Towards Empirical Evaluation of Agent Architecture Qualities

Henry Hexmoor, Matthew LaFary, Michael Trosen

Department of Computer Science, University of North Dakota
Grand Forks, ND 58202
{hexmoor, lafary, trosen}@cs.und.edu
http://www.cs.und.edu/~hexmoor

Abstract. Agent architectures are required to exhibit many qualities such as Autonomy, Timely and Purposeful reactions, and Robustness and Failure Tolerance. We discuss evaluation of agent architectures against these qualities. We present experiments that show effective and adjustable levels in the desired qualities for a given architecture.

1 Introduction

In this paper, we discuss evaluation of agent architectures with respect to system *qualities* or *requirements*. A *requirement* or *quality* is a commonly agreed and expected criterion. Examples are Timeliness, Purposefulness, Autonomy, and Robustness and Failure Tolerance. We will provide methodologies for quantifying these qualities. However, we leave out many other qualities such as Coherence, Flexibility and Adaptability, and Reliability. Elsewhere, we have discussed issues about Resource Usage, Interruption, and Learning [4]. Also, we will not duplicate our discussion of Autonomy given in [5].

Functionalities such as "model-based reasoning", "rule-based system", "fuzzy control", and "natural language user interface" are system abilities selected by system designers who feel these best fit their anticipated class of problems. Within their representational framework and procedures of interaction, functionalities are well-known programming techniques. Comparing architectures by functionality is futile since the need for functionalities varies among domains and experts would disagree on choice of techniques.

An *attribute* is an inseparable characteristic of a system such as "runs on a UNIX platform" and "it has three layers". Attributes are design choices that reflect a philosophy of approach to system building. Some might argue that their attribute is inspired by biology and appeals to our intuitive approval. This argument does not prove superiority of one system over another. By and large, comparing architectures by attribute is also futile since attributes don't have inherent values and their effect on system performance cannot be measured.

There has been some comparison and discussion of architectures made along system attributes and functionalities in the literature [8,9]. These comparisons attempt to match system functionality to domain property. These comparisons may provide guidance for designers in selecting a system architecture or be an after the fact justification of their choices. The comparisons along attributes are seldom used since most research groups are disinclined to use an unfamiliar system for their problem. We are less interested in such comparisons. Instead, we are interested in empirical evaluations of architectures along their own qualities. This will be useful in measuring improvements in a given architecture as well as development of domain-independent metrics.

The merits of testbeds and benchmarks for revealing values of agent design concepts are discussed in [3]. They discuss reactiveness versus planned behavior. They argued about inconclusiveness of experiments but that generalizations and interpretations

are left to researchers. AI has benefited greatly by microworlds such as the blocks world. One problem with testbeds is that they are simplistic and AI customers cannot see the value of such demonstrations. Testbeds that mimic real problems domains such as Business, Flight, or Manufacturing might have less universal appeal but enough appeal to be useful for comparison.

We believe the most beneficial evaluation comes from empirical evaluation of qualities in agent architectures. Brown, et al, preset metrics for interface agents [1]. Their metrics in the areas of adaptivity, autonomy, collaboration, and robustness are ratios measuring system's actual level of appropriate output over maximum expected appropriate output. These measures consider the overall system but do not target effect of specific system design/architecture.

Mali and Mukerjee in [6] define behaviors and tasks, and present concepts for comparing behaviors along dimensions of power, span, task space, usefulness, flexibility, and scalability. Their comparative metrics become guides for knowledge, engineering behaviors at appropriate levels along their dimension of analysis. Mali in [7] goes further and defines goals, environments, and markers. Mali sketches that more complex goals are achieved by adding markers or making behavior sets complex by adding coupling among behaviors. He argues that reactivity and planning do not have to be considered orthogonal but are related along how well the agent treats complexity of the world. With that he implies that there is a spectrum of ways to interact with the world. A knowledge engineer can change the behavior sets or markers to gain more or less reactivity. We agree with Mali's points. However, this does not invalidate multi-layered architectures since they provide concurrent means of interaction with the world. To use Mali's terms, his analysis provides a framework for designing behavior systems that address reactivity along the spatial dimension. Hybrid systems are addressing spatial as well as temporal dimensions of interaction with the world.

2 Timeliness

Generally, we want agents to react fast to take advantage of all opportunities. However, an agent might act faster than the rate at which opportunities appear. In that case the agent's effort is wasted. If we assume agents' opportunities for tasks appear at an equally likely rate in the environment and an agent takes a constant time between successive reactions, we can measure the number of opportunities missed due to slowness of reaction constants. Constant reaction times is not common in real life but here it is used to illustrate an empirical observation.

For concreteness, consider a simplified scenario. Assume an agent is expected to perform N tasks in ascending order. At any time, if the agent has already performed task i, she will only want to perform the $i+1^{st}$ task. For example, this is the case when building a physical structure like a building where it systematically proceeds from the ground level to the top level. Assume that during R units of time, the agent generates only a single response to only one task. R is the reaction constant. The world changes such that R opportunities arise during the reaction constant. One opportunity appears per unit of time. Each opportunity randomly corresponds to one of N tasks. Clearly, if R = 1, the agent is faced with an equally random opportunity to perform a task and no opportunities are overlooked by the agent. Let's consider two specific cases.

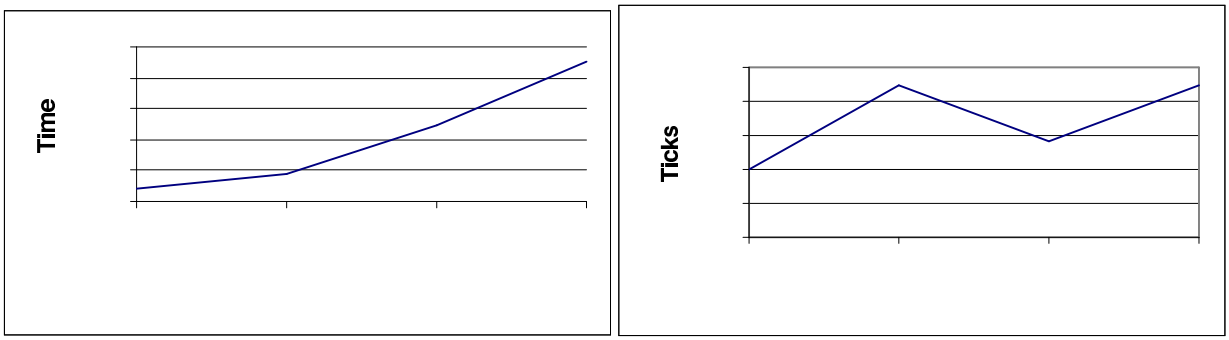

Figure 1 Reactions in case I
Left: the larger R is the longer it takes to complete tasks
Right: system ticks is invariant to R

Case I. The agent only considers the last opportunity in R and if it is not yet done, it reacts to it. Such an agent ignores R-1 opportunities in each time constant for which it does not generate a reaction. We expected that the larger R is the longer it takes to complete N tasks. We empirically tested this expectation and the result is shown in Figure 1 with N = 100.

Ticks in the Figure are the number of reaction constants to complete N tasks. Ticks in the programming term are the loop counts. Time = Ticks * R. The right diagram in Figure 1 shows a near flat line which means the number of ticks is invariant to size of R.

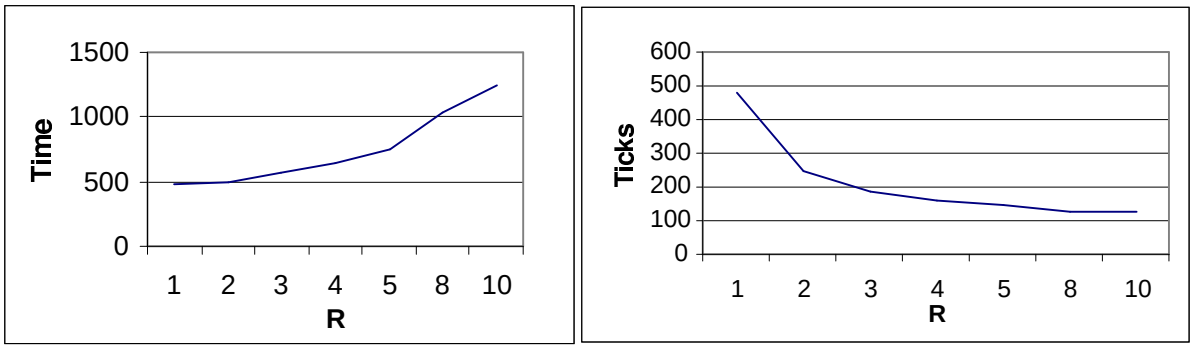

Figure 2 Case II Reactions
Left: the larger R is the longer it takes to complete tasks
Right: the larger R is, the fewer ticks are needed to complete tasks

Case II. The agent selects one of R opportunities in the reaction constant and reacts to it. Unlike case I, this agent considers all R opportunities in a reaction constant and picks one that it has not yet done. It still has to do the tasks in an ascending order. Similar to Case I, this agent ignores R-1 opportunities in each time constant for which it does not generate a reaction. The missed opportunities might be out of sequence tasks, the tasks already accomplished, or repeated opportunities for the same task. As shown in Figure 2 Left, to some extent the phenomenon from

Case I is repeated that is the larger R is the longer it takes to complete N tasks. However, time rises much slower than in case I.

Unlike Case I, ticks are not invariant to R. Our empirical test shows that the larger R is, the fewer ticks are needed to complete N tasks. This is shown in Figure 2 Right with N = 100. The number of ticks falls fast as R increases. So, within low R range (say 1-3), larger reaction constants are not as unfavorable as in Case I. Additionally, if we consider that small R carries a cost overhead, we see that larger reaction constants might be a reasonable compromise.

At R > 4, we see an almost linear part of the curve in Figure 2 Right with very little change but the time returns to a faster rise in Figure 2 Left. R = 4 is a point at which the tradeoff is similar to the curve for Case I reactions.

In architectures that have different layers with each responding to a different opportunity, we can assume different reaction constants per layer. Monitoring mechanisms installed in the agent architecture can detect tradeoffs in reaction constants and help determine them.

3 Impact of Purposefulness on Timeliness

Purposefulness is often at odds with timeliness. If you want to perform tasks fast, you can not afford to be selective. Purposefulness implies goal-directness, deliberative or acting with intention. In many situations, purposefulness pays off when there is interference among tasks, and certain order of tasks may produce a bad consequence or a good one. In this paper we don't want to rehash the tradeoff between reactive versus goal-directed behavior. Nor do we want to measure types of purposefulness. We only want to examine the impact of purposefulness on timeliness. As a purposeful agent, we will consider an agent that uses a simple parameter to discriminate among its opportunities. Whereas some task successions might be desirable, others may detract from the agent's goal or even undo the agent's accomplishments.

For concreteness, let's assume a simple scenario similar to the timeliness in case II, with the added twist that the agent pays attention to task succession. As in Timeliness, the agent still has to do the tasks in the ascending order. We introduce a matrix of task succession biases where we randomly determine that 1/3 of the task pairs are good, 1/3 are bad, and 1/3 are neutral. When the agent is mindless (i.e., not purposeful), it ignores the biases. When the agent is selective (i.e., purposeful), it prefers the tasks in good successions, if there are no good tasks in good succession it picks neutrally ordered tasks, but it avoids bad successions.

Mindless agent is the same one in Timeliness CaseII. We expect that selectiveness will generally slow down the agent. As shown in Figure 3, our empirical tests show that with N = 100, the selective agent takes longer to complete its tasks. However, after R > 10, the curves for the selective agent and the mindless agent merge. This makes the selective agent as timely as the mindless agent. Another observation is that after R>10, the ticks level off. I.e., the agent starts to lose time being selective and both curves become like Case I timeliness. So purposefulness is invariant to further changes in R.

In this scenario, our mindless agent may incur a cost due to selection of bad consequences. However, our experiments show that the shape of the curve does not change at all. If each time a mindless agent is punished for selecting a task in a bad sequence, we may consider that it wasted a reaction constant. We observed about 30 such bad choices in 100 tasks. Our curve for the mindless agent in Figure 3 shifts up by about 30 units (the shifted is not shown to avoid clutter).

What if a bad choice cost 5 time constants. That would shift the curve for the mindless agent in Figure 3 up by 150 units (again not shown to avoid clutter). In that case, even at R = 2, the selective agent would finish faster (270 ticks) than the mindless agent (370 ticks).

Our experiments are very simplistic but they show the general impact of purposefulness. Monitoring mechanisms installed in the agent architecture can detect tradeoffs in reaction constants and help determine them when selectiveness would have a negative impact on timeliness or otherwise improve it.

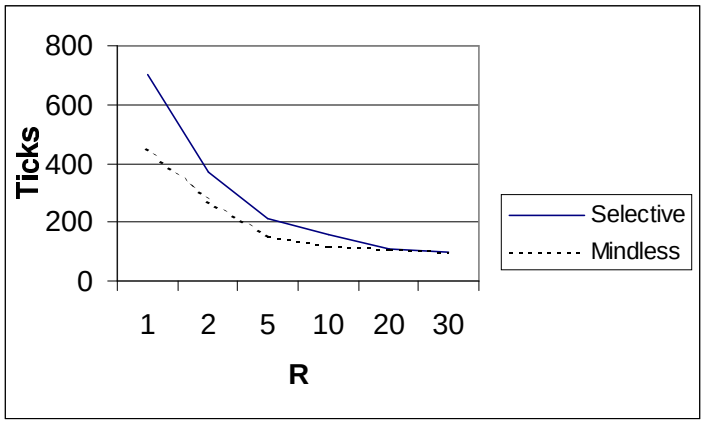

Figure 3 Timing of the Selective agent versus the Mindless agent

4 Autonomy

Generally, we want agents to have a desired level of autonomy over their tasks. We believes that there is not much point arguing about the level of autonomy in rocks, thermostats, or robots. The quality of autonomy is like love and affection. There is no good absolute measure. What matters is the relative autonomy with respect to other agents, about given tasks, at given times, and relative to domain constraints.

We believe there are two sharply contrasting ways to view autonomy. In a setup in which a human is the user of computer programs, issues of autonomy refer to ways in which the human is allowed to view workings of the programs. Viewed this way, sophisticated user-interfaces are needed to monitor and interrupt the programs. This reminds us of the human quarreling with Hal in 2001: A Space Odyssey. In a setup where a number of automated systems share a goal, issues of autonomy refer to ways in which agents arbitrate their tasks. Viewed this way, sophisticated multi-agent models are needed to model agent decision making. This reminds us of the members of a soccer team continually deciding their actions taking team-member efforts and other mental attitudes into account. The latter is the viewpoint in this paper.

In order to approach quantification of autonomy we suggest that a set of features be identified. A *feature* is a generic parameter using which the system can adjust its interaction in the environment. Two of the features we find useful in autonomy are the level of *competence* an agent has for a given task, and the level of *priority* an agent might hold for achieving a given task. We believe there are other features affecting autonomy that we will leave out. Among the features

we leave out are social obligation and conventions among agents, and consideration about resources.

For concreteness, we will consider agents with different competence for tasks over tasks with differing priority and across multiagent system with three different utility functions. We want to observe the rate of accumulation of task priorities. Naturally, it is desirable that tasks with highest priorities be done first. So the utility function that promotes fastest priority accumulation is the most desirable.

Let's assume a simple scenario for agents where their competence level is in the range 1-10. High values are high competence and low values are low competence. Furthermore, we select two sub-ranges, say 1-3 (low competence) and 6-9 (high competence). Given a task, consider that we can assign a probability of the agent having high competence in the task. We'll call the probability value of the agent it's Cando. For example, if an agent's Cando is 90%, given a task, there is a 90% chance of the agent having a high competence for the given task. In our experimental setup, higher competence simply means faster task execution. The least competence level takes one unit of time an proportionally, most competence level takes ½ time unit.

Let’s assume each task has a random priority in the range 1-10. We allowed agent to have an individual perception of these priorities. A random value in the range of -2 to +2 is added to the actual task priority to model the agent’s perception of that task’s priority. For instance, if a task has an assigned priority of 8, an agent may perceive that to be a priority in the range 6-10. The differing perception of priorities by agent not only adds a degree of realism to our setup, it becomes useful in separating performance of two utility functions discussed further in this section.

The agent's policy for choosing a task varies according to the agent's rank relative to other agents. The policy of task choice affects the order of task execution. We consider three schemes for the choice of tasks by the agents. The Fully-autonomous scheme is where an agent acts independent of others. Each agent selects the task that has the highest product of priority and competence. The Boss scheme is where the boss agent assigns tasks to other agents. The boss agent computes a product of task priorities (boss’s perception) and the agent's competence for a task. The Cooperative scheme is where agents first unify their task perceived priorities by averaging them. They then individually use the product of average priority and competence for task selection. Autonomy of agents involved in these schemes is affected by the policy. For example, an agent in the Cooperative scheme has a different autonomy relationship to the tasks than an agent in the Fully-autonomous scheme. We will not measure a precise degree of autonomy with respect to other agents or with respects to tasks. It is enough for use to identify agent autonomy that corresponds to the policy.

We chose different numbers of tasks and agents at different Cando levels. We ran many experiments and repeated each 2000 times. Figure 4 shows a representative set of two graphs. The graphs are drawn with actual values of task priorities are based and not agent perceived values. For 7 tasks and 7 agents, on the average, it works out that priority accumulation builds up to little over 30 in our experiments. Figure 4 includes a case for Random, which refers to arbitrary assignment of tasks to agents regardless of agent competence levels or task priorities. Averaged over many runs it is the lowest curve. As shown in Figure 4, the Fully-autonomous policy is inferior to the Boss and the Cooperative schemes. This is because agents who are not the most competent for the task might perform the tasks. The Cooperative scheme is slightly superior to the Boss scheme since averaging agents perceptions of priorities comes closer to the actual priorities than the boss agent's perception of task priorities. This effect is attributed to superior

matching of tasks to agents in the case of boss scheme. So a bunch a low competence agents don't need a boss as much as a bunch of high competence agents.

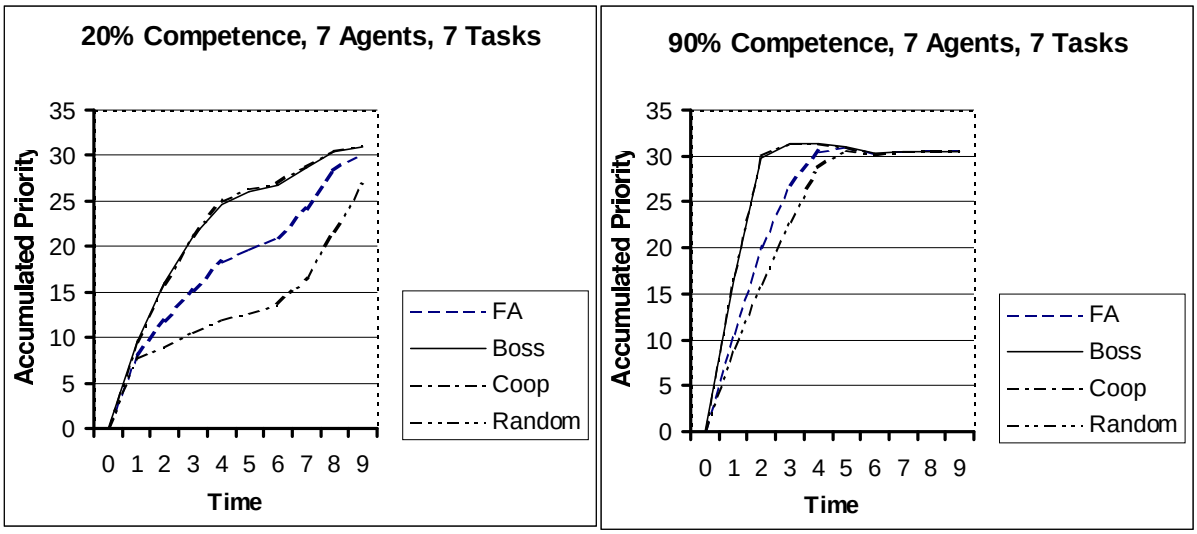

Figure 4 Multi-agent Autonomy
Left: 20% competence, Right: 90% comeptence

Our experiments lead us to believe that in multi-agent architectures, the system should adaptively change task arbitration to speed accumulated priorities. Changes in agent competence as well as task priorities should be monitored. With high levels of agent competence, task arbitration is best as the Boss scheme.

5 Robustness and Failure Tolerance

Generally, we want agents that complete tasks even when there are faults in the system and failures in the agents. Within a single agent, we expect it to compensate for failures in one aspect of its decision making by having different centers or layers that can in part or in whole make up for the failure. In a multi-agent system, we expect other agents to take up the slack for an agent's failing. In this paper we will focus on multi-agent systems and examine that shortcomings in agents are alleviated in situations where agents continually examine new their choice of tasks. In the remainder of this section we will focus on agent failures and will not consider faults. If we measure the system performance against the magnitude or the number failures we want a system that will minimize its performance variance with changes in the overall magnitude or the number of failures.

For concreteness, consider the setup we described for our Autonomy experiments. We can compare groups of agents with different Cando levels. We do not model failures explicitly. We also have not modeled permanent failures where the agent may not be able to do any task or be permanently stuck with a single task. Let's assume that low competence in agents is due to their failure in tasks. Agents having low Cando levels take longer to complete their tasks than agents with high Cando levels. This can be due to the time it takes to deal with failures. This follows the principle that for a given agent competence and failure are inversely related. Although we have not modeled severe failures we believe our results will hold with more severe failures. The

system performance for us will be the time to accumulate all 30+ task priorities from Figure 4. We will call this "time to peak".

Figure 5 shows the result of comparing groups of agents at different Cando levels. As in Autonomy experiments, this data corresponds to 7 agents and 7 tasks. The general trend for all curves is downward. This implies that with generally failure-prone agents (low competence agents), the time to complete tasks is high and vice versa.

Expectedly, the Random scheme, which is to ignore competence levels and priorities, produced the longest time, the top curve. The next curve shows the Fully-autonomous scheme. The Boss curve is below the Fully-autonomous curve and the Cooperative is the lowest curve. With regards to autonomy, the Boss and the Cooperative are better than other curves. Also, we see the Cooperative scheme is clearly better than the Boss scheme. In summary, the relative merit of these schemes is the same as discussed in the previous section on Autonomy.

Clearly, the Cooperative scheme provides the best performance. However, it is the most susceptible to failures (competence levels). The changes in performance in Figure 5 are in order Random (4 time units), Fully-autonomous (4.9 time units), Boss (5.8 time units), and Cooperative (6.4 time units). With regards to failure-tolerance, the Random scheme is the best and the Cooperative is the worst. The shape of curves in the Boss scheme and the Cooperative schemes are nonlinear and change the most around 40% competency level. This leads us to believe that Boss and Cooperative schemes are more susceptible to changes in competence (and failure) levels.

The tradeoff between performance and failure-tolerance can only be decided with respect to the domain characteristics. In domain with remote and time sensitive and hazardous materials, failure-tolerance is more desirable than performance so a full-autonomous scheme might be used. In domains like grazing where we want to cover a large area quickly, performance is more important and the Cooperative scheme will be more suitable.

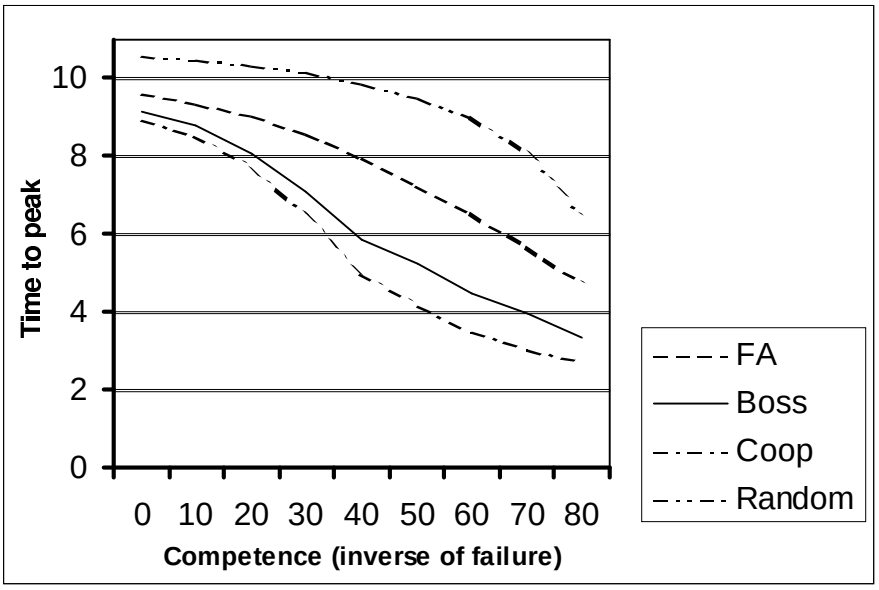

Figure 5 Performance versus failures

6 Conclusion

We have discussed system qualities and evaluation methods for Timeliness, Purposefulness, Autonomy, and Robustness and Failure Tolerance. Our discussions and illustrations have been based on empirical and domain-independent evaluations. We have not specified the fit between environmental parameters and architectural features. We believe agents need to have internal monitoring systems that measure the levels in these qualities and adjust them for the best fit.

We have shown that under certain circumstances, being selective can help with timeliness. A cooperative scheme that accounts for agent competence and task preference helps with timeliness of a multiagent system. Finally, we discussed a tradeoff between failure-tolerance and timeliness and see that a multiagent system of Full-autonomous agents is less susceptible failure-tolerance and a Cooperative team.

References

1. Brown, S., Santos, E., Banks, S., Oxley, S., (1998). Using Explicit Requirements for Interface Agent User Model Correction, International Conference on Autonomous Agents (Agents '98).

2. Gat, E., (1998). Three-Layer Architectures, D. Kortankamp, P. Bonasso, R. Murphy (eds), 1998. Artificial Intelligence and Mobile Robots, MIT Press.

3. Hanks, S., Pollack, M., Cohen, P., (1993). Benchmarks, Testbeds, Controlled Experimentation, and the Design of Agent Architectures, AI magazine 14(4): 17-42, MIT press.

4. Hexmoor, H., (1999). Autonomy in Spacecraft Software Architecture, FLAIRS-99 track: Artificial Intelligence Applied to Spacecraft Autonomy, Orlando, FL.

5. Hexmoor, H., LaFary, M., & Trosen, M. (1999). Agents with Adjustable Autonomy. AAAI-99 spring symposium on Agents with Adjustable Autonomy, D. Musliner, and B. Pell (organizers and editors). Stanford, CA.

6. Mali, A., & Mukerjee, A. (1998). Metrics for Evaluation of Behavior-based Robotic Systems, IEEE International Conference on Robotics and Automation (ICRA), Belgium.

7. Mali, A. (1998). Tradeoffs in Making the Behavior-based Robotic Systems Goal-Directed, IEEE International Conference on Robotics and Automation (ICRA), Belgium.

8. Scerri, P. and Reed, N. (1999). Requirements for a General Agent Architecture for Agent-Based Simulation Environments, In workshop on **Autonomy Control Software**, PP. 102-108, Seattle, WA.

9. Wooldrige, W. and Jennings, N. (1998). Pitfalls of agent oritentated development. In Proceedings of Autonomous Agents '98, pages 385-391, ACM Press.