Designing a Multi-Agent Portfolio Management System

Keith Decker, Katia Sycara, and Dajun Zeng
The Robotics Institute, Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213
(decker,sycara,zeng+)@cs.cmu.edu

Abstract

The voluminous and readily available information on the Internet has given rise to exploration of Intelligent Agent technology for accessing, filtering, evaluating and integrating information. In contrast to most current research that has investigated single-agent approaches, we are developing a collection of multiple agents that team up on demand, depending on the user, the task and the situation, to access, filter and integrate information in support of user tasks. We are investigating techniques for developing distributed *adaptive* collections of information agents that coordinate to retrieve, filter and fuse information relevant to the user, task and situation, as well as anticipate user's information needs. Our approach is based on (1) *case-based task and situation models*, (2) *flexible organizational structuring*, and (3) *reusable agent architecture*. We are currently implementing the system in the domain of financial portfolio management. While this paper focuses on the big picture, the final section will describe the current implementation and point to our work on detailed technical solutions.

Introduction

Due to advances in technology, diverse and voluminous information is becoming available to decision makers. This presents the potential for improved decision support, but poses challenges in terms of building tools to support users in accessing, filtering, evaluating and fusing information from heterogeneous information sources(Kuokka & Harada 1995; Arens *et al.* 1993; Collet, Huhns, & Shen 1991). Most reported research on Intelligent Information Agents to date has dealt with a user interacting with a single agent that has general knowledge and is capable of performing a variety of user delegated information finding tasks (e.g., (Etzioni & Weld 1994; Maes 1994)). For each information query, the agent is responsible for accessing different information sources and integrating the results. We believe that, given the current computational state of the art, a centralized agent approach has many limitations: (1) a single general agent would need an enormous amount of knowledge to be able to deal effectively with user information requests that cover a variety of tasks, (2) a centralized information agent constitutes a processing bottleneck and a "single point of failure", (3) unless the agent has beyond the state of the art learning capabilities, it would need considerable reprogramming to deal with the appearance of new agents and information sources in the environment, (4) because of the complexity of the information finding and filtering task, and the large amount of information, the required processing would overwhelm a single agent. Because of these reasons and because of the characteristics of the Internet environment, we employ a distributed collaborative collection of agents for information gathering.

We are currently working on a system where each user is associated with a set of agents which have access to the task and situation models and keep track of the current state of the task, situation, environment and user information needs. Based on this knowledge, the agents decide what information is needed and initiate collaborative searches with other agents to get the information. During search, the agents communicate with each other to request or provide information, find information sources, filter or integrate information, and negotiate to resolve conflicts in information and task models. The returned information is communicated to display agent or agents that possibly combine it with information from other sources (e.g. the user) and/or filter it for appropriate display to the user.

This paper focuses on the design of such a system of agents for the task environment of financial portfolio management, and on the key issues that we will be addressing. These issues include: (1) gathering and integrating diverse information sources with collaborating software agents, (2) case-based user, task and situation models, (3) adaptive integration of planning, coordination, scheduling, and execution. In our system, case-based reasoning provides meta-level control and activation of agents. Depending on the task, user and situation, case based retrieval selects current planning goals, information needs and information gathering goals. Based on the plans and information gathering goals, agent teams are activated "on demand" to access, integrate and filter information to fulfill these goals. New information can be incorporated in the case base and may give rise to new plans and information gathering goals (and as a result activation of potentially different agent teams). The system has three types of agents, task agents, information agents, and interface agents. Task agents have information about tasks and associated information gathering goals. Information agents have models of information sources, information access strategies and associated task agents to whom the information should be returned. Interface agents interact with the user receiving user specifications and delivering results. They acquire, model, and utilize user preferences to guide system coordination in support of the user's tasks. This work is a continuation of our previous work on multi agent information access, filtering and integration of everyday organizational tasks (Sycara & Zeng 1995).

Of the three types of agents, information agents are the most well-defined. Each information agent—by definition—shares a set of reusable behaviors that give it the ability to process one-shot and periodic queries, monitor for certain conditions, advertise it's capabilities appropriately, and react reasonably in the event of a local or external data source failure. If a domain content ontology has been fixed beforehand, an information agent can be created quickly with little programming—it requires a database definition, and some code to access the external data source (i.e. web page, newsgroup, or normal database) and create local database records. Information agents are described in more detail in (Decker, Williamson, & Sycara 1996).

The Portfolio Management Domain

To evaluate our domain independent agent control, organization, coordination and architectural schemes, we have chosen financial portfolio management as a task domain. This is the task of providing an integrated financial picture for managing an investment portfolio over time, using the information resources already available over the Internet. This task environment has many interesting features, including: (1) the enormous amount of continually changing, and generally unorganized, information available, (2) the variety of kinds of information that can and should be brought to bear on the task (market data, financial report data, technical models, analysts' reports, breaking news, etc.), (3) the many sources of uncertainty and dynamic change in the environment.

The overall task in the portfolio management domain, as stated by modern portfolio theory (Markowitz 1991), is to provide the best possible rate of return for a specified level of risk, or conversely, to achieve a specified rate of return with the lowest possible risk Risk tolerance is one of the features that characterize the user of our system; other features include the user's investment goals (long-term retirement savings? saving for a house?) and the user's tax situation. In current practice, portfolio management is carried out by investment houses that employ teams of specialists for finding, filtering and evaluating relevant information. Current practice as well as software engineering considerations motivate our multi agent system architecture.

Previous work in the portfolio management domain (see (Trippi & Turban 1990) for one collection) has focused on the portfolio selection process (i.e., "stock picking") as opposed to portfolio monitoring—the ongoing, continuous, daily provision of an up-to-date financial picture of an existing portfolio. A multi-agent system approach is natural for portfolio monitoring because the multiple threads of control are a natural match for the distributed and ever-changing nature of the underlying sources of data and news that affect higher-level decision-making processes. A multi-agent system can more easily manage the detection and response to important time-critical information that could appear suddenly at any of a large number of different information sources. Finally, a multi-agent system provides a natural mapping of multiple types of expertise to be brought to bear during any portfolio management decision-making. Existing DAI techniques for resolving conflicting opinions, negotiation, and argumentation can be brought to bear on these problems (Sycara 1989).

The overall portfolio management task has several component tasks. These include eliciting (or learning) user profile information, collecting information on the user's initial portfolio position, and suggesting and monitoring a re-allocation to meet the user's current profile and goals. As time passes, assets in the portfolio will no longer meet the user's needs (and these needs may also be changing as well). Our initial system focuses on this ongoing portfolio monitoring process.

The task of monitoring individual portfolio assets gives rise to a variety of concurrent goals, such as monitoring an asset currently being held (should we continue to hold it? sell some or all of it?), or monitoring the buying or selling of an asset. Buying and selling at this high level are not instantaneous transactions, but also require careful planning and monitoring of plan execution. For example, let us assume that in the process of monitoring the system comes to recommend that an asset be sold. The way in which it is sold will be determined in part by the reason for the sale—perhaps the asset is no longer performing as required for it's role in the portfolios asset allocation mix. Perhaps it is performing *too* well, and there is a growing possibility that the asset has reached a peak. In this second case it is often prudent not to sell one's entire holdings all at once, but to sell in phases and place the appropriate standing orders to protect the user from a sudden downturn (while avoiding worry over simple daily price fluctuations) Thus the goal of selling an asset is not one that requires only a simple short sequence of actions, but one that requires careful planning and monitoring of that plan as it is executed, over an extended period of time.

Case Based Situation and Task Models

We are proposing to model the information gathering task as a planning task where planning and execution are interleaved and search is guided by user, task and situation models. We believe that cases that incorporate user, task and situation models can effectively provide instantiations of the situationdependent task structure and the associated team of information gathering agents. The case base contains cases of successful and unsuccessful information gathering episodes and information evaluation. After each information gathering cycle the case base is updated. Thus, learning is integrated with problem solving and is achieved automatically. This feature makes Case-Based reasoning very preferable in application domains with open and dynamically changing world model (Sycara, Zeng, & Miyashita 1995). Case based reasoning offers three general advantages. First, since it relies on reusing specific experiences rather than reasoning from a general world model, it provides for more efficient planning. Second, since it can start with few cases in memory and incrementally acquire new cases based on a reasoner's interactions with the world, it makes few assumptions about the completeness and correctness of world knowledge. Third, previous past failures warn the planner about the possibility of failure in current circumstances and, hence help avoid future failures.

Each case will be automatically labeled with identifying information such as user's name (a user could be a software agent), time of creation, time of modification, etc. So, a case will carry a complete audit trail of its origin and modifications. This information is essential for careful analysis of all the planning knowledge that is exchanged during system operation, and we will rely heavily on it during the initial stages of knowledge acquisition and development. In addition, this information can be used to evaluate the performance of the system under different planning scenarios, since each item will be clearly identified.

The library of cases we propose can be viewed as a case-based scheme for meta control that offers agents access to task and situation specific information. The agents use this information to focus search and filtering. The asynchronous information gathering activity of the agents results in new information about the world (new cases) that gets incorporated into the case base. Case base updates result in formation of new memory indices. These new indices, along with any new user inputs (e.g., information gathering requests, change in context) activate a new set of cases that reflect possibly new information seeking goals, and new sets of agents. In this sense, the case base can be viewed as (1) tracking the user's intentions and his/her evolving information needs, (2) reflecting changing situations, (3) recognizing new events, and (4) learning information retrieval tactics/heuristics.

Figure 1 shows an active case base which receives as input notification of events, either directly from the user or from new information that becomes available (e.g., from other software agents). Updates of the case-base with the addition of new information finding episodes and new indices are also considered an event that results in the activation of a new round of reasoning. In the following, we use an example to illustrate the general Case-based agent invocation process. Given a situation, the case-base calls a certain task assistant, say Analyst Tracking Agent, which the reasoning process determines is relevant to meeting the information gathering goals (which in turn might be information requests from Portfolio Manager Agent). Portfolio Manager Agent calls upon Earnings analysis Agent, News Classifier, etc., to locate information from the infosphere directly (in the case of News Classifier) or indirectly (in the case of Earning Analysis Agent). Note that there is no hierarchy of agents here: in a different situation Earning Analysis Agent might have been called directly by the case-base reasoning process. After collaborating to find and filter information, Analyst Tracking Agent updates the case base with new information finding episodes that include a timestamp and the results of the search. If unexpected news ("unexpected" in terms of the current situation) has been found during information retrieval (such as major corporate merge news), the case-based process will interrupt the regular plan execution and take other emergency actions. For example, the case-base process might invoke the interface agent to notify the user right away.

From the perspectives of Case-Based reasoning, there are a number of important research questions that we need to address. The most fundamental question is *what constitutes a case?* A case in case memory describes a specific information scenario including: (1) information needs and goals which might come directly from the user or from other software agents, (2) global features which give an abstract characterization of the situation in which this information gathering operation takes place, (3) features of local nature which describe in detail the information about information sources, inter-agent interactions, etc., (4) information retrieval plan skeleton, (5) feedbacks/evaluations with respect to acquired information and information retrieval effectiveness/efficiency, and (6) potential failures and re-trial, plan modification information. At the most abstract level, we might index an asset monitoring task plan fragment using several broad situation characteristics such as asset type, industrial sector, ownership records, tax status, user profile, etc. These characteristics are represented in the case base and used as indices for the retrieval of plan fragments, associated information gathering goals and associated lists of agents to be activated. We are currently investigating other important issues such as efficient case indexing mechanism, case retrieval/matching approaches, and initial case collection.

Organizational Structure

We propose a general system organization in which agents are directly activated based on the top-down elaboration of the current situation. These agent activations, guided by case-based retrieval according to the current situation, dynamically form an organizational structure that fits in with the user's current profile, tasks, and other situational features. This organization will change over time, but will also remain relatively static for extended periods (for example, while monitoring currently held investments during stable market periods). Information that is important for decision-making (and thus might cause an eventual change in organizational structuring) is monitored at the lowest levels of the organization and passed upward when necessary.

In this type of organization (see Figure 1), "task agents" or "task assistants"(Sycara & Zeng 1995) continually interleave planning, scheduling, coordination, and the execution of domain-level problem-solving actions. Task agents interact with one another and with "information agents" or "information assistants" that encapsulate network information sources. Task agents retrieve, coordinate, and schedule plans based on local knowledge modulated by situational context. A task assistant decomposes an information request into information seeking goals and subgoals and interacts with the information assistants to gather the information. In this architecture, a task assistant does the final filtering and fusing of information before it passes it on to agents above it in the organizational structure (requesting agents). This incremental information fusion and conflict resolution increases efficiency and potential scalability (e.g., inconsistencies detected at the information-assistant level may be resolved at that level and not propagated to the task-assistant level) and robustness (e.g., whatever inconsistencies were not detected during information assistant interaction can be detected at the task-assistant level). In addition, a task assistant composes a new case that incorporates its findings to be stored in the case memory.

In this architecture, information-assistants would have models of their associated information sources, learn the reliability of those sources, as well as strategies for low-level information fusion and multiple methods for responding to information requests. As an example of the latter, a stock ticker monitoring agent might have several methods available to it that trade off time, cost, and quality:

- one or more sources of 15-minute delayed values (with varying reliabilities and average response delays)
- one or more sources of real-time quotations that charge a fee (more reliable response but still not guaranteed)
- the ability to guess a quote based on recent data and simple models (very fast but of low quality).

On the other hand, task-specific assistants have a model of the task domain, executable methods for performing the task, knowledge of an initial set of information-assistants relevant to their task and strategies for learning models of pertinent information-assistants.

Figure 1 shows a top-level portfolio manager agent which receives as input notification of events, either directly from the user or from the case base, or from information that becomes available (e.g., from task and information agents). Given the current situation, the portfolio manager agent:

- instantiates task plans and associated information gathering goals according to the current situation
- coordinates those plans with other agents (this includes task assignment actions that activate task assistants)
- schedules and monitors the execution of its local actions.

In Figure 1, the fundamental, technical, news, and outside-analyst task agents have been activated in this manner. These agents are task assistants that can either locate information via information assistants, or by calling upon other task agents. There is not a strict hierarchy of agents—the same task and information agents may be called upon by different parts of the portfolio management organization. After collaborating to find and filter information, task agents update the case base with new information finding episodes that include a time stamp and the results of the search.

This architecture has potential advantages and drawbacks. The advantages include:

- There is a finite number of task assistants that each agent communicates with.
- Because information processing is done by all the task agents at every level (rather than by having one task agent receive all data from every information agent) we avoid having a single computational bottleneck point.
- The task assistants are responsible for checking information quality, filtering irrelevant information, recognizing important information, and integrating information from heterogeneous information sources for their respective tasks.
- The task assistants are responsible for activating relevant information assistants and coordinating the information finding and filtering activity for their task.

All the above characteristics, by imposing some structure through definition of task assistants, contribute to overall system responsiveness. On the other hand, there are potential drawbacks:

- The portfolio manager is a single point of failure. Such failures can be mitigated by expending the resources to have, for example, a redundant portfolio manager that takes over in case of failure.
- Each task assistant also constitutes a "single point of failure for that task". This can be mitigated by having more than one task assistant (either clones of each other or not). In the case where two different assistants exist for the same task, the task assistants must negotiate to resolve inconsistencies. We propose to explore the use of negotiation strategies for resolving inconsistencies.

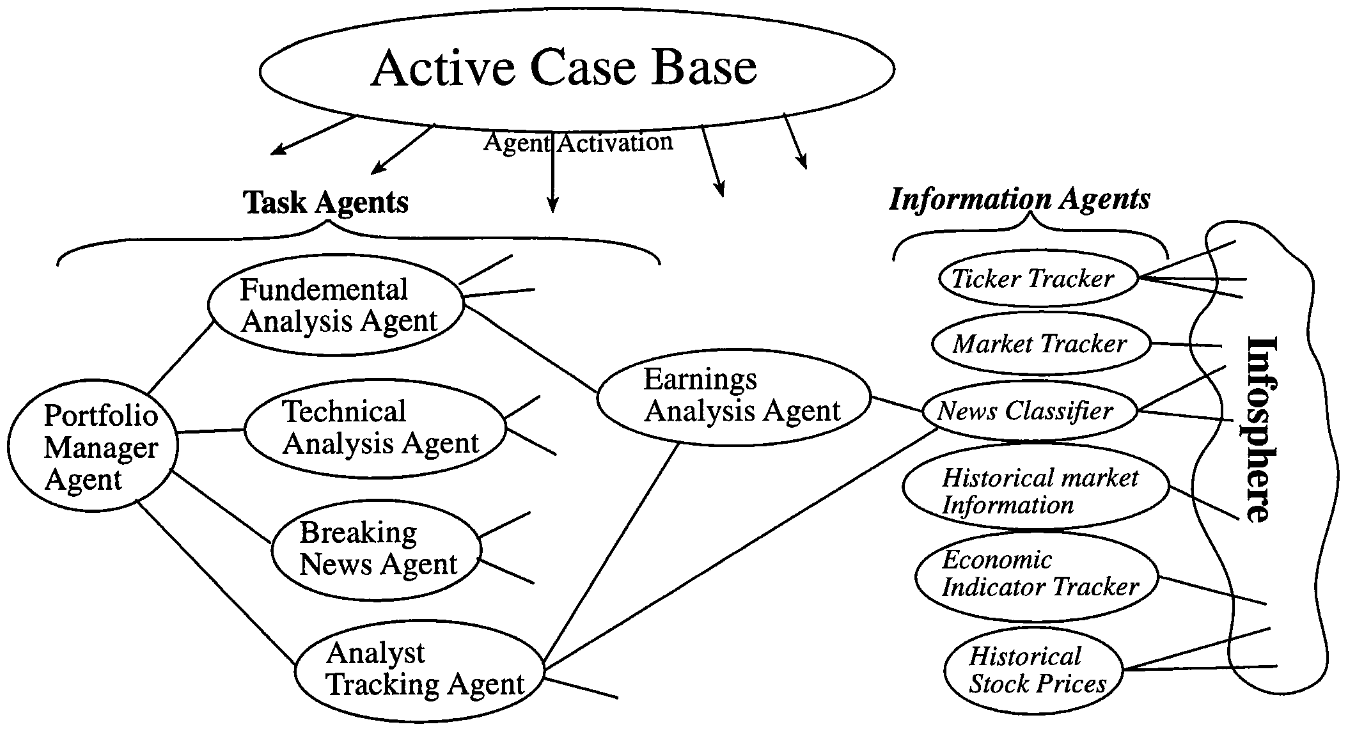

Figure 1: A "direct invocation" agent organization for a portfolio management system.

In the stock portfolio example, task assistants for areas such as earnings analysis might be replicated and allowed to specialize on various industry groups (one agent to handle banking industry earnings, one for manufacturing, etc.). Such agents might begin as clones, but learn specialized case information. In the event of a failure, a non specialist would still be able to retrieve useful plans for a task inside its area of expertise, but outside of its specialty.

The Portfolio Monitoring Task

We can represent the plans that are retrieved using TÆMS task structures (Decker 1995). TÆMS task structures are based on abstraction hierarchies, where task plans are elaborated via a "subtask" relationship into acyclic directed graphs that have actions, called executable methods, at their leaves. Such structures are compatible with most planning representations, and provide the necessary information both for scheduling activities that arise from multiple plans, and for coordinating the activities of multiple agents. We have in fact constructed a decision-theoretic hierarchical task network planner using extensions of this representation framework (Williamson, Decker, & Sycara 1996). The extensions include the ability to represent and reason about periodic tasks. As shown in Figure 2, a top level portfolio management agent interacts graphically and textually with the user to acquire information about the user's profile and goals; as mentioned earlier, we will assume in this paper that the system has gone through an initial usage period and has reached a "steady state" of monitoring the current portfolio.

Such a monitoring task includes gathering opinions from various task experts, integrating this information, and then making or updating the recommendation (such as buy, sell, or hold) for the asset under consideration. These tasks are persistent, in that they are continuously present. An agent will be dealing with many such tasks simultaneously. Gathering opinions from the area experts (fundamental analysis, technical analysis, news, and the opinions of other analysts—the published output of similar human organizations) requires registering with them and then either waiting for new opinions to be received or asking for them directly. An opinion consists of not just a buy/sell/hold recommendation but a short list of positive and negative reasons for holding that opinion, and potentially both symbolic and numeric measures of uncertainty. Information integration involves removing redundant information, resolving conflicts (or declaring them unresolvable), and forming a coherent group opinion, that can then be used for decision-making (in the light of the user's risk tolerance, investment goals, asset allocation, tax status, and so on). The conflict resolution process may involve negotiation between the agents involved.

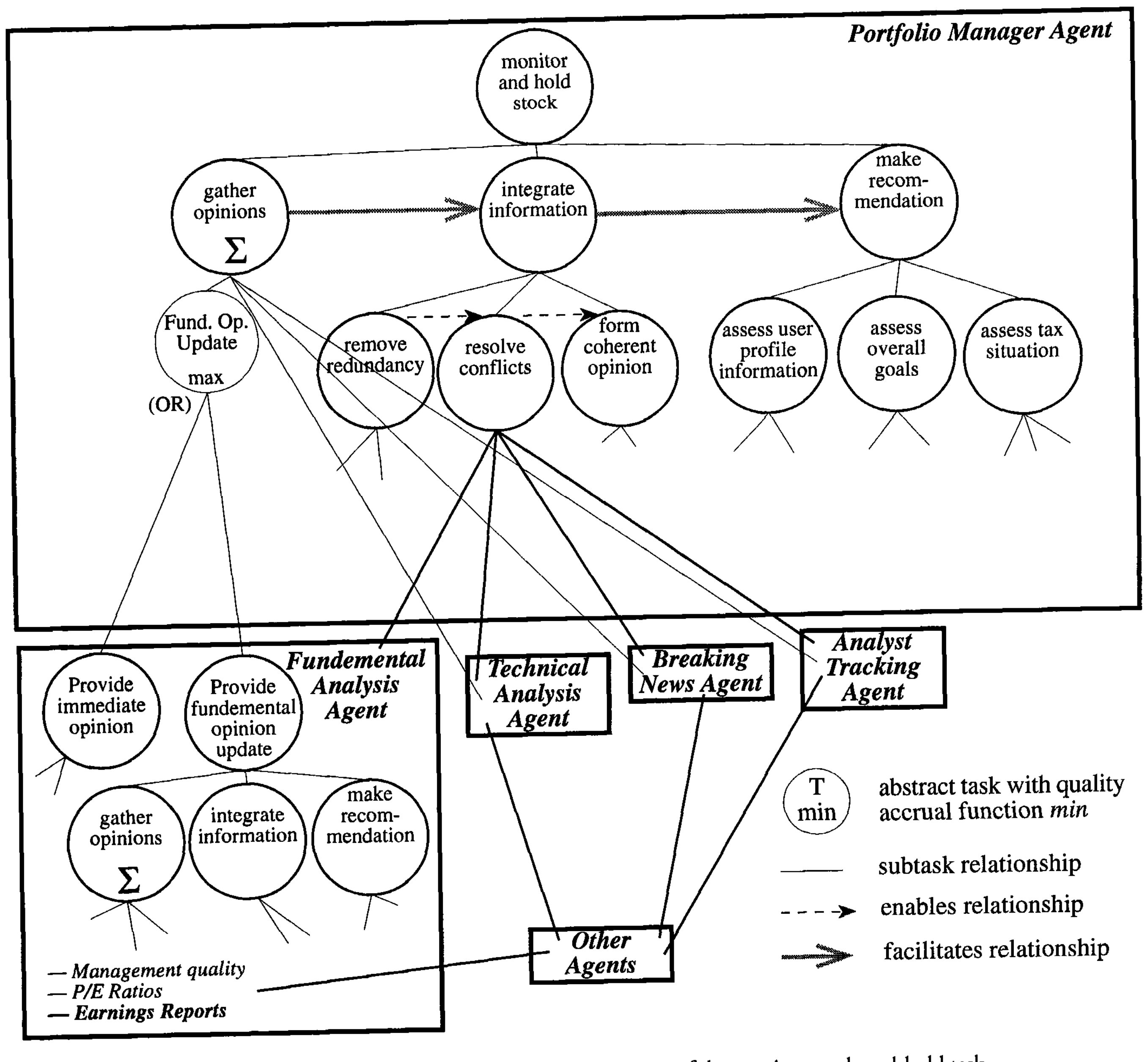

Figure 2: A task structure representing high-level portions of the monitor-stock-and-hold task.

An Example of Coordination: Earnings Report Interpretation

One interesting subproblem in portfolio management is acquisition and interpretation of earnings reports and earnings estimates. The earnings analysis task is a complex one that includes estimating the impact of one company's earnings on other companies in a sector, the information contained in one company's earning report that actually releases information about all companies in a sector, timing in the release of earnings reports (especially for smaller companies), and the differences in actual earnings versus expectations. It is important to track revisions in earning estimates over time, as they often give important clues as to future price moves.

Figure 3 shows a relationship between the abstract plans of the earnings analysis agent and the human analyst opinion tracker agent. The earnings analysis agent initially needs to get data on a companies current and historical earnings patterns, and then it needs to keep up to date on new earnings reports as they are released. Not only does it need to track the new earnings of the company in question, but also the earnings of other companies in an industry sector. For example, a change in the portion of earnings attributed to sales is often applicable to all companies in a group, unlike changes to costs (sales minus earnings) (Joh & Lee 1992).

The analyst tracking agent gathers, from news and other sources, existing and updated analyst reports on a company, including revised earnings estimates (often part of a larger report). This part of the data, if transmitted to the earnings analysis agent, can somewhat speed up (i.e., facilitate) the process of gathering earnings expectations. We have demonstrated the use of general coordination mechanisms, called the GPGP (Generalized Partial Global Planning) approach, that can easily coordinate such task structure interactions (Decker & Lesser 1995). In this instance, the soft-predecessor-coordination-relationship mechanism will cause the analyst tracking agent to commit to the transmission of a completed analyst report form to the earnings analysis agent, which can then easily extract the portion dealing with the updated earnings estimate.

Agent Architecture

The portfolio manager and task assistant agents have an internal agent structure called DECAF (Distributed, Environment-Centered Agent Framework)—a general, reusable, core agent control architecture (Oates *et al.* 1995). The term *architecture* here refers to the internal control structure of a single agent, as opposed to the term *organization* that refers to the control and communication structure of a group of agents.

The important features of the DECAF architecture are:

- A set of clearly defined control modules (planning, coordination, scheduling, decision-making, and monitoring) that work together to control an agent.
- A core task structure representation that is shared by all of these control modules. This core structure can be annotated and expanded with all manner of details that might be "understood" only by one or a few control modules, but there is a core, shared representation.

Briefly, the main control functions consist of a *planner* that creates or extends the agent's view of the problem(s) it is trying to solve, called the *task structure.* The *coordinator* notices certain features of that structure, and may annotate it, expand it, communicate parts to other agents, or add scheduling constraints to it. The *local scheduler* takes the rough plan and creates a low-level schedule or schedules that fix the timing and ordering of actions. The *execution monitor* takes care of actually executing the next desired action (perhaps including pre-emption of the action in true real-time execution).

Previous work has focussed on the design of the coordinator and the local scheduler (Decker & Lesser 1995). Details of the implementation of these components can be found in the cited papers. We have extended this work to include more sophisticated execution monitoring, using such techniques as the TCA (Task Control Architecture) approach (Simmons 1994).

Current Status

We have already built a large part of the underlying basic organizations and achitecture as described here. Organizationally, our agents usually form dynamic unstructured teams using a central "matchmaker" that accepts advertisements from new agents about thier capabilities, queries about who might provide certain services, and unadvertisements when agents leave the open system. Recently we have developed "brokering" agents that act as central points of contact for certain types of services (centralized markets). A hybrid organizational approach allows the use of both subforms, i.e., the use of a matchmaker in order to find an appropriate broker. More details can be found in (Decker, Williamson, & Sycara 1996).

Architecturally, we have developed several versions of the basic agent internals described here. Currently, all of our agent classes (information, task, and interface agnets) are based on this shared architecture. Although the local scheduler is considerably simpler than the one described in (Decker & Lesser 1995), the decision-theoretic hierarchical task network planner is more complex and capable (Williamson, Decker, & Sycara 1996).

WARREN, our multi-agent portfolio management system, currently consists of 6 information agents: two stock ticker agents using different WWW sources, a news agent for Clarinet and Dow-Jones news articles, an agent that can extract current and historical sales and earnings-per-share data from the SEC Edgar electronic annual report database, another for certain textual portions of annual reports, and of course the matchmaker. Two task agents provide

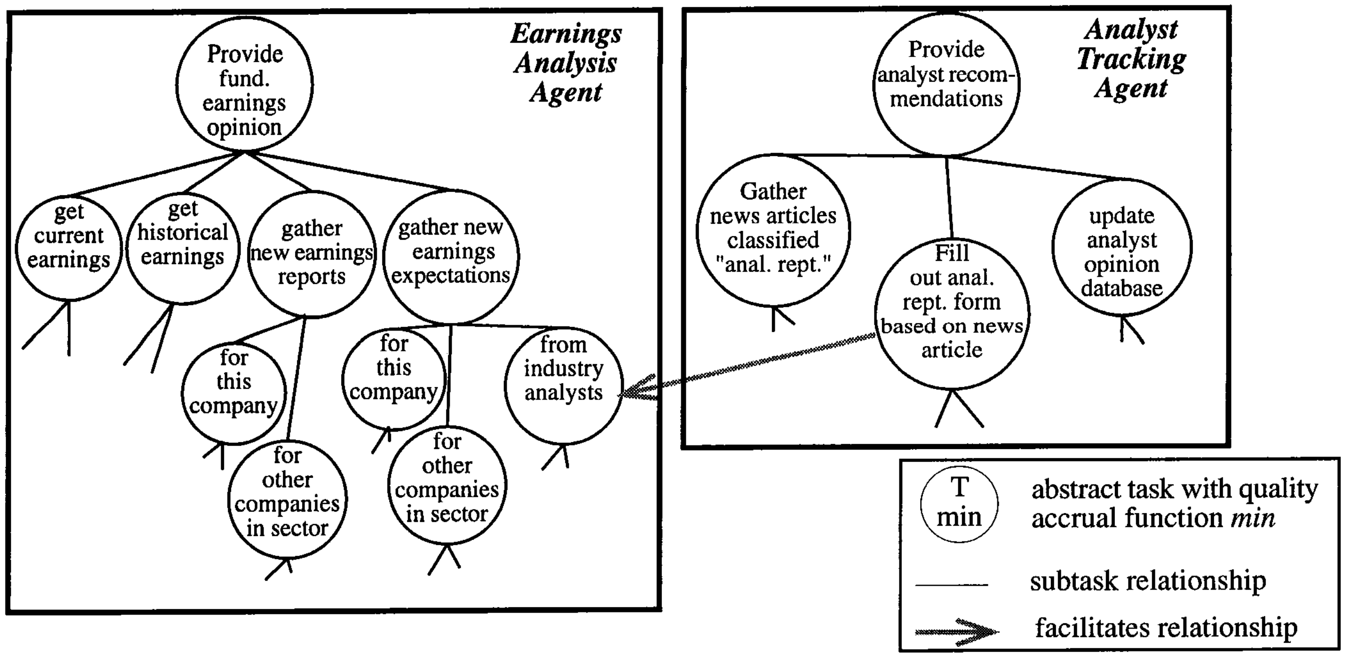
Earnings Analysis Agent
Provide fund. earnings opinion
get current earnings
get historical earnings
gather new earnings reports
gather new earnings expectations
for this company
for other companies in sector
for this company
for other companies in sector
from industry analysts
Analyst Tracking Agent
Provide analyst recom-mendations
Gather news articles classified "anal. rept."
Fill out anal. rept. form based on news article
update analyst opinion database
T min
abstract task with quality accrual function *min*
subtask relationship
facilitates relationship

Figure 3: A task structure showing one coordination relationship between tasks in the domain of the earnings report agent and the human analyst tracking agent.

1. a simple graphical integration of stock prices and news stories about a single stock over time on a WWW page—the body of the news stories can be accessed by hyperlinks, and the information is stored persistantly so that that it survives the task agent going off-line

2. a simple fundamental analysis of a stock with respect to its historical sales and earnings data, along with an accompanying graph.

Finally, a portfolio interface agent can be associated with each user of the system. The Portfolio agent (via a WWW interface) displays the standard information about a user's portfolio, allows the user to (simulate) buying and selling shares, and displays recent pricing and news information. It also provides access to the reports produced by the two task agents, either continuously updated or on demand.

More information and demos can be found on the WWW at `http://www.cs.cmu.edu/~softagents /warren/warren.html`.

Conclusions

We have presented the overall framework and design decisions made in our multi-agent system for the management of financial portfolios through information access, filtering and integration. Within this framework we will explore research issues of agent coordination and negotiation and case base structuring for user, task and situation modeling. In addition, there are a number of learning-related research issues we want to explore. How do we formulate the learning task in the context of multi agent interactions where procedural and control knowledge must be learned? Concept learning has been the focus of most machine learning research (e.g., (Michalski & Tecuci 1994)). Learning of control knowledge has been explored using case-based reasoning (e.g., (Kambhampati & Hendler 1992; Miyashita & Sycara 1995)), and reinforcement type learning techniques (e.g., (Sutton 1988)). This research has been conducted almost exclusively in a single agent setting. We want to explore strategies for multiple agent learning of control knowledge during agent interactions. Within each formulation of the learning task (e.g. as a case-based learning, or reinforcement learning), there are additional more specific issues to be explored. For example, for multiple agent case-based learning, new case indexing and retrieval algorithms might be necessary. In addition, the number of training cases that must be incrementally acquired through agent interactions for reliable learning is an open issue.

Acknowledgments

This work has been supported in part by ARPA contract F33615–93–1–1330, in part by ONR contract N00014–95–1–1092, and in part by NSF contract IRI–9508191.

References

Arens, Y.; Chee, C. Y.; Hsu, C.-N.; and Knoblock, C. A. 1993. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems* 2(2):127–58.

Collet, C.; Huhns, M.; and Shen, W. 1991. Resource integration using a large knowledge base in Carnot. *Computer.*

Decker, K. S., and Lesser, V. R. 1995. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems,* 73–80. San Francisco: AAAI Press. Longer version available as UMass CS-TR 94-14.

Decker, K.; Williamson, M.; and Sycara, K. 1996. Intelligent adaptive information agents. In *Proceedings of the AAAI-96 Workshop on Intelligent Adaptive Agents.*

Decker, K. S. 1995. *Environment Centered Analysis and Design of Coordination Mechanisms.* Ph.D. Dissertation, University of Massachusetts.

Etzioni, O., and Weld, D. 1994. A softbot-based interface to the internet. *Communications of the ACM* 37(7).

Joh, G., and Lee, C. 1992. Stock price response to accounting information in oligopoly. *Journal of Business* 65(3):451–472.

Kambhampati, S., and Hendler, J. A. 1992. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence* 55(2-3):193–258.

Kuokka, D., and Harada, L. 1995. On using KQML for matchmaking. In *Proceedings of the First International Conference on Multi-Agent Systems,* 239–245. San Francisco: AAAI Press.

Maes, P. 1994. Agents that reduce work and information overload. *Communications of the ACM* 37(7).

Markowitz, H. 1991. *Portfolio selection: efficient diversification of investments.* Cambridge, MA: B. Blackwell, second edition edition.

Michalski, R., and Tecuci, G. 1994. *Machine Learning: A multistrategy Approach,* volume IV. Morgan Kaufmann Publishers.

Miyashita, K., and Sycara, K. 1995. Cabins: A framework of knowledge acquisition and iterative revision for schedule improvement and reactive repair. *Artificial Intelligence* 76(1–2).

Oates, T.; Prasad, M. V. N.; Lesser, V. R.; and Decker, K. S. 1995. A distributed problem solving approach to cooperative information gathering. In *AAAI Spring Symposium on Information Gathering in Distributed Environments.*

Simmons, R. 1994. Structured control for autonomous robots. *IEEE Trans. on Robotics and Automation* 10(1).

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine Learning* 3:9–44.

Sycara, K., and Zeng, D. 1995. Task-based multi-agent coordination for information gathering. In Knoblock, C., and Levy, A., eds., *Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments.* Stanford, CA: AAAI.

Sycara, K.; Zeng, D.; and Miyashita, K. 1995. Using case-based reasoning to acquire user scheduling preferences that change over time. In *The Proceedings of the Eleventh IEEE Conference on Artificial Intelligence Applications (CAIA '95).* Los Angeles: IEEE.

Sycara, K. 1989. Multi-agent compromise via negotiation. In Huhns, M., and Gasser, L., eds., *Distributed Artificial Intelligence,* volume Volume 2. Pittman.

Trippi, R., and Turban, E., eds. 1990. *Investment management: decision support and expert systems.* New York: Van Nostrand Reinhold.

Williamson, M.; Decker, K.; and Sycara, K. 1996. Unified information and control flow in hierarchical task networks. In *Proceedings of the AAAI-96 workshop on Theories of Planning, Action, and Control.*