
DOI 10.1109/ICSEA.2010.55

Agent interactions are important for the efficiency, performance and overall quality of multi-agent applications [17].

- **Reaction** is the ability to react to a stimulus from the environment, according to stimulus/response behavior [18]. Agents react appropriately according to the context in which they operate [19].

IV. PRO-ACTIVITY MEASURES

First let us make some points about the measures and then present the proposed measures for pro-activity attributes.

A. *Considerations about measures*

Fig. 1 shows the formulae and their associated curves that are used to normalize the pro-activity measures.

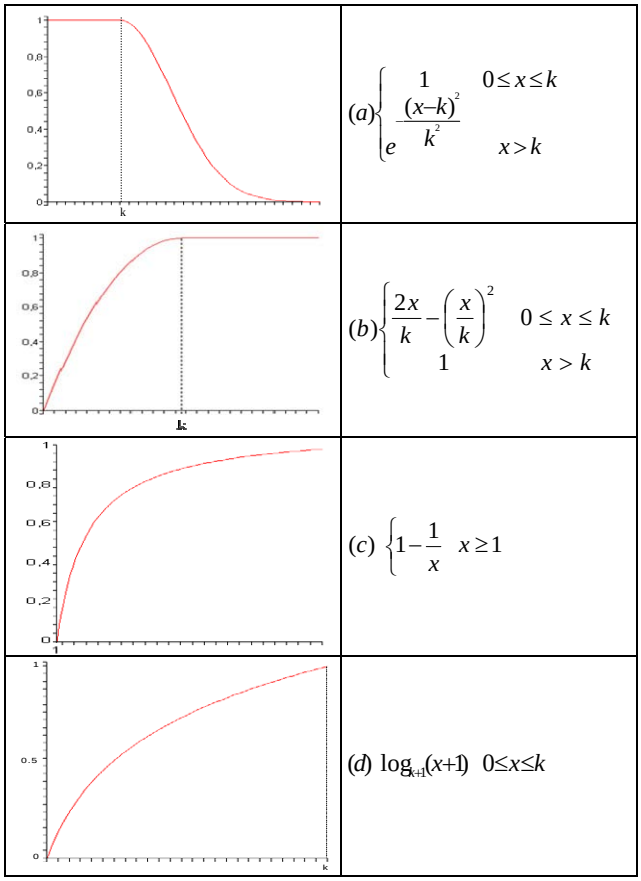

Figure 1. Formula types used in the measures

We interviewed several agent-oriented software experts about how each pro-activity measure should behave. This way we were able to determine which the best curves for the measures were. They are illustrated in Fig. 1.

The measures should be evaluated in a controlled environment, called the benchmark. The benchmark specifies the conditions in which the system under evaluation should be run for each dynamic measure. This assures that the evaluated measures are repeatable and comparable [20].

The measure for each attribute is a function of one or more parameters. The results of each measure are normalized in the interval [0, 1] (where 0 is a poor result and 1 is a good result for the measure).

Curve (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 to be optimum). As x grows, the value of the measure gradually descends to zero, describing an exponential curve. Curve (b) indicates that the measure grows, describing a parabola, as x increases up to a value defined by the parameter k. At this point, independently of the increasing value of x, the measure reaches the maximum value 1. Curve (c) indicates that the value of the measure grows rapidly, as x increases. Curve (d) indicates that the measure grows, rapidly at first, and, as it progresses, its value increases until it reaches the value 1 when x reaches the value k.

The formulae depend on the argument x, where x is a value defined for each measure. The constant k is a parameter that the software engineer can configure to fine tune formula performance for each particular case.

B. *Proposed measures for pro-activity attributes*

The proposed measures for evaluating the attributes defined for the software agent characteristic of pro-activity are as follows. The measures that we have defined are based on research on the agent paradigm, other measures selected from other paradigms (procedural, object-oriented, etc.) and adapted to agent-oriented software, and new measures proposed here.

Initiative: this attribute can be measured using the following measures.

- *Number of roles* (*NOR*): *NOR* measures the number of potential roles that agents perform. Roles are defined in the system design phase and may be allocated dynamically or change in the course of the agent execution, requiring the implementation of additional functions to satisfy the needs of the different roles. A new function does not necessarily need to be implemented for each new role, but some complex roles have to achieve more goals, calling for the implementation of more functions [10]. Let us define the number of agent roles as *NR*. The *NOR* measure describes curve (a) in Fig. 1, where x is the value *NR*. If $NR < k$, *NOR* is optimal, because agent initiative is greater if the agent assumes fewer roles, as it has sufficient abilities to take the necessary actions and achieve the goals defined for these roles during execution. But, the agent starts to lose initiative above the value k, because, as a result of having to perform more roles, the number of activities that the agent has to execute to achieve the goals defined for the above roles increases, and there is no guarantee that it will be able to achieve all the goals during execution. The software engineer should define the value of parameter k based on experience and depending on the application,

estimating the mean number of roles that each agent should be capable of performing without loss of initiative.

- *Number of goals (NOG)*: *NOG* measures the number of goals achieved by the agent during execution with respect to the number of allocated goals. Let us define the number of goals achieved by the agent during execution as *NG*. The *NOG* measure describes curve (d) in Fig. 1, where *x* is the value *NG*. The value of parameter *k* is the number of goals to be achieved by the agent. A lower number of attained goals than allocated goals indicate that agent initiative is low. This could be due to the agent not performing all the actions necessary to achieve the goals. As *NG* increases, agent initiative grows until *NG* reaches *k*, when the agent achieves all of its goals.
- *Messages to achieve the goals (MAG)*: *MAG* measures the agent's initiative to achieve its goals by communicating with other system agents. Let us define *MG* as the fraction of the total number of executive messages to the total number of messages that are sent during agent execution, divided by the total number of goals achieved. If *EM* is the total number of executive messages, *TM* is the total number of messages and *n* is the total number of goals achieved, then let us define *MG* as (1). The *MAG* measure describes curve (b) in Fig. 1, where *x* is the value *MG*. As the value of *MG* increases, agent decision-making or problem-solving initiative increases, enabling the agent to attain its goals during execution. When $MG \geq k$, this measure is optimal, and agent initiative increases because more messages aimed at achieving its goals are sent.

$$MG = \frac{\frac{EM}{TM}}{n}. \quad (1)$$

We suggest that the software engineer should define the value of parameter *k* based on experience and depending on the application by setting an agent-dependent parameter value that identifies the optimum or least number of messages to be exchanged to maximize initiative.

Interaction: this attribute can be measured using the following measures.

- *Methods per class* (*MC*): *MC* measures the number of the methods implemented within the agent enabling it to achieve its goals (number of implemented public methods or offered services, not including internal methods). If the agent has many different methods for achieving a goal, it will be able to interact better and will have a better chance of acting pro-actively to achieve its goals [10]. MC is a static measure and does not account for the calling of the methods. Let us define *m* as the number of methods implemented within the class (assuming that $m \geq 1$). The measure *MC* describes curve (c) in Fig. 1, where *x* is the value of *m*. It holds that the more different methods the agent has, the greater the agent interaction is, as the agent can take advantage of these methods and the services they offer to achieve its goals, thereby increasing agent pro-activity.
- *Number of message types* (*NMT*): *NMT* measures the number of different types of agent messages that the agent can process. This increases the agent's ability to interact with the user and other agents in the environment, thereby increasing its pro-activity. We will comply with FIPA standards for the agent message type [21]. The more message types an agent can handle, the better developed its interaction capability will be, because the agent has more options of communicating with the user and system agents [10]. If *IM* and *OM* are the number of unique incoming and outgoing message types, respectively, then we define *MT* as (2).

$$MT = IM + OM \quad (2)$$

The *NMT* measure describes curve (b) in Fig. 1, where *x* is the value *MT*. If $MT < k$, agent interaction with users and environment agents is low, as there are few existing alternatives for communication with the agent. If $MT \geq k$, the rate of agent communication with the environment improves, thereby improving interaction with other system agents and users, since they have a better chance of contacting the agent. We suggest that the software engineer should determine parameter *k* considering the minimum number of message types enabling environment communication with the agent.

Reaction: this attribute can be measured using the following measures.

- *Number of processed requests* (*NPR*): *NPR* measures the agent's ability to react to the number of received and resolved requests during execution. Let us define the number of received requests (requiring an action to be taken in response) during execution as *MN*. The measure for *NPR* describes curve (a) in Fig. 1, where *x* is the value *MN*. The fewer requests the agent receives, the greater its ability to react because it has more time to process and attain its goals, thereby increasing its pro-activity. As the number of received requests increases, the value of *NPR* falls, because the agent's ability to respond during execution decreases. We consider that the software engineer should determine parameter *k* based on experience

and depending on the application ensuring that a suitable number of processed requirements is estimated for the agent to react properly. The message types associated with the requests made by the agent considered by this measure are as proposed in the FIPA standard [21].

- Call for Proposal (action of calling for proposals to perform a given action)
- Propose (action of submitting a proposal to perform a certain action, given certain preconditions)
- Request (the sender requests the receiver to perform some action)
- Request When (the sender wants the receiver to perform some action when some given proposition becomes true)
- Request Whenever (the sender wants the receiver to perform some action as soon as some proposition becomes true and thereafter each time the proposition becomes true again).

- *Agent operations complexity* (*AOC*): *AOC* measures the mean complexity of the operations to be performed by the agent to achieve its goals. The more complex the operations to be performed are, the less able the agent is to react and pro-actively achieve its goals and vice versa. We consider that, to evaluate this measure, the software engineer might use any complexity regarded as suitable for achieving a good result, such as, for example, cyclomatic complexity [22] or similar. If C_i is the complexity of the operations associated with the i^{th} goal and n is the number of agent goals, then the mean complexity per goal is defined by *PC* (3).

The measure for *AOC* describes curve (a) of Fig. 1, where x is the value *PC*. The value of *AOC* is optimal as long as the complexity of the operations to be performed to achieve agent goals is low. This increases its pro-active ability to react until it reaches the parameter k, as of when the value of *AOC* decreases because the agent is less able to react to achieve its goals. The value of parameter k will depend on the complexity used to evaluate the operations. With respect to cyclomatic complexity, there are several studies that propose different ranges for the upper bound of module-associated risk. A value proposed after several studies is 11 [22], [23], [24], which is a possible baseline value for the software engineer to use to set parameter k.

$$PC = \frac{\sum_{i=1}^{n} C_i}{n} \tag{3}$$

V. Case Study

A. Measures and Evaluation

We have to calculate a single value for each attribute from the attribute measures assessed in a normalized range between 0 and 1, and then use these attribute values to calculate a single value for the pro-activity characteristic.

To determine a single value for each attribute, we will have to take the value of each measure weighted as follows [2]:

$$Y^k = \sum_i \omega_i^Y Y_i^k \ , \tag{4}$$

where Y^k is the name of attribute k, Y_i^k represents the value of the i^{th} measure of this attribute and ω_i^Y is the weight (between 0 and 1) of the measure i^{th} within attribute Y^k. The sum of all the applicable weighted measures must be 1 for every $Y^k \in [0, 1]$.

After outputting the values associated with each attribute, we proceed similarly to determine a single value associated with the characteristic (in this case pro-activity) using the following weighting:

$$\mathrm{C} = \sum_{\mathrm{k}} \omega_{\mathrm{k}}^{\mathrm{C}} \mathrm{Y}^{\mathrm{k}} \ , \tag{5}$$

where C is the characteristic, Y^k is the value of each attribute used to assess this characteristic, and ω_k^C is the weight of attribute k within characteristic C. Applying his or her experience in the application environments, the value of the different weights used in the calculations will have to defined by a specialist following a procedure designed for the purpose. Some of most commonly used procedures are:

- Direct estimation: assign weights directly, based on subjective opinions, and then they are standardized.
- The Method of the Relative Units: use binary comparisons of sub-groups of criteria to sharpen provisional estimates and improve internal consistency [25].
- The Method of Entropy: relate a criterion's relative importance directly to the average intrinsic information generated by the set of alternatives with respect to this criterion and by the subjective allocation of the importance that it is afforded by the decision maker [26].
- The Analytic Hierarchy Process: a multi-criteria decision-making technique where the attributes are organized hierarchically, using pair-wise comparison matrices in order to fully evaluate the characteristic in question [27].

Taking the results of each measure associated with the system agents and considering the importance of these agents within the system under study, which is defined by the system development engineer, the measure of full system

pro-activity is calculated by aggregating the values for each agent.

B. Case Study

As a case study we have used an intelligent agent marketplace. This marketplace includes several kinds of buyer and seller agents that cooperate and compete to process sales transactions for their owners. There is also a facilitator agent that acts as a manager of the marketplace [28].

There are three types of buyers and sellers: Basic, Intermediate and Advanced. They share the same negotiation capacities, but 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 agent pro-activity in this system should be greater if agents have better strategies for buying and selling.

Table I shows the measures associated with the attributes of the pro-activity characteristic and the values for each weight, associated with each attribute measure. We used the Analytic Hierarchy Process (AHP) [27] to evaluate the weights, where the attributes were organized according expert opinions on these measures. They considered that the number of goals (*NOG*) was more important to initiative, followed by messages to achieve the goals (*MAG*), and then the number of roles (*NOR*). Also, they considered that methods per class (*MC*), was more important to interaction than number of message types (*NMT*), whereas agent operations complexity (*AOC*) was much more important to reaction than number of processed requests (*NPR*).

Table II shows the values of the weights of the pro-activity attributes. We also used the AHP to evaluate the attribute weights. In this case, the experts considered that initiative is much more important than the interaction, and reaction is more important than interaction.

Using Equation (4), Table III shows the values of the measure for each attribute, calculated by aggregating the measures output from the values of Table I. The last row of Table III contains the value of the pro-activity characteristic, calculated by aggregating the measures for all the attributes using Equation (5).

Fig. 2 is a bar chart with the value of the system measures, calculated by aggregating the values of the attribute measures for all the agents using the values from Table II. The bottom bar in Fig. 2 shows the value of the system measures, calculated by aggregating the values of the attribute measures for all the agents.

TABLE I. PRO-ACTIVITY MEASURES WEIGHTS

Initiative		Interaction		Reaction	
NOR	0.25	MC	0.56	NPR	0.38
NOG	0.45	NMT	0.44	AOC	0.62
MAG	0.30				

TABLE II. PRO-ACTIVITY ATTRIBUTE WEIGHTS

Attributes	Weights
Initiative	0.66
Interaction	0.11
Reaction	0.23

TABLE III. PRO-ACTIVITY ATTRIBUTE VALUES

	Basic Buyer	Better Buyer	Best Buyer	Basic Seller	Better Seller	Best Seller
Initiative	0.82	0.88	0.81	0.82	0.80	0.89
Interaction	0.88	0.92	0.94	0.91	0.88	0.86
Reaction	0.94	0.99	0.99	1.00	1.00	1.00
Total Pro-Activity	0.85	0.91	0.87	0.87	0.85	0.91

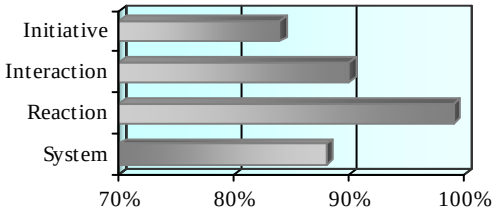

Figure 2. System pro-activity attribute values

From Table III we find that Basic Buyer and Best Buyer agents are less pro-active than the Better Buyer, because they have different buying strategies, and their ability to react is greater than their ability to interact with the agents in the system. The Best Seller agent has a greater pro-activity value (91%) than the Basic Seller (87%) and Better Seller (85%), because it has a better strategy for achieving its goals, although, as for the Buyer agents, the evaluated measures indicate that the Seller agents' ability to react is greater than their ability to interact with the agents in the system.

From Fig. 2, we conclude, with respect to the attributes, that the system scores highest on reaction, at 99%, followed by interaction, at 90%, and, finally, initiative, at 84%. Initiative is influenced by the fact that the agents do not achieve all their goals through cooperation due to the rules that they each apply to achieve their objectives.

Finally, the system's pro-activity value is 88%, that is, the pro-activity of the system agents as a whole is quite high. The weights fine tune the system results, which are more in line with the software engineer's idea of pro-activity

VI. CONCLUSIONS AND FUTURE WORK

We presented a first approximation to a set of measures of agent-oriented software considering the pro-activity characteristic. This characteristic has been decomposed into different measurable attributes, considering initiative, interaction and reaction, and we show the measures considered for its evaluation.

We developed a standard case study to evaluate the attributes associated with this characteristic on each of the participating agents. The values are aggregated using the weights associated with each attribute measure. The designed agents exhibit a high level of pro-activity (88%). The reaction attribute ranks top, whereas initiative scores the lowest values.

Pro-activity measures are different from the measures of the other characteristics. Further research is required to examine whether there is a possible correlation between the defined agent characteristics. For example, intuitively, agent

initiative could quite conceivably have an influence on autonomy, and therefore we need to look at whether any such relationship exists.

In the future we intend to conduct a comprehensive study of the agent-based system, analyzing the measures of each characteristic of each agent type present in the system and their contribution to the measure of system quality. To do this, we propose to build a quality evaluation model, and evaluate this model on several software agent applications, considering the different characteristics, and their attributes, present in agents. Additionally, as a future line of research, we intend to review the attributes (and their measures), with the aim of identifying other alternative and complementary attributes.

Finally, we intend to run a comparative study using other published works (like research published in [9] or [10]). The aim of this study is to establish whether or not there is any type of correlation among different sets of pro-activity measures or whether the measures proposed here are compatible with, or complementary or supplementary to, other authors' measures. In a preliminary review, we have found that Cernuzzi and Rossi [9] provide a framework for evaluating agent modeling methods rather than actual measures, whereas Shin [10] provides just one measure of pro-activity based on the frequency of knowledge discovery. It is commonly accepted that there is no figure of merit (a single measure is unlikely to cover all aspects of one quality characteristic [29]).

REFERENCES

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

[2] J. L. Fuertes, "Modelo de Calidad para el Software Orientado a Objetos," PhD thesis. Facultad de Informática, Universidad Politécnica de Madrid, Madrid (2003).

[3] ISO/IEC, "Software engineering - Product quality – Part 1: Quality model," International Standard ISO/IEC 9126-1:2001 (2001).

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

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

[6] F. Alonso, J. L. Fuertes, L. Martínez, and H. Soza, "Towards a Set of Measures for Evaluating Software Agent Autonomy," Proc. of the 7th Joint Meeting of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering, Amsterdam, Netherlands (2009).

[7] M. Wooldridge, "An Introduction to Multiagent Systems," John Wiley Ltd., Chichester (2002).

[8] Z. Lin and K. Carley, "Proactive or Reactive: An Analysis of the Effect of Agent Style on Organizational Decision-making Performance," Intelligent Systems in Accounting, Finance and Management, Vol. 2 (1993), pp. 271–287.

[9] L. Cernuzzi and G. Rossi, "On The Evaluation Of Agent Oriented Methodologies," Proc. of the Conference on Object-Oriented Programming, Systems, Languages & Applications - Workshop on Agent-Oriented Methodologies, Seattle, USA (2002), pp. 21–30.

[10] 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).

[11] R. So and L. Sonenberg, "Situation Awareness in Intelligent Agents: Foundations for a Theory of Proactive Agent Behavior," Proc. of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (2004), pp. 86–92.

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

[13] H. S. Nwana, "Software Agents: An Overview," Knowledge Engineering Review 11(3), (1996), pp. 1–40.

[14] S. Covey, "The Seven Habits of Highly Effective People," 15th anniversary edition. Free Press, Old Tappan, NJ (2004).

[15] D. Rousseau and B. Moulin, "Mixed initiative in interactions between software agents," Proc. of the 1997 Spring Symposium on Computer Models for Mixed Initiative Interaction. AAAI Press, Menlo Park (1997).

[16] B. Far and T. Wanyama, "Metrics For Agent-Based Software Development," Proc. of the EEE Canadian Conference on Electrical and Computer Engineering, Montréal, Canada (2003), pp. 1297–1300.

[17] B. Far, "A collective view and methodologies for software agents' interaction," Proc. of the Canadian Conference on Electrical and Computer Engineering, Volume 3, Issue 2-5 (2004), pp. 1249–1252.

[18] A. Orro, M. Saba, and E. Vargiu, "Using a Personalized, Adaptive and Cooperative Multi Agent System to Predict Protein Secondary Structure," Proc. of the First International Workshop on Multi-Agent Systems for Medicine, Computational Biology, and Bioinformatics, Utrecht, The Netherlands (2005), pp. 170–183.

[19] B. Qiao, K. Liu, and C. Guy, "A Multi-Agent System for Building Control," Proc. of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (2006), pp. 653–659.

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

[21] Foundation for Intelligent Physical Agents, "FIPA Communicative Act Library Specification," Geneva, Switzerland (2002).

[22] T. J. McCabe, "A Complexity Measure," IEEE Transactions on Software Engineering (1976), pp. 308–320.

[23] M. Dixon, "An objective measure of code quality," Technical report, Energy Group, Beverly, Massachusetts (2008).

[24] J. Ferrer, F. Chicano, and E. Alba, "On the Correlation between Static Measures and Code Coverage using Evolutionary Test Case Generation," Proc. of Decision-Making in Software Engineering, Vol. 3, No. 1 (2009).

[25] C. Churchman, R. Ackoff, and E. Arnoff, "Introduction to Operations Research," Wiley (1957).

[26] M. Zeleny, "Linear Multiobjective Programming," Springer Verlag, New York (1974).

[27] T. L. Saaty, "How to Make a Decision: The Analytic Hierarchy Process", European Journal of Operational Research, 48 (1990), pp. 9–26.

[28] J. Bigus and J. Bigus, Constructing Intelligent Agents using Java, 2nd edition. John Wiley & Sons, New York, NY (2001)

[29] V. R. Basili and D. H. Hutchens, "An Empirical Study of a Syntactic Complexity Family," IEEE Transactions on Software Engineering, 9(6), (1983), pp. 664-672.