The Agent Repository -
Supporting Collaborative Contextualized Learning
with a Medium for Indirect Communication

by
Martin Frank Rausch

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirement for the degree of
Master of Science
Department of Computer Science

1996

This thesis for the Master of Science degree by

Martin Frank Rausch

has been approved for the

Department of Computer Science

by

Gerhard Fischer

Alexander Repenning

Clayton Lewis

Date : _____

Abstract

Rausch,, Martin Frank (M.S., Computer Science)
The Agent Repository -
Supporting Collaborative Contextualized Learning
with a Medium for Indirect Communication
Thesis directed by Associate Professor Alexander Repenning

When building a simulation with Agentsheets and Visual AgenTalk designers are in a dialog with a model they construct. The dialog could be enriched if in addition to the designer's own constructions the simulation would also include foreign constructions.

Sharing constructions is one way of facilitating Distributed Constructionism, a framework that was proposed by Mitchel Resnick. The ideas of Constructionism as introduced by Seymour Papert are combined with new interactive media to promote collaboration and co-construction in a distributed manner.

To support Distributed Constructionism for educational purposes existing design environments like Agentsheets must be extended to support discussion, collaboration and sharing of constructions. With AgentShare and the Agent Repository an extension to Agentsheets is introduced that reduces the effort necessary to participate in a distributed design activity and enables users to benefit from the distributed knowledge base that a community of users represents.

As Ian, one of our first test users at a middle school in Boulder, puts it: "You could share it with about anyone else who has Agentsheets. You could get with some friends that you meet on the web even, and you could all make a game pretty much, and you would be in different parts of the worlds but you could work on it together." And then Clay: "Like a remote design.."

The study that was performed at a Boulder middle school and in collaboration with a school in Los Angeles suggests that the facets of Distributed Constructionism can not be isolated and used separately. Discussion, direct collaboration and sharing address different aspects and phases of the learning process that is supported.

CONTENT

FIGURES

Introduction

Traditional computer-supported design activity is predominantly an individual effort. The idea of supporting collaboration has existed for quite some time but not until the recent explosion of network infrastructure were the sufficiently powerful technologies available to pursue these ideas. With the advent of the World Wide Web as omnipresent medium design models and frameworks have to be modified to take the new potential into account.

As computers were interconnected and communication mechanisms became increasingly sophisticated a transition towards social cooperative design became possible that changed the way knowledge could be interchanged and reused. My interest is in social computer-supported design where the cooperating parties are not motivated externally. I claim that if we can und erstand what motivates a person to collaborate who has no external motivation, but does it simply because it seems beneficial we can as a consequence support collaboration where it is required more effectively. By eliminating the external forces that might distort our results, such as a successful collaboration that had not taken place had the project leader not given out the objective to cooperate, I try to identify results that are influenced rather by the provided infrastructure.

Performing a study at two middle schools such intrinsically motivated design activities will be studied in an educational context. The connection between learning and design is established in the philosophy of constructionism.

The basic idea of constructionism is to construct knowledge representations while constructing artifacts. The learning takes place

in the context where the knowledge is relevant. Thus it supplies motivation. Breakdowns help to detect misconceptions about the underlying model. Sharing artifacts that other users have created and examining solutions they have found for their problems can induce a new shared understanding of a problem domain.

Problem Statement

With Agentsheets users test their ideas about the world in a simulation environment. Users stand in a dialog with their model. But in a closed model where everything in the model was created by the same person there is little room for unpredictable and unexpected behavior although this kind of behavior characterizes systems in the real world.

In order for the simulation to give the user new insights a connection to other users' simulation must be possible. Introduction of foreign artifacts into the previously closed simulation will result in behavior that was not anticipated and that may require modifications of the model.

An interactive medium for Agentsheets artifacts was developed to connect Agentsheets simulation environments and open them up to foreign artifacts. This medium is called the Agent Repository.

My motivation for making connections between Agentsheets environments is to serve educational purposes.

One of my null-hypotheses about such a medium was that it would help people to **learn** about the programming environment. Looking at other people's agents would help them build an understanding of the model behind a simulation.

From an educational perspective, a medium for sharing parts of a simulation should reduce the necessary effort that teachers have to invest to replace isolated programming efforts with collaborative, contextualized project work. A **project** would help to guide the

students efforts and to give the curricular content relevance in a concrete design situation.

Collaboration within a class has proven to be beneficial for the students. The results that can be achieved by a coordinated group effort satisfy the students more than solitary projects. In a later chapter evidence for this claim will be presented. An open question that will be investigated is whether collaboration outside of the class has **analogous benefits** on the larger collaboration-scale and if the benefits are perceived by the students in a similar manner.

In order to benefit from an interactive medium such as the Agent Repository a **critical mass** of contributors and users must be established. The usefulness of an interactive medium is determined by the community that is connected. Only if the medium is endorsed by a *sufficient fraction* of members of the community will it be able to reach a state of universal access.

The thesis will address these issues,

- learning the programming paradigm of Visual AgenTalk,
- using contexts and integrating foreign environments,
- perception of the advantages of collaboration, and
- the social critical mass

that are related to the indirect collaboration via the World Wide Web that is supported by the Agent Repository.

A Scenario

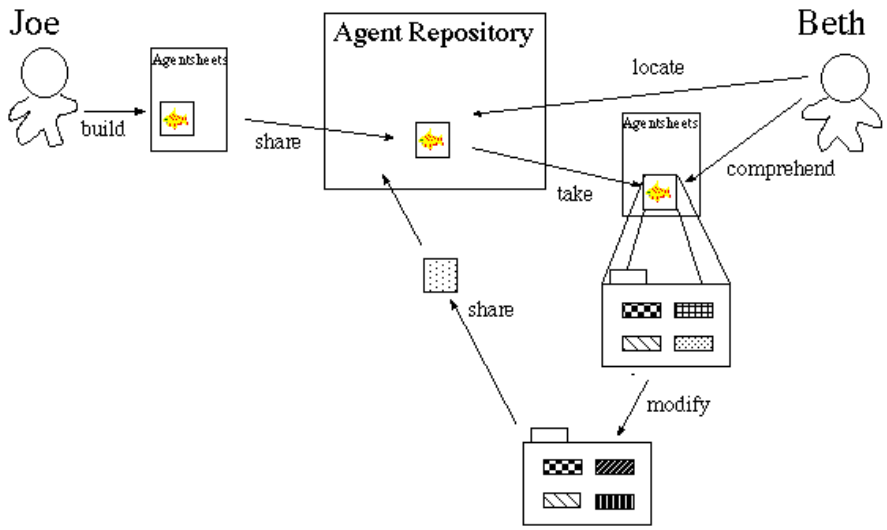

Figure 1: The scenario illustrates how the repository is used. The steps in the design cycle including the repository are (1) build, (2) share, (3) locate, (4) take, (5) comprehend, (6) modify, and (7) share again.

Joe uses Agentsheets to ***build*** *a model of a predator-prey relationship between two species of fish. He seeks to find out effects of changes of the hunting behavior on the relationship between predator and prey population. In simulations he wants to observe implications of changes in his model.*

Figure 2: A fish tank with a couple of 'Yellow Fish' swimming in it.

He starts building two agents representing the two species of fish in his eco-system. He creates a shark for a predator and tuna fish as prey. The shark's behavior is determined by rules. It swims around and eventually becomes hungry and consumes a tuna fish that he spots.

The behavior of the tuna fish includes swimming around and reproduction when meeting another tuna fish.

These behaviors enable Joe to observe the desired predator-prey relationship.

Joe is proud of the model he created and wants to make it publicly available to other Agentsheets users that are interested in similar applications. He decides to ***share*** *his two fish agents. He figures that the two of them need to remain packaged together to preserve the meaning of each of the fish's behavior.*

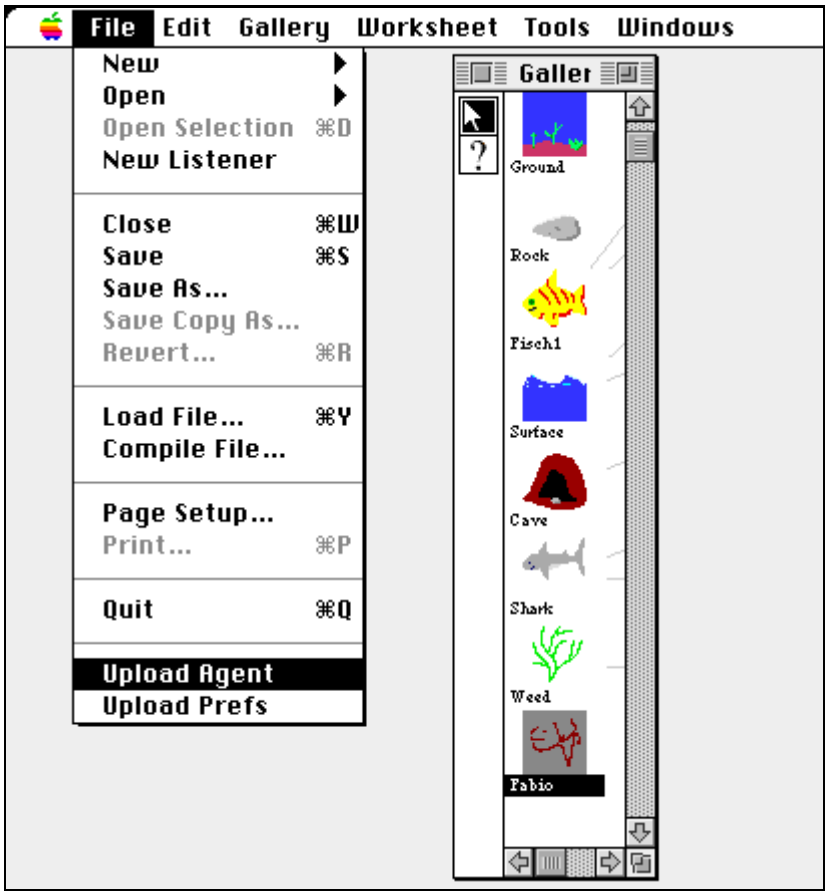

Figure 3: To start the uploading of a selected agent the File menu item "Upload Agent' must be selected.

He selects his two fish agents in the gallery and selects 'Upload Agent' from the 'File' menu to trigger the sharing process (see Figure 3) A dialog window, the contribution form, (see Figure 4) pops up and Joe is asked to provide information that will be submitted with the agents.

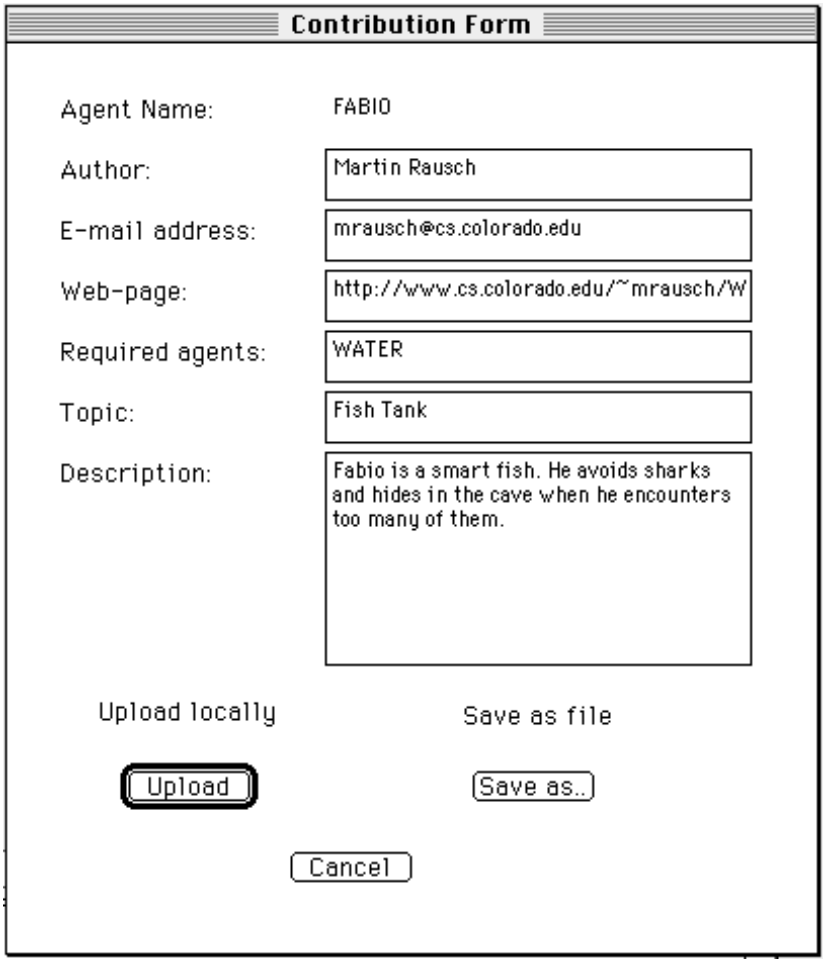

Figure 4: The contribution form allows the user to submit informal information about himself and his or her formal contribution, the agent.

The dialog displays the name of the first agent, the shark. The fields containing name, email address and webpage of the author have been set to the defaults that Joe has put into his upload-preferences. Joe specifies that the shark agent refers to the water background agent and the tuna agent in its behavior by putting these agents' names into the required field. He puts 'fish tank' for the topic of his agents. He derives that name from the name of the project within which he developed his new agents. Then he gives a description of the shark explaining his hunting habits and also briefly describes how he came to design these

agents and what he thinks they are good for, modeling a simple predator-prey model.

After he has filled out the form he selects the 'Upload' button and a second contribution form of the same kind pops up asking for the according information for the tuna fish.

When Joe is finished filling out the second contribution form he selects the 'Upload' button again and shortly thereafter is informed by an alert box that the upload process was successfully completed.

Beth is interested in modeling ecosystems, because she is a teacher and wants her students to learn about dependencies in ecosystems by manipulating simulations of such systems. Beth uses Agentsheets to build simulations. She decides to use the Fish Tank project as starting point of her development knowing that there is a repository of agents on the Web that contains a section where she might be able to ***locate*** *agents. Browsing through the Fish Tank page she finds Joe's two agents. She decides to* ***take*** *them out of the repository into her gallery to have a closer look at them.*

She drags the identifier 'shark' out of the page onto the margin of her gallery where she drops it.

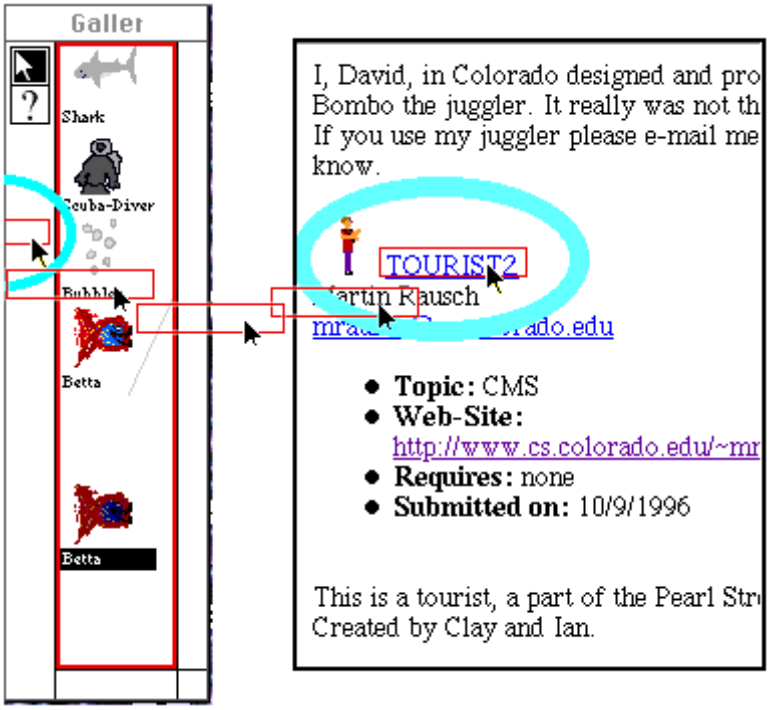

Figure 5: To include a new agent, that was located in the repository, into the local context, the name of the agent is dragged and dropped out of a webpage onto the margin of the gallery.

The shark agent is now part of her gallery. After she has dragged the tuna fish into her gallery, she puts an instance of each new agent into her worksheet. By running the simulation she can grasp an idea of what the agent's behavior is. In order to better ***comprehend*** *what the agents are doing she opens their behavior windows. She inspects the rules to see what they are doing and how.*

Beth decides that she needs to refine the agents. She wants to include into the model the fact that baby fish are the easiest targets for sharks, and that old fish are slower and thus easier to catch than young adult tuna fish. She wants to ***modify*** *the tuna fish's behavior to account for three stages in its life, very young, adult, and senior.*

After she has modified the tuna fish agent adding rules to the original behavior that relate its age to its appearance and its

appearance to its general behavior, she submits her 'improved version' to the repository annotating it with a reference to the fish agent it is based on and the additions she has made to account for age.

Overview

The thesis is organized in three parts that reflect the approach to the problem, a study to test the approach, and the conclusions where the approach is evaluated in the light of the study's results.

The first chapter will present the approach to the problem and the setting in which the investigation was done. In the chapter 'Background' the basic concepts of sharing Agentsheets artifacts will be introduced to the reader. 'Design issues' and 'The Role of Sharing in Constructive Design' will outline the relation of the Agent Repository to the design process.

After introducing the work environment used in the study different perspectives of the undertaking will be given. The chapter 'Related Work' will put my work in a context with Distributed Constructionism and MUDs, make a connection to the field of Computer Supported Collaborative Work and how its findings apply to my study, and focus in particular on the theory of social critical mass.

The second part 'The collaboration with CMS and OCS' will describe a study that was conducted in cooperation with two teachers, one of them in a Boulder middle school the other in the Open Charter School in Los Angeles. Setup and methods are described that were employed to encourage. An analysis of the problems that occurred and a summary of the experiment will follow in the subsequent chapter.

Then 'Conclusions' are drawn from the observations as documented in part two. Finally, 'Future work' will show the

directions for development, the next steps of pushing the idea of Distributed Constructionism with Agentsheets further.

Background

In this chapter the basic concepts essential for understanding the mechanics and purpose of the Agent Repository are introduced. Familiarity with Agentsheets and Visual AgenTalk is assumed. For an outline of the Agentsheets design environment and its programming language Visual AgenTalk you can refer to 'Appendix B'.

A description of the Behavior Exchange, the component of the design environment that supports collaboration via the World Wide Web and the Agent Repository, the part of the Behavior Exchange that contains agents follows.

Agentsheets		
Programming	**Application**	**Collaboration**
Visual AgenTalk		*Behavior Exchange*
Projects	Worksheets	Projects
Agents		Agents
		Agent Repository
Rules		Rules
Commands		Commands

Figure 6: An Agentsheets design environment: Agentsheets integrates (1) a programming, (2) a simulation, and (3) a collaboration environment.

Behavior Exchange

The Behavior Exchange is a unique website. Its design objective is to enable Agentsheets users to personalize the functionality of their programming environments, to find powerful ideas of how to achieve

certain behaviors and share their own constructions with the community of Agentsheets designers.

The difference to comparable web servers, such as the catalogs discussed in the 'Design Issues' chapter, is the innovative access mechanism that is employed. Instead of using the file transfer protocol (FTP) to download a BinHex-encoded file, decoding and possibly decompressing it - like code sharing is supported for traditional non-visual programming languages - all the interested user really has to do is drag and drop the relevant item - i.e. an agent - into a gallery of Agentsheets. The component is immediately usable.

Another unique feature of the Behavior Exchange is the fine granularity of sharable components.

- Projects,
- Agents,
- Rules, and
- Commands

can all be shared among the community of designers. This fine granularity of artifacts makes sharing a more integrated part of programming by making it easier to comprehend foreign components. The shared components are not huge complex structures but small building blocks.

The dynamic nature of Agentsheets tolerates small incompatibilities. When for example a rule refers to an agent that is not part of the gallery, i.e. if that rule came from the repository, the Agentsheets system will replace that reference by a random valid reference to one of the available agents. This may result in unexpected and at first inexplicable behavior but it does not crash the system.

Although all language components can be shared through the Behavior Exchange, the new sharing mechanism employed for agents is not yet implemented for the rest of them. Projects, commands and rules are at the present moment still retrieved in the traditional fashion described above. With the traditional sharing approach the user must leave the design environment in order to share constructions. The Agent Repository represents a first step towards integrating sharing into the design process. One objective is over time to extend the mechanisms employed for sharing agents to all language components of Visual AgenTalk.

From a high-level perspective, the purpose of the Behavior Exchange is to form the base technology necessary to promote social computing.

Ideally an individual user is not confined to refer to manuals or help systems but can in addition consult other users' creations. Taking other modes of communication into account virtual spaces, such as chatrooms or MUDs, could also be employed for solving problems and resolving breakdowns. The recent evolution of the internet and its rapid rate of expansion has brought new possibilities of collaboration into the reach of a large number of people. Research has been done on how to support learning on demand and critiquing [Fischer, 1991a & Fischer et al., 1991]. One perspective for the future in which the Behavior Exchange could play a role might be to move away from local help systems towards shared knowledge bases in a community of practice in which knowledge is formalized and represented as computational artifacts.

Agent Repository

The Agent Repository is the part of the Behavior Exchange that contains agents that every user with internet access can drag and drop out of a webpage into an Agentsheets gallery. There a user can modify and reuse it. If the user has the AgentShare extension agents can be added to the repository as well.

The Agent Repository is a unique combination of formal and informal information. The Agentsheets components constitute the formal, machine-interpretable information. In addition to this, the Agent Repository contains informal annotations that are attached to the formal artifacts. These informal annotations are depictions or animations of the agents, references to webpages, that are specified by the author, and textual, informal descriptions. These descriptions can contain any kind of meta-information the author wants to include about the agent. They are not reviewed by any mechanism or person. The responsibility to make the informal information comprehensible and useful is in the hands of the person that submits the construction.

And with respect to the work, that needs to be done to create a web representation of the artifact. Informal annotations are all that is left to the contributor. And this part can not be automated. The creation of the agent's depiction, the layout on the webpage, and the management of the resources are handled by the system.

Behavior Exchange and within the exchange the Agent Repository form a medium for sharing artifacts, that can be annotated informally. A social context for the formal constructions can be supplied with every submission.

Design Issues

In this chapter the mechanisms employed for sharing agents and the considerations that influenced the layout of the repository webpages will be described. The main issue in designing the Agent Repository was the organization and display of the submitted artifacts in an informative and meaningful way to support location [Fischer et al., 1991 & Fischer & Reeves, 1992] through browsing. The fact that browsing does not scale up well will require a search mechanism as a future development in order to support location once the repository has expanded to a more substantial size. The study at the middle school that is described later on has already indicated that the repository can grow very fast and the point where browsing is no longer a feasible location method is not far away.

Through analysis of existing web-based catalogs, identifying their strengths and weaknesses, I deduced the design that was applied in the repository.

In two examples of web-based repositories it is shown how the issues of documentation and the division of work and benefits are addressed.

Example I: The MCL Community

A good example for a successful group memory is the MCL[1] community. With the digitool website[2] there exists a forum for MCL programmers, where code is shared, questions can be asked and

[1] MCL=Macintosh Common Lisp
[2] http://www.digitool.com

answers posted. In these pages different methods of internet-based information exchange are integrated into the WWW site: a newsgroup for stating new problems, FTP[3] for reuse of code, an archive of past newsgroup discussions and a Frequently-Asked-Questions summary.

This site can certainly serve as a model for what I want the Agent Repository to become, a forum for Agentsheets designers. Its biggest shortcoming is the FTP[3] server holding code contributions. The contributed files are compressed and put in a folder named *"contrib"* mostly without explanations. Only a small number of entries has additional text files that are associated by using a similar file name. For the most part all information the user has for comprehension of the file's content has to be deduced from its name.

[3] FTP=File Transfer Protocol, an internet protocol to send and receive binary or text files

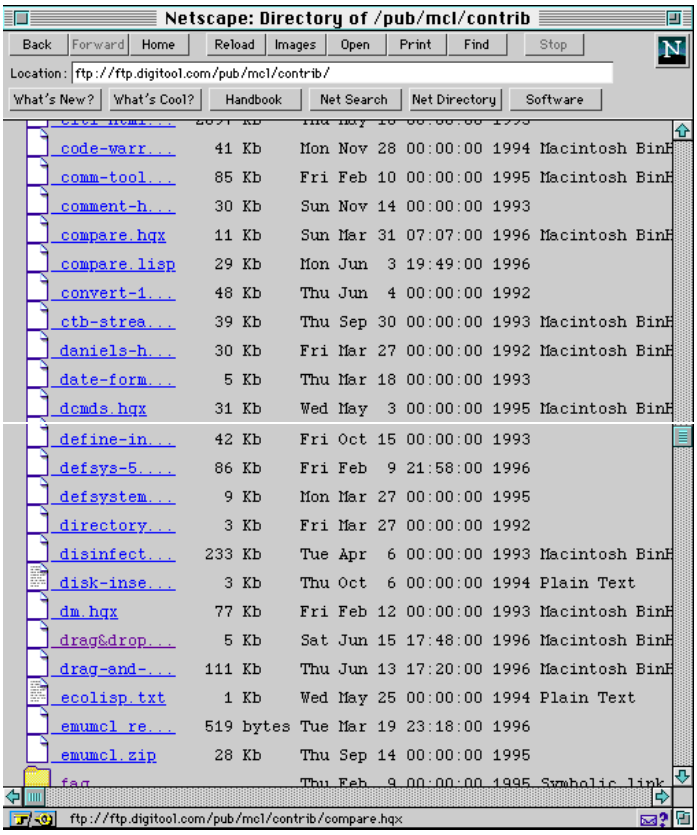

Figure 7: The source code page at Digitool does not provide informal information about the stored formalized artifacts (here Lisp programs).

Example II: University of Texas Macintosh Archive

The University of Texas Macintosh shareware archive[4] provided another source of ideas. In contrast to the first example this website emphasizes retrieval and distribution of shareware. New items are added rarely.

Everything on this server is represented in HTML[5]. A download button triggers the FTP[3] download. The user navigates by pointing and clicking.

[4] http://wwwhost.ots.utexas.edu
[5] HTML= Hypertext Markup Language

Every program is documented with information like download size, author, a description of what it does etc. The programs can be browsed by categories (applications, communication, graphics etc.), by author or title.

That way different search modes are supported and the problem of comprehension of complex artifacts (here: application programs) is addressed with a unified and informative documentation style.

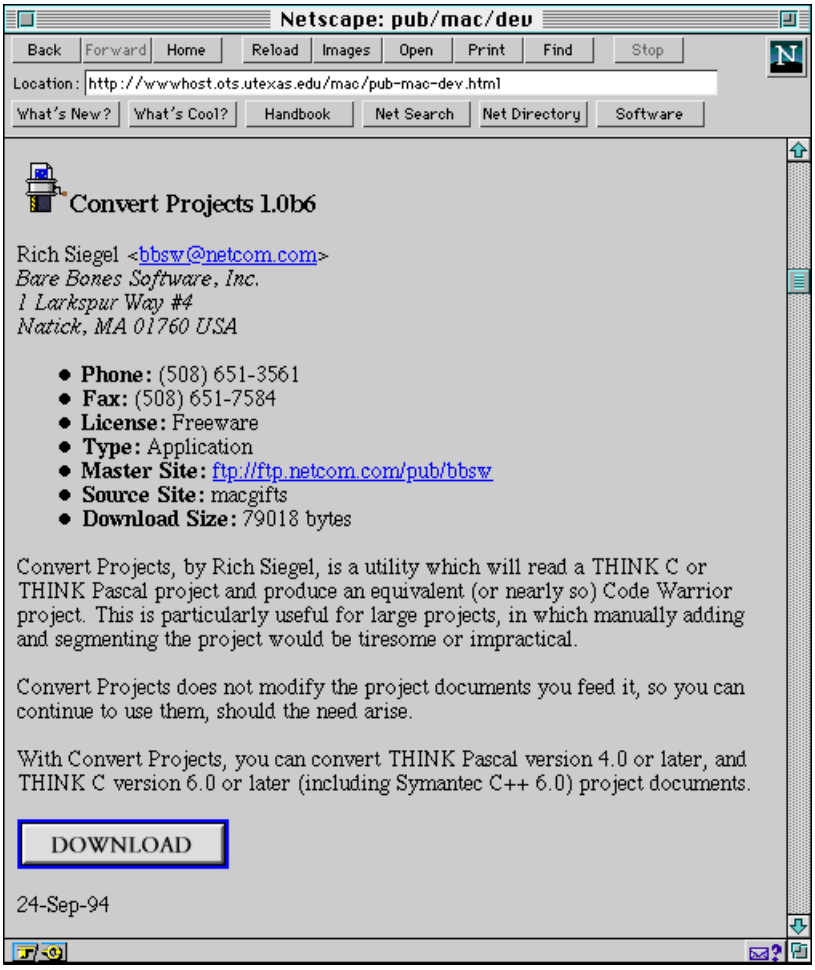

Figure 8: University of Texas Macintosh shareware archive

If the digitool MCL site would adapt the good documentation found at the University of Texas Macintosh archive, it would be an extremely valuable and attractive source of information for MCL programmers.

The tension between who has to do the work and who benefits can be illustrated with the examples that were presented here. The highly dynamic nature of the MCL site can be explained by the low threshold that contribution has. For the user community, in this case Lisp programmers, the FTP upload does not present a problem. Documentation is neither supported nor encouraged by the structure of the code repository. The lack of informal information about the contributions decreases the usefulness of the repository substantially. Location and comprehension are not addressed at all.

On the other hand, the UT Mac archive is administered by someone whose job is to support comprehension of the stored artifacts. And as mentioned above, the website evolves slowly which does not require the administrator to add annotations frequently. Much more work is put into the informal annotation and the larger effort is compensated by increased usefulness.

The two example live on two extreme sides of a spectrum. The Agent Repository's design objective was to be somewhere midway.

In these two examples the consequences of external as opposed to internal motivation can also be observed. The creators of the MCL website have not tried to supplement their mechanism (FTP) with additional incentives for good documentation. The internal motivation of the Lisp programmers was not big enough, in most of the cases, to contribute an extra file with additional information. Looking at the resulting webpage, one can understand why. The

difference in appearance between a well-documented source and a source without documentation is negligible. A documentation text file will only result in another entry in the list of files (see Figure 7).

The external motivation of the administrator of the UT Mac site ensures that he will be interested in the success of his work. The resulting webpage is something the quality of his work is measured by.

With respect to the motivational issues the Agent Repository has an ambitious goal: to be far closer with respect to usefulness to the UT Mac site that is maintained through external incentives than to the MCL site that is conceptually much closer.

The approach that should encourage the effort necessary to submit informative meta-information is to make the effort transparent in the webpage. A well-documented agent should stand out among uncommented agents. Users should understand that when they contribute artifacts their constructions will be more noticeable if they are well documented.

Having motivated a structured repository I will describe the layout of the Agent Repository as it was implemented.

The Organization of Agents in the Repository

The Behavior Exchange consists of webpages that are connected by hyperlinks that build a hierarchical structure in which the language components are organized by their type (commands, rules, etc.). Agents are collected in a substructure, the Agent Repository

The root page of the Agent Repository (see Figure 9) is divided into two display frames. The left frame contains hyperlinks to the pages that are dedicated to different topics. Those topic pages are then displayed in the right frame. Which topics have their own pages is determined by the administrator of the repository. Contributors have the possibility to specify an attribute “topic” with each agent they submit. The administrator can define a mapping from this space of possible topics to the pages of the website. According to this mapping each new addition to the repository is assigned its destination page.

Figure 9: In the Agent Repository, each agent is listed with its depiction, its attributes and description

The keywords he does not explicitly include are mapped to "Miscellaneous". When the administrator notices that a big enough number of contributions with keywords related to a specific topic has been added to "Miscellaneous" he can add a new topic page to the repository.

The role of the administrator is not clearly defined through the implementation of the system. The system for the most part is running autonomous. The adaption of the mapping between topics and webpages right now is extended manually. But using a system like GIMMe [Lindstaedt, 1996] that uses latency semantic indexing this part of the process may be automated as well. The system could then run unsupervised for some time. To speak in terms of the SER[6] model [Fischer et al., 1994] the process of reseeding would still have to be supervised. Once agents frequently are frequently submitted to the repository the problem of an abundance of redundant information will have to be addressed by such a model. The agent repository will have to be trimmed back to a reasonable size and as much redundancy removed as possible.

Categorizing the agents into topics is at the present the only support for locating agents. The topics are intended to keep agents close together that are likely to be usable in the same context. The structure is consistent with the idea of learning in a context. For a beginner, that has decided on a context in which to start his or her exploration into the programming world the repository will have the relevant constructions all in one page. Even with very little understanding about how to analyze a retrieved agent the user can just by making the

[6] SER=Seeding, Evolutionary Growth, and Reseeding

connection between his local context and the topic in the repository find a number of agents to test out. Chances are that the agents that fall into the same category will fit into the local context and their behavior is more likely to resemble some behavior the user is already familiar with.

The Role of Sharing in Constructive Design

In order to be able to evaluate the Agent Repository and to explore its usability the use as intended by the designer is compared in this chapter with the actual use as observed in the middle school study. I will try to identify the differences and furthermore try to find out the reasons for these differences in use of the repository.

Questions in the anticipated use of the Agent Repository as a group memory are

- at which point in the construction process the agents are added and
- when during the design it makes sense to retrieve them.

The phase of design in which submission to the repository makes the most sense is the point when the construction already exhibits powerful behavior but is still on a level that is general enough to allow for the construction to be used in different contexts. The observation about agent design that justifies this claim is that agents are usually designed incrementally. Beginning with the most basic behavior i.e. the rules of how an agent moves on the worksheet, agents are refined to have more complicated, more specific, or less frequently observed behavior after the more generic tasks are implemented.

In the middle school study, for example, agents fell into two categories. There were passive background agents without any behavior and active agents that had behavior. Active agents were at first programmed to move about the screen. In the Fish Tank scenario (s.a.) this would translate into the rules that make the shark swim

around in the tank. In a second design phase an agent would get behavior that was more specific. In the case of the shark that could be his hunting behavior.

Supporting my earlier argumentation about when agents are 'ready to share' is the collection that has accumulated in the Agent Repository since its installation. Most fish, for example, 'know' how to swim and have in general one more distinguished property or skill, such as age, appetite, communication etc.

Following this, the point at which to retrieve an agent then is analogous. Agent reuse, the idea behind the repository means that a user does not have to 'reinvent the wheel' every time he or she creates a new agent. For some generic behavior the user might want to look for existing agents in the repository that can be combined and refined to fit more specific requirements.

The reuse of agents requires skills to comprehend other people's intentions and their ways to implement them. These skills include a good understanding of the language the artifacts were formalized with.

This should not be perceived as a shortcoming but as a source of motivation to acquire such skills.

In the next chapter the repository is related to the theory of Distributed Constructionism, the concerns that have been raised by researchers in the field of Computer-Supported Collaborative Work are discussed, and implications of the theory of a social "Critical Mass" are presented.

Related Work

In this chapter I will tie my work on the Agent Repository in with related work.

In Multi-User Dungeons (MUDs) multiple users connect to a server and enter a virtual space or room in which they communicate and build constructions and exhibit them in the space. MUDs are facilitated to support Distributed Constructionism by collaborating on artifacts directly. In my opinion, their potential is best explored in the early phases of learning when a basic understanding of the system model is constructed.

CSCW[7] researchers have investigated the peculiarities of tools for collaboration for a long time. How their concerns apply to the Agent Repository and where the system is different from traditional CSCW applications are showed in the following section.

In the third part of the chapter the Agent Repository is related to the "Theory of Critical Mass" as introduced by sociologists. "Critical Mass" describes the circumstances under which a new interactive medium is either rejected by a community of potential users or develops towards universal acceptance.

Sharing Constructions

"Constructionism is based on two types of 'construction'. First, it asserts that learning is an active process, in which people actively construct knowledge from their experiences in the world. People do

[7] CSCW = Computer-Supported Collaborative Work

not *get* ideas; they *make* them. (This idea is based on the *constructivist* theories of Jean Piaget.) To this, constructionism adds the idea that people construct new knowledge with particular effectiveness when they are engaged in constructing personally meaningful products. They might be constructing sand castles, LEGO machines, or computer programs. What is important is that they are actively engaged in creating something that is meaningful to themselves or to others around them." [Resnick and Rusk, 1996]

In "Distributed Constructionism" Mitchel Resnick [Resnick, 1996] divides constructionist activities into three categories: discussing constructions, sharing constructions, and collaborating on constructions.

Discussing and collaborating on constructions are best supported by systems that use direct communication. The time scale, for which the design process can be supported by direct communication systems only, is short-term. The information, i.e. design rationale, that evolved in course of a discussion or collaboration is not automatically persistent and therefore often lost for later comprehension.

This problem was for text-based direct communication addressed by the design of the GIMMe system [Lindstaedt, 1996]. A modified GIMMe system could store argumentation and even textual represented artifacts, as those in MUDs, and would allow later reuse and relocation through latency semantic searching.

To capture the design knowledge that was put into the creation of artifacts indirect communication [Fischer,Grudin,Lemke 1992] that is

more independent of spatial and temporal vicinity is better fit than synchronous direct communication.

The Agent Repository facilitates indirect communication through storing knowledge represented as behavior in agents and making that knowledge accessible. Sharing completed artifacts is a less interactive process than collaborating on-line as it is described in the next section. It appeals to a more experienced user group since it requires a greater amount of knowledge about the system model. Without such a model the purpose of a shared artifact remains hidden and the artifact will be hard if not impossible to reuse. In addition, the permanent availability of the information makes the user more independent from time constraints.

The opening scenario described the process of sharing agents. It used the fish tank project as its context. In the following box you can find out why the fish tank is often cited in connection with the Agent Repository and used as an example to think with and talk about.

Why Fish Tank ?

"The original idea for the Vivarium, the ecology-in-a-computer concept, came from Ann Marion, now the Vivarium Program Manager, when she was working with Alan (Kay) at Atari. One of their projects was to try and do intelligent autonomous Warner Bros. cartoon characters, to send Bugs Bunny and Elmer Fudd into the forest, and have them play out a cartoon as a result of their personalities. Ann, however, sought to infuse life into more realistic creatures engaged in social interaction with each other and within their environment.

She chose to model an aquarium, with fish that would chase and eat one other and reproduce. It was an arduous task, and one that we now seek to make as easy as child's play.

Today we use the animal in a biological ecology as a metaphor for an agent in an information ecology." [Yaeger]

Originally out of the Vivarium program [Kay, 1991], the Fish Tank has become one of many Agentsheets projects. Because it is quickly comprehensible, as experience has shown, it has become the prime introductory example when exposing new users to Agentsheets for the first time. It not only demonstrates behaving agents and their interactions but also allows a first glimpse into the world of programming one's own agents. With the behavior limited to a small set of conditions and actions it does not overwhelm the user with details.

It furthermore offers a wide variety of possible behaviors of newly introduced agents. Their rules might give them distinguished features with respect to animation, modes of interaction with other species or within their own species, etc. This provides for the opportunity of many people to extend the population of the tank.

Into this uniform environment agents can be introduced and their interaction with other agents in that environment can be explored. The simulation is a good toy study for Distributed Constructionism because the domain does not necessarily require a long introduction. Most people are at least somewhat familiar with fish tanks

Figure 10: A worksheet of the Fish Tank project

Using the synchronous approach a first attempt on investigating Distributed Constructionism in practical application was started by Amy Bruckmann.

MediaMOOSE - Distributed Constructionism and MUDs

In this section a different approach of applying Distributed Constructionism, collaborating on constructions in textual virtual reality spaces, is put in contrast to sharing agents in the Agent Repository.

Amy Bruckmann studies Distributed Constructionism using MUDs - text-based virtual reality environments [Bruckmann, 1994]. Her experiments and interviews have indicated that people find learning to program more enjoyable in the social context of virtual spaces. MUDs provide an environment for collaborative work and learning of several users on one or more constructions.

My thesis investigates an approach to Distributed Constructionism that utilizes indirect communication rather than the highly interactive media utilized in MUDs. Thus it encourages the sharing of completed artifacts rather than its collaborative design, but by no means excludes the sharing of prototypes or not-yet-completely-operational agents.

The target phase for its use is not when people first learn how to use a programming environment but when they have reached enough expertise to understand constructions they did not create themselves.

A MUD is a virtual text space in which a number of people are present. In addition to communicating with each other directly, like in a chatroom, they are able to create objects and interact with them in that space. To add a new object to the space a user has to write a program to specify that object. As soon as it is compiled users in the space can be affected by and interact with that object. The programs -

their own and other people's - talk back to them. Users present their own programs to an audience for feedback or are inspired by other users' programs.

In such an environment, shared understanding of difficulties in designing objects can be built when users contribute their own experience and solution models. If the diversity in the areas of expertise is wide and at the same time a consensus is found on the vocabulary in which to state problems and solutions such an environment can help solving the difficulties that individual users brought into the collaboration space.

One could imagine to employ a MUD for collaborating in a shared Agentsheets worksheet. The participants could control agents in the worksheet and directly communicate with other human participants or even with fully automated agents. The fully automated agents would blend in with the user-controlled agents just like in textual MUDs.

The drawbacks though are the same as in text-based MUDs, the dependency on other people's presence in the shared space when one needs their help and the transient nature of the constructed consensus and knowledge. A MUD is not designed to create a persistent group memory. It is constructed for on-line direct collaboration. But again, an Agentsheets GIMMe might offer a possible solution for the problem of capturing and relocating argumentations, that in case of an Agentsheets MUD could be animated sequences of interacting agents.

I see MUDs as one component of several that enable design environments like Agentsheets to become more supportive of collaboration. Multi-User Dungeons are not an alternative to repositories but a supplement to establish a community of practice.

Guidelines from Computer-Supported Collaborative Work (CSCW)

CSCW has investigated collaboration via networks for quite some time. Most of the work was done before the World Wide Web gained its present popularity and omnipresence. The focus was on work in commercial corporations.

Nevertheless, many findings from this research area are valid in the design of systems to support distributed constructionism as a means for learning and education.

Jonathan Grudin stated eight challenges [Grudin, 1994] to be addressed by designers of systems to support collaboration. In the following section, each challenge is answered individually on how the Agent Repository and its access mechanisms address the problems stated.

1. Work vs. Benefits

The amount of costs and benefits depend on preferences, prior experience, roles, and assignments

But in general, the disparity in effort and benefit works against acceptance in many situations and helps explain the failure of many systems intended to support collaboration.

Often doing the additional work becomes somebody's job.

Demonstrating an application's collective and indirect benefits can help.

The work involved in making the Agent Repository both, usable and useful, is done by the user that makes a contribution. The contributor is responsible for the documentation of its artifact. At its present stage, the repository does not offer computational support for

location and comprehension of agents. The only search mode supported is browsing. Even in the short term study with a small number of students contributing to the repository it became clear that more computational support for location is required.

Comprehension does not necessarily have to happen in the webpage. With the cost in time and effort of retrieving an agent as low as it is, a rough idea of what an agent can be used for might be good enough.

The process of making constructions public cannot be fully automated. The informal description of an agent, its purpose and functionality, is essentially only known to its author. The best a computational environment can do is free the user from as much formalization work as possible. The creation of the HTML-code to represent the agent in the webpage or the creation of a depiction of the agent are examples of tasks that involve the formal information that can be automated.

The beneficiaries of an Agent Repository that is rich in resources are those users who are in a position of enough expertise to comprehend the agents' formal structure once they are retrieved.

2. Critical Mass and Prisoner's Dilemma Problems

Most groupware is only useful if a high percentage of group members use it.

Designers can reduce the work required of all users, build an incentives for use, and suggest a process of use that provides or emphasizes individual and collective benefits.

The issue of critical mass is addressed in detail in the following section.

3. Social, Political and Motivational Factors

The computer is happiest in a world of explicit, concrete information. Central to group activity, however, are social motivational, political and economic factors that are rarely explicit or stable.

The informal annotations that are submitted with the formal structure, the agent, are intended to capture the social, political and motivational factors that may be important for an agent.

The description that is attached with the agent provides a way of expressing thoughts that are technically not related to the agent. The reference to a webpage is another way of attaching informal meta-information to an agent.

4. Exception Handling

A wide range of error handling, exception handling, and improvisation are characteristic of human activity.

Exceptional situations using a medium occur when the user is not aware of the assumptions the medium relies on. The communication mechanisms of the Agent Repository are based on some standard assumptions that might limit its use in exceptional situations.

The assumption that the other agents that are referenced in an agents rule are in the context that the agent is imported in is a soft constraint. If the assumption is wrong the agent will not display its desired behavior, but the system stays intact.

More limiting is the assumption that all commands within the rules of an agent are interpretable by the system. Visual AgenTalk has a set of standard commands that can be assumed present in all system,

but the open architecture of the language allows extensions to the language, i.e. new commands. If such a specific command is referenced in the rules of a shared agent a system that does not include the command in question will crash.

A solution to this approach would be to allow packaging. Instead of allowing only individual agents to be uploaded one by one, an extension that allows to put together 'context packages' that may contain required agents, commands and resources, could solve some of these compatibility problems.

The challenge would be to extend sharing to packages without making the mechanisms complex and harder to comprehend.

5. Designing for Infrequently Used Features

If possible, add groupware features to an already successful application rather than launch a new application with a fanfare that creates expectations of heavy use.

The sharing features are additions to the Agentsheets design environment. The hopes though are to promote social computing and thus to make sharing a Frequently Used Feature.

6. The Difficulty of Evaluation

Groupware evaluation methods are less precise. Field observations are complicated by the number of people involved over time at each site, the variability in group composition, and the range of environmental factors that affect the use of the technology.

I do not fool myself into believing that the results that I could collect in the short time I was working with the two groups of students are representative. I think that interesting phenomena can be

observed and reported and the two groups I was working with had two very different situations in which they used the repository. The problems that they encountered came from a wide range.

7. The Breakdown of Intuitive Decision Making

Project management applications primarily benefit project managers. Meeting schedulers and meeting management systems benefit those who convene meetings.

But in the case of groupware, managers often underestimate the down side, the unwelcome extra work that an application will require of other users, resulting in neglect or resistance. For example, a group decision support or work management application can require many people to learn to enter data, it can record information that participants prefer not to have disseminated, and it can block other means of influence decision making, such as private lobbying.

The Agent Repository is understood as a medium for sharing Agentsheets constructions, in its case agents. Nobody has to do the extra work of contributing his agent unless he wants to make it public. The Agent Repository is not governed by any authority. Therefore the problem does not seem to apply here.

8. Managing Acceptance: a new Challenge

Finally, someone should be prepared to prevent premature rejection by anticipating and dealing quickly with early problems, and follow-through support should be in place to handle the posthoneymoon period, when the group's curiosity wanes and work returns to center stage.

The results of the attempt to prevent premature rejection are partly captured in the next chapter about the collaboration with the schools. In the conclusions at the end of the thesis an explanation will be offered why it took about two months before the repository caught on and finally appeared to be accepted and used.

Critical Mass Theory for Interactive Media

One of the biggest concerns with the Agent Repository is acceptance by its target community. Like any medium of information exchange, the repository will only find broad acceptance if it contains useful resources, and contribution will only be attractive if the repository has broad acceptance.

The topic of motivational factors of groups using a new application has been studied in sociology for quite some time. The “Theory of a Critical Mass” that is required to promote a new system’s universal acceptance was introduced by Oliver, Marxwell, and Teixeira. [Oliver et al., 1985]

In [Markus, 1990], this theory is applied to the introduction of new interactive media. A number of factors determine the likelihood that a new medium will find universal acceptance. Some of the factors represent resources, i.e. infrastructure, equipment, required effort and skill, other factors are the ‘Production Function’ and the heterogeneity of the community in question. The requirements in term of equipment, effort and skill are summarized in the term operational access. The lower the threshold to achieve operational access the higher the likelihood of universal acceptance. In case of the Agent Repository operational access requires a TCP/IP network connection, the AgentShare extension, and a web browser. The required skills are

familiarity with mechanisms like drag and drop and menu-based user interfaces.

The effort one has to make when retrieving consists of the location of the agents in the page and a drag and drop operation. Contribution requires more effort. Here I see the most crucial point for the success of the repository. The documentation of agents is the most costly operation in terms of effort and at the same time it is the most important to ensure comprehensibility. Informative documentation is a prerequisite for reuse of agents in the repository.

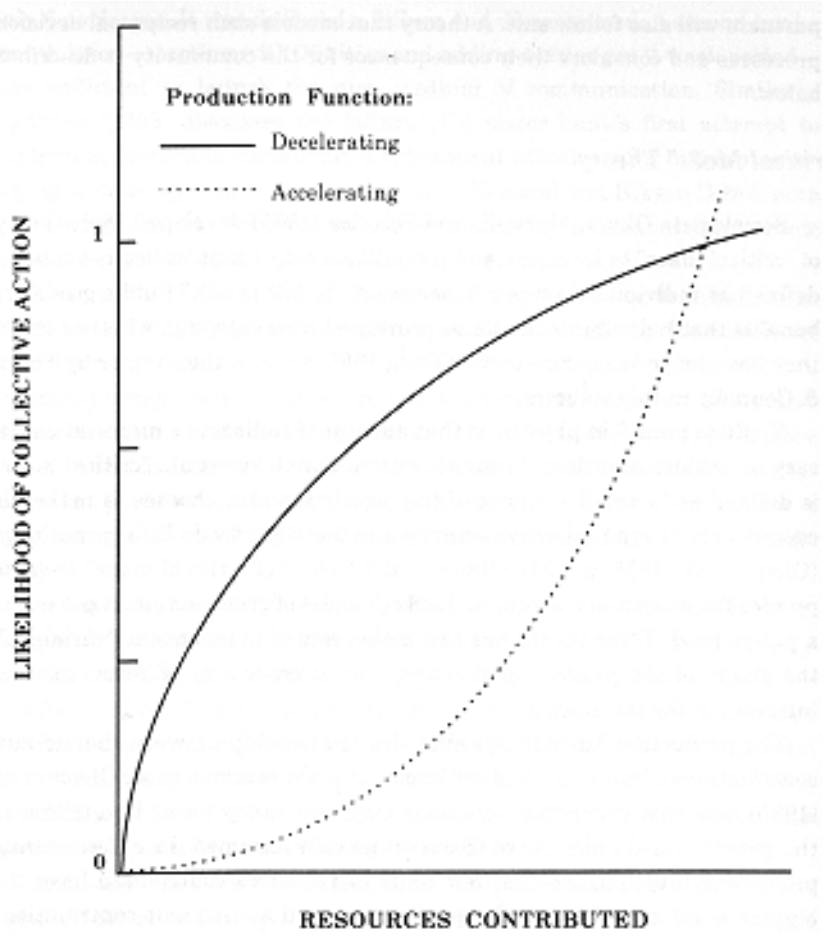

Figure 11: The shape of the Production Function [Markus, 1990, p.202] indicates at what point in the diffusion of the medium contributions have the biggest impact on the common good.

But operational access is not the only factor that determines the likelihood of universal acceptance. The production function, the heterogeneity of the community as well as the resources that the initial users will contribute and what they are worth to the community determine success or failure just as much.

Figure 11 shows two possible shapes of the so-called 'Production Function'. "The production function specifies the relationship between individuals' contributions and achievement of the common good."[Markus, 1990] It characterizes at what point a contribution has the greatest effect on the usefulness of the medium. An accelerating function means that the first contributions have the biggest impact on the benefit the medium yields. A decelerating production function means that later contributions will increase the common benefit of the medium more than early contributions.

Markus argues that for interactive media the production function is accelerating since "each contribution of effort increases the ability of prospective users to benefit."

Heterogeneity refers to the assumption that not all users are equivalent in their potential to contribute to and benefit from the medium. According to Markus, heterogeneity is an important factor of successful diffusion. Not only depends the likelihood of success on the number but also on the expertise of the early contributors. A new

medium might offer new possibilities to interact with experts that otherwise could not be consulted.

The chances of a fast diffusion of the new medium are improved when the incentives of using the medium are sufficiently high. Users can be attracted gaining benefits from retrieving information that would otherwise be hard or even impossible to get. If this point is reached a 'feedback loop' is possible because a bigger audience means a bigger incentive to make one's own resources publicly available.

If the Agent Repository succeeds in gaining acceptance in the community of Agentsheets users depends on all the factors stated here. In the next chapter, a study is described that tried to promote contributions in order to evaluate where the Agent Repository falls short.

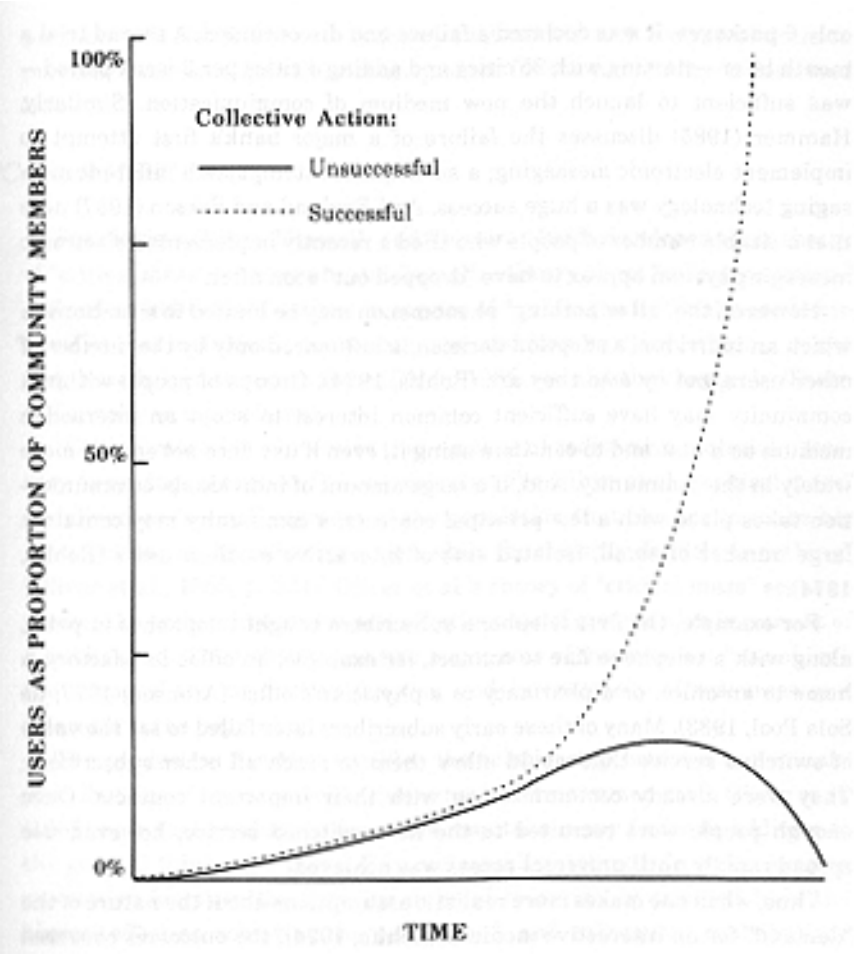

Figure 12: The "Critical Mass" Theory [Markus, 1990, p. 199] states that success of an interactive medium depends on it diffusion among the user community. If at a certain time in the diffusion process a certain treshold is not reached, the medium will not succeed. The point in time as well as the fraction that is the threshold are research topics.

The Collaboration with Centennial Middle School in Boulder and the Open Charter School in Los Angeles

To investigate the issues of indirect long-term collaboration and distributed constructionism with the Agent Repository I performed a study in which two groups of middle school students that were separated by several thousand miles - including the Rocky Mountains - could communicate and collaborate with each other on the construction of a simulation.

Settings at the Schools

The two schools involved in the study were Centennial Middle School (CMS) in Boulder, Colorado and the Open Charter School (OCS) in Los Angeles, California.

The setting at CMS was the 'Friday Afternoon Computer Club'. Regularly on Friday afternoon technology teacher Scott Dixon opens the computer lab to students who are interested in exploring new technologies especially the internet. The students are free to work on whatever they want with whatever tools they want to use out of the collection of applications the school has acquired. Only playing games is limited in order to not have the computer club degenerate to a free arcade.

Students in the computer club have been test users for earlier studies, i.e. in connection with the LEGOsheets project [Gindling, Ioannidou, et. al., 1995]. We could benefit from that in recruiting students for our project because we could build on the reputation established by the LEGOsheets group for offering exciting projects.

The group of people that were interested in working with us at Centennial Middle School consisted of around 12-14 students, grades

five to seven, with little fluctuation of students dropping out or joining late. Alex Repenning, Andri Ioannidou and myself met with the students once a week for two and a half hours. Two students lost interest in the first month and left the group. During the same period two student joined the group. Attendance was not mandatory so that the number of students varied every week.

The goal with respect to my thesis was to eventually initiate collaboration between two groups of students via the repository. The plan was to build a project in Boulder and to provide the components to other users. The targeted users that were to reuse the Boulder components were the class in the Open Charter School.

The Open Charter School in Los Angeles educates students from Kindergarten to 6th grade. Classes are not as strictly separated by age as in common public schools, but embrace a range of students of different ages. The particular group I was involved with consisted of 3rd, 4th and 5th graders and was called the 'Blue Cluster'. The Blue Cluster is educated by BJ Allen-Conn and Donna DiBernardo. Each cluster has an unique emphasis.

"The theme in Blue Cluster is 'Considering Multiple Perspectives'. Blue Cluster stresses independence and time management skills as students study the marine environment and our country's history. The curriculum encourages children to make appropriate choices, both academically and interpersonally. Students participate in whole group lessons, small group lessons, cooperative groups and interest groups.

The social studies curriculum takes a multicultural perspective on California and United States history, concentrating on the experience of women, men and children of different ethnic and religious groups. As students study the marine environment, they analyze the unique characteristics and adaptions of plants and animals to that environment. Students participate in the SALWECO project via the

Internet, creating a two-dimensional world on-line. In doing this, they must consider complex ramifications of life in two dimensions, such as how a hinge might work or how an arm might grasp objects. The Blue Cluster curriculum helps students consider multiple perspectives in problem solving, in analyzing historical and social problems, scientific situations, as well as mathematical situations. Technology is integrated throughout the curriculum."[8]

As a consequence of the objective that is pursued in Blue Cluster Agentsheets was one of several perspectives of how to build simulations. The class was divided into three groups. One group working with Apple's Cocoa[9] system, one group working with MIT's MOOSE Crossing [Bruckmann, 1994] and the third group working with Agentsheets. The Blues Cluster has a history of very experimental curricula.

The school's profile description says, "We pioneered the movement to integrate state-of-the-art technology with a humanistic, interdisciplinary curriculum. Between 1986 and 1993, in a joint venture with Apple Computer, Inc., we participated in the innovative and exciting Vivarium Project [Kay, 1991] to explore the educational role of technology. The Project explored how teachers and young children use personal computers as an extension of themselves and their learning endeavors.

Our partnership with Apple provided us with a strong technological foundation upon which to build. Our association with Apple is currently carried on through a Consortium of local schools. We continue to seek ways to make computers a part of everyday school life by working with other corporations, foundations and research / educational institutions." [OCS, 1996]

For the collaboration with Centennial Middle School the students in Los Angeles together with their teacher, B.-J. Allen-Conn, learned to program Agentsheets applications with Visual AgenTalk. Before

[8] I cannot quote a source, because this text segment was taken from a handout that was part of an informational package provided by the Open Charter School. For a copy you might try the same contact address as (OCS 1996)

[9] http://cocoa.apple.com

they could start and tackle this challenge, teacher and students had to face yet another impediment first, the installation of Agentsheets and its proper operation. I tried to support them as best I could, being connected on-line with them via IRC[10] whenever they convened for class and trying to solve their problems and answer their questions. But the amount of technical complications were greater than anticipated.

Unlike in Boulder the setup in Los Angeles was in the format of class time assigned to programming.

"At the Open Charter School, learning is doing, seeing, questioning, listening, experimenting, analyzing and, most of all, applying knowledge in ways that enrich our students' lives, the community and the environment.

- No one subject is taught in isolation, instead each activity is interdisciplinary.
- Our integrated curriculum is activity-based.
- Classes are team taught and organized into multi-age clusters.
- Students assume responsibility for their learning, their behavior and the materials they use.
- The environment is the school's thematic emphasis and forms the core of our curriculum

Although little is traditional about the School, when standardized tests are administered, our students outperform their counterparts at other schools." [OCS, 1996]

The 'Pearl Street' Project

During the first two Friday afternoon sessions the students at CMS were introduced to Agentsheets and Visual AgenTalk. They were not constrained on what they were to do with it. One group of students

[10]IRC=Internet Relay Chat, a protocol that allows multiple users to directly communicate on a 'channel' that is broadcast to all participants.

started to develop a game involving fighter planes and missiles. Other students worked by themselves on small simulations and games, most of them consisting of a maximum of five background agents without behavior and one or two active agents that could be controlled to move on the screen.

Roles in Web-Based Collaboration

One interesting aspect of the study was the emergence of roles that users take on when they become part of a new community on the World Wide Web. The internet with all its modes of communication, nicknames, accountnames etc., allows its users to take on new roles and even new identities in virtual communities [Turkle, 1995].

With respect to the study at the middle school the roles that I expected to differentiate were “experts” and “learners”, experts being the users that have knowledge about programming Visual AgenTalk, learners lacking that skill.

I further expected the students in Boulder to be more likely to take on the role of experts in the exchange with the students from Los Angeles, just because they had a head start using Agentsheets, since the Friday afternoon sessions started about two weeks before the collaboration with the Open Charter School. In addition to that the Boulder students were instructed by experts, Alex Repenning, Andri Ioannidou and myself, whereas the students in Los Angeles learned along with their teacher.

Roles were not restricted to collaboration between the two groups of students, but were also important within the group we were working with. As the project started it turned out that a small number of

students was interested in the programming aspects of Agentsheets while the majority was involved creating the depictions and using Agentsheets as a painting tool to make stencils and put them into a worksheet. These static worksheets would then be animated by the "programming experts".

Our study did not take place in a social vacuum. The students in our group knew each other before and they had their roles in the computer club before they started working with us. The "authorities" took control of parts of the projects and assigned tasks to other students. The "subordinates" did not seem to mind. They would put in their own perspectives and find a compromise between the "master mind's" idea and their own.

After the students had worked unconstrained in the first two weeks I decided to have them reflect on their assessments and achievements so far. Only one group had stuck together. The majority of students had dispersed into their own little projects.

In the reflection it became clear to these solitary designers that the group with its self-coordinated efforts had achieved a great deal more than the individuals' projects. They had been more productive quantitatively and qualitatively. While the individual designers had been working with around five agents in a project and a lot of the projects were similar, this group had about 15-20 agents that were part of one big project. Thus, the outcome of the group project so far was more interesting since more diverse than the other projects'. This was perceived by all the students as they expressed their perspective of the situation.

Through this experience, the students were receptive when asked whether or not they were interested in dedicating all their efforts to pursue one big project in which all of them would participate. Several possible topics for a project were discussed and my request to the students was to build something that was meaningful to them and at the same time left the possibility to share components of their project with students from another part of the country.

Finally, we agreed on Pearl Street[11] as theme for a simulation game.

Figure 13: Killer Dog

The Virtual Dog

One of arguments supporting 'Pearl Street' as the theme for the project was that the students were more likely to be motivated to build a simulation of something that was from their own experience.

That students are motivated to express experiences from their own living situation was supported by a small project built by two students in the first two weeks of the study, "Adventure to School". The game's scenario was a crosswalk on the way to school. The character of the game, a small boy, had to make sure he would walk to school without coming too close to the 'Killer dog' character (see Figure 13, p. 54). The students put the 'Killer dog' agent into the Agent Repository. A couple of days later I looked into the repository and found the 'Killer Dog'. The student had submitted a reference to what I assumed to be his

[11] Pearl Street is a mall in downtown Boulder where street performers present their skills to shoppers and tourists walking by.

homepage. My curiosity and general desire to learn more about the students drove me to the referenced homepage. To my big surprise, I did not find the student's homepage, but the homepage of his dog that had striking resemblance to the 'Killer Dog' created with Agentsheets.

Obviously, the student had incorporated part of his own living situation into his simulation.

On a larger scale, incorporating part of the students' environment should hopefully provide for intrinsic motivation in a group project.

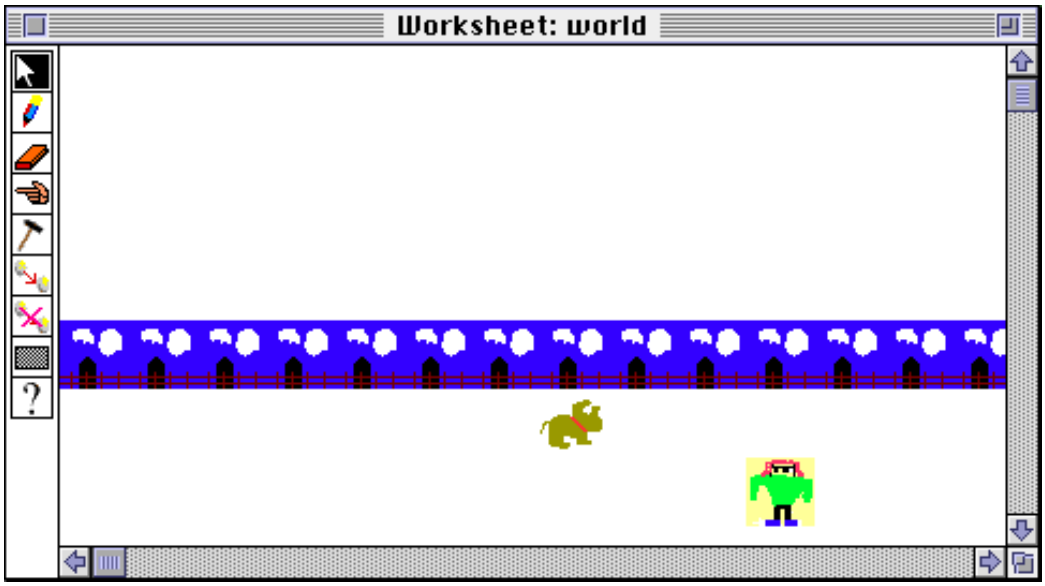

Figure 14: 'Bob' is trying to avoid the 'Killer Dog' on this 'Adventure to School' worksheet.

This next episode should illustrate an instance of reuse that could be observed in the middle school. It displays reuse of agents as it appeared and can be compared to the intended use of the repository as stated in 'The Role of Sharing in Constructive Design'.

On Fish, Toxic Bubbles, and Modification of Shared Agents

At one point in time a group of University of Colorado students taking a class "Designing the Information Society of the Next

Millennium" got an introduction to Agentsheets and Visual AgenTalk as an example of an end-user programming environment. The fish tank being the basic introductory project served as the context for their first attempts to build agents.

In a relatively short time a wide variety of agents was designed by the class. The creativity of the students did not only produce new kinds of fish, i.e. a colorful fish that looked a lot like a Siamese fighting fish (or 'betta'), but also a scuba diver that would emote air bubbles when the space bar was held down. All these agents were uploaded to the Agent Repository at the end of the class session.

When I showed the students at Centennial Middle School the collection of agents the following Friday they were at once intrigued by the scuba diver and the betta. They dragged and dropped them out of the webpage and tested their behavior within a fishtank worksheet.

But they were not content with the displayed behavior. The betta fish would just sink to the bottom of the tank and then swim off the right edge of the tank and disappear into the depth of the unused worksheet[12].

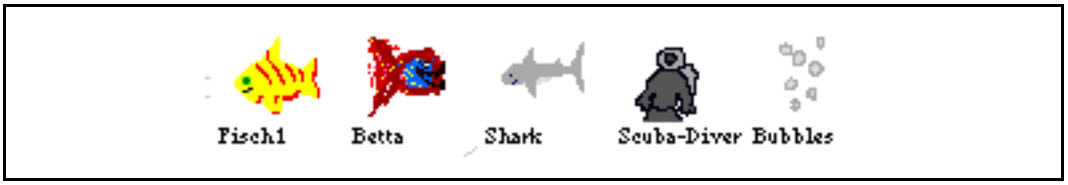

Figure 15: The agents

[12] When a worksheet is created it is allocated a grid of 100 by 100 cells. Users usually only use a small fraction of that given space. Most worksheets only have meaningful content in the area visible in the window.

They analyzed, **comprehended**, and then modified the betta's behavior so that it would stay within the window area. The original designer had included a rule for the betta that if one betta would see another specimen above itself then a new betta would be instantiated. Since the bettas behaved so poorly swimming out of the tank immediately this rule never showed any effect. Now that the fish stayed in a contained area they eventually met other kinsmen and reproduced. The CMS students were thrilled. Soon a large fraction of the tank was filled with bettas since they multiplied incessantly.

Figure 16: The scuba diver and the bettas are added to the tank

The fish tank was already populated by other species of fish, one of them a predator, a shark. The sharks would swim around and encountering a yellow fish eat it with a certain probability. The Middle School students quickly grasped the idea that since the sharks did not have behavior related to the newly introduced betta fish they would not just start hunting them.

They **modified** the sharks' rules in analogy to the rules that were already there concerning the consumption of other species. The rule that stated that a shark eats a fish on sight with a certain probability was copied and modified so that it stated 'on sight of a betta' where the original had said 'on sight of a yellow fish'. The yellow fish, the original prey for the sharks, did not reproduce. Thus a simulation always 'ended up' with no more yellow fish, only sharks in the tank. Now they had extended the simulation to a model of predators and prey and they interactively participated in the simulation. The bettas would multiply and the sharks would hunt them but the ratio would always be off. Either the whole population of bettas would be eliminated by sharks or the sharks could not keep up with the numbers of new bettas that were created and the tank would 'overflow' with bettas.

The students tried to regulate the simulation by either adding more sharks and removing bettas or vice versa depending on what was necessary to balance the ecosystem. They manipulated the simulation as it was running 'by hand' using the paint and eraser tools to add or remove fish.

Figure 17: A simulation of sharks and bettas

But that is not were they stopped. The scuba diver would emote bubbles whenever a user would hold down the space bar. The bubbles would rise to the surface and disappear. The students changed the rules that made the bubbles rise to the surface so that they would erase a betta or a shark when they hit one. Now they had transformed the simulation into a shoot-'em-up game.

Using an existing project and modifying agents they found in the website they had learned to build a simulation of an ecological system, here predator-prey, and had applied their own creativity to extend other people's creations.

The actual use of the repository as found in this episode was similar to the use that I as the designer of the repository had intended it.

The basic behavior of sharks and bettas was modified to display more **specific** behavior, i.e. the shark was tailored for a simulation involving bettas. The bettas had not been included in the shark's

context of his rules before. In a similar manner behavior was added to the scuba diver's bubbles.

"Wouldn't it be cool if the bubbles were toxic and would kill the fish when they hit 'em?" -- Ian

Coming back to the Pearl Street Project, the students had started planning a game, a scavenger hunt. The main character, a tourist, had to collect certain items and be aware of pickpockets, reckless bikers etc. The had divided the project up into the design of passive background agents and actual characters of the game, i.e. the tourist, the pickpocket, a street performer etc.

Programming above C-Level

One of the students who had been in the Agentsheets designer group from the start one day came to us before a Friday session and said he was sorry but a friend was going to teach him C and that was why he could not work with Agentsheets anymore. We were a bit frustrated (whoever knows the title screen of Agentsheets knows why) but had to let him go.

Only two weeks later, he was back. He rejoined the group and made a lot of contributions to the 'Pearl Street' project. I think that two factors brought him back from C to Agentsheets. One argument has to do with the work done by Clayton Lewis on design environments that include graphical objects as primitives so that rapid development of executable prototypes with interesting behavior is supported [Lewis, 1987]. The student had created several applications with Agentsheets in the first weeks with the group. The shift to C appealed to him

because, he said, it was ‘beneficial for his career’. But producing results similar to an Agentsheets simulation are out of reach of a C beginner.

But, in addition to that, the Agent Repository played a role in his comeback. At the time the student left the group the student designers had been encouraged to start to put some of their agents ‘on the web’. David, one of them, had created an animated agent of a juggler. He was very proud of his creation and showed everybody the webpage with his name and his agent. In the following weeks since then, the repository gained status. More students looked at it and it became important who was in it and with what. The repository was moving toward a critical mass among the students.

That realization became clear when I received an email one day from the student who had left the group earlier preferring to learn C. He asked me to include his name in the credits of the ‘tourist’ agent, the main character of the Pearl Street simulation. Until then, the students had not really cared what the descriptions had said who created what agents. Finally, it seemed the repository had gained importance and the students started to care.

This development supports my claim that the repository can motivate a critical mass of users to contribute resources and make it more valuable to every user.

At the same time the students in Boulder were working on the Pearls Street Project, BJ Allen-Conn at the Open Charter School in Los Angeles started to teach her students programming Agentsheets with Visual AgenTalk.

Talking about Artifacts ...

The collaboration with the school in Los Angeles started with phone conversations between me and the teacher, Mrs. Allen-Conn. The intention was to connect the students at Open Charter School (OCS) with the group in Boulder and watch them communicate through the repository. The goal was to observe how the students would reuse their counterparts' agents. But logistical problems i.e. with the installation, led to delays in the setup. In cooperation with Mrs. Allen-Conn I tried to help with the solution of those problems. On a weekly basis, a connection was established during class time in Los Angeles. The teacher met me on an IRC[13] server and support was given on-line. For less pressing problems, email was used.

Without any conscious effort direct communication naturally sneaked its way into the collaboration and it proved to be useful in preventing frustration in the early phase of the collaboration.

\

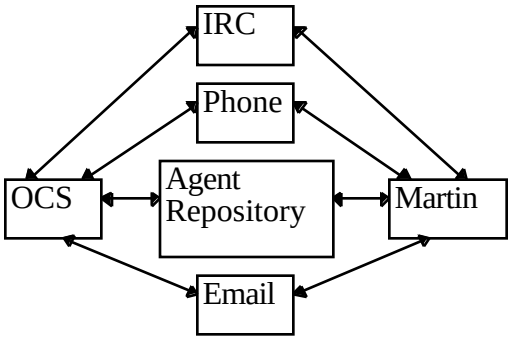

Figure 18: Over the course of the collaboration with OCS direct communication mechanisms proved to be more appropriate for solving the problems encountered.

[13] IRC=Internet Relay Chat, a protocol that allows multiple users to directly communicate on a 'channel' that is broadcast to all participants.

At one instance the students at OCS came across a problem when creating additional animals for the fish tank. The problem was that the fish they had created swam out of the screen area. And although they had created rules that checked for the edge of the tank the fish would not stay within tank.

The source of the problem was the ordering of the rules. The rule that checked for the edge of the tank was listed below the rules of moving around the tank. The fish kept moving because they never applied the rules that checked for the tank's edge.

The dialog between Mrs. Allen-Conn and me will illustrate the problems that arise when an informal description of a formal artifact must be facilitated to solve a problem in the formalized system.

Here is an excerpt from the IRC[10] chat between Mrs. Allen-Conn (bj) and me:

bj: How does a child make a square that means nothing

Martin: Means nothing? Has no behavior? I don't quite understand?

bj: If I want the fish to notice that he is at the end of the water, then if the fish doesn't see blue which is nothing, then her should not go anywhere or at least change his appearance

Martin: There is an 'empty' condition in the palette, that tests for 'nothing'

bj: Ok I probably have missed it

bj: I found the condition about empty. Talk me through what the rule should look like. I can not get it to stop at the end of the tank

Martin: On top of your behavior you have a number of rules (2 or 4 presumably) that check for empty and if they see empty on the right they change their depiction (to the one facing left) and move to the left one cell

Martin: Then come the movement rules (the ones you already have)

Martin: An empty rule looks like this (i.e. detect right edge) :

Martin: Condition : Empty (arrow right); Action : Change (the dot) (fish facing left) & Move (arrow left)

bj: The empty condition does not say if I see....it simply say empty. So I click on if empty is to the right then change my appearance to another appearance. Is this the correct way to create the rule

Martin: Yap

...

bj: Martin

bj: one of my kids just made a rule that says if I see ground beneath me then change my appearance to a different appearance and it will not work. What should the rule look like??

Martin: Condition: See (arrow down) (ground depiction) Action: Change (the dot) (whatever depiction)

bj: The problem is that I have this program that you and I are talking from on one computer and the children are working at 16 different stations all over the room. I would have to copy over their project via the server to this computer. The example that you gave for the change of appearance depiction is exactly what we have done and it won't work. It just swims off the world below the ground depiction

Martin: Maybe you have to move the rule up.

Martin: The rules are tested from top to bottom, if this particular rule is the last one, there is a good chance it is never tested. You can drag and drop it higher that will make sure it fires.

bj: I got it .. This is the same sequencing problem we run into all the time with Cocoa. The kids have to put their rules in the right order.

The description and framing of the problem took almost five minutes. Had I been physically present I would have grasped very quickly what the problem was - the sequence of rules. But the translation into text and my translation back into rules added so much "noise" to the program that even a simple problem like this took about five minutes to be resolved where seconds should have been enough.

Had my repository worked more efficiently at the time this intermediate translation could have been avoided. I would have told Mrs. Allen-Conn to submit the agent in question and could have retrieved it and examined its behavior in my Agentsheets environment. For such and similar occasions an extension to the structure of the repository would be useful - an area (page) where one could upload agents to that would be erased regularly, thus serving more immediate needs than the rest of the repository. Another advantage of such a 'scrap page' would be the lower resistance of people to share agents 'under construction'.

The class in Los Angeles is still learning to program with Visual AgenTalk and has not started their project yet. They want to build a simulation of Venice Beach. The class in Boulder's Centennial Middle School has created many autonomous agents that populate the simulation worksheets of Pearl Street. A collaboration between the two schools as planned has not yet become reality.

Conclusions

In this chapter I will show results that relate to the issues stated in the ‘Problem Statement’.

The Shift from Direct to Indirect Communication

In my opinion, the problem with indirect media such as the Agent Repository, in the early phases of learning to program is that the learners’ model of the system model is still vague.

Therefore they do not yet have a sense for how to reflect on and resolve breakdowns. Consequently, they cannot comprehend artifacts that they did not create themselves. But the comprehension of another user’s behavior rules is the key skill to use an indirect medium like the Agent Repository.

With more experience an analysis of a breakdown or the comprehension of unknown artifacts is easier due to a more accurate mental model of the processes that led to the breakdown. From a constructivist’s perspective, the construction of such a mental model constitutes the learning.

A beginner lacks not only the necessary knowledge to reason about a breakdown but even the understanding necessary for knowing where to start an analysis.

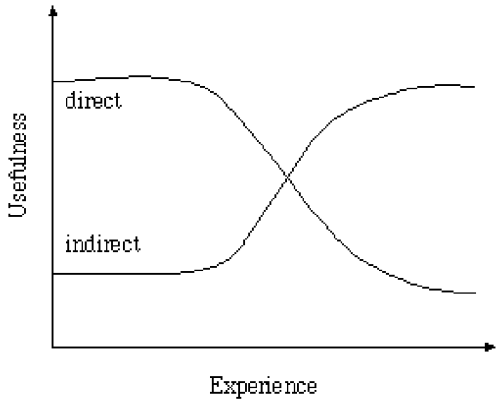

Figure 19: The usefulness of direct and indirect communication depend on the users' experience. An inexperienced user benefits more from direct communication than an experienced user, whereas the relationship seems to be reciprocal for indirect communication.

Direct communication has the advantage for an inexperienced user being able to react to a breakdown immediately. Instant feedback is necessary to avoid frustration. Instead of an analysis that would explore the solution or explanation space, the beginner is able to remain in the problem space and describe his or her trouble to more experienced users in the virtual space of synchronous communication. Human support in this phase is essential because the help must take the mutual ignorance into account. Questions and suggestions must be formulated in a language that takes the limited understanding of the inexperienced user into account.

In my IRC conversations with Mrs. Allen-Conn this was a frequent problem. I would refer to an Agentsheets component in an explanation of a procedure, but would not be aware that these terms, i.e. 'action palette', were not as firmly embedded in her vocabulary as they were in mine. This was often the source of confusion. The

episode that was recounted in the last chapter led me to the conclusion that for this kind of introductory instruction an Agentsheets MUD would be much more useful than an Agent Repository. I could have looked at the behavior in question the moment a problem would come up being in the same virtual space as the problematic agent.

At two occasions I tried to employ the Agent Repository for purposes of illustration. I created agents whose behavior was designed to illustrate for example the use of variables. I would email Mrs. Allen-Conn that I had put an agent into the repository that would help her understand how variables can be utilized in Visual AgenTalk. It was weeks later, that I realized that she did not have the AgenTalk programming skills yet to even understand what I had done.

The students in Boulder did not use the repository in the first two months. Although I encouraged them to upload their agents, which they did, they never went back to the repository to retrieve anything.

Recently, as they became more experienced in programming Visual AgenTalk they relied increasingly on agents from the web. They also took a stronger interest in their own creations to publish them in the repository. As an example the episode that was described in the previous chapter can be used, where one of the students sent me email asking me to add his name to the authors of an agent that he had created with another student. The other student had uploaded the agent so that he was credited as author.

To summarize the hypothesis, to which these experiences have led me, I think that direct and indirect communication address different

information exchange needs that occur in different phases of a user's learning process.

In fact I would go one step further, and say that indirect communication replaces the direct communication as the preferred mode of communication with growing experience.

Context supports Learning, Projects help to Focus

One assessment that I have made through the course of the study at CMS is that working on projects helps to focus the students' attention.

The first piece of evidence for this result was found in the first weeks of the collaboration with CMS. The group that stuck together to create a fighter plane game not only was more productive than the individuals' projects but they also grasped the idea of rule-based programming faster than the rest of the students. That was not too surprising since they had each other to reflect upon what they were doing as well watching what other group members did, i.e. one student had animated agents and shared his knowledge with other group members when they faced the same challenge.

But besides the benefits that come with group work, the common context in which all group members are working is the basis for a shared understanding and a shared language.

This proved to be beneficial to Harold, a newcomer that joined the group very late. Building on the established shared knowledge and the focused work on the Pearl Street project in the group he grasped the ideas behind Visual AgenTalk very fast. After only two weeks he was among the most proficient programmers.

Most students worked in roughly the same simulation environment, only different with respect to the agent or agents they were working on. Harold could go around and look at anyone's screen and immediately be familiar with the context of the project having seen it on every screen he had looked at before. That way he was free to concentrate on the particular aspects that the student he observed was working on and could pick up something new from about everyone.

Advantages of Collaboration in Class and between Classes

As described in the previous chapter, the students came to the realization that collaboration was beneficial for them when we encouraged them to reflect on the progress of the first two weeks. We did not have to point out to them what the difference was.

In my hypothesis with respect to inter-class collaboration between CMS and the Open Charter School I was assuming that such collaboration would yield the same kind of benefits making the design more efficient and productive.

Unfortunately, the students in Los Angeles so far did not reach a level of expertise that would allow collaboration. Mrs. Allen-Conn had talked about having the students in Los Angeles build a simulation of Venice Beach and then have both groups of students exchange their artifacts and import them into their own simulations. An actual exchange of agents never happened.

Nevertheless, the hypothesis is supported by the work that the students did with the fish tank project. In addition to the episode of the previous chapter on one occasion a student who had not worked with the group before expressed interest in Agentsheets. The students

were now proficient enough using Agentsheets that we let the explain the system and its programming to newcomers.

Interestingly, the student demonstrating Agentsheets dragged the scuba-diver agent from the repository into the gallery of the fish tank project. Then he started explaining how to program with Visual AgenTalk adding rules to the scuba diver.

This supports my earlier claim that agents are taken out of the repository for their basic behavior and then modified to meet special needs. It also shows an indirect collaboration between the creator of the scuba diver and the student and how the student could benefit from something somebody else had built and shared on the web.

Critical Mass

The questions concerning the critical mass can only be answered with the limitations due to the small test user community and the short time of actual employment. In earlier sections I have described the Qualitative use of the repository. In this section numbers will demonstrate the expansion of the repository. In Figure 20 and Figure 21 the number of contributed agents is graphed over the time span of the study.

The first three numbers reflect a seeding process. The agents built in the information society class were submitted first. The twelve agents labeled ‘CMS seed’ were an attempt to give the students an idea of the kinds of agents that would make up a simulation of a city.

In the following weeks only a very small number of agents were submitted as the students at CMS were learning Visual AgenTalk and the Pearl Street Project was still in an early stage.

On October 25th, the use of the repository jumps to another level. This was first general upload. Most agents that were part of the Pearl Street Project were added to the webpage. From then on the usage remained on a high level.

Uploading agents to the repository became an integral part of the sessions. Towards the session's end the students would upload their agents.

Submission Date	Source of Agents	Number of Agents
Sep, 24th	"Information Society" class	9
Sep, 26th	CMS seed	12
Oct, 4th	New Fish	4
Oct, 8th	David Fier's Juggler	1
Oct, 9th	Tourist for Pearl Street	3
Oct, 15	New Fish	1
Oct, 25th	Pearl Street Project	16
Nov, 1st	Pearl Street Project	21
Nov, 8th	Pearl Street Project	14

Figure 20: Quantitative Evaluation of the Agent Repository.

The answers to why the change in use of the repository happened so abruptly and what caused the change must be further investigated. At this point these questions remain open.

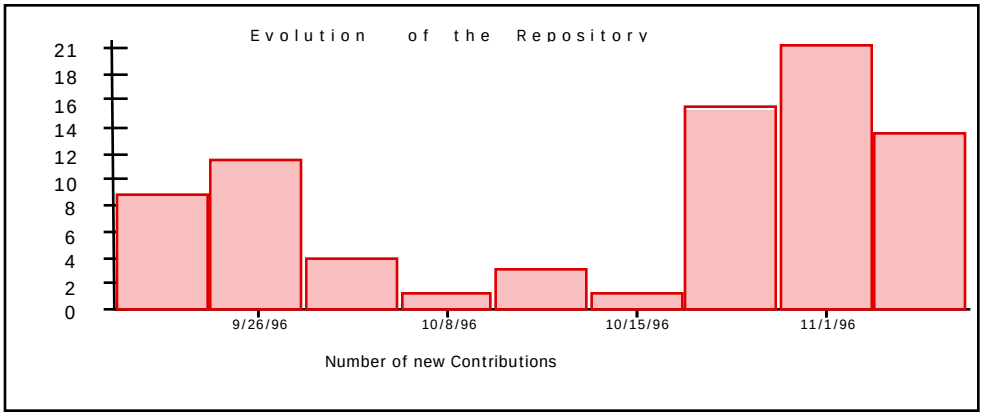

Figure 21: Graph of the Quantitative Evaluation of the Agent Repository

Future Directions

CL-HTTP

One easy yet very effective improvement in the future would be to combine the code that updates the repository pages with a Macintosh-based HTTP (Hypertext Transfer Protocol) server. With CL-HTTP, there exists such a server that is implemented in Common Lisp (CL).

The advantage of a server implemented in Lisp for my purposes would be that one could call Lisp functions instead of CGI (Common Gateway Interface) scripts. So, a reference to a URL could trigger the dynamic creation of a web page including all those agents that are currently in the repository - written in Lisp. In contrast to the present implementation the question of intervals at which to update the agent-pages would be removed since every request for an agent-page would result in an image of the agent repository at the very moment a request was made.

There would still be the question of when to check the email account for new income agents but this updating happens one level further away from the webpage user and in addition is done much faster than updating webpages and thus less likely to be critical.

The increased flexibility would most likely lead to a longer response time for the web-pages containing agents because of their dynamic creation. I would have to carefully weigh the benefits against the costs and see what seems more important to the users, a speedy download or more accuracy of the web-pages. At this point I assume that the accuracy is more valuable. So far the user behavior suggests that a user uses only one agent-page while working on a project in Agentsheets

and that the higher cost of a longer response time would be an acceptable trade-off.

Besides making the repository respond faster to new submissions, the new version would have another advantage. With the agents better accessible through Lisp functions some very useful features could be added, i.e. search capabilities, detecting dependencies between agents automatically, putting the same agent under more than one topic, control different versions of the same agent and eventually view even the behavior of the agents in the web-browser, thereby diminishing the importance of the textual description. Search capabilities will gain importance once the repository achieves more universal acceptance and will start growing at a much faster rate. The problem of location is at the present state not critical yet, but will become more important as the number of agents in the repository increases and makes browsing more and more impractical.

Employ Mechanisms for other Language Components

One obvious development for the Behavior Exchange is to employ the mechanisms of the Agent Repository to other language components such as commands.

As stated above, packaging several agents and resources like commands or sounds together is another extension that would increase the system's usefulness.

Use a GIMMe-like system for location and placement of agents

Semantic indexing as used in the GIMMe system could improve the current system in two ways. It could help automating the process of mapping an agent's topic to the appropriate webpage.

In addition to that, the mechanism could be used for better support of locating agents in a semantically enriched search engine.

An Agentsheets MUD

The study and especially the collaboration with OCS in Los Angeles have showed that synchronous communication mechanisms are better fit for the early stages of learning Visual AgenTalk. A promising idea is to combine Amy Bruckmann's results with the end-user programming approach of Visual AgenTalk.

An Agentsheets MUD would address the issues of collaborating on constructions.

References

Bruckmann, A. (1994)
"Programming for Fun: MUDs as a Context for Collaborative Learning."
National Educational Computing Conference, Boston, MA, International Society for Technology in Education
Available via anonymous FTP from media.mit.edu in pub/asb/papers/necc94.{ps.z, rtf.Z, txt}

Fischer, G. (1995)
"Turning Breakdowns into Opportunities for Creativity"
Journal of Knowledge-Based Systems, special issue on Creativity and Cognition

Fischer, G., McCall, R., Ostwald, J., Reeves, B., Shipman, F. (1994)
"Seeding, Evolutionary Growth and Reseeding: Supporting Incremental Development of Design Environments"
Human Factors in Computing Systems, CHI '94 Conference Proceedings, ACM, 1994, pp. 292-298

Fischer, G., Grudin, J., Lemke, et al. (1992)
"Supporting Indirect Collaborative Design with Integrated Knowledge-Based Design Environments"
Human-Computer Interaction, Vol. 7, 1992, pp. 281-314

Fischer, G., and Reeves, B. (1992)
"Beyond Intelligent Interfaces: Exploring, Analyzing, and Creating Success Models of Cooperative Problem Solving"
Journal of Applied Intelligence 1, pp. 311-323

Fischer, G, (1991a)
"Supporting Learning on Demand with Design Environments"
International Conference on the Learning Sciences, pp. 165-172

Fischer, G. et al. (1991b)
"The Role of Critiquing in Cooperative Problem Solving"
ACM Transactions on Information Systems, Vol. 9,No. 3,April 1991, pp. 123-151

Fischer, G., et al. (1991)
“Cognitive Tools for Locating and Comprehending Software Objects for Reuse”
Proceedings of the 13th International Conference on Software Engineering, Austin, Texas, 1991, pp. 318-328

Kay, A. (1991)
“Computers, Networks and Education”
Scientific American, September, pp. 138-148

Gindling, J., Ioannidou, A., Loh, J., Lokkebo, O., and Repenning, A. (1995)
“LEGOsheets: A Rule-Based Programming, Simulation and Manipulation Environment for the LEGO Programmable Brick,”
Proceedings of Visual Languages, Darmstadt, Germany, 1995, pp 172-179

Grudin, J. (1994)
“Groupware and social dynamics: Eight Challenges for Developers”
Communications of the ACM 37(1), pp. 92-105

Lewis, C. (1987)
“NoPumpG: Creating Interactive Graphics with Spreadsheet Machinery:,”
University of Colorado at Boulder, Technical Report, CU-CS-372-87, Dept. of Computer Science

Lindstaedt, S. (1996)
“Towards Organizational Learning: Growing Group Memories in the Workplace”
CHI ‘96, Doctoral Consortium, Vancouver, BC, Canada

Markus, M. (1990)
“Towards a ‘Critical Mass’ Theory of Interactive Media
in Fulk, J. and Steinfield, C. (Eds.)
Organizations and Communiation Technology
Sage Publications, pp. 194-218

OCS (1996)
“The Open Charter School profile”
can be aquired from
Open Charter School
6085 Airdrome Street
Los Angeles, California 90035
Telephone: (203) 937-6249; Fax: (213) 937-2884

Oliver, P. E., Marxwell, G., & Teixeira, R. (1985)
“A Theory of Critical Mass I. Interdependence, Group Heterogeneity, and the Production of Collective Action.”
American Journal of Sociology, 91(3), pp. 522-556

Repenning, A., Ambach, J. (1996)
“Tactile Programming: A Unified Manipulation Paradigm Supporting Program Comprehension, Composition and Sharing”
Proceedings IEEE Symposium of Visual Languages, Boulder, Colorado, 1996, pp. 102-109

Repenning, A., (1993)
"Agentsheets: A Tool for Building Domain-Oriented Dynamic, Visual Environments"
University of Colorado at Boulder, Ph.D. dissertation, Dept. of Computer Science, 171 Pages.
(Available as Tech Report CU-CS-693-93)

Resnick, M. (1996)
“Distributed Constructionism”
in Proceedings of ICLS ‘96, International Conference on the Learning Sciences, 1996, E.C. Edelson, E.A. Domeshek (Eds.), AACE, pp. 280-284

Resnick, M., Rusk, N. (1996)
“The Computer Clubhouse: Preparing for Life in a Digital World”
IBM Systems Journal, Vol. 35, NOS 3 & 4, 1996, pp. 431-439

Turkle, S. (1995)
“Life on the Screen: Identity in the Age of the Internet”
Simon & Schuster, New York, 1995

Yaeger, L
“Vivarium History: The Vivarium Program”
http://www.research.apple.com/people/Yaeger/VivHist.html

Appendix A : Technical Notes

The Agent Repository requires two software modules. One module runs on an assigned Macintosh servicing the repository, creating GIF pictures, update pages etc. and the other is an extension to Agentsheets to allow users to upload their agents to the repository from wherever they are.

The programs were written in Macintosh Common Lisp and two TCP/IP protocols were employed, SMTP (Simple Mail Transfer Protocol) and POP (Post Office Protocol).

Appendix B : Agentsheets & Visual AgenTalk

Agentsheets & Visual AgenTalk

Agentsheets [Repenning, 1993] is a programming substrate for creating domain-oriented programming and simulation environments. The construction paradigm employed by Agentsheets consists of a large number of autonomous, communicating agents organized in a grid called the agentsheet. Agents support different communication modalities such as animation, sound, and speech.

Visual AgenTalk [Repenning, Ambach, 1996] is a tactile end-user programming language substrate. Tactile programming allows users to conceptually grasp meaning by physically grasping language components. Comprehending by touching.

By blurring boundaries between application worlds (the objects within a simulation), programming worlds (the objects of the language - commands, rules, etc.) and collaboration worlds (the objects that can be exchanged by users) Agentsheets creates a unified manipulation paradigm.

Programs can be used and adapted via drag and drop from web pages. This part of the environment is called the Behavior Exchange.

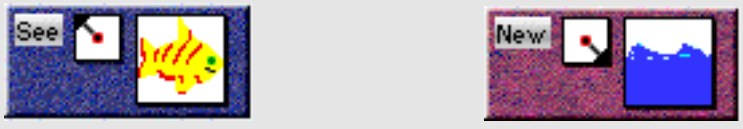

Figure 22: A condition (blue) & a action (red)

Agentsheets projects can be created using Visual AgenTalk. Each project consists of one or more galleries (see Figure 3). The galleries are collections of agents. These agents in turn are instantiated in one of the workspaces of the environment, the worksheets (i.e. Figure 10). When using Visual AgenTalk agents are defined by their appearance in connection with their behavior (see Figure 23). The behavior of an individual agent is defined by a set of methods in which rules are grouped together. Each rule represents a number of actions that are executed if the conditions of the rule are evaluated to be true (see Figure 22).

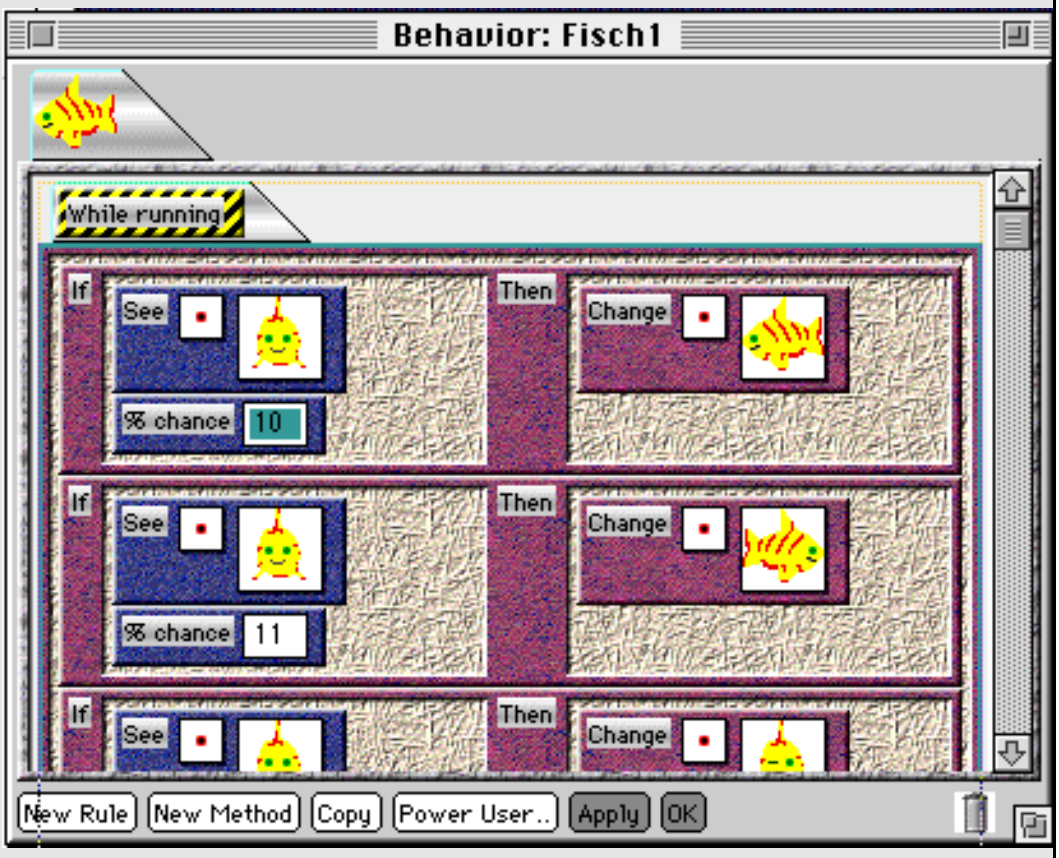

Figure 23: A behavior browser

The Agentsheets language architecture is open and supports the creation of domain-oriented languages. Not only can one create one's

own agents and projects but the language itself - the commands and the control structures - are open to be modified and supplemented.