International Journal of Engineering Science Invention Research & Development; Vol. I Issue II August 2014
www.ijesird.com e-ISSN: 2349-6185

A Study of Agent Oriented Metrics

[1]Ms. Minakshi Sharma, [2]Mrs Sapna
[1]*Assistant Professor, CRM, Jat college Hisar,*[2]*Assistant Professor, CRM, Jat college Hisar, INDIA*
[1]Minakshi_cselect@yahoo.com , [2]Sapna_mahar@yahoo.com

***Abstract*: For better quality of software, software engineering strongly recommends applying metrics in software development. Software metric are used to analyse and examine various software characteristics. The paper includes a set of well known and commonly applied traditional and objects oriented software metrics which could be applied to agent oriented programming and a set of agent oriented metrics. Agent oriented metrics will consists of some traditional, object oriented metrics and some pure agent oriented metrics.**

***Keywords*— Object-Oriented Metrics, Agent Oriented Metrics, Measurement.**

I. INTRODUCTION

Already there are a lots of metrics have been researched, developed and used in ensuring software quality and allow statistical analysis. Software metrics provides estimates of the resources needed during development. Metrics allows formal evaluation of software product design and establish software product standards. Software metrics are divided into Product, process and resources. In this paper we will focus on the product metrics. We will first discuss existing popular metrics used in traditional and object-oriented programming and see how they can be applied to the different aspects of software agent.[1] has listed some software quality factors: Correctness, reliability, efficiency, integrity, usability, maintainability, testability, flexibility, portability, reusability and interoperability. Basically, software metrics should be able to examine the efficiency of the design implementation, architectural complexity, code understandability, software testability and maintainability. Software metrics should be able to examine the efficiency of the design implementation, architectural complexity, code understandability, function usability, component reusability and interoperability, software correctness, portability and maintainability. There are well established set of software metrics and many of them emphasized on software object. Agent-Oriented Software Engineering is the one of the most recent contributions to the field of Software Engineering. Agent Oriented Software Engineering (AOSE) is a new programming paradigm that has evolved from Object Oriented Software Engineering (OOSE). There are similarities in concept between the two technologies such as data abstraction, encapsulation, cohesiveness and coupling. However, agents tend to focus more on the autonomy of one-self and the interaction among different agents. In terms of metrics, Object Oriented (OO) metrics focus on the object class, information hiding, inheritance, polymorphism and coupling. Agent metrics emphasize in measuring agent characteristics such as social ability, autonomy, reactivity, adaptability, intelligence, learning, proactively, goal-oriented and mobility. Therefore, a complete Agent Metrics Suite should include a subset of OO metrics and a set of pure Agent metrics[2].

The structure of this paper is as follows: Section 2 gives a brief description about measurement and metrics, why they are needed. Section 3 explains a brief introduction about the evolution of metrics from traditional to agent oriented metrics. Section 4 explains approaches of metric which includes some popular traditional metrics, some well known object-oriented metrics that can be applied to agents and a brief introduction about the agent metrics based on their characteristics and some product performance metrics especially applied for AOSE and finally conclusion is given in Section 5.

II. MEASUREMENT

First of all we must discuss what is measurement, one of the definitions of measurement is as: *"Formally we define measurement as a mapping from the empirical world to the formal, relational world. Consequently, a measure is a number or symbol assigned to an entity by this mapping in order to characterize an attribute.[3]"*

A. Developing a set of metrics

IEEE Standard 1061 lays out a methodology for developing metrics for software quality attributes. The standard defines an *attribute* as "a measurable physical or abstract property of an entity" A *quality factor* is a management oriented attribute of software that contributes to its quality". A metric is a measurement function, and a software quality metric is *"a function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality"*.
Before developing a sit of metrics for a project, we have to create a list of quality factors that are important for it:

- Associated with each quality factor is a direct metric that serves as a quantitative representation of a quality factor.
- For each quality factor, assign one or more direct metrics to represent the quality factor, and assign direct metric values to serve as quantitative requirements for that quality factor.
- Use only validated metrics to assess current and future product and process quality.

Standard 1061 lays out several interesting validation criteria that are: Correlation, Consistency, Tracking, Predictability, Discriminative power and Reliability. These validation criteria are expressed in terms of quantitative

relationship between the attribute being measured and the metric.

B. Why do we need Software Metrics

Question arises is why we should spend time and effort on software metrics, so here is some of the benefit of incorporating metrics in the software development.

- Specify and implement tools or aids for assessing software product quality.
- Resources allocation for product development.
- Formal evaluation of software product design.
- Formal statistical analysis of product.
- Define the standards for the software products developed in its organizational unit.

III. EVOLUTION OF METRICS:

Metrics are key components of many engineering discipline, software engineering is no exception. Software metrics is often used to measurements for computer software. Since 1950s Software Engineering is greatly researched on, software metrics were developed continuously so, become an important research area in the field of software engineering. Software engineering community and ISO9000-3 have recognized the need of metrics very well.

In very beginning there very not proper development cycle for designing software, but when requirement for software programs get more complicated, software engineering came into picture and Software Development Life Cycle was introduced. At the same time first set of software metrics was proposed and accepted. These software metrics was proposed for structural programming and known as traditional metrics. Some of the more appropriate traditional metrics that can be applied to software agents are listed in section 4.

Software Engineering evolves over time and Object Oriented Software Engineering (OOSE) grew popularity over the years. "Pure" Object Oriented metrics were introduced to measure the essences of OO software concepts and notions like encapsulation, inheritance and polymorphism. Some of the OO metrics suites are described in section 5 that can measure OO software projects.

Somehow it is true; agents are derived from the objects. Agent Oriented Software engineering (AOSE) is an evolution of a new software engineering paradigm though it borrowed a numerous features from OOSE. However, agents tend to focus more on the autonomy of oneself and the interaction among the different agents. Some features of the object-oriented paradigm are not too useful in agent programming. AOSE has introduced new features and concepts that differentiate it from OOSE. Apart from the normal encapsulation, agents encapsulate their behaviour within themselves. They would be able to make their own decision and their behaviours are non-absolute. Agents focus and depend greatly upon their environment and counterparts in performing their task and goals through collaboration and interaction. This introduced a more complex form of interaction messages to facilitate the transferring of information among the agents. Artificial Intelligence is also used to embed in some of the agents. This would include the knowledge discovery and learning ability. AOSE has introduced.

IV. APPROACHES OF SOFTWARE METRIC

We have listed here three approaches for software metrics, which are Traditional metrics, Object Oriented and Agent Oriented metric. These three approaches are not altogether different form each other but overlap with each other as shown in figure 1.

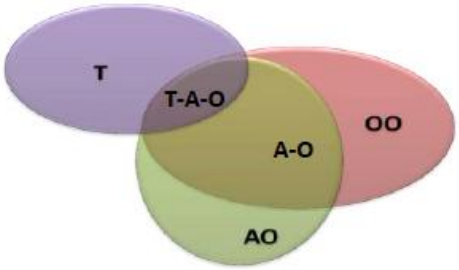

Fig. 1 Relationship between Traditional, Object-Oriented and Agent Oriented Metrics.

In Figure 1. T depicts a set of traditional metric, OO stands for a set for object oriented metrics , A is set of metrics for agent oriented metrics, A-O is the set of metrics common to both object and agent , TAO include the set of metrics common to Traditional object and agent.

A. Traditional Metrics

We will list some of the more appropriate traditional metrics that can be applied to software agents. In an object-oriented system, traditional metrics are generally applied to the methods of a class. We could also use the same metrics for the methods in the agents.

1) Complexity measurement: It is used to evaluate the application of an algorithm. It could be of method level, class level or system level. These are Conditions Count(COC), Variable count(VAC), Inner Method Call(IMC), Outer Method Call (OMC), Cohesion Ratio Metrics(CRM), Cyclomatic Complexity Number (CCN), Knot Measure(KNM), Line Of Code(LOC), NLE(Nesting Levels).
2) Size Measurement: The size of the agent a measure of understand ability, reusability and maintainability. It is a measurement of the number of codes in the agent. These are Line Of Code (LOC), Average Module Length (AML), Executable Statements (EST), Executable Size(ESI).
3) Comment Measurement: It is a measure of understand ability, reusability and maintainability. Includes Delivered Source Instructions(DSI),

Comment Line Per Method(CLM), Percentage of Commented Methods(PCM).

4) Attribute Measurement: It is a measure of testability, reusability and maintainability. It includes Variable Count(VC), Live variable(LV), Binding among Modules(BAM).
5) Method Measurement: It is a measure of agent functionality, maintainability and testability. It includes Method Count(MC), Number of Parameters per Method(NPM),
Average Method Size(AMS).

B. Object Oriented Metrics:

In recent years, OO technologies have emerged as a dominant software engineering practice and are often heralded as the silver bullet for solving software problems. As OO technologies has some new characteristics, such as data abstraction, encapsulation, inheritance, polymorphism, information hiding and reuse, traditional software metrics do not readily lend themselves to the OO notions. Therefore, new ways of measuring OO software are largely researched on. After years of research, many OO metrics are proposed, each targeting at a specific phrase of the OO development life cycle. In the following subsections, some of the more established metric suites are briefly discussed.

Chidamber and Kemerer, OO metrics suite[5]

With an aim to measure the key notions of OO software, Chidamber and Kemerer developed a set of six metrics to identify certain design traits in OO software, like inheritance, coupling and cohesion etc. The various metrics in the C&K OO metrics suite are hereby summarized.
1. Depth of Inheritance Tree (DIT). This metric measures the maximum level of the inheritance hierarchy of a class.
2. Number Of Children (NOC). This metric counts the number of immediate subclasses belonging to a class.
3. Lack of Cohesion in Methods (LCOM). This metric is intended to measure the lack of cohesion in the methods of a class.
4. Weighted Methods per Class (WMC). The sum of the complexities of the methods in a class.
5. Coupling between objects (CBO). The number of other classes whose methods or instance attribute(s) are used by methods of this class.
6. Response for a Class (RFC). The sum of the number of methods in the class and the number of methods called by each of these methods, where each called method is counted once.

Metrics for Object Oriented Design (MOOD) by Abreu[6]

MOOD is a metrics suite that targets specifically to obtain measurements for the design phrase. The emphasis behind the development of the metrics is on the features of OO design, namely inheritance, encapsulation and coupling. Each metrics in MOOD suite is expressed as a quotient where the numerator is the actual use of a particular mechanism (i.e. inheritance, information hiding, coupling and polymorphism) in the system being measured and the nominator is the maximum possible use of the same mechanism. The value of each metric would then range from 0 (total absence), to 1(maximum possible presence). The six metrics are summarized as follows:

1. Attribute Hiding Factor (AHF). This metric is the ratio of hidden (private and protected) attributes to total attributes and is proposed as a measurement of encapsulation and information hiding.
2. Method Hiding Factor (MHF). This metric is the ratio of the total inherited methods to total methods defined. Similar to AHF, it is proposed as a measurement of encapsulation and information hiding.
3. Coupling Factor (CF). CF is defined as the ratio of the maximum possible number of couplings in the system to the actual number of couplings not imputable to inheritance.
4. Polymorphism Factor (PF). PF is defined as the ratio of the actual number of possible different polymorphic situation for a class to the maximum number of possible distinct polymorphic situations for the class.
5. Attribute Inheritance Factor (AIF). AIF is defined as the ratio of the sum of inherited attributes in all classes of the system under consideration to the total number of available attributes for all classes.
6. Method Inheritance Factor (MIF). MIF is defined as the ratio of the sum of inherited methods in all classes of the system under consideration to the total number of available methods for all classes.

C. Agent Oriented Metric

1) Existing framework for agent software Metrics- Before proceeding further, we must have a look what an agent is. *"An agent is an encapsulated computer system that is situated in some environment, and that is capable of flexible, autonomous action in that environment in order to meet its design objectives".[7]*

Characteristics of agent: agents have their own will (*autonomy*), they are able to interact with each other (*social ability*), they respond to stimulus (*reactivity*), and they take initiative (*pro-activity*). in addition agents can move around (*mobility*), they are truthful (*veracity*), they do what they're told to do (*benevolence*), and they will perform in an optimal manner to achieve goals (*rationality*).

Not a single agent metric alone can determine the efficiency and quality of the agent since each agent type perform different roles in their own environment and require different characteristics and behaviours

There are a number of papers regarding software agent metrics. Most of the papers have described the overview of what to measure but have not illustrated the detail of how to measure them.

Metrics for agents intelligence

It has suggested that software agent intelligence can be divided into behaviour intelligence, knowledge discovery, mobility, interactions and cooperation. [1]

Metrics for agent mobility

For the performance of mobile agent, it has suggested four metrics: time of creating and launching messenger agents, message sending and setup time, message travelling time and message round-trip time. [8]

Metrics for agent complexity

It has suggested subjective and objective algorithmic complexity for Multi-Agent System. The subjective metric is based on a modified version of Function Points and the technical complexity. The objective metric used communicative cohesion metrics to decide the applicability of cyclomatic complexity in determining the system complexity.

Metrics for agent oriented modeling methods

It has identified agent attributes and grouped them into three different perspectives: agents' internal characteristics, interaction process attributes, and those more directly inherent to the design and development process. For the internal attributes, they have suggested the agent autonomy, reactivity, pro-activeness, beliefs, goals, intentions. For interaction attributes, they have suggested the agents' organization relationships, conversations, interface, interests, and interactions with environment and agent subsystems. As for the other process requirement, modularity, decomposition, dependence, abstraction, system view and communication support have been explored[**9**].

Agent product performance metrics

It defines a whole set of performance metrics related to the product, process and resources technologies and components of the software agents. The product performance metrics measured the agents and the system in terms of their design, description and working level[10, 11]. We will discuss these metrics in section 7 in detail.

2) Product Performance Metrics[12]

We may classify Product Performance metrics in two parts which are Agent and System. These two are further divided in various levels as shown in figure2

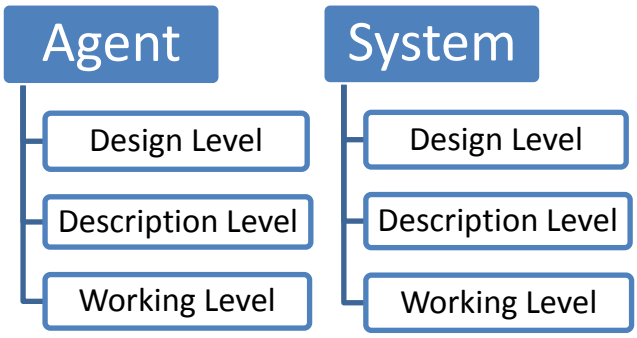

Fig. 2 Classification of Product Performance metrics

Agent design level:

(i) Software agent size (implicit performance): The size considers the functional size and the physical size of a software agent. A large agent size can cause a low performance and mobility. These are **E**xecutable **St**atements, **E**xecutable **Si**ze, **L**ines **O**f **C**ode, **A**verage **M**odule **L**ength.

(ii) Software agent component structure (structure performance): The structure depends on the kind of the agent (intelligent, reactive, deliberative etc.), the agent interface is related to the kind of agent coupling (as fixed, variable or evolutionary). The structure affects the coupling effects and changeability. These are **L**ack of **C**ohesion between **M**ethods, **C**ohesion **R**atio **M**etrics, Conditions **C**ount, **L**oop **C**ount, **M**ethod **H**iding **F**actor, **and Attribute H**iding **F**actor.

(iii) Software agent complexity (immanent performance): The complexity is divided in the computational and psychological complexity and should be measured on both concrete aspects. A high computational complexity leads to a weak performance. These are **C**yclomatic **C**omplexity **N**umber, Conditions **C**ount, **V**ariable **C**ount, **I**nner **M**ethod **C**all, **O**uter **M**ethod **C**all, **C**ohesion **R**atio **M**etrics, **Kn**ot **M**easure, **Lo**op **C**ount, Nesting **Le**vels, **L**ive **Va**riables, and **N**umber of **P**arameters per **M**ethod.

(iv)Software agent functionality (action performance): This aspect considers the appropriateness of the agent compared to the requirements. A high functionality can injure the performance and the chosen object-oriented implementation paradigm. These are **M**ethod **Co**unt, Number of **P**arameters per **M**ethod, **R**esponse **F**or a **C**lass.

System design level:

(i) Agent system size (potential performance): It includes the potential number of (active) agents and their contents; on the other hand, the size is related to the environment. A small agent system size can cause an overhead and reduce the application area.

(ii) Agent system component structure (architecture performance): This metric includes agent hierarchies vs. classless, the degree of parallelism, the kinds of organizational functions (representational, organizational, cognitive, interaction, productive, preservative). The system structure relates to the distributed performance and system changeability.

(iii) Agent system complexity (entropy performance): One of the measure aspects leads to the degree of the organizational dimensions (social, relational, physical, environmental and personal). It influences the system applicability.

(ii) Agent system functionality (model performance): It considers the realization of all of the functional system requirements. The distribution of the functionality in the system components affects their efficiency.

Agent description level:

(i) Software agent development description level (change performance): It considers the completeness of the development documentation (including comment, tests and change supports). The description level determines the maintainability of an agent. These are **C**omment **L**ines per **M**ethod, **P**ercentage of **C**ommented **M**ethods, **R**esponse **F**or a **C**lass.

(ii) Software agent application description level (usability performance): It includes the quality (readability, completeness, online support etc.) of the user documentation. This evaluation considers the usability of a software agent.

(iii) **Software agent publication description level (distribution performance):** This metric considers the public relations for using the software agent and involves the system description. A high publication level supports the spreading of the agent use.

System description level:

(i) Agent system development description level (maintenance performance): It considers the integration of the agent concepts and dynamics and their sufficient documentation. It affects overall system maintenance.

(ii) Agent system application description level (using performance): It considers the user documentation of all aspects of the system applications related to the different user categories. A good application description is a precondition for an efficient use of the whole system.

(iii) Agent system publication description level (marketing performance): Publication metrics evaluate the user acceptance and marketing aspects of the agent-based system application. A good system publication supports the spreading.

Agent working level:

(i) Software agent communication level (communication performance): It considers the size of communication and the level of the conversation required to sustain the activities. High communication intensity can affect a flexible application. These are **R**esponse **F**or a **C**lass, **C**oupling **B**etween **O**bjects, **N**umber of **P**arameters per **M**ethod.

(ii) Software agent interaction level (interaction performance) AI: It is related to the agent context and environment and their different kinds of actions (as transformation, reflecting, executing, modification, commands, perception, deliberation). It expresses the activity of an agent. These are **R**esponse **F**or a **C**lass, **C**oupling **B**etween **O**bjects.

(iii) Software agent learning level (learning performance "intelligence of the mind") AI: This metric evaluates the skills, intentions and actions of extending the agent facilities itself. It is based on the type of an agent and his roles in the system. (how much can be learn "discovery of data/knowledge", effect upon the agent, time and effort needed "bring latency to agent and may impair overall goals achievement", value of these new knowledge) These are **A**ttribute **H**iding **F**actor, **V**ariable **C**ount, **L**ive **Va**riables.

(iv) Software agent adaptation level (adaptation performance): The adaptation metric considers facilities of agent changing in order to react on new conditions in the environment. It determines the stability and complexity of the agent implementation. More complex algorithm is needed for the agent to adapt to different environment conditions. It could be measured by the time the agent survived and active in the system.

(v) Software agent negotiation level (negotiation performance): The measuring is directed on the evaluation of the facilities like the agent intentions, conflict resolution, and realized commitments for successful negotiations. It determines the success of an agent activity relating to common tasks. It can be measured only during runtime.

(vi)Software agent collaboration level (collaboration performance): It is oriented to the agent facility to work together with other agents. A high collaboration of an agent classifies his roles in the given tasks. These are **C**oupling **B**etween **O**bjects, **C**lass **C**ou**p**ling.

(vii) Software agent coordination level (coordination performance): It considers the agent facility of managing any agent tasks. A high level determines the role of the agent in an administration hierarchy. Some agent may have a lower coordination level than the other; it has to depend on the nature of the agent type.

(viii) Software agent cooperation level (cooperation performance) AI: It considers the volume and efficiency of an agent relating to a common task. It determines the effectiveness of common tasks realizations. We could count the average number of messages that are exchange during runtime before the task can be done.

(ix) Software agent self-reproduction level (reproduction performance): The number of destroyed agents related to repaired agents is counted. It determines the stability of a software agent itself. (Including error handling facilities) It could be measured only at run-time.

(x) Software agent performance level (operation performance): It considers the task related performance of an agent. A high agent performance is related to all kinds of agent activities. It could be measured only at run-time. It measures all agent aspect and threshold must be set before this summary metrics can be computed and consolidated from the various agent metrics.

(xi) Software agent mobility level (mobility performance): This aspect considers the efficiency relating to the agent movement. It considers the efficiency relating to the agent movement and includes The time of creating and launching messenger agents (mobile agents with minimal content), The time to create and post messages, The size of the message agents and messages, The time of agent taken for travelling along different nodes before returning to its host, with minimal content and interactions with the nodes and The synchronization time to exchange a message between two hosts.

(xii) Software agent specialization level (suitability performance): The metric consider the degree of specialization and the degree of redundancy of an agent. A high specialization can lead to high performance. The metric is Weighted **M**ethods per **C**lass.

(xiii) Software agent competition level (competition performance) AI: Opposite of cooperation level. It considers the determination and rights of the agent to say no to request from other agent that may detrimental to its goals or tasks. We count the number of request rejected by the agent during runtime.

System working level:

(i) Agent system communication level (advising performance): It counts the number of ACLs between the different kinds of software agents and their different roles and actions. It characterizes the intensity of the conversations and describes the agent collaboration.

(ii) Agent system interaction level (team performance): It considers the average types of interactions relating to the agents and their roles in the environment of the agent based system. Many interactions are based on a high cooperation

(iii) Agent system knowledge level (knowledge performance): It measures the results of agent learning for agent-based system based on the different kinds of agents (as tropistic and hysteretic agents). This aspect determines the knowledge-based foundation of the agent-based system.

(iv) Agent system living level (life performance): This metric is based on the agent adaptation which keeps the adaptation level of the whole agent-based system. It based on the adaptability of the agents and characterizes the system maintenance effort.

(v) Agent system conflict management level (conflict solution performance): The system success is based on the agent negotiation and considering the relations between the different kinds of a fair play in the realization of the system tasks. A high conflict management level leads to high system stability.

(vi) Agent system community level (community performance): It considers the level of different agent communities based on the agent collaboration. A high community level is caused on collaboration for different classes of system application.

(vi) Agent system management level (management performance): This system metric is based on the agent coordination level related to the whole agent system structure. An efficient management determines a good agent organization level.

(vii) Agent system application level (application performance): This metric is related to the application area and the different agent roles in their cooperation. It is based on effective task-oriented agent cooperation.

(viii) Agent system stability level (stability performance): The stability measure is based on the agent self-reproduction. A high stability level includes the agent self-reproduction and other system error handling facilities.

(ix) Agent system performance level (processing performance): The handling with object to realize special tasks through the different agents is considered. This level includes the agent performance and the performance of the environment.

(x) Agent system flexibility level (flexibility performance): The mobility behavior of all agents is considered here.

(xi) Agent system organization level (organization performance): The different agent's roles are considered. This level leads to an efficient distribution of the agent roles and their administration.

V. CONCLUSION AND FUTURE WORK

We have presented an overview about software metrics, i.e. traditional, object oriented and agent oriented metrics suggested by various papers. As per requirement of time different methodologies are evolved and for each methodology new set of metrics are suggested and accepted accordingly. A comprehensive and complete agent metrics measuring suite is still yet to be established. We have applied some of the existing above mentioned Traditional and Object Oriented metrics in agents because there are much similarity between agents and the other programming paradigm software, especially object-oriented software. There are some special characteristics of AOSE which are not by the available metrics; we have to derive new metrics to measure these agents' behaviours. It is unquestionable that more trial and time have to be spend in determine the usefulness and efficiency of these new metrics.

REFERENCES

[1] *L. Pouchard, "Metrics for Intelligence: the Perspective from Software Agents" (2001)in NIST SPECIAL PUBLICATIONS Performance Metrics for Intelligent Systems Workshop, P; 123-126 .*

[2] *Bimlesh Wadhwa, Koh shin Jang and Teo Ee Nam, "Object and Agent Metric Approach"(2003).*

[3] *Cem Kaner, Senior Member, IEEE, and Walter P. Bond," Software Engineering Metrics: What Do They Measure and How Do We Know?",in 10th International Conference on Software Metrics, p.1-12, 2004.*

[4] *Dr. Linda H. Rosenberg, Lawrence E. Hyatt, "Software Quality Metrics for Object-Oriented Environments", (1994).*

[5] *R. Chidamber and Chris F. Kemerer. "A Metrics Suite for Object Oriented Design", (1994).in Software Engineering,IEEE Transaction on Vol.20 ,Issue 6,paper 476-493, june1994.*

[6] *F. Brito e Abreu "The MOOD Metrics Set" Proc. ECOOP'95 Workshop on Metrics, (1995).*

[7] *Nicholas R. Jennings, "Agent-Oriented Software Engineering" , (2000)in IEA/AIE '99 Proceeding of 12th international conference on industrial and engineering applications of artificial intelligence and expert systems:multiple approaches to intelligent systems ,Springer-Verlag New York,Inc.Secaucus,NJ,USA1999 ISBN:3-54066076-3.*

[8] *M. D. Dikaiakos, and G. Samaras, "Performance Evaluation of Mobile Agents: Issues and Approaches", In Dumke, Rautenstrauch, Schmietendorf and Scholz (eds.), Performance Engineering. State ofthe Art and Current Trends, Lecture Notes in Computer Science Series, State ofthe Art Survey, Vol. 2047, Springer, May 2001, p. 148-166.*

[9] *L. Cernuzzi & G. Rossi, "On the Evaluation of Agent Oriented Modeling" (2002) in Proceedings of Agent Oriented Methodology Workshop, Seattle paper 20/02/11 Vol 29, p106.*

[10] *C. Wille, R. Dumke and S. Stojanov, "Performance Engineering in Agent-based Systems Concepts, Modeling and Examples" in IWSM'01 August 28-29,(2001),paper ,p 109.*

[11] *R. Dumke, R.Koeppe and C. Wille,"Software Agent Measurement and Self-Measuring Agent Based System", Fakultat fur Informatik,Otto-von-Guericke-Universitat,Magdeburg,2000,paper Preprint No.*

[12] *Cornelius Wille, Nick Brehmer, Reiner R. dumke, "Software measurement of Agent Oriented System", (2001)*