# Building a Testbed for Studying Service

**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

Performing a task for a user is one part of assistance. As a prior step, a helper must identify which of a possible number of tasks should be assisted. We intend to study the requirements that enable service to be rendered and understand the factors that impact the delivery of assistance. Toward this end, we extended a simulated environment to use as our testbed for developing and evaluating approaches to service. Our new testbed, MÆDEN, supports multiple agents, communication between agents, and records detailed logs of agents' behavior. We used the testbed in preliminary studies of servant and recipient behavior and interaction. Results hint at the usefulness of the testbed.

## Introduction

Providing assistance to a computer user is an exciting and rewarding prospect, both for Artificial Intelligence researchers as well as for end users. Much work is focused on advancing the ability of artificial agents to perform tasks on the behalf of human users or to suggest to human users how to proceed in particular situations. If such work represents the engine that is progressively moving us forward, we want to raise the questions, where are we trying to go and how do we steer to get there? More specifically, we want to learn how to most effectively target assistance when we have a variety of tasks that could be aided. For a user to receive assistance, some of the questions associated with this include what are the user's goals, what are the user's capabilities and limitations, and what are the user's needs. This view of the preliminary questions suggests *service* as a model for this endeavor. However, an exploration into service reveals how little we know about its nature, its requirements and its benefits.

Thus, we want an experimental testbed as much for studying how to deliver service as for discovering the nature of service and addressing these other questions. Toward this end, we implemented MÆDEN, a simulated environment that satisfies a set of requirements we analyzed as necessary to study these questions.

Having assistants persist over extended time periods raises a number of interesting questions. However, the requirements for a testbed remain the same when those requirements are selected with an eye toward service in the first place. As long as the testbed does not preclude or mask necessary characteristics, the burden is on the designer of the agent to support things like trust, transparency, initiative and perseverance. Although we intend to address these issues in ongoing work, in this paper, we focus on the development of an appropriate testbed.

We begin the paper by viewing assistance in the light of competition and cooperation and then consider *service* as a useful model for studying the delivery of assistance. Next, we briefly consider previous work on providing service and argue the need for a new model such as service. Then we turn to our search for and development of a testbed that would prove to be suitable for studying the questions related to service. We conclude with a summary of preliminary results, some initial conclusions from our work to date, and our plans for continuing work.



## Service as a Model of Assistance

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. Certainly, an assistant may have self-interests and legitimately pursue those interests in the course of providing assistance. However, typical adversarial models where each agent seeks to maximize its own utility at the expense or the interests of other agents do not yield immediate insights into the nature of assistance nor do they provide guidance on how to deliver assistance.

A cooperative model of behavior would seem to be more promising, but it also turns out to be problematic for our purposes. Cooperative systems tend to have a global objective (even if multi-valued) and all of the cooperating agents try to optimize their behavior with respect to this objective. However, the ideal assistant attempts to achieve the objective of the recipient of assistance, even at the expense of its own interests. We could attempt to finesse this problem by giving the assistant the objective function of the agent receiving help, but this would raise thorny questions of identity that we would rather avoid.

Thus, we have a situation where an ideal assistant is not trying to maximize its own ends at the expense of others, nor merely trying to maximize some collective good, but is

willing to sacrifice its own ends for the benefit of another. Although it may be possible to capture each situation by appropriately weighting the objectives of respective agents, we think there is value in approaching the problem of assistance using service as the model.

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 focuses attention on the benefit accrued to the service recipient. Although assistance is closely related, the focus tends to shift to the actions of the service provider rather than those of the recipient. 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 Assistants and Service

There is a long tradition within Artificial Intelligence 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, the first author explored several approaches to modeling user behavior and adapting a system's behavior to exploit the predictive power of the learned models (Iba & Gervasio, 1999; Gervasio, Iba & Langley, 1999).

While that work 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 in the first place. Possibly more problematic still, it seems that the adaptive modeling components were selected and developed opportunistically rather than strategically. The intention was always to help the user, but the effort overlooked the problem of 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 is found in Management Science but focuses on quantifying the value of service received (Heskett, Sasser & Schlesinger, 1997).

Having been trained as computer scientists, we approach the study of service with a strategy of developing agents that provide help and agents that need help and placing them in a controllable environment. Working with humans as the recipients of service provides the ultimate test. However, it also introduces a number of problematic issues. First, experiments must run at the speed of the human interaction. Second, repeatable tests with alternative conditions of help are impossible and the scientist must instead rely on large groups of users. Finally, we might like to control the reactions of a user to given offers of help in order to evaluate the consequences of the recipient's behavior; with humans, this is again problematic. For all these reasons, we chose to study artificial helper and recipient agents within a simulated environment.

## A Simulator for Studying Service

### Background

In order to address our questions about assistants and the nature of service, we needed a testbed. Although a number of options were available, we started with the EDEN simulator (Perkins, Paine & Chattoe, 1992). Initially developed in Poplog in 1992, it was later reimplemented in a client-server architecture using Java and Poplog where the simulation and user interface was coded as a Java applet client that connected to a Poplog server running the agent controller. The environment consists of a series of problems where an autonomous agent tries to locate and consume a bit of food before running out of energy. The agent can sense its immediate surroundings, move about, and pick up and use tools found in the world. Obstacles include impenetrable walls, doors that can be opened with keys, etc. Some of the sample worlds consist of problems involving a sequence of obstacles and tool interdependencies that require sophisticated reasoning skills to solve.

The originators of EDEN intended to stimulate the study of embodied agents within a reasonably constrained world. The primary goals were to by-pass 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. In principle, the design of an agent was left to a researcher. Test-worlds of varying complexity were provided and new ones could be created via configuration files. The simulator reported to the agent its initial situation as specified in the configuration file. The agent would attempt certain 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 track the changes to 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 current position and it would be the responsibility of the agent to note that the action was unsuccessful.

### Desirable characteristics

Since our goal is to explore the nature of service, we want to identify the characteristics of testbeds that facilitate this study. A suitable testbed should support tasks with variable difficulty and resource constraints on problem solving. 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. We also expect some type of reward, and additionally may want some type of currency for exchange between agents. Our testbed must also be instrumented so that we can log and subsequently analyze the behavior of our service providers under different experimental conditions. We discuss each of these briefly.

A useful testbed must clearly support multiple tasks of varying difficulty but the difficulty may be varied in a number of ways. A problem's difficulty can be measured in solution length (either length of time or number of steps), solution path frequency (how many different solution paths there

are), and solution path specificity (the proportion of solution paths to non-solution paths). We can manipulate these factors either by changing the layout of the artificial world or by altering the availability of resources to the agents.

Perhaps the most important feature of the testbed for the purpose of studying service is that it support relationships between multiple agents. If one agent is to assist another, the testbed must support interaction between agents and the environment as well as between two agents. Although agents may require sophisticated internal models of their relationships with other agents, in order to provide service the environment itself need only support some form of communication between agents. In bees, we have an example where movement alone is sufficient for some communication, but we expect a useful testbed to support the exchange of both messages and objects. Then, capturing the complexities of relationship is left up to the agent designer. Finally, we would like to have the testbed support relationships between agents whether they are artificial or human. In other words, there should be an interface to the testbed's environment that can be meaningfully used by humans.

It should be no surprise that we expect our testbed to reward agents when they complete a problem. However, we also want to think of reward, or potential reward, in terms of currency and allow agents to exchange this currency. The purposes of such exchange could quite naturally involve fee-for-service, but other exchanges could be supported as well. Once again, the testbed need only support the exchange itself and need not be concerned with the modeling implications of different types of exchanges.

Another critical requirement of a testbed for studying service (yet most frequently missing), is adequate instrumentation. In order to evaluate what factors or behaviors led to favorable or unfavorable outcomes for an agent seeking assistance, we must be able to analyze in detail what the agent and the helpers actually did. This requirement is especially necessary if humans are interacting with artificial agents or with other humans. In general, one should be able to completely reconstruct the situation at which point a particular action was taken.

## An extended simulator

The EDEN simulator does not support many of the desirable characteristics for studying service. Thus, we implemented MÆDEN (Multi-agent EDEN) by extending EDEN to support multiple agents, communication between agents, and detailed logging capabilities. We started working from a rational reconstruction coded by Glenn Iba in 1994 and implemented these extensions. More recently, we completed a reimplementation of the simulation engine in Java with a client/server model.

Enabling the MÆDEN environment to handle multiple agents was the most important, but also the most difficult of the changes. 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. In our first version of MÆDEN, agents were part of the simulator process itself. The biggest disadvantage is that agent control programs must be implemented in the same language as the simulator – Common Lisp. However, we are currently reimplementing the system in Java with a client/server architecture that separates the simulation engine from the controller code and the two communicate with each other through sockets. At this stage, agents may be written in any language that supports a socket interface. Such agents may be run on any networked machine that can reach an instance of the Java simulation engine. This architecture is the opposite of the approach taken in EDEN II where the server is controlling the (single) agent. Our design explicitly supports multiple agents connecting to the same world. Figure 1 shows a screen shot of a single agent in a sample test world.

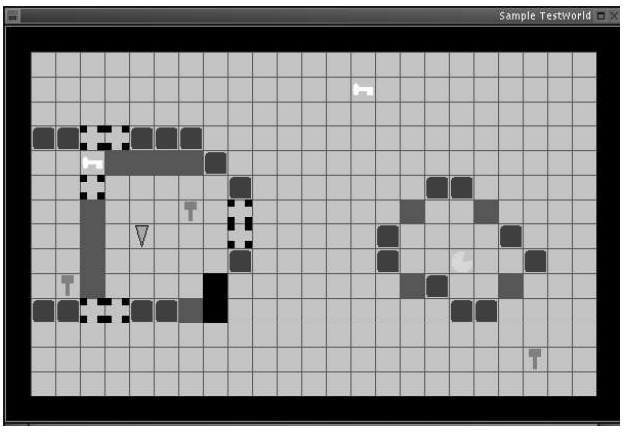


Figure 1: A screen shot of MÆDEN's display showing (on the left) a triangular agent facing south surrounded by obstacles and (on the right) the circular food also surrounded by obstacles.

Although primitive communication can take place through mutual observation of behavior, we wanted to support a more detailed communication through message passing between agents. MÆDEN supports both speech acts and auditory senses. Messages travel a limited distance and, if heard, include coarse direction and distance information. An agent may choose to speak or shout a message, influencing the distance the message will carry. The content of speech acts is left up to the agents and their designers.

The original EDEN simulator provided very little support for evaluating an agent attempting to solve a world. The only quantitative measure of success was the amount of energy used during a run. But even this had to be carefully extracted since the initial energy level could vary from one problem to the next and the simulator reported the remaining level. Although energy is still a primary measure in MÆDEN, we created an extensive logging facility that allows us to return to a situation later and analyze what a given agent was doing. From this, we hope to assess the eventual benefit of agents' actions. Naturally, it is even more helpful if logs from the agents' controller processes can be synchronized with the simulation logs. However, even without such views of agent reasoning, we can infer much from which actions agents perform and what communication events take place.

We implemented a number of other minor extensions to EDEN. 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. Although currency exchange can play an important role in the delivery of service, having this capability available to agents broadens MÆDEN's usefulness to the study of other models of interaction. That is, the exchange of currency does not entail a particular model of multi-agent interaction but rather supports a broader range of models than without it. The other extensions, such as variable action costs and bread-crumb markers among others, are detailed in the MÆDEN documentation. We anticipate making the simulator publicly available during the Spring of 2005 (see http://www.westmont.edu/~iba/maeden/).

## Preliminary Experiments and Results

### Setup and framework

Having implemented a first version of the MÆDEN environment, we developed some simple agents and conducted several experiments intended to demonstrate the potential of the testbed. For these initial tests, we were mainly interested in three dependent measures. The first, survivability, was simply the number of problems that could be solved by a main agent (the one receiving help) for any given condition of assistance. The second measure was the amount of time and energy consumed by the main agent in the course of a problem attempt. And the third was the amount of currency expended in obtaining assistance. We sometimes combine these two into net-resource consumption. We think of the *value* of the provided service in inverse relation to the net-resources consumed.

We designed our initial agents as a configurable collection of capabilities, such as wall-following, opening doors, building maps, etc. We did not choose this architecture because we thought it was particularly ideal for building assistants but simply because it allowed us to define a range of agent types with varying capabilities using the same structure. So far, our experiments have tested only the extremes of the spectrum. Our main agent, the one needing assistance, has the ability to move toward the food and ask for help when needed; the default helper agent can perform all of the skills and respond to requests for help.

In our framework, help takes the form of a request, an exchange of currency, and a set of activities by a helper agent. If the helper is successful in finding the food, it returns and guides the main agent to the food. Note, the support for currency exchange in the simulator and our use of it in simulating service does not entail a competitive model of interaction (although such models or others may yield insights).

### Test summaries

As a first demonstration of the testbed in use, we hypothesized that an unskilled agent could receive beneficial assistance when facing tasks that would otherwise be impossible. When paired with a helper that had the necessary skills, we expected the unskilled agent to successfully complete the impossible tasks. Unsurprisingly, the results supported this hypothesis in terms of both survivability and net-resource consumption.

We also demonstrated that the value of assistance increased when helpers dedicated more of their resources to helping the main agent. In other words, as the cost of help increased, the value to the main agent decreased. Again, this is an unsurprising result that tells us nothing about service (other than that our agents and testbed are functioning as one would hope). However, in the context of these runs while also varying the main agent's strategy of asking for help, we noticed an interaction between the main agent's commitment to requesting help and the overall benefits received. A conservative agent would sometimes solve a problem without help just through random chance but more often miss out on opportunities to receive help, whereas a liberal agent would sometime pay for help it did not need but would accomplish the tasks more frequently. This has an intuitive appeal to our understanding of service and we think it may point to the importance of the role played by the recipient of assistance in service contexts. We intend to explore this more deeply in future work.

In a related vein, we considered the relationship between the task difficulty and the value of obtaining assistance. We predicted that as tasks became more difficult, the value derived from service would become more significant. As it turns out, we were not able to support this prediction. In fact, for the most difficult tasks, the value tends to decrease. These results appear to arise because we have not combined the measure of completing a task with the net-resource consumption. Looking at the completion rate, we see that the main agent is surviving the most difficult worlds when assisted by a helper even though the value (as given by resource consumption) is going down. Further details on all of these experiments may be found elsewhere (Iba & Burwell, in press).

## Conclusions and Next Steps

We can conclude from our initial experiments with MÆDEN that there is much to be learned about service and that artificial agents in a simulated environment can begin to shed light on some of the relevant questions. Those questions include how can we measure the value of service and what factors influence its delivery. The few preliminary tests run so far suggest a number of additional tests that we expect will yield insight into the nature of service. Although these results are not specific to assistance being delivered over prolonged periods, we anticipate that the lessons learned will inform such contexts.

As preliminary work, we have many tasks before us for ongoing work. We are currently finishing the Java version of MÆDEN and anticipate a first release in the Spring of 2005. Improvements on our initial study will include varying the capabilities of the agents and helpers and implementing selective help-acquisition schemes. Our agent design lends itself nicely to different subsets of skills and consequently to different levels of expertise. We predict that the main agent, even with more skills, can still receive benefit even from helpers that are less skilled. We also intend to implement

alternative help-request schemes that vary the willingness to ask for help and how much will be paid.

It seems obvious that helping users in settings over extended time periods demands adaptation to characteristics of the user. We certainly intend to explore agent types that can build models of each other. The implementation of MÆDEN was a necessary precursor to that effort but we hope to address it as the simulator stabilizes. And in that context we hope to gain greater insight into the nature of service and to formalize its elements.

## 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). Studying service: An exploration of the costs and benefits of assistance. In *Proceedings of the 18th International FLAIRS Conference.* Clearwater, FL: 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 Cailin Andruss and Annie Evans for commenting on earlier drafts. 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.