Building Brains for Rooms: Designing Distributed Software Agents

Michael H. Coen

MIT AI Lab
545 Technology Square
Cambridge, MA 02139
mhcoen@ai.mit.edu

Abstract

This paper argues that complex, embedded software agent systems are best constructed with parallel, layered architectures. These systems resemble Minskian Societies of Mind and Brooksian subsumption controllers for robots, and they demonstrate that complex behaviors can be had via the aggregates of relatively simple interacting agents. We illustrate this principle with a distributed software agent system that controls the behavior of our laboratory's Intelligent Room.

Introduction

This paper argues that software agent systems that interact with dynamic and complex worlds are best constructed with parallel, layered architectures. We draw on Brooks' subsumption architecture (Brooks, 1985) and Minsky's Society of Mind (Minsky, 1986) theory to dispel the notion that sophisticated and highly capable agent systems need elaborately complex and centralized control.

Towards this end, we present an implemented system of software agents that forms the backbone of our laboratory's "Intelligent Room" (Torrance, 1995). These agents, known collectively as the *Scatterbrain*, control an environment very tenuously analogous to the intelligent rooms so familiar to Star Trek viewers --- i.e., rooms that listen to you and watch what you do; rooms you can speak with, gesture to, and interact with in other complex ways.

The Scatterbrain consists of approximately 20 distinct, intercommunicating software agents that run on ten different networked workstations. These agents' primary task is to link various components of the room (e.g., tracking cameras, speech recognition systems) and to connect them to internal and external stores of information (e.g., a person locator, the World Wide Web). Although an individual agent may in fact perform a good deal of computation, we will focus our interest on the ways in which agents get connected and share information rather than how they internally manipulate their own data. And while the Intelligent Room is a fascinating project in itself, we will treat it here mainly as a test-bed to learn more about how software agents can interact with other computational and real entities.

Our approach has also been modeled on a somewhat unorthodox combination of the Brooks (Brooks, 1991) and Minsky approaches to core AI research. As pointed out in (Kautz et. al., 1994), it is difficult to find specific tasks for individual agents that are both feasible and useful given current technology. Many of the non-trivial tasks we would like software agents to perform are simply beyond the current state of the art. However, taking our cue from Minsky, we realize interesting and complex behaviors can be had via the aggregates of simpler ones; groups of simple agents can be combined to do interesting things. We also found Brooks' subsumption architecture useful for guiding the creation of the Scatterbrain, particularly for building parallel layers of behaviors that allow the room to process multiple events simultaneously and to change contexts quickly. In many ways, the room is similar to a disembodied robot, so it comes as no surprise that the robotics community can provide insight into how the room's brain should be designed. We argue, however, that this does not preclude insights obtained in creating the Scatterbrain from applying to other distributed software agents systems. Rather, as argued by Etzioni (Etzioni, 1994, 1996; Etzioni et al. 1994), even agents who solely interact with the online world (and don't have cameras for eyes and microphones for ears) can be viewed as a kind of simulated, virtual robot. More important than its connection with the real world, what the Scatterbrain shares with Brooks' robots is its organizational structure and its lack of central processing; all of the Scatterbrain's agents work together in parallel with different inputs and data being processed simultaneously in different places.

The next section of this paper describes the Intelligent Room's physical infrastructure. After this, we introduce the room's most recent application, a *Tour Guide* agent that helps a person present our lab's research to visitors.

---

[2]This material is based upon work supported by the Advanced Research Projects Agency of the Department of Defense under contract number F30602—94—C—0204, monitored through Rome Laboratory and Griffiss Air Force Base.

Next, we present in detail the room's software agent architecture, including the design and implementation of several components of the Scatterbrain and Tour Guide agents. We also contrast our approach with several earlier monolithic efforts taken in our lab to program and control the behavior of the Intelligent Room.

Part of the motivation for this work has been to push the envelope of software agent design. Much has been made over the lack of obvious "killer applications" for software agents. After all, how many automated meeting schedulers does the world need? We are interested in exploring new realms of complex interactions for software agents which in and of themselves constitute these "killer apps" that have been seemingly so elusive from the single-agent perspective. Minsky argues that societies of agents can do things that seem inordinately complex when their behavior is viewed as the work of a single entity. Our experiments with fairly large assemblies of software agents mark an early attempt towards establishing that this is indeed the case.

The Intelligent Room

The Intelligent Room project explores new forms of interaction and collaboration between people and computers. Our objective is to create a new kind of environment capable of interpreting and contributing to activity within it. On a grand scale, we are examining a paradigmatic shift in what it means to use a computer. Rather than view a computer as a box with a keyboard and monitor used for traditional computational tasks, we envision computers being embedded in the environment and assisting with ordinary, traditionally non-computational activity. For example, if I lose my keys in the Intelligent Room, I'd someday simply like to ask the room where I left them.

The Intelligent Room is an excellent environment in which to conduct core AI research according to the criteria of both Brooks (Brooks, 1991) and Etzioni (Etzioni, 1994), The room is "physically" grounded in the real-world. The room's cameras and microphones allow it to deal allow it to deal with the kinds of complex, unpredictable and genuine phenomena that Brook's argues is essential for a core AI research testbed. However, the room also processes abstract, symbolic information that actually represents something extant, thereby satisfying Etzioni's desiderata. For example, if a person asks the room, *What is the weather in Boston?*, the room needs to recognize more than a meaningless *weather* token - it needs to get that information and display it to the user. This is done using a variety of information retrieval systems available on the World Wide Web.

This section first describes the room's physical infrastructure. We then present the room's most recent application, a *Tour Guide* agent that helps a person present our lab's research to visitors. In the next section, we discuss in detail the room's software agent architecture, including the design and implementation of the Scatterbrain and Tour Guide agents.

Infrastructure - From the bottom up

Figure 1 diagrams the room's physical layout. The Intelligent Room's infrastructure consists of several hardware and software systems that allow the room to observe and control its environment in real-time. The positions of people in the room are determined using a multi-person tracking system. Hand pointing actions are recognized by two separate gesture recognition systems. The one used in the application described below allows the room to determine where someone is pointing on either of two images projected on a room wall from high-intensity, ceiling mounted VGA projectors. A speech recognition system developed by (Zue, 1994) allows the room to listen to its inhabitants, and it is used in conjunction with a speech generator to enable to the room to engage in sustained dialogues with people. The room interfaces with the START natural-language information retrieval system (Katz, 1990) to enhance its ability to understand complex linguistic input. The room also controls two VCRs and several other video displays in addition to the ceiling mounted projectors. A matrix switcher allows arbitrary connections between the room's audio/visual inputs and outputs.

The room's hardware systems are directly interfaced with low-level C programs to insure their real-time operation. For example, the room's tracking cameras have 30 Hz frame rates and their data streams need to be synchronously processed using direct operating system calls.

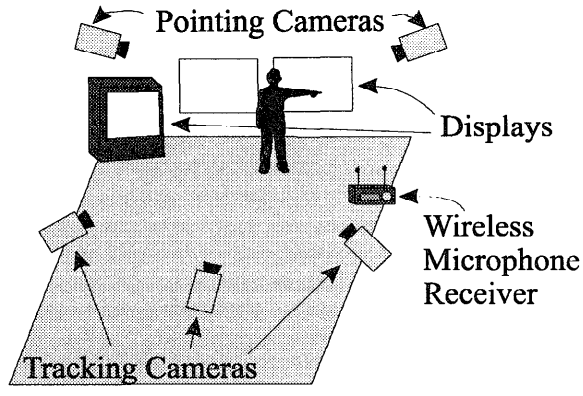

Figure 1 - Intelligent Room Floor Plan

The Tour Guide Agent

The room's most recent application provides support for someone giving tours of our laboratory. These tours typically involve a group of visitors meeting with a graduate student who discusses and answers questions about the lab's research and shows several video clips. Rather than have these presentations given in an ordinary

conference room, we have decided to have them in the Intelligent Room so the room can assist the human tour guide. A typical dialogue between the room and student tour guide is:

Tour guide: *Computer, load the AI Lab tour.*
Room: *I am loading the AI Lab tour.* Right projector now displays a Netscape browser window with a special Lab Tour home page.
Tour guide: Using hand, points at link to a research project displayed on the wall and says, *Computer, follow this link*
Room: Loads the indicated page into the browser.
Tour guide: *Computer, show me the Intelligent Room home page.*
Room: Loads the URL corresponding to the name of the page. Then says, *I have a video clip for this research. Would you like to see it?*
Tour guide: *Computer, yes.*
Room: Moves appropriate video casettetape to correct position and starts the clip playing on left projector.
Tour guide: (watches video for a few seonds) *Computer, stop the video. Computer, play the Virtual Surgery clip.*
Room: Performs requested action. Stops video when clip is done.
Tour guide: *Computer, how many graduate students are there at the AI Lab?*
Room: *I am asking the START system for the answer...The Laboratory's 178 members include 17 faculty members, 26 academic staff, 29 research and support staff, and 106 graduate students.* Also displays web page with elaborated answer.

Other applications include a control center for planning huricane disaster relief and an intelligent living room.

Control Architectures

The room has been discussed so far at its most concrete and most abstract: namely, its hardware infrastructure and its high-level software applications. How these applications are actually created on top of this infrastructure, i.e., how the room actually works, is the subject of this section.

Monolithic Control

In its early stages of development, each of the room's components was wrapped inside a TCP client or server that connected with a monolithic C-language program that controlled the room's behavior. Figure 2 contains this controller along with each of the programs it connected with. (Included in parentheses with each component is the name of the workstation it ran on.)

From a conceptual point of view, the most serious flaw with the centralized controller was that it failed to distinguish between basic functioning common to all room contexts –such as noticing when someone came through the doorway – and unique activities associated with a particular room application. Furthermore, adding new functionality to the room required modifying the monolithic controller and manually determining the interactions and conflicts between old and new room

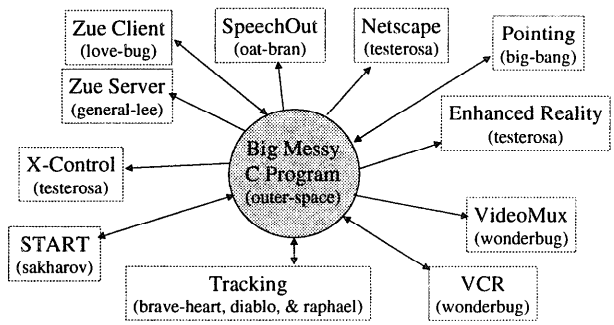

Figure 2 - The Monolithic Controller

functions. There was no way to modularly add new room capabilities on top of old ones and assume everything would continue working as expected.

Also, directing the information flow among the room's various components –one of the main functions of the controller – was overly difficult in a language like C. We needed higher-level mechanisms for describing how room information moved among its producers and consumers.

From a practical point of view, the monolithic controller also made it difficult to reconfigure the room dynamically or restart pieces of the room independently of others. We often found while working on the room that in order to restart one component, it was necessary to restart the entire room. This was particularly frustrating because starting the room required the coordinated activity of several people to start particular programs (in a predetermined order) and configure various room hardware. It was also difficult to move components of the room to different workstations because that required modifying hard-coded machine dependencies in the code.

SodaBot

Although we managed to use the monolithic approach for several very simple applications, it seemed unlikely to scale to the more complex interactions we had in mind for the room. Our initial dissatisfaction with this architecture led to the adoption of the SodaBot software agent platform (Coen, 1994) for duplicating the functionality of our initial monolithic room controller with a system of distributed software agents.

SodaBot provides both a programming language and runtime platform for software agents, and it simplifies writing interactive programs by providing high-level primitives for directing flows of online information. For example, it provides mechanisms for writing agent-wrappers that interface with preexisting software either via text-based or graphical user interfaces (X-windows and Windows 95/NT).

For example, we created a SodaBot *Netscape Agent* that controls interactions with a Netscape browser. It offers functions to other agents such as those listed below.

Function	Purpose
New (host)	Runs a new browser on given host
Load(url)	Loads URL in browser
Page_watch()	Arranges for notification (of URL) to another agent whenever browser loads new page
Link_watch()	Arranges for notification to another agent when a new page is loaded containing its URL/anchor text pairs
Text	Returns text of current page
Page(direction)	Moves browser scroll-bar in given direction

For the Intelligent Room, we use SodaBot agents as *computational glue* for interconnecting all of the room's components and moving information among them. Initially, we simply duplicated the room's monolithic controller using SodaBot's high-level programming language. Most notably, SodaBot simplified description of room functioning and interaction with remote TCP-based clients and servers by removing networking and hardware details. However, this new room controller, dubbed the *Brain*, was still a computational bottleneck, and we had yet to distinguish between a general behavioral infrastructure for the room (i.e. its core functionality) and the more complex, application specific interactions we built on top of it. This led to the development of the room's current control system, the Scatterbrain, which is the subject of the next section.

Distributed Room Control

The Scatterbrain (Figure 3) is platform on top of which room applications can be layered. In the figure, each circle represents a distinct SodaBot software agent that is wrapped around and interfaced with an external application. (The layer containing these "base applications" is not shown.) Each of the Scatterbrain agents is responsible for a different room function. For example, the *SpeechIn Agent*, runs and interfaces with our speech recognition systems. Once started, *SpeechIn* allows other agents to submit context-free grammars corresponding to spoken utterances they are interested in. As they do, it updates the speech recognition systems to contain the current set of valid utterances. When a sentence is heard by one of the speech systems, SpeechIn then notifies those agents who indicated they were interested in it. As another example, the *Netscape Agent* connects to the *Display Agent* to make sure that when web pages are loaded, the browser is actually displayed somewhere in the room where people can see.

The Scatterbrain agents are distributed among 10 different workstations and rely on SodaBot interagent communication primitives to locate and communicate with each other. The lines in the figure represent default interactions the room manifests in all applications, such as having various agents connect with the speech recognition agents and making sure the tracking system notices when someone comes in the room. Essentially, the Scatterbrain implements the Intelligent Room's reflexes.

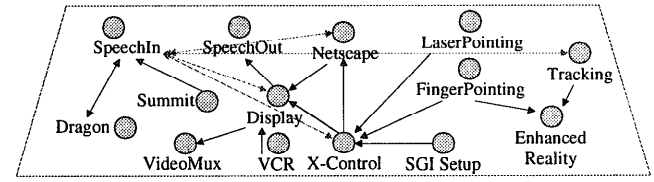

Figure 3 – The Agents of the Scatterbrain

The room no longer has a central controller. A small startup agent runs all of the Scatterbrain agents which then autonomously move to the machines on which they are supposed to run. All the Scatterbrain agents then work together in parallel with different inputs and data being processed simultaneously in different places. This makes the room robust to failure in one of its sub-systems and allows us to restart sections of the room independently. Also, the SodaBot system allows real-time data connections between agents to be broken and resumed invisibly. For example, if the *Tracking Agent* is updating another agent in real-time, either one of them can be stopped and restarted and they will be automatically reconnected.

Layerd on top of the Scatterbrain, we created higher-level agents that rely on the Scatterbrain's underlying behaviors. Figure 4 contains the room's intermediate information-level applications such as a *Weather Agent* that can obtain forecasts and satellite maps for particular places. By relying on the previously described interaction, if the Weather Agent uses the Netscape Agent to display information, it doesn't need to be concerned with insuring the browser is displayed in a place where the user is looking.

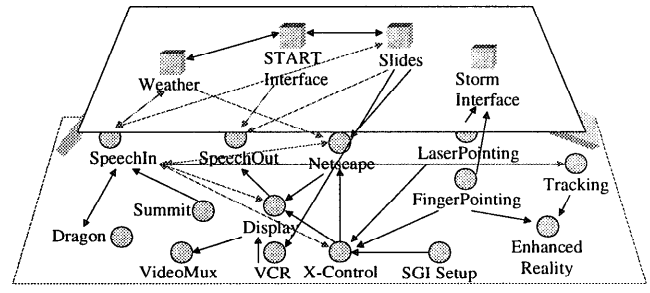

Figure 4 – Information Agents

We then created specific room application agents that relied on the lower-level, general-purpose agents in the room. Figure 5 contains a diagram of several room applications and how they connect to the room's underlying architecture. Note that all of the objects in the figure represent SodaBot software agents and many of

them connect to non-displayed external applications. The next section explores two of the application agent interactions in more detail.

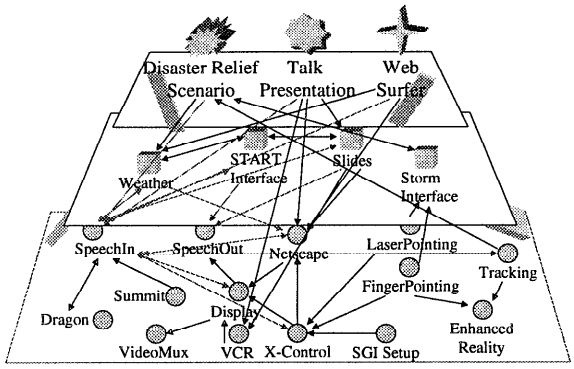

Figure 5 – Intelligent Room Software Agents

Agent Interaction

This section examines how we can get the room to exhibit interesting behavior by layering agents on top of each other. We examine two separate room behaviors and then discuss how they combine to produce greater functionality.

We have a system in the room called *Storm*, used in a disaster relief planning scenario, that can display scalable maps of the Carribean. People can interact with Storm using pointing and speech. For example,

User: *Computer, display Storm on the left projector.*
(User now points at Puerto Rico.)
User: *Computer, zoom in.*
(User now points at San Juan.)
User: *Computer, what is the weather here?*
(The room then displays a weather forecast for San Juan inside a Netscape browser on the other projector.)

To see how this scenario works, we first examine pointing recognition as an example of simple agent interaction. We then look at a more complex scenario from the Tour Guide agent presented earlier.

By default, the room's projectors are set by the *Display Agent* to show portions of the screens of two of our SGI workstations. If someone points someplace close to one of these projected displays, the display's mouse cursor moves to that position. Although this seems like a trivial process, there is a fair amount of effort behind it, as shown in Figure 6. The person moving his finger is reflected in the camera images received by the neural network pointing software. This reconciles the images to produce new pointing information. These new data are passed to the *FingerPointing Agent* which is responsible for handling all such events in the room. By default, the Scatterbrain has all pointing events on the each display sent to an agent called the *X-Server* that controls the actual SGI workstation generating the display. This X-Server agent then moves the mouse cursor to the appropriate position, which is reflected in the displayed image. However, the Storm Agent overrides this default behavior and redirects pointing events on the Storm display to itself. Upon receipt of a pointing event, it updates the Storm application's internal cursor, which moves intelligently between salient geographical features. For example, pointing near San Juan will cause the Storm program to register the city with the Storm Agent, rather than a point three pixels to its left. Finally, note that the various agents are responsible for translating between the room's many coordinate systems, as shown along the connections.

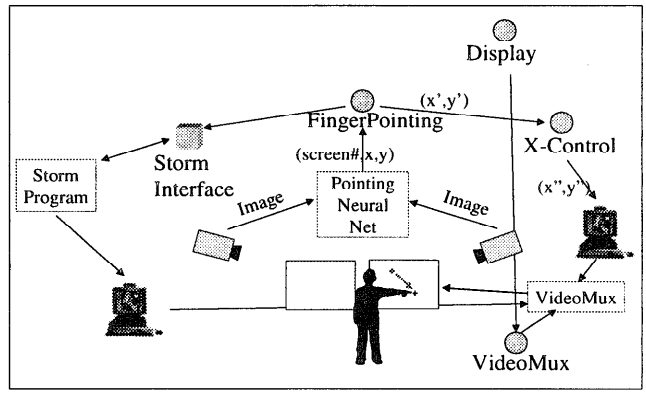

Figure 6 – Pointing in the Room

When someone in the room says *What's the weather here?*, the *SpeechIn Agent* notifies the room's *Disaster Relief Planning Agent* because this utterance is contained in the grammar that agent had registered when first run. The *Storm Agent* is then contacted to determine what geographical entity is closest to where the person was pointing close to the time they asked the question. (Low-level room events are time-stamped by agents in order to facilitate multimodal reconciliation.) This process is shown in Figure 7.

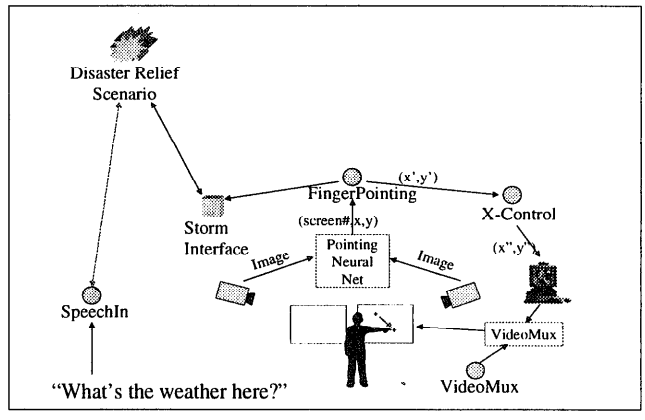

Figure 7 – Multimodal Resolution

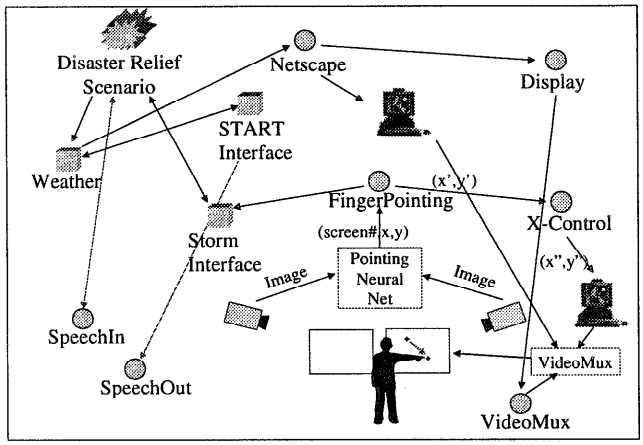

Figure 8 – Loading Weather in Browser

When the Disaster Relief Agent is told what region's weather is being queried, the *Weather Agent* is then asked to display the requisite information. After consultation with the *START Agent* to find an appropriate URL which contains this information, it asks the *Netscape Agent* to load the given page as shown in Figure 8, which also displays the complete agent interaction for handling the user's question.

A separate interaction from the Tour Guide Agent presented earlier is shown in Figure 9. Here's someone asks the room to load a particular web page, e.g., *Computer, load the Intelligent Room home page.*

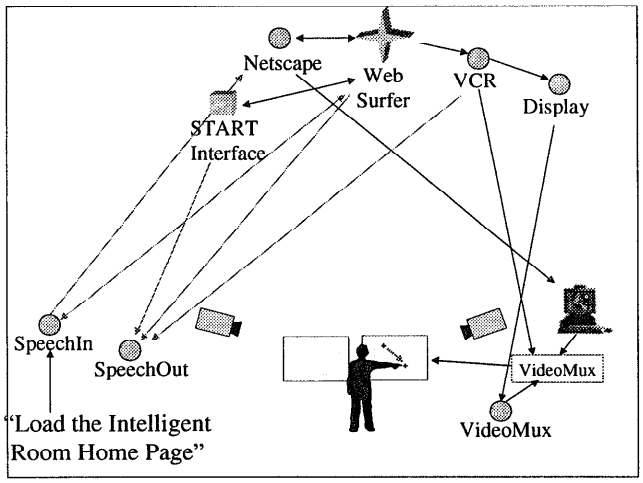

Figure 9 – Video Notification

After the Netscape Agent receives this request from the SpeechIn Agent, it loads the URL in the netscape browser. Whenever the Netscape Agent loads a new page, it also notifies the *Web Surfer Agent* that it is doing so. The Web Surfer Agent consults with the Start Agent to check if it has any additional information about the content of the newly loaded web page.[3] In this case, it announces that it has a relevant video. If the user indicates he wants to see the clip, the VCR agent announces that it is cueing to the appropriate tape position and then plays the segment.

The Scatterbrain architecture combines these two behaviors to allow the room, for example, to notify us if we have additional information about things being referenced during other interactions. For example, the room can volunteer to show video clips about San Juan when a person asks for the weather there. This entire interaction is contained in Figure 10, which simply overlaps Figures 8 and 9.

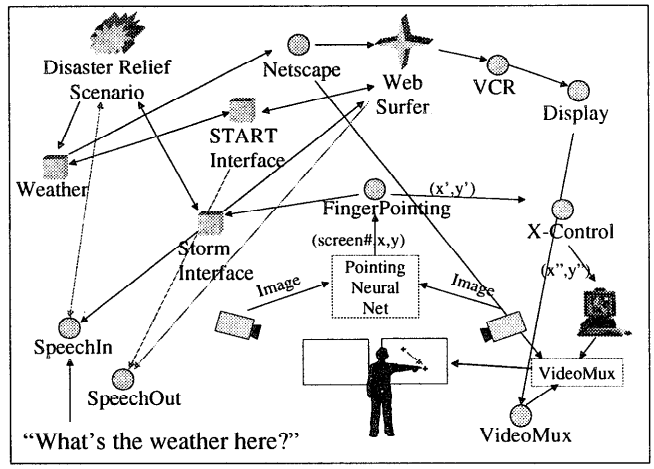

Figure 10 – Combining Behaviors

One of our primary interests is making the room an active partner in applications rather than a passive servant that simply responds to explicit commands. The video-notify behavior discussed here is an early effort towards this. By layering behaviors on top of the Scatterbrain that are indirectly triggered by room activity rather than by direct user instruction, the room can autonomously become involved in its ongoing activities and thereby appear more spontaneous and intelligent to users.

Note that although the Scatterbrain is not actually a subsumption system, the influence of subsumption architecture is clear. The room is controlled by multiple layers of behaviors in which higher-level agents rely on the activity of lower-level ones. When appropriate, these higher-level agents can also override the specific behaviors of other agents. The Scatterbrain architecture also supports combination of agent behaviors to get enhanced functionality.

[3] Note that this information is not contained within the page itself.

Conclusion

Motivated by Minsky's Society of Mind and Brooks' subsumption approach to building robots, we have argued that software agent systems that interact with complex and dynamic worlds are best created from distributed collections of simple agents with parallel, layered architectures.

The complexity of the overall system comes from the interactions of these agents, even though no individual agent is in itself particularly complex and no single agent centralizes the system's control. This approach allows us to build robust, reusable, and behaviorally sophisticated systems that are capable of interacting with the ever-changing real and online worlds. To demonstrate this approach, we presented the Scatterbrain – a distributed collection of software agents that control our laboratory's Intelligent Room.

Acknowledgements

Development of the Intelligent Room has involved the efforts of many people. Professors Tomas Lozano-Perez, Lynn Stein and Rodney Brooks were principally responsible for the room's conception and construction. Mark Torrance led the project during its first year and wrote the room's earliest monolithic controllers. The room's many vision systems are due to the efforts of Jeremy De Bonet, Chris Stauffer, Sajit Rao, Tomas Lozano-Perez, Darren Phi Bang Dang, JP Mellor, Gideon Stein, and Kazuyoshi Inoue. Polly Pook contributed to the design of the room's distributed computation and has worked on modeling the room's functionality as a cognitive process. Josh Kramer wrote large sections of the Scatterbrain and participated in the development of the SodaBot system. All of the above mentioned were also responsible for designing room applications, and many of the above hacked on various room components. Kavita Thomas, along with help from Mark, Polly, and Tomas, configured Victor Zue's speech recognition system. (Jim Glass provided assistance in getting the system running.) Boris Katz and Deniz Yuret provided much support in interfacing with and customizing the START natural language system. Mike Wessler created one of the room's earliest applications and wrote invaluable graphical interfaces for much of the room's hardware.

References

Brooks R. 1985: A Robust Layered Control System for a Mobile Robot, AI Lab Memo 864, Massachusetts Institute of Technology. Cambridge, MA.

Brooks R. 1991: Intelligence without Representation, in Special Volume: Foundations of Artificial Intelligence, Artificial Intelligence, 47(1-3).

Coen, M. 1994. SodaBot: A Software Agent Environment and Construction System. AI Lab Technical Report 1493. Massachusetts Institute of Technology. Cambridge, MA.

Etzioni, O. 1994: Intelligence without Robots, AI Magazine, Winter 1994.

Etzioni, O.; Levy, H.; Segal, R.; and Thekkath,C. 1994. OS Agents: Using AI Techniques in the Operating System Environment. Technical Report 93-04-04. Dept. of Computer Science. University of Washington. Seattle, WA.

Etzioni, O. 1996: Moving Up the Information Food Chain: Deploying Softbots on the World Wide Web, in Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI Press/MIT Press, Cambridge, MA, pp.1322-1326, 1996.

Kautz, H.; Selman, B.; Coen, M.; and Ketchpel, S. 1994. An Experiment in the Design of Software Agents. In Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI Press/MIT Press, Cambridge, MA.

Katz, B. 1990. Using English for Indexing and Retrieving. In *Artificial Intelligence at MIT: Expanding Frontiers.* Winston, P.; and Shellard, S. (editors). MIT Press, Cambridge, MA. Volume 1.

Minsky, M. 1986. *Society of Mind.* New York. Simon and Schuster.

Torrance, M. 1995. Advances in Human-Computer Interaction: The Intelligent Room, In Working Notes of the CHI 95 Research Symposium, May 6-7, 1995, Denver, Colorado.

Zue, V. 1994. Human Computer Interactions Using Language Based Technology, IEEE International Symposium on Speech, Image Processing & Neural Networks, Hong Kong..