Follow this and additional works at: http://scholar.colorado.edu/csci_techreports

Recommended Citation
Ellis, Clarence A.; Barthelmess, Paulo; Quan, Bo; and Wainer, Jacques, "NEEM: An Agent Based Meeting Augmentation System ; CU-CS-937-02" (2002). Computer Science Technical Reports. Paper 881.
http://scholar.colorado.edu/csci_techreports/881

This Technical Report is brought to you for free and open access by Computer Science at CU Scholar. It has been accepted for inclusion in Computer Science Technical Reports by an authorized administrator of CU Scholar. For more information, please contact cuscholaradmin@colorado.edu.
Abstract. This document introduces project Neem, its goals and directions, and its motivation. Project Neem at the University of Colorado is concerned with improving distributed multimedia meetings. A major novel aspect of this research is the use of intelligent artificial agents as full-fledged meeting participants. Goals of this interdisciplinary project include the enhancement of group interaction understanding, and the creation and testing of a prototype distributed meeting environment. Research methods include theoretical modeling, meetings analyses, prototype implementation and testing in real world environments, and meeting metrics development and application. Tools for understanding the social and organizational context of these meetings include the SYMLOG methodology applied by social agents, and the IBIS methodology applied by organizational agents. The work described herein is part of an ongoing effort within the Collaboration Technology Research Group at the University of Colorado to understand human collaboration, and to assist and enhance that collaboration via technology.

Introduction

Meetings are ubiquitous. They come in many sizes, shapes, and flavors. Organized business meetings are one form of interaction that can at times serve
the purposes of informing participants, of mediating differences of opinion, of gathering expertise to make informed decisions, and of allowing participants to have better understanding of other participants and their context. If humankind is a fundamentally social animal, then meetings serve not only technical purposes, but fundamental social purposes. We know that social interactions during meetings are important; are tightly interwoven with business; and are critically important to success of meetings.

However, many meetings do not live up to the expectations and hopes of their participants. The literature has identified numerous problems and breakdowns that can occur within meetings. Many meetings are fraught with problems. The average business manager and technical professional spend nearly one fourth of their total workweek in meetings. And the frequency of meetings are growing. Studies suggest that over 50% of the productivity of typical business meetings is wasted [16]. In a Bell Telephone Lab study, technical engineers frequently said that meetings were one of their biggest wastes of time. They wished that meetings would stop interfering with their work. By improvement of the business meeting venue, they might discover that productive meetings are an important component of efficient, effective work.

There is a lot of literature discussing problems and some potential solutions of meeting failures and fiascos [18,32]. Poor meeting planning, poor organization and execution, and poor meeting follow-up are cited as causes of ineffective meetings. In a study of 1300 managers and technical professionals, they were asked to report on problems with their meetings [16]. The 16 top offenders were reported as:

1. getting off the subject;
2. no goals or agenda;
3. too lengthy;
4. inadequate preparation;
5. inconclusive;
6. disorganized;
7. ineffective leadership;
8. irrelevance of information discussed;
9. time wasted during meeting;
10. starting late;
11. not effective for making decisions;
12. interruptions;
13. individuals dominate / aggrandize discussion;
14. rambling, redundant, or digressive discussion;
15. no published results or follow-up action;
16. no pre-meeting orientation

Compounding this is the obvious, and huge toll of travel to / from meetings. Given that most large organizations are distributed, and many are multinational,
the time and money invested in meeting travel is considerable. If effective
distance meetings could replace this travel, then much savings would accrue. The
rapid growth of Internet availability, and personal computers has lead to the
desktop conferencing phenomenon. Organizations that have tried this in the past
have met with very limited success. The Neem project is addressing meetings,
and meeting improvement within a distributed setting. If successful, it can
represent a significant step forward.

Especially, recent studies of distributed organizations show that attempts to
conduct significant business via distributed, technologically augmented meetings
frequently fail to produce the desired results [4]. Adequate understanding of
meeting factors and meeting metrics in distributed situations is noticeably
lacking. This is an area needing further research. For example, it is sometimes the
case that participants at a meeting convey their agreement, pleasure, or confusion
via facial expressions and body language [17]. However in many distributed
meetings, these communication elements are lost. Our prototype system provides
alternative channels to allow communication elements such as these and others to
be available.

This document is fundamentally concerned with business meetings and their
improvement / augmentation. It describes project Neem, a research project
ongoing within the Collaboration Technology Research Group at the University
of Colorado aimed at understanding some of the social, structural, and
organizational factors at play within medium size distributed business meetings.

The organization of this document is as follows: Section 2 discusses related
work and our philosophy. Section 3 presents a scenario of the Neem system in
action. Section 4 discusses roles, agents and tools and how each is employed in
Neem. Section 5 discusses the Neem architecture. Section 6 discusses the
implementation architecture. Section 7 presents current status and Section 8 our
summary and conclusions.

Related Work and Our Philosophy

Over the years, there have been a large number of efforts to improve meetings.
Large gatherings in many settings and societies have found that there must be
structures to enable effective interaction. Even in primitive societies, societal
norms included who could speak, and who could perform certain actions at group
meetings [19]. There is an acknowledged need for agreed upon structures and
protocols in large meetings. Robert’s Rules of Order [32] were conceived to help
fulfill this need.

As soon as reliable computing technology became available, experiments
began with electronic blackboards, and other meeting assistance technology [24].
The field of GDSS (Group Decision Support Systems) has built technology, and
studied its effect within face-to-face meetings. Many meeting room tools have now matured, and found good utility within meetings [27]. Previous meeting technology research work was performed at MCC during the 1980s; co-author Ellis was head of that effort (Project NICK [13]). That effort was enlightening, and successfully applied technology, but did not attack distributed meetings, and did not employ agents. Likewise, several projects in universities and in industrial labs are exploring various facets of meetings, but we have found none that is implementing social and organizational agents within distributed meetings.

In recent times, numerous companies have presented offerings to allow distance meetings to transpire over the Internet. Netmeeting and Placeware are interesting examples of product offerings ranging from local software to full service meeting control centers. Our project Neem endeavors to take the next step by incorporating active augmentation within distributed meetings. Active augmentation is needed in medium to large meetings to provide adequate facilitation.

Our philosophy includes a belief in active meeting augmentation, and a belief that well researched, well thought out technology can truly make a significant positive difference. Specifically we think that different functionality and goals can coexist within a meeting. For example, there should be a healthy tension between fast pace (efficiency) and careful quality of interaction (effectiveness); likewise, between convivial agreement and critical questioning of assumptions. Different perspectives can conveniently be represented in Neem as different roles played by different participants – some of them automated agents. For example a social agent may want to take time to enjoy side discussion that informs all; as opposed to an organizational agent that may urge the group to keep on its scheduled agenda, and to keep on time. Our Neem hypothesis includes the statement that agents should be full fledged participants in all meetings of the group. They can help add continuity, and perspectives.

In keeping with this hypothesis we want each agent to have sensors (e.g. spoken language understanding unit) and effectors (e.g. spoken output directed to the group). The Center for Spoken Language Research (CSLR) at the University of Colorado has produced excellent base technology for this [http://cslr.colorado.edu], and is our partner in Neem research. Although they will be clearly and obviously identified as non-human, we want our agents to be anthropomorphic and anthropopathic. Ideally these agents will be able to detect emotion in others, and to express emotion [33]. Agents will not be dictators, but tunable meeting assistants.

The team involved in this research work resides within computer science, business systems, cognitive science, communications, artificial intelligence, anthropology, and other disciplines. Our research work is quite interdisciplinary, and builds upon work of many colleagues and forerunners. One of the investigators recalls experiences in the 1960 with Weizenbaum’s Eliza program.
The Eliza agent simulated a psychiatrist (via a terminal typing interface), but actually had minimal understanding of the real world objects being discussed by the patient. Eliza was never-the-less successful in guiding the conversation, and giving the participant a feeling of being heard. In a similar fashion, our goal is to implement a set of multimedia meeting agents that collect meeting data, have a small understanding of the meeting, and have high impact in steering the distributed meeting in good directions. Foundations upon which we build include the following:

- Spoken language processing systems (We are closely collaborating with the Center for Spoken Language Research at the University of Colorado);
- Literature of meeting management and improvement, ranging from popular folk tips [10] to Robert’s Rules of Order [32];
- Distance meeting technologies and studies – teleconferencing, videoconferencing, desktop conferencing;
- Social theories of interaction adapted to distributed meetings;
- Organizational theories adapted to distributed meetings;
- Agent theories and technologies;
- Group user interfaces.

**A Scenario**

The following scenario of a program committee paper selection committee illustrates some of the functionality, the technology, the support level, and the interaction interface that we envision for our Neem prototype. In this scenario, papers that have been submitted to a conference are in the process of being evaluated at a distributed real time program committee meeting. This meeting will be followed by asynchronous interaction using electronic decision tools to evaluate the borderline papers. The follow-up interaction includes paper evaluation conversations that involve all interested program committee members (anonymous) and paper authors and also our automated agents. The technology and the agents augment the meeting, as well as the pre- and post-meeting activities. In this example, there are three artificial agents with voice input/output capabilities:

1. Kwesi – an information agent;
2. Kwabena – a social agent;
3. Kwaku – an organizational agent

The program committee meeting involves committee members who connect to the system from their home or office at a designated time. Simple desktop conferencing tools include whiteboard, chat tool, audio (and potentially video) tools, shared artifact viewers, a whisper function, and conversation tools. Fancier
set-ups, including real-time animation of 3-D models, will also be pursued in later versions of the system.

All participants view and manipulate various conference documents and lists. For example, upon login, all participants see a virtual conference room and conference table with icons of all participants that are present and their roles shown graphically. An agenda tool lists the items to be discussed during the meeting and time estimate bars for each item. As the meeting progresses, actual time bars accumulate next to each item. Thus participants can see at a glance how much time has been allocated to each item (and how much time was supposed to be allocated) in real time during the meeting.

Submitted papers are discussed and categorized one by one. In the case of one controversial paper, the program committee, after its deliberations, receives a (spoken) message from agent Kwesi; he has searched 350 web sites and found several published documents on the exact topic of the submitted paper. None of the program committee members had been aware of these documents. The documents substantiate that this is a very important topic, and also that the topic is not a completely new one. Given this finding, the program committee decides to move this paper to the category of “deferred” until they have reviewed the related documents after the meeting. They thank Kwesi and move on to discussion of the next paper. This is an example of a research agent that is willing to spend hours searching out relevant information. Given the ever-increasing volume of information on the Internet, this is a very valuable agent.

After a while, the social agent, Kwabena, suggests that the committee has been working diligently for two hours, and people might like to take a break. Kwabena also notes that several participants have expressed this wish by pushing their “break time” buttons. The user interface for all participants presents half a dozen soft buttons such as the “I would like to speak” and the “break time” buttons. Any of these buttons can be pushed using the mouse by any participant at any time. Everyone agrees to take a 15-minute break. Kwabena also suggests a “get-acquainted game” that can be played by participants during the break. Committee members laugh at this and decline. Unlike a typical human, Kwabena is never offended by being over-ruled. Kwabena is tunable; whenever the program committee feels that Kwabena is interrupting too much, they can turn down Kwabena’s interaction quotient via another button. This is an example of a social agent that attempts to be responsive to the sensitivities of the group, and that is controllable by the group.

After the break, roll call is again taken, and the future vision papers are evaluated. Numerous virtual reality papers have been submitted; they are quickly rejected. Kwabena suggests that the program committee hear from John, who (Kwabena discovers from the program committee profile pages) is a virtual reality specialist; he has not had a chance to speak during the meeting. John points out some strengths and weaknesses of various papers, and advocates one
particular virtual reality paper. He recommends that although there are minor flaws, it should be accepted. It would be stimulating and more balanced if virtual reality had some representation within the conference, John says. It is decided to take this issue into an asynchronous interaction mode so that people will have time to do some reading and some reflection. People will then contribute their opinions using the asynchronous gIBIS tool [7].

The meeting comes to a close. The chair thanks everybody. Kwaku reminds all of rapidly approaching deadlines, and asks everyone to check the online issue base frequently. Participants then fill out an online questionnaire to evaluate the interaction. Meeting minutes, including action items, are instantly available. Important interactions continue after the meeting in an asynchronous fashion.

Evaluation Issues

A significant issue is the evaluation of the committee interactions, and the prototype system. We are exploring an evaluation framework that basically consists of a technology space (T-space), an actor space (A-space), and an interaction space (I-space). The framework supports evaluation via a set of fuzzy mappings between these spaces. See [11] for more detail. Besides the usual metrics of efficiency and effectiveness, we are particularly interested in the metric that we call “group positioning.” As one example, a great decision meeting that alienates the participants may be judged excellent on the efficiency and effectiveness dimensions, but very poor on the group-positioning dimension because the group may quickly dissolve itself.

Roles, Agents and Tools

Our view of meetings is that they are interactions between various participants who play various roles. Roles are important. As described by Biddle [5], roles allow well understood division of work among a group, and mediates expectations of who will do what. Roles are a convenient mechanism for associating privileges (authority) and responsibilities (blamee) to participants. A role can be associated with a person, an agent, or a subgroup. Roles always exist within interacting groups, whether they be formal and explicit, or informal and implicit. Roles such as “devil’s advocate” and “social matchmaker” are quite important although they are frequently informal and implicitly assumed by appropriate people.

In neem, meeting augmentation is accomplished through two complementary functionalities, role support tools (RST) and agents. Tools are designed to provide custom support for roles. Some of the participants playing roles are humans; others are automated agents (Figure 1).
A key theme of our Neem system is its use of active, anthropomorphized agents as full-fledged meeting participants. Every agent has input capabilities, computational capabilities, and output capabilities. Agents have visible characteristics called personalities. Agent actions are driven by their goal sets. Different agents typically have different goal sets. These sets may be disjoint or overlapping. Thus, different agents may have common goals, and may also have conflicting goals.

For example, Kwaku has goals of keeping the meeting on time, and keeping discussion focused upon the specified agenda. He keeps track of time, of who speaks, and of the agenda. If the meeting falls behind time, then Kwaku discourages informal chitchat about peripheral topics. Kwabena has a goal of building common ground among participants, and knows that informal discussion about peripheral topics is important for this. Thus there are conflicting goals here. Agents have voices, so the result in this situation may be two spoken statements to all participants. In a strong voice, Kwaku says, "Hey gang, we are behind time! Let's move on to the next agenda item." In a gentle voice, Kwabena says, "You all know it's important for us to have a common understanding of our team assumptions and direction. Whenever necessary, please take time to have clarifying discussions."

Agent inputs may include feedback, i.e. observation and estimation of the effect of their actions upon the group. Of course other agents are members of the group, so instead of independent statements evoked by the two agents in the above example, the agents might coordinate, negotiate, and decide to NOT speak until after the chit-chat has continued for more than some agreed upon amount of time (e.g. 10 minutes).

A major thrust of our Neem research is to explore functionality to assist in normal meeting tasks. At times, certain functionality is best provided by tools rather than agents. Various tools that are available to users provide information,
and in some cases an expression outlet, for users. According to the roles played by various participants, the information may be available to some, and not to others.

For example, Neem implements a talk queue that optionally appears on every user's screen. The queue is visible only if a user opens it, and can optionally be used by anyone to put their name into the queue of participants waiting to speak to the group. It provides background information that is available to users and to agents, but does not actively push this information onto users conscious foreground. It also provides an expression outlet for users that want to speak. There are positive and negative aspects of implementing this functionality as a passive tool rather than as an agent. I can indicate that I have something to say without interrupting the current spoken flow of the meeting.

Consider the mood buttons tool implemented by the Neem user interface. Any user, at any time, can push any of the mood buttons on their screen. Buttons have labels such as "I am confused" and "need-a-break" and "this discussion is very upsetting to me." The tool will store button push information in some data store. Should this information be made available to the current speaker? Neem will explore various choices, such as making this available to analysts after the meeting only; or making this instantly available to selected administrative roles only (e.g. the social agent, Kwabena); or making this part of the active ground available to all participants in real time.

Neem explores the use of knowledge and shared culture that is normally in the background (exogenous context) by actively bringing it to the appropriate users' attention when certain thresholds are exceeded. In this example, there might be colored mood globes that are visible in the background on users' screens. Using the interaction analysis theory of Bales [3], the globe might change colors as the meeting progresses in informative ways. For example, as the discussion “heats up” and becomes argumentative for some people, portions of the globe turn orange, then red.

Agents and role support tools are described more in the next section.

Neem Architecture

In the previous section, we discussed roles and their support through tools and agents. Role support tools (RST) make available a broad range of information, which includes both typical meeting information and exogenous context information. Agents play some of the roles. Agents are active, visible, and have human-like characteristics such as voices for speaking up at any time during the meeting. We now discuss details of the RST and agents architecture.
Role Support Tools

Participants cooperate through RST. Awareness is conveyed to them through changes on tool elements. Agents are a kind of participant. As such, they also are made aware of the interaction by sensing changes that take place on/through RST. They make other participants aware of their "actions" through RST as well, as everybody else does. We will use the term user to refer to human users and agent to refer to automated entities. Participants will refer to users and/or agents.

Each participant interacts with a potentially different set of RST, depending on their roles. Agents are no exception to that. They heavily use some RST, while others are not interacted with at all. Because of the roles agents play, some RST that wouldn't make sense in their absence are introduced to the desktops (e.g. mood buttons). This is not a privilege of agents, though - other (human played) roles cause new RST to be introduced to support them. The chair's desktop, for instance, includes RST that give her additional control over the meeting. Facilitators, note takers and other participants playing special roles (such as the agents) also have at their disposal a custom set of RST that directly support role specific tasks.

There is no necessary one-to-one correspondence between RST and information that is required by participants to reach their goals/fulfill their roles. RST are built with UI guidelines in mind. One tool can provide a wealth of information required to reach many goals of different roles and conversely, some goals require information that is distributed among more than one tool.

User's workstations are populated by one or more RST (Figure 2). RST are composed of a set of multimedia elements: audio, video and widgets such as buttons, lists, text fields, etc. In particular, agents make use of the audio and video elements embedded in RST to send synthesized voice and images whenever anthropomorphized interjections are deemed appropriate.

Most of the RST present on user desktops are shared, i.e., operations performed on them are broadcast to all participants or a subset of them. Examples of such RST include e.g. chat tool and video displays. The set of active RST can change as a meeting progresses. Active RST are those that are available for use at any given moment. RST that are useful in a specific phase of a meeting, such the initial introduction, can be progressively removed to make space for others that become more relevant in later phases. Launching and removal of RST can be commanded by the users themselves, if their role thus allows, or automatically by the system. In the latter case, the command to remotely launch or remove RST in one or more desktops is generated by the system in response to some sensed situation, e.g., a perceived transition to the next meeting phase.
RST include, among others:
- Chat, whiteboard, file transfer and application sharing tools: conventional data conferencing interaction tools;
- Video and audio input and output: the usual tools in videoconferences;
- Room tool: a round or oval table, surrounded by clickable participant icons. Positions are associated with roles, e.g., chair, note taker, and so on. The icons can be clicked to select participants to which one wants to whisper, i.e., send a private message;
- Agenda tool: displays agenda items and their expected duration. Actual time spent in each of the items is shown in real time as the meeting progresses;
- Subgroup whisper tool: allows private communication between participants, both human and automated;
- Talk queue: shows the order of speakers and allows participants to manifest their wish to talk. The order of speakers can be rearranged by the participants;
- Action items tool: registers details of group decisions regarding agreed upon actions to be taken by participants in a future date;
- Voting tool;
- Discussion tool: supports IBIS-based discussions;
- Information board: space where participants can post information relevant to the meeting. Agents also use this tool to post minutes, rules, and other meeting history;
- Personal feedback tool: allows people to provide feedback on attitudes of other members of the meeting.
- Buttons tool: lets participants express how they feel about the meeting at each moment;

**Agents**

Agents are the basic automated entities that can be employed to play some role in a meeting. Agents thus have abstract goals, e.g. social, organizational and informational augmentation and well-being. These goals correspond to roles of the social facilitator, the organizational assistant and informational assistant.

The social agent, for instance, has as its mission keeping the social ambiance at its best at all times. For that to happen, an agent needs to embed notions of what is desirable, how to gauge current status and how to steer meetings into the desired direction in case deviations are detected. Agents' functionality can be compared to that of a thermostat - they measure the environment, compare current readings to a preset goal and introduce changes to the environment in an attempt to correct deviations between observed and desirable parameters. Information gathering, transformations and output are based on models:

- A model of measurement: which data to gather and how to interpret it;
- A model of desirable behavior;
- A model of corrective actions: what to do once deviations between the measured and modeled desirable behavior are detected.

Since agents are at the service of very abstract objectives, such as "keeping the social ambience at its best", their goals are not directly reachable. These higher-level goals are all encompassing descriptions of myriad of different aspects that all contribute in varying degrees to the overall goal. Agents, therefore, reach their goals by employing sub-agents, or nested agents. We will refer to this nesting by saying that an agent can contain one or more agents. In this case, it is called a container agent. Conversely, an agent can be contained in a higher-level (container) agent.

An agent is enclosed in an environment. An agent reacts to stimuli generated by the environment in which it is contained, applies transformations and produce responses that influence this same environment (Figure 3-1). Input stimuli signal events of interest, e.g. actions performed by participants on one or more RST. Responses in turn command, e.g. state changes in one or more of the RST, usually resulting in some perceivable change in user interface elements (e.g., text output in some designated area, color changes, or a voice message). Agents maintain a private internal state (knowledge base). At a minimum, this internal knowledge base registers the received stimuli and the generated responses, but agents can also register other state in its knowledge base as needed.

All agents have the same structure (operate in an environment, have a set of input stimuli, apply transformations, output responses, have a private internal
Contained agents have more concrete goals that represent a decomposition of the higher-level goal of the agent in which they are contained. Such decomposition is applied recursively until the goals are simple enough to allow implementation. In practice, as with most divide-and-conquer approaches, actual design is based on analysis as well as synthesis, i.e., agents can be combined into higher-level ones, as well as decomposed into simpler ones as described. This process is strongly empirical and constitutes perhaps one of the main research questions in this project.

Figure 3 – 1) agent structure; 2) contained agents.

Contained agents react exclusively to stimuli that come from their container's knowledge base. In other words, the internal state of a container agent is the environment in which the contained agents operate. Since the container's knowledge base registers the stimuli received by it plus eventual additional state, contained agents can be made to react either to the stimuli received by their containers (Figure 3-2-a) but also to other types of changes in the container's internal state (Figure 3-2-c). Conversely, contained agents' outputs can correspond to one or more of the container's responses (Figure 3-2-b), or this output can only affect the container's $kb$ (Figure 3-2-d). Containing agents can still deal with inputs and outputs on their own, without having to employ a contained agent (Figure 3-2-e and f).

Responses that are based on a larger context, i.e. requires collecting the actions of many participants, maybe over a set period of time, make use of the mechanisms we just described. This is the case when an agent has, e.g., to collect different participants' button clicks on the "I'm confused" button over a period of 2 min before some reaction can be generated. Contained agents that deal with individual button clicks have as their output changes to their container's state, i.e. they effect changes in the context (e.g. incrementing a counter), without generating any visible external response. When this context indicates need for action (number of clicks > threshold), a response can be generated, causing an external effect, e.g. change color of the group mood-o-meter.
Higher-level agents may share contained agents, i.e. there is potentially an intersection between the sets of contained agents that compose an agent. 

Agents can present themselves to users as anthropomorphized entities. In practice that means that they can interact with users through language-based RST that require the production of phrases that are either spoken or output to some textual area of the shared user interface. Agents' outputs have the challenging aspect, that they should be consistent with the agent's personality as defined at design time.

**Implementation architecture**

We now describe the architecture of the prototype that is being built as part of the project. 

The prototype is based on an extensible, open-ended platform (Figure 4). The focus of the platform is on providing generic support to distributed meetings augmentation as described in previous sections.

![Figure 4 – Architectural components.](image)

The system's functionality can be logically divided into interactors, extractors, effectors, translators, and processors. A coordination mechanism binds together the various components. Interactors and processors are the main functional components, and are responsible for the support of the synchronous interaction through RST and agents respectively. These two complementary functional blocks communicate to each other by (indirectly) posting messages to the coordination mechanism. Remaining functional components act as bridges between these main elements. We next describe in more detail each of these functional components.

Interactors are distributed RST built according to the ITU H.323 protocol. This protocol establishes a set of services that can be employed as a basic
multimedia conferencing support layer. It includes, among others, services for conference creation, handling client connection and disconnection, file transfer, whiteboard, application sharing, video and audio communication and transfer of data between two or more connected clients. Also included is support for remote launching of applications in selected connected clients. As such, it offers an adequate substrate upon which RST can be built. Client software based on this protocol is readily available in many platforms (e.g. NetMeeting on Windows and SunForum on Solaris).

The interaction is monitored by extractors, which collect events of interest, convert them if needed, and post them to the coordination mechanism to be processed by the system (details later in this paper). The goal of the system is to have the extractors collect as much information from the interaction as possible, to make it available to some potentially interested agent.

Some extractors are very simple and do nothing more than post messages whenever the events they monitor takes place. This is the case for extractors that monitor widgets such as buttons, lists and other standard interface elements. Other extractors are more complex, e.g. the audio and video extractors. Audio channels are monitored to detect voice exchanges. The voice stream is then converted to text to make it amenable to processing. We would like in the future to extract other important aspects carried by this stream, such as voice intonation. We would also like to extract body language cues from the video streams. We anticipate incorporating such features as technology matures and makes feasible their use in real time.

Effectors play a symmetric role to extractors. Effectors cause the interactors to display the results of the computations performed by the agents. Effectors can be simple, just commanding a user interface element to modify its state, e.g. by changing its color, or by displaying a message. Other effectors are responsible for conveying to the human users the anthropomorphic appearance of agents, by making use of synthesized voice and images. Again, the architecture is open to technological enhancements, that can be incorporated as they become available, to add more realistic voice production, intonation and better facial expressions to the animated characters.

Effects can be caused to happen on all users' desktops, on a selected group, or just on a single one. A message, for instance, can be displayed privately to a single user, to a selected group or to all participants. Similarly, voice output can be whispered directly to a single participant, or to a subgroup or all participants. The dynamic launching and removing of interactors that we mentioned previously is yet another effect that can be selectively caused to happen on participants' desktops.

While extractors post specific messages to the coordination mechanism, effectors do exactly the opposite: they are triggered when certain kinds of messages are inserted into the coordination mechanism. The same message can
trigger the activation of one or more effectors, making it possible for multiple
effects to take place in response to the same message. A widget can be
commanded to change color or blink and at the same time a voice message can be
generated as reactions to the same message.

We now turn our attention to the functional components responsible for
supporting agent logic - processors. Processors provide support for the
computation of the appropriate reaction to observed events, i.e., they realize
agents' functionality. The stimuli that drive the agents, as well as their responses,
are posted to the coordination mechanism. This mechanism constitutes for
practical purposes the environment in which higher-level agents are embedded.

At the core of the agents support functionality is a production system
mechanism that is used to implement agent logic. Production rules fire when
conditions related to observations are met. Resulting actions can either command
the activation of some effector, or cause changes to the system's state.

Auxiliary processors are employed to deal with more focused tasks, such as the
information miner. The information miner searches databases and the web
according to the requests it receives. Agents can request services from these
additional processors, if required. Other processor modules can be introduced as
needed.

Translators do simple conversions between formats necessary to accommodate
the processors to the coordination mechanism that bind together all other
functional components.

The coordination mechanism provides an intermediate common language that
serves as a bridge between the different system's elements. Different components
interact by posting and reacting to messages that are traded through the
coordination space that is mediated by this mechanism. Depending on the
complexity of an agent, different routes are followed within the system. We will
show the flexibility of the proposed mechanism by analyzing a few such routes.

Let's suppose there is a trivial agent, acting on behalf of higher-level agents,
that senses clicks on a tool's button and causes as response that a message be
displayed on some textual area within another (or the same) tool. In this simple
case, the following sequence of activations takes place:

1. The button click is sensed by an extractor;
2. The extractor posts a message signaling this simple event to the
   coordination mechanism. Since the response is equally simple, an effector
can serve this message directly, without the intervention of other system
modules. In other words, the associated agent has empty state and applies
a direct one-to-one (empty) transformation.
3. An effector reads the message from the coordination space.
4. The effector causes the text to be displayed by the appropriate user
   interface widget.
Notice that even in this simple case group awareness is being promoted. When a participant clicks on the button in question, other participants are made aware of that by receiving the message that is propagated to all of them through associated effectors, through calls to the underlying H.323 compliant substrate.

A more complex case involves the functionality of the processors. An agent can require that a set of events take place before a reaction occurs. The mood-o-meter, for instance, displays a color that indicates group feeling about a meeting, varying gradually from green to red, as the perceived group feeling gets worse. In order to gauge that, an agent needs to collect events from all participants, or from a significant number of them. A single participant contributes only partially to the general group feeling that needs to be displayed. In this case:

1. Extractors sense clicks on mood buttons.
2. The extractors post messages to the coordination mechanism as explained. Since no direct response is appropriate, no effector is fired at this time.
3. These messages are collected by translators, which transform them to conform to what is expected by the processors, if required.
4. Productions fire and collect the evidence presented by the many users, modifying the processor's execution context.
5. Once a threshold is reached, e.g., the majority of participants manifested confusion or disagreement, a reaction is generated.
6. This reaction is translated to conform to the coordination's mechanism protocol, if needed.
7. The message containing the reaction triggers effectors.
8. The effectors command a color change on the mood-o-meters of all participants.

The decoupled nature of this mechanism makes it easier to dynamically replace components and to add new ones, which is a necessity given the evolutionary aspect of the prototyping effort, and the wish to take advantage of new technology as it becomes available. The development strategy is heavily based on reuse. This architecture is realized by available H.323 protocol stacks (currently NetMeeting SDK), and CU Communicator natural language processing modules. The Coordination mechanism is realized by the DARPA Communicator Hub (derived from MIT's Galaxy-II). The rich functionality provided by H.323, CU Communicator and DARPA Communicator are explored in detail in [21], [36], and [29] respectively.

Current Status

The first data gathering phase of this work has already begun. It involves observation and videotape analysis of working groups. One identified application
area of Neem is conference program committees. We focus upon the social and organizational dynamics of selected conference committees. What are their tasks, what are their interactions, what are their problems, and how do they perceive the interaction? These initial studies focus on groups that do not utilize any meetingware technology. How do you measure the goodness of a meeting? We are developing a meetings metrics document. This research complements our theory building, and our prototype building. Later phases of this thrust will study and analyze conference groups using our meetingware prototype, currently under development. This will provide evidence toward a proof of concept for our architectural ideas, for our evaluation framework, as well as proving ground for our interactive group agents.

Summary and Conclusions

This document has introduced project Neem, its goals and directions, and its motivation. Project Neem at the University of Colorado is concerned with improving distributed multimedia meetings. A major novel aspect of this research is the use of intelligent artificial agents as full-fledged meeting participants.

We are working with groups of consenting conference organizers as our test bed. We will utilize interviews, questionnaires, and videotaped observations of their actions and interactions to assess our prototypes. The meetingware system also collects information and measurements about one to one communications, and subgroup interactions. Techniques appropriate for analysis of this thrust include organizational analysis, distributed cognition studies, questionnaires, interviews, observation, technology usage analysis, and cognitive walkthroughs. With this data, it is possible to perceive the existence of cliques, of gatekeepers, and of patterns of communication. It is possible to estimate the cohesiveness of the groups and other salient characteristics using theories from the social network analysis domain. The results of these analyses will be useful within our ongoing systems and agents design.

We believe the time is ripe for a spectacular success in a similar fashion to the Eliza agent success, but in the distributed group interaction domain. With minimal domain knowledge, our agents will coax and steer the meetings in positive directions. If our hypotheses of active meeting augmentation and anthropomorphic agents are validated then this could open the door to enormously more effective distributed meetings.
References


