Enabling Trust with Behavior Metamodels

Scott A. Wallace
Washington State University Vancouver
14204 NE Salmon Creek Ave.
Vancouver, WA 98686
swallace@vancouver.wsu.edu

Abstract

Intelligent assistants promise to simplify our lives and increase our productivity. Yet for this promise to become reality, the Artificial Intelligence community will need to address two important issues. The first is how to determine that the assistants we build will, in fact, behave appropriately and safely. The second issue is how to convince society at large that these assistants are useful and reliable tools that should be trusted with important tasks. In this paper, we argue that both of these issues are be addressed by behavior metamodels (i.e., abstract models of how the agent behaves). Our argument is 1) based on experimental evidence of how metamodels can improve debugging/validation efficiency, and 2) based on how metamodels can contribute to three fundamental components of trusting relationships established in previous literature.

Introduction

Computational systems that can communicate with humans and perform tasks reliably and competently have been a long standing goal of the Artificial Intelligence community. These intelligent assistants (agents) hold the promise of a simplified life and of increased productivity. Little by little, research and engineering are helping to bring this promise within reach. Over the past two decades, computers have rapidly been incorporated into nearly all facets of our daily lives. Computers operate the cell phones we carry in our pockets, the PDAs that help us track our appointments, and they monitor and adjust the settings of our automobiles' engines. Nonetheless, the futuristic robotic assistants envisioned in the 1960s are nowhere to be seen. Moreover, although computers have helped to automate many simple tasks (e.g., paying reoccurring bills) intelligent software assistants that perform complex tasks autonomously on our behalf are still far from common place.

As researchers, we are constantly looking for methods to improve the state of the art: how can the methods of yesterday scale to the problems of tomorrow? Perhaps one reason intelligent assistants are not more common is that research has not yet advanced to the point that we can (even in theory) make assistants that perform useful and complex tasks autonomously. However, another possibility exists that is rarely explored in the AI literature: perhaps we, as members of society, are unwilling to delegate important tasks to autonomous computerized assistants.

We believe that as research and engineering allows us to build more and more complicated autonomous software, this second issue of societal acceptance will become increasingly salient. There are at least two aspects of the societal acceptance problem that, though related, deserve to be explored individually. The first problem is how do we, as the engineers of these systems, convince ourselves that next generation assistants will be useful, reliable, and practical tools for society at large. The second, related, problem is how to convince the rest of society to become willing users of these new technologies.

In this paper, we take the stand that the first problem (convincing ourselves) arises not simply because researchers cannot create useful or reliable tools, nor because researchers are inherently unwilling to use the tools they develop. Instead, we believe that an important gap exists between promises made by theoretical developments and the practical constraints of implementation.

At first glance, the second problem (convincing society) may seem to be one simply of marketing. Yet research in psychology, sociology and human-computer interaction suggest that some of this issue's roots may be much deeper. In particular, a large body of literature cites trust as a critical factor in developing human-human and human-computer relationships. Although marketing may help to influence trust, it is likely that there are more appropriate methods to help end-users establish trust in tomorrow's intelligent software assistants.

In this paper, we explore these two problems in greater detail. We begin by examining the gap between the promise of theory and the practical constraints of implementation. Then, we explore notions of trust and why trust may be a critical factor in the acceptance of future intelligent assistants. In the section entitled Metamodels, we show how both problems can be addressed in a unified way using abstract representations of behavior. In particular, we show how metamodels can improve the efficiency of the validation process leading us to the conclusion that metamodels can help create more robust and reliable agents. Next, we show how metamodels support three fundamental components of trusting relationships thereby enabling trust between agents

and their users. Finally, we describe our current work with metamodels and conclude with directions for the future.

The Gap Between Theory and Practice

The gap between the promises of theory and the constraints of practice is perhaps most evident when we consider the problem of determining whether an intelligent assistant has been implemented correctly. To perform complex tasks, these assistants require complex supporting software. Simply encoding the requisite knowledge to describe potential goals, actions, and the relationships between them is a difficult engineering problem even without considering the agent architecture that turns this knowledge into executable form. Thus, even if we know how to build the intelligent assistants we desire, the task may be so difficult that it cannot be managed without compromising its very integrity.

As we argued in (Wallace 2005), successfully constructing complex intelligent agents requires overcoming challenges in both specification and implementation phases of development. In particular, it is often hard to specify precisely what constitutes correct behavior for an intelligent assistant since this specification may exist only within the mind of a human domain expert. Implementation, even assuming a well defined specification is difficult and error prone simply due to the shear size and complexity of the supporting knowledge base.

These difficulties are compounded by the fact that it is not always clear how to apply software engineering techniques to maintain control of the development process. This is due, in part, to the fact that many common programming approaches used in artificial intelligence (e.g., rule based system and logic programming approaches) are substantially different from those used in mainstream software development. Moreover, many techniques for storing knowledge, such as conditional probability tables, are easy to build with automated techniques, but difficult for humans to inspect and assess.

In traditional knowledge-base validation techniques (Kirani, Zualkernan, & Tsai 1994; Tsai, Vishnuvajjala, & Zhang 1999), a domain expert and knowledge engineer find and correct errors by examining an agent's behavior in a number of test scenarios. Although this is still the de facto approach for identifying errors in many complex intelligent systems, it has several weaknesses. It is time consuming; each test requires the domain expert's participation. It is error prone; problems in the agent's behavior are likely to be relatively few and far between and the domain expert must be constantly engaged in order to identify the remaining flaws. Finally, because of the time and cost associated with this approach, it is likely that only a very small amount of potential behavior can be tested.

Reducing the time and cost of validation would allow the agent to be tested in a larger number of scenarios. This, in turn, would increase the reliability of the overall system. As we will see in later sections, metamodels can be useful tools to help decrease the cost of validation by increasing the speed at which errors can be identified and corrected.

Enabling Trust Between Users and Agents

For society to become willing users of intelligent assistants, these assistants need to hold the promise of improved quality of life or improved productivity. Moreover, the improvement gained by using these agents will need to be evaluated against the potential harm that they could cause (i.e., if they fail to perform their tasks). This same situation exists when dealing with a human assistant.

In our everyday lives, we need to work with other people, many of whom we have never met, in order to accomplish our goals. Even tasks as mundane as ordering a new computer over the phone requires the interplay of many different people for the transactions to complete successfully. Thus, in order to function in today's world, we must be able to evaluate the rewards that can be obtained by relying on others against the potential risks of such associations.

In most situations, an individual cannot accurately assess the real risk and reward of establishing a particular relationship. It is the construct of *trust*, that allows people to navigate these situations without complete knowledge (Lewis & Weigert 1985; Fahrenholtz & Bartelt 2001; Fogg & Tseng 1999). Indeed, Luhmnann, who is widely cited in modern trust literature, argues that a critical function of trust is to reduce complexity (Luhmann 1979). Thus, it is trust that keeps people from the impossible task of attempting to analyze all possible outcomes that might result from forming a particular relationship (Simmel 1964; Lewis & Weigert 1985).

Models of Trust

Trust has been the focus of a numerous studies and reports in sociology, psychology and computer science literature. In this paper, we focus our attention on three models of trust that have received recent attention but have much older roots.

The first model of trust, described by Ratnasignham in (Ratnasingham 1998), is based on the work of Lewick and Bunker (Lewicki & Bunker 1996) and Shapiro et al. (Shapiro, Sheppard, & Cheraskin 1992). The model identifies three stages of trust that develop between two parties over the course of time. The first stage is called deterrence-based trust. Here, the relationship is typically in early stages of development and trust is linked more to a threat of punishment that to a promise of reward. As the relationship continues to develop, it may enter a second stage based on knowledge. At this point, trust is linked to past experiences that allow the trustor to understand and predict the behavior of the trustee. The third, and final, stage of trust is identification based. In this stage, trust stems from empathy and common values, similar to the trust that is established between friends and family.

A second model of trust is described by Lewis and Weigert in (Lewis & Weigert 1985). These authors identify three distinct components which occur to different degrees in any single trust relationship. The first is cognitive trust, based on a knowledge of the other party's character or their past actions. Lewis and Weigert note that knowledge alone is not sufficient to cause trust. Rather, in their mind, trust

is the construct that results from taking a "cognitive 'leap' beyond the expectations that reason and experience alone would warrant" (Lewis & Weigert 1985, pp. 970). In addition to this cognitive basis, the authors also indicate that trust always has an important emotional component. This serves as a critical link between parties; breaking the trust bond causes emotional distress (wrath or pain) that neither party wishes to experience. The final component of trust is behavioral and is affected by the way in which each party treats each other. Namely, if the parties act as though they trust one another, this helps to establish a bond of true trust; the converse also appears to be true.

The final model of trust we examine is that presented by Fahrenholtz and Bartlet (Fahrenholtz & Bartelt 2001) based on the work of sociologist B. A. Misztal. In this model, there are three high-level trust *forms*, each of which has three manifestations called practices. The practices are essentially behaviors that a person uses in lieu of grounded information; that is, these behaviors serve to manage complexity.

The first form is habitus; its first practice is that of habits all of which can be viewed as a form of default behavior that reduces the need to react or perform deep reasoning about what to do in a particular situation. The second practice is reputation, a mechanism that helps place reasonable limits on trust based on mutually recognized tokens such as diplomas. The final practice is memory, the imperfect process of recording history, which can be reshaped over time.

The second form in this model is passion. Passion operates independently from reason and is based on familiar associations between two people. The three practices of passion are identified as those dealing with family, friends and society.

The final form is policy. Its first practice is solidarity, which is the practice of taking the goals of a larger group based on mutual understanding. The second is toleration which is the acceptance of difference, and places limits on how much difference can be allowed in a trusting relationship. The final practice is legitimacy which results from existing beliefs and describes the degree to which institutions are valued.

Common Traits

Although each of the three models described above have distinctive features, they also share some common traits. Equally interesting is the fact that although these models are designed to describe trust between people, research in Human-Computer Interaction has demonstrated how many of these common traits are also relevant to relationships between humans and machines. Below, we explore these common traits.

The first trait shared by the models above is the notion that trust is somehow related to *predictability* and *understanding*. Knowledge based trust, cognitive trust and the trust that results from habits and memory are all examples of this trait. Echoing this, experiments involving human use of automated (software) aids have indicated that an understanding, or at least an explanation, of why the aid may make mistakes increases the users' level of trust in the aid and their likelihood to rely on it (Dzindolet *et al.* 2003). Similarly, intelligent systems that can justify their actions have been shown to engender greater trust in their end-users (Miller & Larson 1992).

The first and third models also indicate that *similarity* between parties is important for trust. The notions of empathy, common values, solidarity, and passion are all examples of this trait. Fogg and Tseng surveyed studies of computer credibility, a notion closely related to trust that the authors describe as "trust in information" or believability of a source (Fogg & Tseng 1999). They cite two studies that point toward an indication that users find computer programs more credible when they are considered a part of the same group as the user (e.g., from the same company, team, etc.). Bickmore and Cassell describe another experiment that indicates ties between similarity and trust. Here, an intelligent agent's interface was built to allow two conversational strategies. The first strategy used "small talk" to establish connection with a user while the second used only topically relevant discourse. The study found that behavioral similarity between the user and the agent increased trust. Specifically, users who attempted to initiate a conversation with the agent were more trusting of the agent that used small talk while those that didn't initiate a conversation were more trusting of the other agent (Bickmore & Cassell 2001).

Finally, the first and second models both cite *consequences* as important components of the trust relationship: deterrence-based trust and emotional trust illustrate this trait. The fact that machines are not capable of emotion may provide a serious roadblock to establishing credibility with future intelligent assistants. Although research has shown that personification of the interface can help establish trust with its users, personification alone is not enough to sustain trust (Bickmore & Picard 2005). Personification may initially help to establish an intelligent system as an emotional being, yet it is fair to say that nearly everyone recognizes that even today's most sophisticated intelligent systems are incapable of real human emotion. Decisions made by these systems are necessarily based on cold calculation. Therefore, any notion of consequence is similarly calculated and cannot have any special, emotional, significance to the agent. In the near term, it is likely that the threat of consequences will be aimed more towards an agent's developer than toward the agent itself.

Enabling Trust in Tomorrow's Assistants

Just as trust must be established between two people before potentially risky relationships are formed, so too will end-users require trust in their intelligent assistants. While there are no simple answers for how to ensure that such a bond can be formed, in the following section we look at one technique, namely the use of high-level behavior models, that could help end-users better understand and more accurately predict the behavior of future intelligent assistants.

Behavior Metamodels

Behavior metamodels provide a simplified description for how a particular system behaves. They are literally a model of the agent, which itself is a model that represents how to

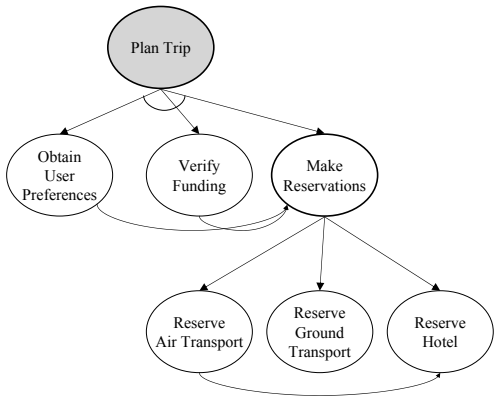

Figure 1: A Hierarchical Behavior Representation

perform a task. While the properties of a metamodel are entirely open ended, we focus on models that have three features: 1) they are inexpensive to produce; 2) they are substantially easier to understand than the agent's internal implementation; and 3) they are consistent with the agent's actual behavior (albeit at a high level of abstraction). As we will argue in later sections, metamodels that meet these requirements can be used both to help establish trust with end-users and to help improve agent validation.

In previous work (Wallace & Laird 2003), we examined a particular type of metamodel called a hierarchical behavioral representation that is inspired by AND/OR trees, HTN planning (Erol, Hendler, & Nau 1994) and GOMS modeling (John & Kieras 1996) to encode the variety of ways in which particular tasks can be accomplished.

An example HBR is illustrated in Figure 1. The hierarchy is an AND/OR tree with binary temporal constraints representing the relationships between the agent's goals and actions. In this representation, internal nodes correspond to goals and leaves correspond to primitive actions (in the figure above, only a partial tree is shown). A node's children indicates the set of sub-goals or primitive actions that are relevant to accomplishing the specified goal. The manner in which sub-goals should be used to achieve their parent goal is encoded by the child's node-type constraint (AND vs OR) and the binary ordering constraints between siblings. In the figure above, *Plan Trip* is an AND node while all others are OR nodes. Nodes can be associated with preconditions (not shown in the figure above) that specify properties of the world state that are consistently true when a particular goal is pursued. Binary temporal constraints between siblings are represented with arrows. Thus, the figure illustrates that to complete the *Plan Trip* goal, the agent must obtain the user's preferences and verify sufficient funds are available before making reservations. Making reservations, in turn, may involve a combination of flights, ground transportation and hotel rooms. However, according to the model, if both a flight and a hotel room are to be reserved, the flight should be reserved first.

The HBR meets each of our three requirements. It is inexpensive to produce; although it may be produced by hand, it is also easily constructed from observational traces of behavior using machine learning techniques (Wallace & Laird 2003). In addition, because the HBR is also a very simple structure, it is much easier to understand than a standard rule base. Finally, although the simplicity of the model necessarily means that some details of behavior are lost, it has been shown to be effective at capturing enough nuances to distinguish between correct and incorrect behavior (Wallace & Laird 2003). In the following section, we present new results indicating just how much these metamodels can increase the speed of identifying and correcting errors in an agent's behavior. Following this presentation, we discuss how these same metamodels can be used to enable trust with end-users.

Metamodels: Reducing the Theory/Practice Gap

Metamodels may help reduce the gap between theory and practice by allowing developers to create more reliable and more robust intelligent systems. While the HBR model discussed in the previous section has been shown to be effective enough to allow developers to identify flaws in an agent's behavior, the time savings of this contribution has not been previously quantified.

To examine this aspect of performance, we conducted an experiment in which five users attempted to find and correct flaws in an agent's behavior given HBRs that represented the agent's current (flawed) behavior, and HBRs that represented correct, expert-level behavior. Agents were implemented in the Soar (Laird, Newell, & Rosenbloom 1987) architecture and each participant had at least six months of Soar programming experience. Participants identified two behavior flaws, with and without the aid of the HBR metamodel. In the unaided situation, participants relied on standard debugging tools and techniques that they were already in the practice of using. Once the flaw was identified, the participants corrected the agents' knowledge using Visual-Soar, the standard Soar development environment. Thus, in the experiments presented below, there are two conditions: aided and unaided. Condition is a within-subject variable, which is to say that each participant experiences both.

Our test-bed agent performed a simulated object retrieval task. A "correct" agent was implemented in 78 Soar rules, and used in conjunction with a simulated environment to create a HBR representing its aggregate behavior. Note that in normal use, observations of correct behavior are likely to come from human experts. However, by creating a correct agent first, it is possible to describe precisely how flawed agents differ from the ideal (both in behavior and in their implementation). This property is critical for the experiment.

Table 1 illustrates salient properties of the correctly implemented agent and two flawed implementations created by modifying one of the agent's rules. The correct agent (column 1) is capable of exhibiting four distinct behaviors (row 3) based on its interactions with the environment. The average length of its behavior traces (i.e., the number of goals and actions required to complete the task) is 67 (row 5). We constructed two flawed agents so that each participant could

	Correct	Flawed-A	Flawed-B
Modification	N/A	New Proposal	Deletion
Manifestation	N/A	Intrusion	Commission
Num. Behaviors	4	12	8
Consistent BTs	N/A	4	4
Avg. BT Length	67	69	68

Table 1: Properties of correct & flawed agents

Participant	Unaided	Aided	Initial Method
1	A	B	Aided
2	B	A	Aided
3	A	B	Unaided
4	B	A	Unaided
5	A	B	Unaided

Table 2: Assignment of participants

perform a debugging task with and without assistance from the HBR (each time using a different flawed agent). One of our primary desires was to ensure that both flawed agents were similarly "different" from the correctly implemented agent. That is, we wanted to ensure that the debugging task was equally challenging in both cases. The first flawed agent (column 2) was created by adding a single new proposal rule (row 1). The effect of the new rule was to introduce inappropriate goals into the otherwise correct behavior (row 2). The flawed agent exhibited twelve distinct behaviors (row 3), four of which were the correct (desired) behaviors (row 4). The second flawed agent (column 3) was created by removing a preference rule that ranked two competing goals against one another. The net effect was that this agent would inappropriately replace one goal for the other. This lead to eight distinct behaviors, four of which were consistent with correct behavior, and four of which where incorrect.

It is worth noting that the flaws introduced into these agents are minor by most standards. In this experiment, flawed behavior does not result deadlocks or infinite loops. Indeed, when viewed in the classical sense, these agents are not necessarily "flawed". They are successful in achieving the desired final state (finding the lost object). The distinction between the correct and flawed agents is similar to the distinction one might make between an experienced plumber and a home-improvement aficionado. Both may be able to get the job done (thus achieving the same final state), but the events that transpired along the way and the efficiency of the job may be remarkably different. In this sense, the home-improvement aficionado would be a inadequate replacement for the experienced plumber just as our flawed agents are inadequate replacements for the correct agent upon which their behavior is based.

Because none of the participants had used, or even seen, the graphical representations of HBR models, an initial tutorial was required. For this, each participant was given an overview of how the HBR modeled behavior. After this discussion, users were assisted in identifying four different errors (misplacement, intrusion, omission, commission) in an exceedingly simple mock environment using the HBRs. In addition to this tutorial, participants were asked to read a short informational summary. This provided an overview of the experiment including a description of the debugging task, a summary of the agent's behavior, and a plain English description of some salient operators used by the agents. This was intended to familiarize the users with the domain and the agent's design without requiring each participant to build their own agent from the ground up. Finally, the participants were alerted that only one rule in each of the agents' knowledge bases had been modified, removed or added in order to generate the behavioral deviations.

After the participant had read the informational summary, they were randomly assigned an agent to validate. We attempted to mitigate bias by varying the order in which the aided and unaided tests were presented as well as the pairing between the agent and the validation method. Table 2 indicates each participant's (agent,validation method) pairings. For each experiment, we asked the participants to indicate when they were ready to modify the agent's knowledge and to articulate what changes they believed were required. This allowed us to measure the amount of time needed to identify the behavioral flaw as well as the total time required to correct the agent's behavior.

During the first phase of the debugging session, participants identified how the flawed agent's behavior differed from the standard set by the correct (original) agent. Participants were free to look for errors using whatever debugging techniques they had developed during previous Soar programming experiences. Similarly, in the aided case, no specific instructions on how to identify errors we given, participants generalized their tutorial experience to interpret the information in the HBRs and identify what changes would be required to make the flawed agents behave correctly.

The second phase of the debugging session began once the participant determined that they were ready to try modifying the agent's knowledge in order to correct the error. Regardless of whether the error identified with or without aid from the HBRs, participants used the VisualSoar editing environment (a standard part of Soar's development environment) to locate and correct the faulty knowledge. Once the participant had made changes, they reexamined the agent's behavior to ensure that the flaw had in fact been corrected. When the participant was confident that the problem was resolved, the clock was stopped and this was recorded as the total time needed to correct the agent's behavior.

Figure 2 illustrates the total time spent by each participant identifying and correcting errors in the flawed agent's knowledge. Encouragingly, the aided approach yields at least a factor of three performance advantage over standard debugging techniques that are unaided by the HBRs. Not surprisingly, a paired t-test confirms that the performance advantage gained when aided by the HBRs are statistically significant ($p = .0002$). This gives creditability to the belief that using the HBRs would increase designers' ability to identify and correct flaws in the intelligent systems they develop.

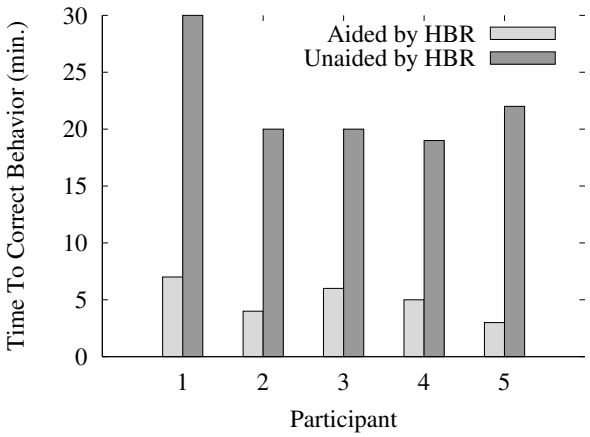

Figure 2: Total time to identify and correct the agent error

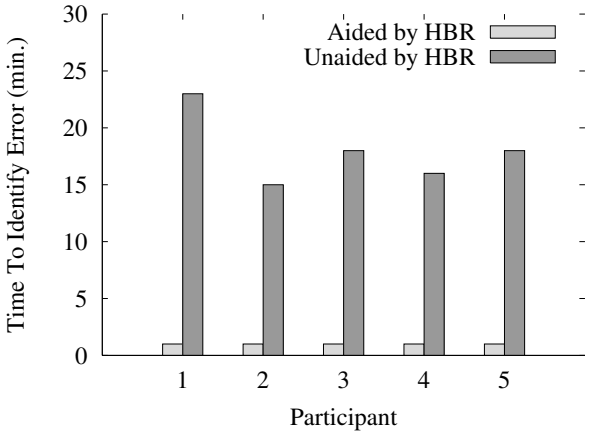

Figure 3: Time to identify the agent error

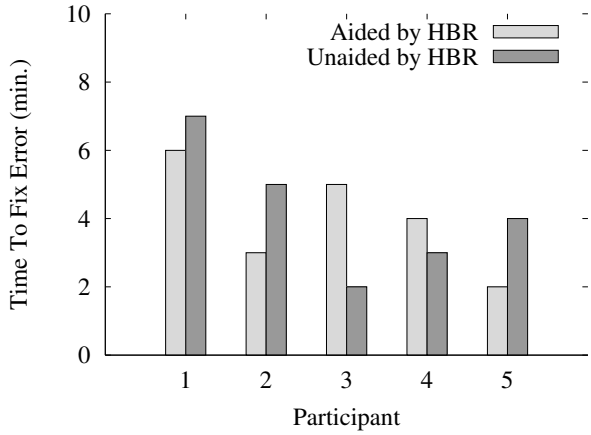

Figure 4: Time to fix the agent error

Figure 3 illustrates the time required by each participant to identify (but not correct) the error in the agent's behavior. Here, time was stopped once the participant verbalized a correct description of what part of the agent's behavior needed correction. Since the metamodel is most useful for the error identification portion of the complete task, this figure illustrates more precisely how much time savings could be realized using the graphical behavior summary provided by the HBR. Somewhat surprisingly, measuring time with minute precision yielded no variance between users aided by the HBRs. Although it is unfortunate that we cannot inspect these measurements in greater detail, we were extremely encouraged by the fact that there was such a substantial difference with and without the aid of the metamodel. Again, a paired t-test indicates statistical significance of these results ($p = .0006$).

Finally, we examined the time required by each participant to fix the agents' flaws once they were identified. This is simply the difference between the values reported in Figure 2 and Figure 3. These times are displayed in Figure 4. The assumption here is that the time required to make the fix should not be dependent on whether the aided or unaided approach was used to identify the error. Examining the figure indicates that there is no obvious trend, and a paired t-test indicates that the differences between aided and unaided performance for fixing a flaw is *not* significant ($p = .85$).

Given the size of our sample, we cannot present a convincing argument that the HBR alone is responsible for *all* performance effects seen above. For instance, we cannon meaningfully compare the overall difficulty of finding Flaw A v.s. finding Flaw B using each technique. However, it does seems safe to conclude that the HBR provides a good deal of information that is useful for identifying behavior differences and for isolating faulty knowledge. By improving the efficiency of the debugging process, HBRs would also allow developers to examine more test scenarios without increasing costs. This, in turn, should increase the developer's ability to reliably construct complex system thereby reducing the gap between the promises of theory and the

limits of practice.

Metamodels: Enabling End-User Trust

Recall that the three trust models examined earlier in this paper cited the following factors as important to establishing trust: *understanding*, *predictability*, *similarity* and *consequences*. Of these, it seems likely that metamodels may help with the first three of these factors.

The experiments reported in the previous section indicated that HBRs were very useful to identify flaws in an agent's behavior. Most likely, HBRs are useful because they allow people (here the experiment's participants) to quickly understand what types of behavior an agent has performed. It seems reasonable to assume that end users will also find it easier to understand the high-level behavior of a particular agent with the aid of a behavior metamodel such as the HBR presented here. Although end users may not be as technically knowledgeable as the participants in this study, and therefore may not receive the same degree of benefit as our participants, it is important to remember that the participants themselves were in no way experts at interpreting the meaning of different HBR structures. There only exposure to these representations was via the short tutorial conducted prior to the debugging task.

If metamodels are successful at helping end-users understand an agent's behavior, we should also expect that through this understanding, end-users will also have an increased ability to predict how a particular agent is likely to behave. Indeed, the very nature of the HBR is to act as a road map that indicates potential ways in which problems can be solved. Thus, although the models may not be particularly useful for identifying the fine details of how an agent will behave, they do provide insight into how the agent decomposes and proceeds to perform complex tasks.

HBRs may also increase understanding and predictability in other ways. In particular, because these models explicitly represent each of the agent's goals and actions, it would be possible to indicate the expected likelihood of the agent successfully completing its task based on the types of goals it is currently pursing. For example, consider an agent that acts as a navigational aid. It's knowledge may be subdivided into goals and actions suited for a variety of situations such as urban or forest environments. In heavily forested areas, the agent may not perform optimally, perhaps due to limitations of its sensors. If performance is related to the goals being pursued (i.e., some goals are more likely to lead to success than others), this information could be stored within the HBR. The result would then be a new representation capable of indicating to the end-user when it may be more or less appropriate to rely on their intelligent assistant. Not only would this help the user understand the strengths and weaknesses of their assistant, it could also serve as a basis for adjusting the agent's autonomy as it becomes less capable of performing its task.

Finally, HBRs, and metamodels in general, may also contribute to trust by increasing notions of similarity between the user and the agent. By improving the transparency of the problem solving process, metamodels not only improve the user's understanding of how the agent reasons, but they also allow the user to draw parallels between their own reasoning and that of the agent.

As argued above, there is good cause to believe that metamodels may positively contribute to three of the four common factors required for trusting relationship. However, one factor that remains unaddressed is the notion of *consequences*. As non-human, and perhaps more importantly, non-emotional entities, the threats normally posed by the violation of trust hold little, if any, water for today's agents. End-users of these systems will likely need to redirect their needs for emotional reparation on to the companies or individuals that design and sell intelligent systems. This possibility brings the value of metamodels full circle. Although metamodels cannot directly address an end-users need to ensure that there are consequences for an agent that breaks a trust relationship, they can help developers limit their exposure to litigation by ensuring that the agents they create are robust and reliable before they end up in the hands of the public.

Future Work: Beyond Trust

So far in this paper, we have discussed the use of metamodels to help developers and end-users understand the behavior of intelligent systems. This can decrease validation costs for developers while also increasing trust amongst end-users. However, we predict that metamodels may be useful for other purposes as well.

Recently, we have investigated the possibility of using metamodels to bring validation out of the laboratory and into the runtime environment. Our system, called S-ASSESS (Wallace 2005) uses a metamodel as a specification of appropriate behavior. S-ASSESS then constrains the agent's behavior during task performance in order to uphold requirements specified by the model while being as consistent as possible with the agent's own preferences. This approach is similar to previous work on safe planning systems (e.g., (Weld & Etzioni 1994) and (Musliner, Durfee, & Shin 1993)) but allows safety constraints to be upheld by reactive systems that don't exhibit distinct planning and acting phases.

As mentioned earlier, metamodels may also provide a natural mechanism to adjust the degree of an agent's autonomy. S-ASSESS facilitates this use as well. During performance, S-ASSESS continually checks to ensure that the agent's behavior will remain consistent with its metamodel. When a threat to the constraint model is identified, a number of options exist. By default, S-ASSESS attempts force the agent to "do the right thing". However, certain types of threats may be more severe than others, offering a natural motivation to reduce the agent's autonomy and fall back to outside expertise. We are just beginning to explore potential applications of this feature in our current work with S-ASSESS.

Conclusions

The future success of intelligent assistants relies on more than the technological and theoretical developments of the Artificial Intelligence community. In particular, we believe that the problem of determining whether complex intelligent

agents will perform their tasks appropriately will become increasingly important in the years ahead. Similarly, there will be a parallel need to convince society at large that intelligent systems are trustworthy and valuable resources. In this paper, we have argued that both of these problems can be addressed by high-level models of an agent's behavior (metamodels). We have demonstrated how metamodels can improve the debugging process, and reasoned that the same models may be useful for establishing trust with end-users.

References

Bickmore, T., and Cassell, J. 2001. Relational agents: A model and implementation of building user trust. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, 396–403.

Bickmore, T. W., and Picard, R. W. 2005. Establishing and maintaining long-term human-computer relationships. *ACM Transactions on Computer-Human Interaction* 12(2):293–327.

Dzindolet, M. T.; Peterson, S. A.; Pomranky, R. A.; Pierce, L. G.; and Beck, H. P. 2003. The role of trust in automation reliance. *International Journal of Human-Computer Studies* 58:697–718.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *Proceedings of the Twelveth National Conference on Artificial Intelligence*, 1123–1128. AAAI Press/MIT Press.

Fahrenholtz, D., and Bartelt, A. 2001. Towards a sociological view of trust in computer science. In *Proceedings of the Eighth Research Symposium on Emerging Electronic Markets*.

Fogg, B. J., and Tseng, H. 1999. The elements of computer credibility. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, 80–87.

John, B. E., and Kieras, D. E. 1996. The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer–Human Interaction* 3(4):320–351.

Kirani, S. H.; Zualkernan, I. A.; and Tsai, W.-T. 1994. Evaluatuion of expert system testing methods. *Communications of the ACM* 37(11):71–81.

Laird, J. E.; Newell, A.; and Rosenbloom, P. S. 1987. Soar: An architecture for general intelligence. *Artificial Intelligence* 33(1):1–64.

Lewicki, R. J., and Bunker, B. B. 1996. Developing and maintaining trust in work relationships. In Kramer, R. M., and Tyler, T. R., eds., *Trust in Organizations: Frontiers of Theory and Research*. Thousand Oaks, CA: Sage. 114–139.

Lewis, J. D., and Weigert, A. 1985. Trust as a social reality. *Social Forces* 967–985.

Luhmann, N. 1979. *Trust and Power*. Wiley.

Miller, C., and Larson, R. 1992. An explanatory and "argumentative" interface for a model-based diagnostic system. In *Proceedings of the ACM Symposium on User Interface Software and Technology*.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1993. CIRCA: A cooperative intelligent real-time control architecture. *IEEE Transactions on Systems, Man and Cybernetics* 23(6).

Ratnasingham, P. 1998. The importantance of trust in electronic commerce. *Internet Research: Electronic Networking Applications and Policy* 8(4):313–321.

Shapiro, D.; Sheppard, B. H.; and Cheraskin, L. 1992. Buisness on a handshake. *The Negotiation Journal* 365–378.

Simmel, G. 1964. *The Sociology of Georg Simmel*. Free Press. Translated, edited and with an introduction by Kurt H. Wolff.

Tsai, W.-T.; Vishnuvajjala, R.; and Zhang, D. 1999. Verification and validation of knowledge-based systems. *IEEE Transactions on Knowledge and Data Engineering* 11(1):202–212.

Wallace, S. A., and Laird, J. E. 2003. Behavior Bounding: Toward effective comparisons of agents & humans. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 727–732.

Wallace, S. A. 2005. S-Assess: A library for self-assessment. In *Proceedings of the Fourth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-05)*, 256–263.

Weld, D., and Etzioni, O. 1994. The first law of robotics (a call to arms). In *Proceedings of the Twelveth National Conference on Artificial Intelligence*.