Studying Service: An Exploration of the Costs and Benefits of Assistance

Wayne Iba and **Nicholas Burwell**
Department of Mathematics and Computer Science
Westmont College, 955 La Paz Road
Santa Barbara, California 93108
{iba,nburwell}@westmont.edu

Abstract

If we are to provide systems that benefit human users, we need a better understanding of the nature of service. In this paper, we describe the necessary characteristics of a simulator for studying service, and then present MÆDEN, our attempt to satisfy those requirements. We then proceed to describe an initial set of agents that we use to explore the nature of service. Our experiments confirm the usefulness of MÆDEN as a testbed, but also point toward insights into service in general. We end with a discussion of future work intended to explore those insights.

Overview

In this paper, we address the problem of service. Because the nature of service is difficult to understand, we face particular challenges when trying to build systems that provide valuable service to users. In this section, we describe our view of service and review previous relevant work. Then in the following section, we describe our work extending a world simulator to support the study of service. Next, we present our design and implementation of agents that give and receive service. We follow this with the results of our first experiments and conclude with a discussion of future work and lessons learned.

Service as a Model of Assistance

Most researchers in Artificial Intelligence, in one fashion or another, are developing systems that are intended to benefit humanity. Yet we understand very little of the nature of service and have trouble identifying why one system is more beneficial to a user than another. If we consider what is taking place when one provides assistance, we discover several limitations of our traditional models. It would seem clear that we are not looking at a competitive situation. Although assistants may have and pursue self-interests, typical adversarial models, where each agent seeks to maximize its own utility at the expense of another's, do not yield many insights into the nature of assistance. It also turns out that we do not have a typical cooperative situation. Cooperative systems tend to have a global objective (even if multi-valued) with respect to which all the cooperating agents optimize their behavior. In the case where one agent is genuinely trying to assist another agent, we have something between a cooperative and competitive situation.

We propose that service is a useful model for evaluating and learning about assistance. First, service tends to emphasize a relationship between a servant and a recipient of service. Second, thinking in terms of service highlights the benefit accrued to the service recipient. Although assistance is closely related, the focus tends to shift to the service provider and what it can do rather than on the recipient and what it needs. Ultimately, we think that the service model will lead to insights that guide the design and implementation of mechanisms that are truly beneficial to end users.

Previous Work on Assistance and Service

There is a long tradition within AI focused on developing artificial assistants that provide help to users. Significant work has taken place just in the area of adaptive user interfaces (Langley, 1999; Webb, 1998). In previous work, Iba explored several approaches to modeling user behavior and exploiting the predictive power of the learned models (Iba & Gervasio, 1999; Gervasio, Iba & Langley, 1999).

Although our previous work and that of others successfully demonstrated an ability to correctly anticipate user actions and preferences, it was never clear whether the end user was better off with the adapted system than without it. Possibly more problematic still, it seems that the adaptive modeling components were selected and developed opportunistically rather than strategically. That is, facets of a problem were selected and assisted based on the ability to anticipate the user rather than on alleviating the user's greatest problems. The intention was always to help the user, but the effort overlooked the problem of determining what help actually is and identifying what action would provide the most help.

Perhaps this oversight is not surprising. Computer scientists are not typically trained as social scientists or psychological counselors. But on further reflection, even social scientists and psychologists do not seem to have a handle on the problem. The most relevant work on the nature of service is found in Management Science, but those studies focus on quantifying the economic value of service received (Heskett, Sasser & Schlesinger, 1997).

A Simulator for Studying Service

In this section, we describe our development efforts aimed at providing a suitable testbed for studying service. First we describe the history of EDEN, the world simulator with which we started. Then we proceed to describe the characteristics of a suitable testbed for service, and then summarize our extensions culminating in MÆDEN, a multi-agent simulator for studying service.

Background

In order to address our questions about assistants and the nature of service, we needed a testbed. We started with the EDEN simulator, initially developed in Poplog in 1992 (Perkins, Paine & Chattoe). The environment consisted of a series of problems, or worlds, where an autonomous agent tried to locate and consume a bit of food before running out of energy. The agent could sense its immediate surroundings, move about, and pick up and use tools it found in the world. Obstacles included impenetrable walls, doors that could be opened with keys, etc. Some configurations of obstacles required sophisticated reasoning skills involving dependencies between tools and action sequences. In 1994, Glenn Iba implemented a rational reconstruction of EDEN in Common Lisp. That reconstruction served as the basis for our recent extensions.

The originators of EDEN intended to stimulate the study of embodied agents within a reasonably constrained world. The primary goals were to ignore low-level sensory and effector issues, while requiring the agent to interact with an environment that was not directly under the agent's control. The EDEN environment managed the interaction between a given agent and a simulated grid-world. The design of an agent was left entirely up to a researcher although it had to be implemented in Poplog. Test-worlds of varying complexity were supplied and others could be created via configuration files. Based on the agent's location, the simulator provided the appropriate sensory information. The agent would attempt actions and the simulator would resolve the consequences of those actions based on the state of the world. For example, if the agent moved forward, the simulator would update the agent's location in the world and would provide sensory information reflecting the agent's new position. However, if an obstacle was in the way, the simulator would maintain the agent's existing position; it would be the responsibility of the agent to note that the action was unsuccessful.

Simulator Requirements

Since our goal is to explore the nature of service, we want to identify the characteristics of a testbed that facilitate this study. A suitable testbed should support tasks with variable difficulty, resource constraints on problem solving, and some type of currency for reward and exchange. Since we know that service takes place in the context of a relationship, we also need to support multiple agents and substantial interactions between them. Finally, our testbed must also be instrumented so that we can log and subsequently analyze the behavior of our service providers under different experimental conditions.

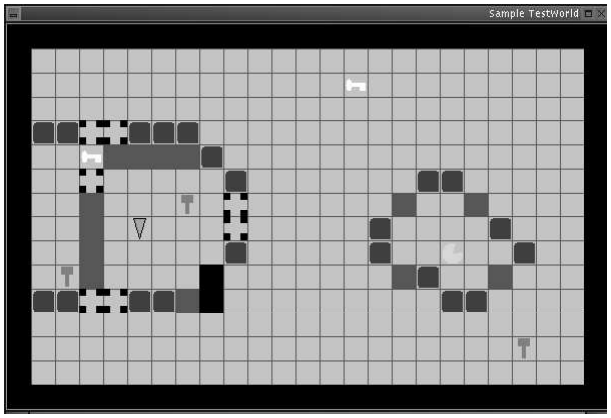

Figure 1: A screen shot of MÆDEN's display with a single triangular agent (on the left) facing south and surrounded by obstacles and the circular food (on the right) also surrounded by obstacles.

MÆDEN: A Service Testbed

Although the EDEN simulator provided most of the basic requirements, it did not support the most important features for addressing service. Thus, we implemented MÆDEN, an extended environment that includes support for multiple agents, communication between agents, and detailed logging.

The most significant change that we made to the simulator was support for multiple agents to simultaneously interact with the world and with each other. The simulator had to provide individualized sensory data to each of the agents, resolve the consequences of attempted actions, and update the world state accordingly. Difficulties arose from these changes, however. The simulator had to deal with conflicts arising when two agents tried to move into the same space, or both attempted to grab the same tool in the same time step.

In our Common Lisp implementation of MÆDEN, agents are part of the simulator process itself. Before a simulation is run, the agent files are loaded by the simulator and integrated directly into its code. The biggest disadvantage is that agents must be implemented in the same language as the simulator. However, we are currently developing the next version of the simulator using a client/server model where the control systems for the multiple agents will interact with a simulator server over common networks. The simulation engine is now written in Java and agent controllers may be implemented in any language that supports the socket interface. Thus, researchers can easily run the Java simulator on a variety of platforms, they can experiment with different agent architectures in their favorite language, and the agents can interact with the MÆDEN server from any machine on the network. Figure 1 shows a screen-shot of a simple world; on the left side, a single agent faces south and must find a way past the obstacles surrounding it, then move toward the right side and get past the obstacles surrounding the goal food.

Note, however, the experiments we describe in this paper reflect results from the Common Lisp version of the simulator.

Although primitive communication can take place through mutual observation of behavior, we wanted to support richer message passing between agents. Therefore, MÆDEN supports speech acts and auditory senses. Messages travel a limited distance and, if heard, include approximate direction and distance information. The message can be heard by any agents in a radius of up to five times the retinal view. Agents have a choice between talking or shouting; talking uses less energy but travels a smaller distance.

The content of messages is left up to the agents and their designers. When sending messages, agents execute a talk or shout action with a message argument; the simulator distributes the message to other agents as appropriate. If another agent is within hearing distance, the message is converted into a packet of information that identifies whether it is a talk or a shout, the ID of the sender, the direction that it originated from, and the content itself. The message packet becomes available to that other agent as sensory data.

The original EDEN simulator provided very little support for evaluating a specific run through a world; the only quantitative measure of success was the remaining energy at the end of a run. We created an extensive logging facility that allows us to analyze the eventual benefit of agents' actions. The logging facility has five levels of detail, and at the highest level effectively logs every change made in the simulation and the resulting effect on each agent. We are especially interested in logging all communication and transactions between agents, but by using scripts, we can easily extract any logged data for analysis. From such analysis, we can determine the effect, if any, that agents have upon each other while navigating the worlds.

We made a number of other minor changes to the simulator. Most notably, we added support for agents to exchange assets for service. In response to a request for assistance, the simulator manages the transfer between helper and the recipient. The currency of exchange is identical to energy but is managed in a separate account. The other extensions are detailed in the MÆDEN documentation. We anticipate releasing the software during the Spring of 2005.[1]

Agents and Experiments

Design of Agents and Service

While the MÆDEN simulator provided an environment to study service, we needed some base-line agents to start addressing service directly. Furthermore, we needed two types of agents: a main agent that receives service and a helper agent that provides it. We started with a basic agent that had very limited capabilities. Gradually the skills increased in both number and effectiveness. The different capabilities included: wall-following, obstacle avoiding, opening doors, digging through walls, exploring, mental map-making, as well as leading and following other agents.

[1]We are making the MÆDEN source code, sample agent controllers, and miscellaneous documentation available at: `http://www.westmont.edu/~iba/maeden/`.

By design, the agents existed as a collection of independent skills. Each capability was a distinct entity that could be included or withheld from agents. This gave rise to a family of agents based on different configurations of skill-sets. Agents select actions to perform based on a currently active skill. Complex behaviors can arise by maintaining a stack of active skills. A currently active skill could be interrupted by environmental conditions and another skill could be pushed onto the stack and executed. If at some point the goal of the currently active skill becomes satisfied, it is popped off the stack and the previously interrupted skill is reactivated.

The default *main agent* was the one receiving the help, or service. It was given minimal capabilities, consisting of only three primitive skills. The agent could sense where the food was and move in that direction, ask for help when against an obstacle, and follow a helper agent to food. The helpers were given considerably more skills, and for our experiments the default service provider possessed all of the capabilities in the skills library.

For these primitive agents, service takes place when the main agent requests help and a helper is within earshot. In this event, a transfer of energy is made from the main agent to the helper. We managed two separate accounts, a true energy account used to support actions and a payment account for getting help. We allowed deficit spending in the second account but not the first. By default, helpers charge a flat-rate fee for services and allocate a portion to providing help. If the allocated portion is adequate to accomplish the task, the helper will proceed to find the food and lead the main agent to the food's location. However, we have begun exploring and implementing alternative payment schemes that allow greater flexibility, more extensive help capabilities, and richer interaction between the helper and receiver.

Empirical Results

Having implemented the extended environment, MÆDEN, we ran several experiments to demonstrate the potential of the testbed for studying service. For these tests, we utilized the two types of agents described above. The main agent has only a minimal set of capabilities: it could either follow its nose, ask for help when encountering an obstacle, or follow a helper agent. The helper agents could exercise any of the skills we created as the situation warranted.

For these first experiments, we were primarily interested in two dependent measures. The first was simply whether the main agent survived. Surviving meant finding and eating the food before the agent's energy was entirely depleted. For a particular experimental condition, we report the survival rate as the fraction of worlds completed over repeated runs. We also considered efficiency as a second dependent measure. We measured efficiency as the main agent's net remaining energy after subtracting any payments for help that may have been made. To date, we have only considered the main agent's efficiency.

Evaluating Survivability. As a base-case, we hypothesized that an agent with many skills could help an unskilled agent solve otherwise impossible tasks. But we also expected that the overall benefit received by the main agent

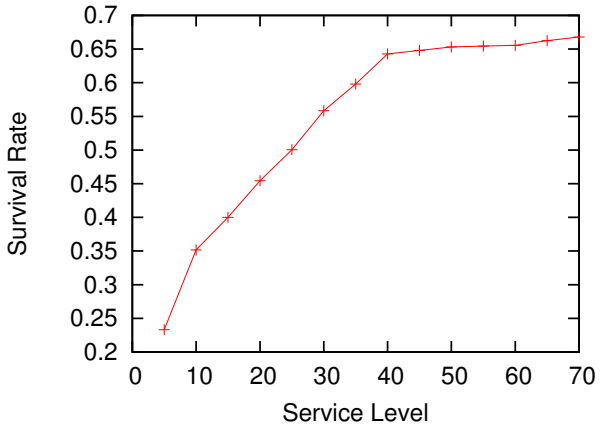

Figure 2: Average survival rate as a function of service level dedicated by the helper agent.

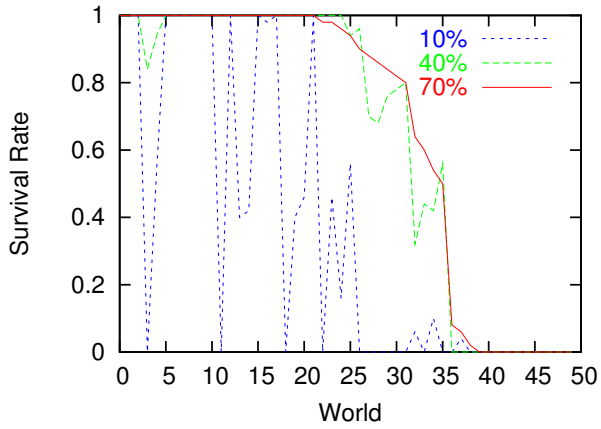

Figure 3: Average survival rate for three service levels, for each world as a function of increasing world difficulty.

would be proportional to the amount of service provided. We tried to control the level of service by having the helper agents allocate some fraction of the payment toward the main agent's requests; the helper would pocket the remainder. We say that a higher level of service is provided when a larger fraction is allocated to helping. Figure 2 shows the results from this experiment. Each data point represents the average success rate over 50 runs for each of the 50 worlds at one particular level of service. We varied the level of service from a low of 5% to a high of 70%. Unsurprisingly, the results supported our initial hypothesis.

We also wondered if there were situations where asking for help would actually be a hindrance. In terms of survivability, this never seemed to be the case, but when considering efficiency (discussed below) we did see this occur in a few cases.

Measuring the Benefit of Service. Our first experiment demonstrated that allocating more of the payment to actual help was more beneficial in terms of survival. But we suspected that differences in problem difficulty would influence the gain in benefit. Specifically, we hypothesized that while the highest levels of service would give the best survival rates, for the worlds with moderate difficulty, the main agent was paying too much. Analyzing our data by individual world, we see this concern was well placed. Figure 3 plots three curves for three levels of service, 10%, 40%, and 70%. Each point represents the average survival rate for the given world over 50 runs at the corresponding service level. We see from the figure that the highest service level (70%) seems to provide the best survival rate. However, the 40% service level nearly approximates the higher level, and in many of the worlds, the 10% level equals the higher ones.

Part of this effect is due to simple worlds where the main agent never requests help. But we can conclude that in many of these cases, the 30% difference in payment (or 60% in a few cases) is an unnecessary loss for the main agent. In other words, the helper can often provide adequate help using only 40% of the payment and thus, the main agent is in some sense over-paying for services. Wanting to understand this issue further, we turned to an efficiency measure for these conditions.

Weighing the Value of Service. Our initial analysis of the data suggested as many new questions as were answered. It seemed necessary to consider efficiency in order to understand which interactions were providing service and which were not. Figure 4 shows data from the same experiments as before, but plots our measure of efficiency at the different levels of service. As before, data points represent the average efficiency for a given world repeated 50 times.

We draw two conclusions from looking at efficiency. First, the comparison between the high and low service levels suggests that the high service level is indeed delivering efficiency advantages. Although the two conditions are the same for a number of the worlds, when they differ the 70% service level almost always provides better efficiency. In the few worlds where the low service level shows a higher efficiency rating (worlds 36-38), we see in Figure 3, a slightly higher survival rate for the 70% service level. This highlights the fact that survival and efficiency are two different objective criteria that must be balanced. We suspect that as we continue to study service, we will find several other factors contributing to the overall value of service.

Determining the Cost of Service. Perhaps a third conclusion may be drawn from the results shown in Figure 4. Even though obtaining a high level of service is better than getting poor service, the resulting efficiency gets worse as the problems get more difficult – and indeed is negative much of the time. This observation motivated us to identify those costs where it is and is not worth obtaining service. Note that we do not have a way to combine the net energy efficiency with the survival outcome. Nevertheless, at what point should an agent simply give up?

In most situations, we can imagine a theoretical reward level that would make purchasing assistance at a fixed cost an overall advantage (ignoring survival). Figure 5 shows a

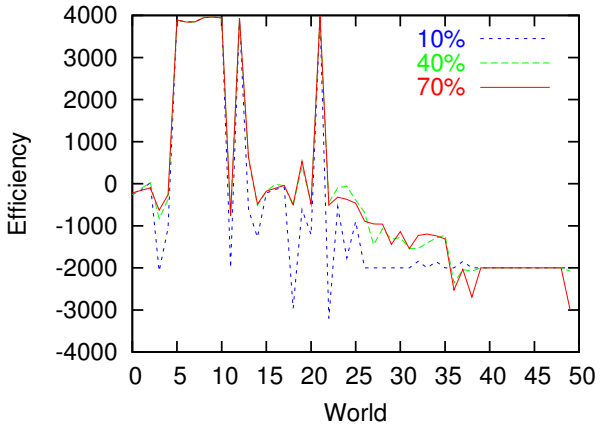

Figure 4: Plotting efficiency (net energy gain or loss) at three levels of service for each of the 50 worlds.

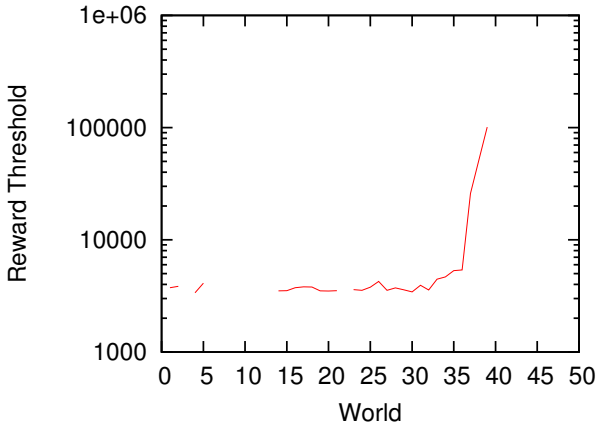

Figure 5: Computed reward threshold as a function of world. The threshold identifies the point above which asking for help in the given world is beneficial.

graph of the necessary reward above which the agent sees a net gain through asking for help. The reward threshold is undefined for a world where the main agent can solve it without help or where the helper itself cannot accomplish the task.

We find it interesting that the reward threshold is nearly constant for the first 30 worlds. Note that the value of the constant is tied to the default reward that we used in our experiments. But comparing Figure 5 to Figure 3 makes the constant threshold somewhat surprising for worlds 20-30. In Figure 3, we see the survival rate steadily declines over those worlds but the threshold remains constant. Only as the survival rate continues to fall radically for worlds above 30 do we see the reward threshold increase significantly. This is another of several questions that we continue to investigate.

Other Explorations. Based on our results described above, we realized that we needed to implement alternative schemes to capture payment for service. Rather than require a flat rate for help, we wanted to give the main agent more flexibility in hiring services.

Thus, we implemented a payment scheme where the main agent has much more control over how much energy is given to helpers. When first seeking assistance, the main agent pays a small amount for the service. Regardless of the helper's internal profit structure (i.e., what fraction of the payment gets allocated to help), there is some chance that it will assist the main agent. If help is not delivered after a period of time, the main agent will offer the helper twice the original amount. This waiting and doubling continues until either the helper leads the agent to the solution or the agent runs out of energy.

We hypothesized that this variable payment scheme would improve efficiency since help is only done when help is needed and less energy is wasted on large payments for simple services. The results showed improved efficiency for some worlds, but no change on many of the others. In the latter cases, it appears that both payment schemes ended up extracting the maximum amount from the receiver because these worlds were so difficult.

But in the cases where efficiency improved, we find an interesting force at work. The main agent continued investing resources in increasing amounts until it received help. We noticed an apparent interaction between the overall benefit received and the recipient's commitment (in terms of expending resources) toward getting help. We think this points to an interesting insight into obtaining service in the real world and we intend to explore this question more carefully in ongoing work.

Ongoing and Future Work

Our planned activities fall into three categories: developing the MÆDEN simulator, developing more advanced agents, and conducting additional experiments with these agents.

Although we are pleased with the progress in MÆDEN and the current functionality, we are making several significant changes in order to provide the simulator to any interested researchers. First, we are now reimplementing the main engine in Java so that MÆDEN can be used on a variety of platforms. This transformation is largely completed and promises to yield additional benefits in terms of improved design choices. Second, we are designing a generic network interface between agents and the simulator. This will again improve flexibility as existing agents can easily be extended to work with the new framework, but new agents may be written in any language that supports a network interface. Finally, we are developing a graphical interface to display simulated worlds as well as a particular agent's perspective view of its surroundings.

In the area of agent designs, we intend to test our current approach more thoroughly as well as explore alternatives. For our first steps, we will extend the inter-agent language that we used for our initial agents. In conjunction with those extensions, we want to implement alternative strategies for negotiating payment for services between agents. We also will be exploring the consequences of having more than one helper. We tested this capability from a software develop-

ment perspective but have not run experiments with a specific hypothesis.

We will also be implementing learning methods that enable helpers to acquire models of the recipient of service. These models would include goals, habits, and voids in skills. We want to evaluate the relative benefits of explicitly communicating needs between agents as well as of observing behavior as a means to acquire information about the main agent's goals.

Finally, there are already numerous experiments that we did not have time to conduct during the research. We anticipate the extensions we have outlined will generate even more. Perhaps the most important outstanding experiment will involve varying the sets of skills in the respective agents. All of our experiments to date have addressed an unskilled main agent seeking help from a highly skilled helper. What will happen when both agents are moderately skilled? We hypothesize that we will find measurable benefit from service even in cases where the helper is less skilled than the main agent. Furthermore, we think that it may be the case that an agent with a helper can solve certain problems where the agent alone would be unsuccessful, even if it had the combined skill-set of the two agents.

Conclusions

We set out to study service with the goal of discovering insights into the nature of service. We are convinced that service is difficult to understand but crucial to building Artificial Intelligence systems that are truly helpful to human users. As a step toward understanding service, we sought a testbed that could be readily used to evaluate different approaches to modeling and delivering service between agents. Not finding a suitable simulator with the necessary characteristics, we opted to build our own environment, MÆDEN, an extension of a single-agent simulated environment. Our extensions centered around supporting multiple agents, communication between agents, and detailed logging utilities.

Partly as a test of the new environment, we developed a set of agents and ran several experiments involving the exchange of service between agents. We implemented a main agent with limited capabilities and a family of helper agents with extensive skill sets. The specific design of the agents was less important than demonstrating the richness of the environment and our approach to modeling service. In our experiments, we varied problem difficulty and service level while measuring survival rate and efficiency. Although our results are based on initial tests, they show significant promise for the general approach by way of the questions and hypotheses they stimulated.

Currently, we are continuing to improve the MÆDEN environment and anticipate releasing it to the community during the Spring of 2005. We also have well defined plans for alternate agent designs and new experiments to evaluate their behavior. Although understanding the nature of service is a daunting challenge, our results to date encourage us that we are following a promising path.

References

Gervasio, M. T., Iba, W. & Langley, P. (1999). Learning user evaluation functions for adaptive scheduling assistance. In *Proceedings of the Sixteenth International Conference on Machine Learning* (pp. 152-161). Bled, Slovenia: Morgan Kaufmann.

Heskett, J. L., Sasser, W. E., & Schlesinger, L. A. (1997). *The Service Profit Chain: How leading companies link profit and growth to loyalty, satisfaction, and value.* The Free Press: New York.

Iba, W. & Burwell, N. (in press). Building a testbed for studying service. In *Proceedings of the Stanford Spring Symposium on Persistent Assistants: Living and Working with AI.* Palo Alto, CA: AAAI Press.

Iba, W. & Gervasio, M. (1999). Adapting to user preferences in crisis response. In *Proceedings of the International Conference on Intelligent User Interfaces.* Redondo Beach, CA: ACM Press.

Langley, P. (1999). User Modeling in Adaptive Interfaces. In *Proceedings of the Seventh International Conference on User Modeling*, (357-370). Banff, Canada: SpringWien.

Perkins, S., Paine, J. & Chattoe, E. (1992). EDEN: Poplog-based AI Microworld. Areas:Testbeds:Eden, CMU-AI Repository.

Webb, G. (1998). Special issue on machine learning for user modeling. *User Modeling and User-Adapted Interaction*, 8. Kluwer.

Acknowledgments

We gratefully acknowledge the support of Westmont's Provost, Shirley Mullen. A Westmont Faculty Development grant to the first author supported the second author and a second student assistant, Chris Phillips. We especially thank Glenn Iba for the Common Lisp version of EDEN from which we started, and Chris Phillips for his assistance with implementing the MÆDEN simulator. We also thank Kim Kihlstrom for commenting on an earlier draft. We appreciate the helpful suggestions and criticisms from several anonymous reviewers; they identified a number of ways to significantly improve the paper, but of course any remaining problems are our own fault.