Performance Evaluation of Agents and Multi-agent Systems using Formal Specifications in Z Notation

Christos Dimou[1], Fani Tzima[1], Andreas L. Symeonidis[1,2], and Pericles A. Mitkas[1,2]

[1] Dept. of Electrical & Computer Engineering, Aristotle University of Thessaloniki
[2] Information Technologies Institute, CERTH, Thessaloniki, Greece
`[cdimou, fani]@issel.ee.auth.gr, asymeon@eng.auth.gr, mitkas@auth.gr`

Abstract. Despite the plethora of frameworks and tools for developing agent systems, there is a remarkable lack of generalised methodologies for assessing their performance. Existing methods in the field of software engineering do not adequately address the unpredictable and complex nature of intelligent agents. We introduce a generic methodology for evaluating agent performance; the Agent Performance Evaluation (APE) methodology consists of representation tools, guidelines and techniques for organizing, categorizing and using metrics, measurements and aggregated characterizations of agent performance. The core of APE is the Metrics Representation Tree, a generic structure that enables efficient manipulation of evaluation-specific information. This paper provides a formal specification of the proposed methodology in Z notation and demonstrates how to apply it on an existing multi-agent system.

1 Introduction

Although Agent technology has produced many interesting findings in the research sphere, not many real-world applications actually use agents in practice. We argue that the reason for this is that, despite the plethora of tools, theories and achievements in the field, there exists no general, standardized evaluation methodology that enables developers to validate and quantify benefits and drawbacks that agents and MAS may exhibit.

We further stress that the engineering aspects of the agent software lifecycle must not be underestimated. Having achieved a more concise understanding of the scope and applicability of agents, rigorous engineering methods have to be used to decide under which circumstances the use of agents is beneficiary, trading-off between appealing characteristics, burdens of development and actual execution performance.

The very complexity and unpredictability of agent systems require an evaluation methodology specifically tailored to address these aspects. Until now, no such methodology has been proposed, mainly due to three basic reasons [11]: a) lack of consensus in definitions, b) interdisciplinary nature of agent computing, and c) lack of maturity of the field.

Within the context of the paper, we present the *Agent Performance Evaluation* (APE) methodology, a complete methodology for evaluating the performance of agents and MAS that aims to:

1. provide structured representation tools for organizing and using evaluation-specific knowledge
2. standardize various steps of the evaluation process, including metrics, measurements and their aggregation
3. incorporate higher level (qualitative) concepts into quantitative evaluation processes
4. reuse evaluation knowledge, so that accumulated experience is readily available to the agent community, and
5. ensure that the definition of the methodology is abstract enough to embrace a wide variety of systems and application domains.

The remainder of this paper is structured as follows: Section 2 discussed related work in the field of Agents and Intelligent Systems (IS) evaluation, while Section 3 provides a formal specification of the APE methodology, using the Z notation [14]. Section 4 demonstrates the applicability of APE in a pilot application domain. Finally, Section 5 concludes the paper with the summary of our contributions and future research directions.

2 Related Work

Currently, efforts for generalized metrics and evaluation methodologies exist only in specific domains, such as robotics and autonomic computing. In robotics, evaluation efforts span from robots and autonomous vehicle navigation (e.g. [12]) to hybrid human-robot control systems (e.g. [13][5]), while in autonomic computing, emphasis is given on the quality assessment of the selected self-managing techniques [6]. Though both fields provide useful metrics and thorough methodological steps for evaluation, they do not provide relevant tools that support process. And what is even more important, they do not deal with the case of knowledge infusion to the autonomous entities they benchmark, not giving room for a generalized evaluation scheme.

There exist two general approaches to the IS evaluation problem: a) the bottom-up and b) the top-down. The former advocates the definition of formal constructs and languages that enable the definition of the appropriate terms and scope of IS. Evaluation thereafter is gradually built upon these formal foundations. To that end, Zadeh [16] proposes the definition of hierarchical descriptive languages for the purpose of evaluating IS; specifically, Zadeh proposes the *Precisiated Natural Language*, which complements the less expressive hierarchy of fuzzy logic, crisp mathematics and natural language.

On the other hand, the top-down approach (e.g. [9] and [1]) argues that existing or newly implemented systems urge for evaluation methodologies and it is therefore preferable to instantly evaluate them at any cost. According to this approach, experiences from different ad-hoc evaluation attempts are generalized

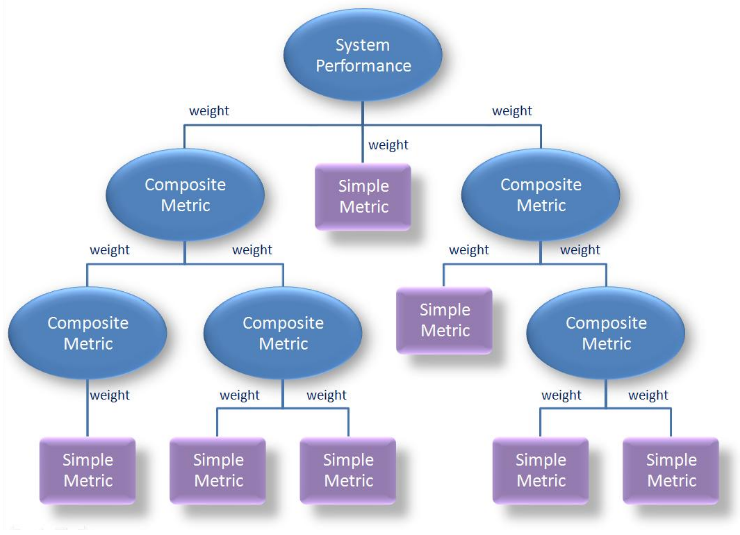

Fig. 1. The Metrics Representation Tree

into a concise domain-independent methodology, that is established at the time that ISs reach a sufficient level of maturity. This approach implies the reduction of the problem scope to purely engineering terms and the definition of terms, such as intelligent behavior, within the context of the engineering concepts of performance, goals achieved etc. In this paper, we adopt the top-down approach, driven by the urging need to evaluate existing or emergent applications.

3 The APE Methodology

3.1 Overview

APE is part of a more general evaluation framework that also addresses the use of automated tools for conducting online collection of measurements during the runtime of a MAS. APE is complementary to the MEANDER framework[3], which provides a set of automated and interactive tools that assist the evaluator in organizing, collecting and processing metrics and experimental measurements. The Evaluation Ontology, also described in [3], complements APE and MEANDER by providing the definition of appropriate evaluation concepts and their relationships. Both APE and MEANDER provide tools for each of the three aspects of evaluation, namely: a) Metrics, b) Measurements, and c) Aggregation. The way that the presented methodology deals with the three evaluation aspects is detailed in the following sections.

Metrics The Metrics Representation Tree (MRT) is a scheme that organizes metrics related to the different aspects of performance evaluation in different levels of abstraction. As depicted in Figure 1, the bottom-most level of the MRT comprises *Simple Metrics* (SM) that are directly measurable and therefore can be directly assigned a specific measured value. SMs are weightedly combined to form *Composite Metrics* (CM), which are not directly measurable. CMs represent higher-level performance concepts, often discussed *qualitatively* in current evaluation efforts. Moving upwards through the tree, CMs as well as SMs are further combined to form more CMs, until the root of the tree which corresponds to the overall *System Performance.* It is worth noting, at this point, that the MRT is not to be used "as is", but allows the evaluator to focus on specific (or more general) aspects of system performance by selecting the appropriate segments of the tree. The MRT allows the evaluator to select parts of the tree and focus on specific (or more general) aspects of the system performance.

Measurement As stated in literature, after having selected the measurement method (for example with respect to the nature of the experimental technique [7]), one must thoroughly provide an experimental design prototype and a data collection procedure. Such an experimental design prototype must describe thoroughly the objectives of the experiments and ensure that these objectives can be met using the specified techniques. The APE methodology does not provide a specific design prototype; it can only provide a set of guidelines that designers may apply to their application at hand.

Having specified the design prototype, data collection must be performed. APE again provides the basic guidelines for the designer to follow, in order to ensure that the data collection process is well defined, while also monitors data collection and looks out for deviations from the use case design. Since measurement techniques heavily depend on the application at hand, APE does not provide details on *how* measurement is conducted. Rather, it emphasizes on the automation of measurement collection, an issue that is dealt within the MEANDER framework, a specific implementation of automated evaluation that follows the APE principles.

Aggregation Following the collection of measurement values and the construction of metric-measurement tuples, aggregation must be performed, in order to summarize the experimental findings into a single characterization of performance, either of single modules, or the system as a whole. Within the context of the proposed methodology, the user must traverse the MRT in a bottom-up manner and, starting from the specific metrics view, he/she must proceed upwards and apply -at each view- aggregation to provide single characterizations for every parent node. The specific techniques one may apply for aggregating the MRT, range from simple average operators to complex multicriteria decision-making techniques. A detailed discussion on aggregation techniques for evaluation purposes can be found in [2], where *Fuzzy Aggregation* is employed for combining diverse and heterogeneous measurement information.

3.2 Specification of APE

Agents and MAS specification APE has been developed upon a formal, abstract specification of agent environments (*System*). In this paper, we employ several basic definitions of agent-related concepts found in [4][10], wherein the authors provide a concise, abstract definition of agents and MAS and proceed on defining more complex types of agent systems.

As discussed in [4], the specification is based on an axiomatic definition of *Attributes*, *Motivations*, *Actions* and *Goals*, gradually proceeding to more complex terms, such as an *Agent* and a *MAS*. Also, it is organized around basic entity definitions (Entity, Object, Agent), their states (e.g. AgentState) and a series of possible operations (e.g. AgentAction). The reader is encouraged to refer to the original publications for more detailed descriptions of relevant terms (not included due to space limitations).

Practically, in the context of a MAS, a *System* consists of one or more *MAS*, a set of *Entity* schemas (further refined as either agents or static objects) and a set of attributes that describe various properties of the *System*. It is induced that all entity properties are members of the *sysAttrs* set: [*Attribute*, *Action*, *Goal*]

$$
\begin{array}{|l}
\hline
\textit{MAS} \\
\textit{entities} : \mathbb{P}\,\textit{Entity} \\
\textit{objects} : \mathbb{P}\,\textit{Object} \\
\textit{agents} : \mathbb{P}\,\textit{Agent} \\
\textit{autonomousagents} : \mathbb{P}\,\textit{AutonomousAgent} \\
\textit{staticobjects} : \mathbb{P}\,\textit{StaticObject} \\
\textit{serveragents} : \mathbb{P}\,\textit{ServerAgent} \\
\hline
\textit{autonomousagents} \subseteq \textit{agents} \subseteq \textit{objects} \subseteq \textit{entities} \\
\textit{agents} = \textit{autonomousagents} \cup \textit{serveragents} \\
\textit{objects} = \textit{agents} \cup \textit{staticobjects} \\
\#\,\textit{agents} \geq 2 \\
\#\,\textit{autonomousagents} \geq 1 \\
\exists\, aa1, aa2 : \textit{agents} \bullet aa1.\textit{goals} \cap aa2.\textit{goals} \neq \{\} \\
\hline
\end{array}
$$

An *Entity* is an abstraction that describes any possible participant in the *System* that may have a set of possible *actions*, *goals* and other *attributes*. An *Agent* is a special case of an *Entity* that is required to have non-zero goals. A *View* is a non-empty set of *Attributes* that describe a subset of the *System*'s attributes. A View can be perceived by an *Agent* at a given time of runtime execution. *AgentPerception* describes the ability of a situated agent to understand changes in its surrounding environment. The AgentPerception schema includes the Agent schema and definitions of a set of Actions (*perceivingactions*) that allow an agent to receive stimuli from the environment, as well as two functions that describe the observable (*canperceive*) and the selected (*willperceive*) stimuli. The constraints part of the schema describe in detail the properties of the above definitions, including their domain and set relationship properties.

Having defined all the above, a MAS is a conglomeration of several types of entities that are hierarchically ordered as entities, objects, agents and autonomous agents. Several different types of agents may participate in a MAS (namely *autonomous agents* and *serveragents*, both specified in [10]). Nevertheless, in our definition it is required that a MAS consists of at least two agents, one of which has to be an autonomous agent. It is also required that at least two agents in the MAS share a minimum of one goal.

Finally, we also define *AgentGroup*, a schema useful when evaluating the performance aspects of a group of cooperating agents. In the definition below, we require that all participating agents share at least one common group goal.

AgentGroup

$$\begin{array}{l}
agents : \mathbb{P}\, Agent \\
groupGoal : \mathbb{P}\, Goal \\
\hline
\#\, agents \geq 2 \\
\forall\, a : Agent \bullet (a \in agents) \Rightarrow (\exists\, g : \mathbb{P}\, Goal \\
\qquad\qquad \bullet\, g \subseteq a.goals \\
\qquad\qquad \wedge\, g \subseteq groupGoal) \\
\forall\, a_1, a_2 : Agent \bullet (a_1 \in agents \wedge a_2 \in agents) \\
\qquad\qquad \Rightarrow (\exists\, g : Goal \bullet (g \in a_1.goals \cap a_2.goals \\
\qquad\qquad \wedge\, g \in groupGoal))
\end{array}$$

Evaluation The core of APE is the *metric* concept, defined as a "standard that specifies a measurable attribute of entities, their units and their scopes" [8]. Metrics are the essential building blocks of any evaluation process, as they allow for the establishment of specific goals for improvement: a specific metric provides an indication of the degree to which the corresponding system attribute has met its defined goal.

The term *Metric* refers to a general performance aspect that is either known to an entiry or is measurable by central observers or non-agent entities (such as logfiles and databases). A *Metric* in its core comprises of a measured *Value* and an *Attribute* (a corresponding performance aspect, often tied to a certain variable in the application code). The *children*, *weights* and *weightedChildren* members of the definition are instantiated only for the case of a *CompositeMetric*, later defined in the specification.

Metric

$$\begin{array}{l}
v : Value \\
attr : Attribute \\
children : \mathbb{P}\, Metric \\
weights : \mathbb{P}\, Weight \\
weightedChildren : \mathbb{P}\, Metric \nrightarrow \mathbb{P}\, Weight
\end{array}$$

A *Value* consists of an actual measured *value*, the *type* (either *simple* or *composite*) and the corresponding *scale* of measurement. The extensible nature of Z notation allows evaluators to define more specific properties of a measured value, based on this generic definition, as, for example, requiring that *scale* be a member of the set $T = \{ORDINAL, NOMINAL\}$. A sample extension of the *Value* schema is presented in Section 4.

Value

$value : Attribute$
$type : Type$
$scale : Attribute$

$Type ::= simple \mid composite$

A *SimpleMetric* in the context of APE is a *Metric* that is of *simple* type, i.e. it has no children and weights, as explained in the previous sections.

SimpleMetric

$Metric$

$v.type = simple$
$children = \varnothing$
$weights = \varnothing$
$weightedChildren = \varnothing$

A *CompositeMetric* is a higher-level metric that is not directly measurable; instead it is constructed from a number of *children* metrics. The contribution of each child to the *CompositeMetric* is signified by a *Weight*, whose schema is defined below; a one-to-one function, (*weightedChildren*), maps children to weights.

CompositeMetric

$Metric$

$v.type = composite$
$\mathrm{dom}\ weightedChildren = \mathbb{P}\ children$
$\mathrm{ran}\ weightedChildren = \mathbb{P}\ weights$
$\#\ children > 0$
$\#\ weights = \#\ children$
$weights = weightedChildren\ children$

A *Weight* in the above schema is simply a real value that signifies the contribution of each child metric value to its parent.

Note that the cardinality of *children* is equal to the cardinality of *weights* and there is a one-to-one relationship between these sets, as described by the above domain and range constraints.

Finally, the *MRT* schema contains *metrics* that comprise both a set of *SimpleMetric* and *ComplexMetric* instances. The *status* member takes values in the set $S = \{EMPTY, PARTIALLY\_BOUND, BOUND,$ $AGGREGATED\}$, signifying an MRT that has zero, some, all Simple Metrics values and all Composite Metrics values, respectively.

The measurability requirement for a *SimpleMetric* is satisfied by ensuring that the *value* property is either in *enAttrs* or *sysAttrs*.

We also require that *value* is not a member of neither *sysAttrs* nor *enAttrs*, thus making a *CompositeMetric* not directly measurable. Instead, CM's *value* is calculated with respect to children values, according to the aggregation function (defined later, in the *Aggregate* schema) chosen for the particular case at hand.

The last four constraints ensure the tree structure of the MRT. We require that: a) all *SimpleMetrics* are children of at least one *CompositeMetrics* and b) there exists a single root node that has no parents and all other metrics are its direct or indirect offsprings.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad MRT \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}} \\
system : System \\
metrics : \mathbb{P}\, Metric \\
sMetrics : \mathbb{P}\, SimpleMetric \\
cMetrics : \mathbb{P}\, CompositeMetric \\
status : Status \\
root : CompositeMetric \\
\hline
cMetrics \cup sMetrics = metrics \\
\forall s : SimpleMetric \bullet s \in sMetrics \Rightarrow \\
\qquad (\exists a : Agent \bullet a \in system.entities \land \\
\qquad s.attr \in a.enAttrs) \lor \\
\qquad s.attr \in system.sysAttrs \\
\forall c : CompositeMetric \bullet c \in cMetrics \Rightarrow \\
\qquad c.attr \notin system.sysAttrs \\
\forall c : CompositeMetric \bullet c \in cMetrics \Rightarrow \\
\qquad (\forall e : Entity \bullet e \in system.entities \Rightarrow \\
\qquad c.attr \notin e.enAttrs) \\
\forall sm : SimpleMetric \bullet sm \in sMetrics \Rightarrow \\
\qquad (\exists cm : CompositeMetric \bullet cm \in cMetrics \\
\qquad \land\ sm \in cm.children) \\
root \in cMetrics \\
\forall cp : CompositeMetric \bullet cp \in cMetrics \Rightarrow root \\
\qquad \notin cp.children \\
\forall m : Metric \bullet m \in metrics \setminus \{root\} \Rightarrow \\
\qquad (\exists p_1 : CompositeMetric \bullet p_1 \in cMetrics \\
\qquad \land\ m \in p_1.children) \land \\
\qquad (\exists m : Metric \bullet m \in metrics \setminus \{root\} \\
\qquad \land\ m \in root.children) \\
\hline
\end{array}
$$

$Status ::= EMPTY \mid PARTIALLY\_BOUND \mid BOUND \mid AGGREGATED$

Evaluation Operations Several operations are required for the application of the APE methodology to a runtime MAS, in order to collect the measured values for metrics within the context of an experiment, integrate these values in the MRT and, finally, aggregate the MRT.

To do so, we define the *Experiment* schema from the APE perspective. Any experiment involves the MRT of the application domain at hand, a set of measurement-collecting entities and a set of experimental parameters, *params*, such as the experiment initiation and termination conditions, the number of iterations, the number of participating entities, etc.

Experiment

$$
\begin{array}{l}
mrt : MRT \\
collectorStaticObjects : \mathbb{P}\, StaticObject \\
collectorAgents : \mathbb{P}\, Agent \\
collectorAgentGroups : \mathbb{P}\, AgentGroup \\
collectorMAS : \mathbb{P}\, MAS \\
params : \mathbb{P}\, Attribute \\
\hline
collectorStaticObjects \subseteq mrt.system.entities \\
collectorAgents \subseteq mrt.system.entities \\
\forall\, ag : AgentGroup \bullet ag \in collectorAgentGroups \Rightarrow \\
\qquad\qquad (\forall\, a : Agent \bullet a \in ag.agents \Rightarrow \\
\qquad\qquad a \in mrt.system.entities) \\
collectorMAS \subseteq mrt.system.mas \\
\forall\, p : Attribute \bullet p \in params \Rightarrow p \in mrt.system.sysAttrs
\end{array}
$$

A valid association among the system, the APE methodology and the experiment, is guaranteed by the fact that all *SimpleMetrics* in the MRT refer to actual system attributes and all measuring entities participate in the system runtime, while all experimental parameters are valid within the context of the system.

BindValue fills-in the measured value of a certain *SimpleMetric*. The schema takes as input a measured *value* and modifies a certain *SimpleMetric* ($\Delta SimpleMetric$). The constraints ensure that input and *SimpleMetric* values share the same type, in order for the correspondence of metric and measurement to be correct.

BindValue

$$
\begin{array}{l}
\Delta SimpleMetric \\
v? : Value \\
\hline
v?.type = v.type \\
v' = v?
\end{array}
$$

Though the above schema binds a value to a single metric, we still need to define the impact of this change to the entire MRT. To this end, we employ the *BindValues* operation, which binds every measured value collected by the measurment collecting entities to their corresponding *SimpleMetric* in the MRT.

Another critical operation on Composite Metrics is *Aggregation*, which is "the process of summarizing multiple measurements into a single measurement" [8]. During the aggregation phase, collected measurements are grouped and combined, by the use of weights of importance, in order to provide an atomic characterization for the evaluated system.

Aggregate

$$\begin{array}{|l}
\Delta \mathit{CompositeMetric} \\
\mathit{calculateCompositeMetric} : \\
\qquad \mathbb{P}(\mathbb{P}\,\mathit{Metric} \times \mathbb{P}\,\mathit{Weight}) \rightarrow \mathit{Value} \\
c? : \mathit{CompositeMetric} \\
\hline
v' = \mathit{calculateCompositeMetric}(c?.\mathit{weightedChildren})
\end{array}$$

This process is implemented, within our specification, by the aggregation operation that calculates the value of a *CompositeMetric* based on an appropriate function on the values of its children. In a similar manner to the *BindValue* schema that promotes changes to the entire MRT, we employ the *AggregateMRT* operation in order to aggregate all CompositeMetrics therein / in the MRT.

Summarizing all the above, the experimental phase of evaluation consists of the following successive steps:

1. the parameters of the experiment and the corresponding MRT are defined
2. the system enters the runtime phase
3. the measuring entities measure the values of the monitored Simple Metrics
4. the collected values are bound to the MRT
5. the MRT is aggregated

which is formally defined as:

$$\mathit{Experiment} \mathbin{\#} \mathit{InitSystem} \mathbin{\#} \mathit{BindValues} \gg \mathit{AggregateMRT}$$

where the # operator denotes a succession of schemata or operations, while the $\gg$ operator denotes a piped execution of operations, in which the output of the lefta-hand schema is the input of the right-hand schema. The *InitSystem* operation that initializes the members of the *System* schema, is defined as:

InitSystem

$\Delta System$
$entities? : \mathbb{P}\, Entity$
$mas? : \mathbb{P}\, MAS$
$attrs? : \mathbb{P}\, Attribute$

$entities' = entities?$
$sysAttrs' = attrs?$

4 Applying the APE methodology

In this section the applicability of APE is demonstrated through *Symbiosis*, a multi-agent simulation environment aiming to study various aspects of learning behavior in agent societies. Details on the *Symbiosis* architecture can be fount at [15] (omitted due to space limitations).

We initiate the evaluation process by defining the appropriate metrics in the MRT structure. This process, currently undertaken by a domain expert, requires the identification of Simple Metrics, the creation of Composite Metrics and the definition of the weighted associations between them.

Name	Symbol	Name	Symbol
Agent Metrics		Environmental Metrics	
energy	-	epoch	N
age	-	resource availability	a
resource consumption rate	rcr	environmental variety	v
trap collision rate	tcr	environmental reliability	r
unknown situation rate	usr	current population	cp
reproduction rate	rr		
knowledge base completeness	kbc		
rule generality	rg		
effectiveness	e		
net effectiveness	e_{NET}		

Table 1. Symbiosis Simple Metrics

Table 1 lists the Simple Metrics as introduced and applied in experiments in work by Tzima et al [15]. Simple Metrics are organized in *Agent metrics* and *Environmental metrics*, depicting that the former are attributes of single agents, measured by Agent measuring entities, while the latter are attributes of the system, measured by the MAS. Note that Agent metrics are calculated as aggregates of all related attributes of a specific agent group, either predators or preys, while Environmental metrics are calculated as aggregates of all related attributes of the Symbiosis grid. For example, prey resource consumption rate (*prey_rcr*) is calculated as the average *rcr* of all prey agents, while *rcr* is itself an average of the series of values (one for each timestep) measured for a specific agent through the execution of the System.

As already discussed, Composite Metrics describe higher-level, qualitative attributes of performance. The domain expert has identified the following composite attributes for Symbiosis:

Freedom of action (FoA). FoA represents the behavioral variety of animats with respect to a changing environment. An animat with a large degree of *FoA* would essentially have a large and reliable knowledge base of rules that indicate the optimal action in a given environment. Ideally, an efficient knowledge base would have a large number of generalized rules. For example, for the case of Preys, we define:

$$\begin{aligned}&\mathit{FoA} : \mathit{CompositeMetric} \mid\\ &\qquad \theta\ \mathit{CompositeMetric} = \lhd \mathit{children} \mapsto \{\mathit{kbc}, \mathit{rg}, v\} \rhd\\ \mathit{where}&\\ &\mathit{kbc};\ \mathit{rg};\ v : \mathit{SimpleMetric}\end{aligned}$$

Adaptability. The ability of animats to learn and adapt in a changing environment. *Adaptability* is proportional to the animat's *effectiveness* that encompasses the efficient use of its energy and environmental resources. Moreover, *Adaptability* is inversely proportional to the unknown situations an animat encounters, another indication of learning and adapting to hostile environments. We therefore define:

$$\begin{aligned}&\mathit{PreyAdaptability} : \mathit{CompositeMetric} \mid\\ &\qquad \theta\ \mathit{CompositeMetric} = \lhd \mathit{children} \mapsto\\ &\qquad\qquad \{\mathit{prey\_e}_{NET}, \mathit{prey\_usr}\} \rhd\\ &\mathit{and}\\ &\mathit{PredatorAdaptability} : \mathit{CompositeMetric} \mid\\ &\qquad \theta\ \mathit{CompositeMetric} = \lhd \mathit{children} \mapsto\\ &\qquad\qquad \{\mathit{predator\_e}, \mathit{predator\_usr}\} \rhd\end{aligned}$$

where
$\mathit{prey\_e}_{NET};\ \mathit{prey\_usr};\ \mathit{predator\_e};\ \mathit{predator\_usr} :$
$\mathit{SimpleMetric}$

In a similar manner, *Security* and *SpeciesDynamics* are defined.

The organization of all the above Simple and Composite Metrics into a MRT is depicted in Figure 2, while a sample instantiation of *PreyAdaptability* would contain relations of the following form:

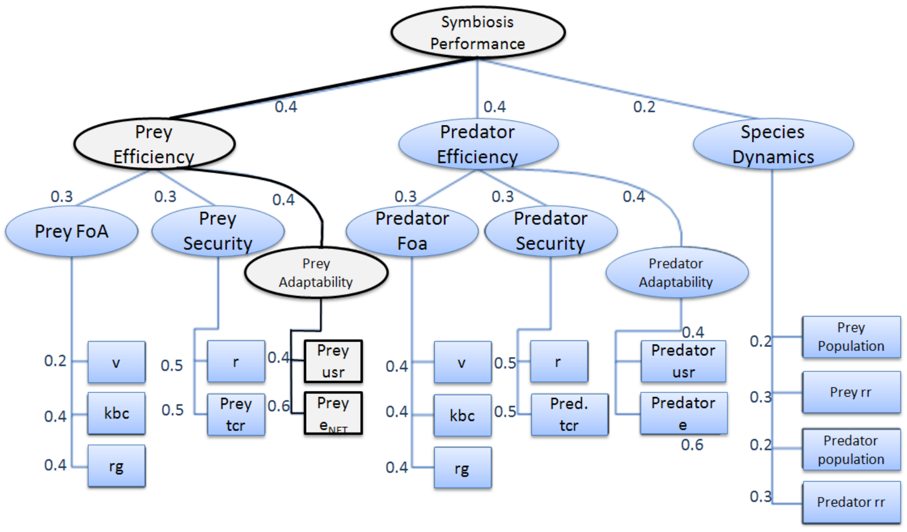

Fig. 2. The Symbiosis MRT

$$
\begin{aligned}
&\mathit{PreyAdaptability} : \mathit{CompositeMetric} \mid \theta\ \mathit{CompositeMetric}\\
&\quad = \lhd \mathit{children} \mapsto \{\mathit{prey\_e_{NET}},\\
&\quad \mathit{prey\_usr}\},\\
&\quad \mathit{weightedChildren} \mapsto \{\mathit{prey\_e_{NET}} \rightarrow 0.6,\\
&\quad \mathit{prey\_usr} \rightarrow 0.4\}\},\\
&\quad \mathit{value} \mapsto \mathit{Aggregate.caclulateCompositeMetric}\\
&\qquad \mathit{children\ \ weights} \rhd
\end{aligned}
$$

In the above relations, SimpleMetrics $prey\_e_{NET}$ and $prey\_usr$ would have been defined as:

$$
\begin{aligned}
&\mathit{prey\_usr} : \mathit{SimpleMetric} \mid \theta\ \mathit{SimpleMetric} =\\
&\quad \lhd\ \mathit{aAttr} \mapsto \{\mathit{prey\_net\_efficiency}\}, \mathit{value} \mapsto 0.151 \rhd\\
&\quad \mathit{prey\_e_{NET}} : \mathit{SimpleMetric} \mid \theta\ \mathit{SimpleMetric} =\\
&\quad \lhd\ \mathit{aAttr} \mapsto \{\mathit{prey\_unknown\_situation\_rate}\}, && (1)\\
&\qquad \mathit{value} \mapsto 0.942 \rhd && (2)
\end{aligned}
$$

Note that subjectivity in metrics selection and association is inevitable when the MRT design is exclusively handled by a domain expert.

Having constructed the MRT, it is now easy to evaluate system performance, given different experiment (agent and environment) parameters. Illustration of results is omitted due to space limitations.

5 Discussion and Future Work

Within the context of this work we have presented APE, a generic evaluation methodology for agents and MAS that provides a coherent framework of tools and guidelines for organizing and using evaluation-related information. The Metric Representation Tree (MRT) defined provides a structure for defining and hierarchically organizing both simple and composite metrics. Through the use of the MRT, APE extends the current evaluation practices by quantifying higher-level evaluation concepts (composite metrics) and relating them to existing, directly measurable performance aspects (simple metrics). This way, qualitative discussions often found in literature can now be integrated in the evaluation process in a quantitative manner.

APE supports both the measurement and aggregation phases of evaluation. During the former phase, measured values are bound to the simple metrics of the MRT, while during the latter phase, all composite metrics are being calculated based on appropriately defined weights and aggregation functions. The evaluator is able to focus on any subtree of the MRT in order to isolate aspects of system performance or examine the entire MRT for obtaining a single system-wide characterization of performance.

Along with the MEANDER automated evaluation framework, APE is a complete, domain-independent methodology that addresses the issue of evaluating complex and unpredictable agent systems, by enabling evaluation at different levels of granularity.

The definition of the metrics and weights in the MRT is currently undertaken by domain experts and, subjectivity is, therefore, inevitable. However, there are several techniques for automating this procedure. For example in [2], we propose a training method for the MRT that assesses the validity and certainty of the selected metrics and weights, based on historical performance data. Towards this direction, we envision a community of experts for each application domain that would initially define MRTs to be continuously refined by using historical performance data on this domain. Thus, evaluators would have access to readily-available, domain-specific MRTs, being relieved from the burden of re-inventing metrics and therefore avoiding current ad-hoc evaluation practices.

References

1. James Albus, Elena R. Messina, and John M. Evans. Measuring performance of systems with autonomy: Metrics for intelligence of constructed systems. In *In Proc. of the First International Workshop Performance Metrics for Intelligent Systems*

(PerMIS) Workshop, NIST SP 970 (eds. E. Messina and A.M. Mystel), pages 1–30, 14-16 August 2000.
2. Christos Dimou, Manolis Falelakis, Andreas L. Symeonidis, Anastasios Delopoulos, and Pericles A. Mitkas. Constructing optimal fuzzy metric trees for agent performance evaluation. In *In Proceedings of the 2008 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-08)*, 9-12 December 2008.
3. Christos Dimou, Andreas L. Symeonidis, and Pericles A. Mitkas. An integrated infrastructure for monitoring and evaluating agent-based systems. *Expert Systems with Applications*, 36(4):7630–7643, 2009.
4. Mark D'Inverno and Michael Luck. *Understanding Agent Systems.* SpringerVerlag, 2004.
5. Michael A. Goodrich, Erwin R. Boer, Jacob W. Crandall, Robert W. Ricks, and Morgan L. Quigley. Behavioral entropy in human-robot interaction. In *Proc. of the Fourth International Workshop on Performance Metrics for Intelligent Systems (PERMIS)*, 13-15 August 2002.
6. Markus C. Huebscher and Julie A. McCann. Evaluation issues in autonomic computing. In *Proceedings of Grid and Cooperative Computing Workshops (GCC)*, pages 597–608, 2004.
7. Barbara Ann Kitchenham. Evaluating software engineering methods and tool, part 2: selecting an appropriate evaluation method technical criteria. *SIGSOFT Softw. Eng. Notes*, 21(2):11–15, 1996.
8. Klaus Krippendorff. *A Dictionary of Cybernetics.* The American Society for Cybernetics, Norfolk VA, USA, 1986.
9. Christopher Landauer and Kirstie Bellman. Refactored characteristics in intelligent computing systems. In *Proc. PERMIS02*, pages 303–306, 2002.
10. Michael Luck and Mark d'Inverno. Structuring a z specification to provide a formal framework for autonomous agent systems. In *ZUM '95: Proceedings of the 9th International Conference of Z Usres on The Z Formal Specification Notation*, pages 47–62, London, UK, 1995. Springer-Verlag.
11. Raj Madhavan and Elena Messina. Permis 2000 white paper: Measuring performance and intelligence. In *In Proc. of the First International Workshop Performance Metrics for Intelligent Systems (PerMIS) Workshop, NIST SP 970 (eds. E. Messina and A.M. Mystel)*, 14-16 August 2000.
12. Andrew L. Nelson, Edward Grant, and Thomas C. Henderson. Competitive relative performance evaluation of neural controllers for competitive game playing with teams of real mobile robots. In *Proceedings of the 2002 PerMIS Workshop, NIST Special Publication 990,*, pages 43–50, 13-15 August 2002.
13. Jean Scholtz, Brian Antonishek, and Jeff Young. Evaluation of human-robot interaction in the nist reference search and rescue test arenas. In *Proc. of the Fourth International Workshop on Performance Metrics for Intelligent Systems (PERMIS)*, 13-15 August 2002.
14. Michael J. Spivey. *The Z notation: a reference manual.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.
15. Fani A. Tzima, Andreas L. Symeonidis, and Pericles A. Mitkas. Symbiosis: Using predator-prey games as a test bed for studying competitive co-evolution. *International Conference on Integration of Knowledge Intensive Multi-Agent Systems, 2007. KIMAS 2007.*, pages 115–120, 30 2007-May 3 2007.
16. Lotfi A. Zadeh. In quest of performance metrics for intelligent systems – a challenge that cannot be met with existing methods. In *In Proceedings of the Performance Metrics for Intelligent Systems (PerMIS) Workshop, NIST SP 990 (eds. E. Messina and A.M. Mystel)*, pages 303–306, 13-15 August 2002.