

Modeling a Virtual Food Court Using DECAF

Foster McGeary and Keith Decker
Computer and Information Sciences
University of Delaware
{mcgeary,decker}@cis.udel.edu

Abstract

Modeling of economic phenomena principally proceeds on two levels, the microeconomic and the macroeconomic. Microeconomic theory explains economic decisions made at the level of the firm and the individual. Other fields of study also explain these decisions. In particular, organizational theory explains the organizational decisions made by individual participants. Since voluntary organizational decisions are most likely made by self-interested parties, organizational decisions in an economic market are subject to explanation by (at least) two different fields of study. This paper describes initial work on economic modeling, including the modeling of voluntary organizational contracts, of a set of business entities set in the market context of the food court at an upscale suburban mall, the Virtual Food Court (VFC).

DECAF is an multi-agent system building toolkit that provides this effort with (1) a GUI to easily establish the control mechanisms needed to express exchanges (“conversations”) between participants and (2) the operating system features to perform the message sending and receiving that models the actual transactions. DECAF provides high-level language support and pre-built middle-agents for building multi-agent systems across typical TCP/IP-based networks. By taking an “agent operating system” approach rather than the “API” approach to creating agents, DECAF has been shown to help researchers or students get to the “interesting” parts of a multi-agent system or simulation more quickly.

1. Introduction

Economic entities exhibit many of the characteristics associated with agents. First, each economic entity is (generally) viewed as a rational decision-maker attempting to maximize its own welfare. An economic entity can thus be thought of as a rational computational agent accomplishing a goal. Second, each economic entity/agent is concerned with and in control of its own

welfare, and is thus an autonomous agent. Third, each agent is viewed as intelligent because it engages in purposeful activity expected to achieve goals or objectives. Having made these three observations, we use the DECAF agent toolkit [5] to model potential individual consumer and individual business behaviors explicitly, giving us a collection of economic agents which we expect to behave as received economic theory instructs us. Our agents have limited rationality (unlike traditional economic agents), can engage in certain specific well-defined behaviors, and have explicit representations of how to compute their own self interest. We then let the individual agents communicate and bargain with each other, where each agent only makes those transactions, chosen from among those available to be made with other autonomous agents, that maximize its own individual best interests, as best it can. The aggregate behavior of such agents should provide the same results as received economic and organizational theories (as far as such theories rely primarily on economic behavior). Our modeling approach provides us with the ability to prohibit certain selected behaviors, and thus to identify those key individual behaviors which are essential to the production of a particular behavior.

VFC models diners, workers, and entrepreneurs. These economic entities are caricatures of the participants in transactions that take place within a simplified shopping mall food court. They exhibit sufficient behaviors to allow VFC to contain a labor market, markets for food service equipment, and markets for food products. For example, accepting a contract to perform labor and forming an organization are reciprocal events. Because both of these are voluntary actions, we believe it necessary to model and explain both sides of the transaction simultaneously. This is what we do in VFC, planning to extend our results to model organizational structures more complicated than a simple employment contract while still basing the analysis on the need for there to be reciprocally voluntary contracts. We expect that such models will be expanded to include aspects of governance [7] and non-economic social forces [4] as we ex-

plore the long term control and stability of such structures.

VFC exhibits behavior in a model of bounded rationality. Information is only available through reporting mechanisms or communications with other entities. Not all information is available to all decision makers. Limits are placed on the amount of information that can be analyzed and the amount of time available to analyze it. These limits are a reasonable reflection of actual decision making, which is often bounded by time. DECAF provides a convenient mechanism for modeling economic activity as the actions of a set of agents. In particular, DECAF's model of software agency includes as primary features the need to support reasoning under deadlines, reasoning about alternatives, and reasoning about tradeoffs between multiple activities.

2. Our Principal Interests

Management of an arbitrarily large number of interacting agents and organizing them into workable units, while still recognizing and respecting their individual autonomy, is a significant problem. Since multi-agent system toolkits such as DECAF do not limit the number of agents, since the various agents can reside on different machines, and since the computational location of an agent is of no concern for modeling purposes, simple computational scalability does not loom as a significant problem. However, when these agents must interact in complex, changing ways, the scalable control and coordination of such systems becomes a pressing issue. One principal interest is, then, how to organize and at least partially control large dynamic multi-agent systems.

DECAF allows us the opportunity to create an artificial world with all the essential characteristics of the economic market place. With this tool, we can study and explain the "management" of numerous autonomous agents to simulate the economic activities which characterize the agent-selected activities that result from the mutually independent desires of each agent to maximize its own well being. To fully achieve such an explanation, we must further explain how autonomous agents communicate and interact with one another within constraints imposed by physical reality, to achieve organizations. We put quotes around the word "management" because, as Mintzberg would have it, there is mutual adjustment between agents. The only decisions are those of individuals, and thus organization is an observable result of the communication and actions of individual autonomous agents. Explaining how organizations come about out of large numbers of autonomous individuals, in a changing environment, with changing technologies and changing demands, is our principal long term goal. VFC serves as the testbed for those ideas as a means of

explaining how businesses organize.

3. Overall Operation of the Virtual Food Court

As with the general economy, activity in VFC is consumer (Diner) driven. Each Diner has the goal of maximizing its own satisfaction from food purchases. Diners' preferences for food are known to the Diner and are revealed through requests for items and selections from among the available foods. Entrepreneurs act as vendors and attempt to maximize their incomes (the difference between sales revenue and total labor and preparation costs). Information on labor and food purchases is available through "governmental" bureaus that record and publish some (but not all) information related to labor use and to food consumption. As a result of the interaction of diners, workers, and entrepreneurs, restaurants are formed (organized) to serve the needs of the diners. The restaurants, in turn, serve the income desires of the entrepreneurs. Alternative forms of ownership and organization are expected to evolve in later versions of the VFC. Notions of market purchases and contracts are used in VFC. Market purchases are simple exchanges of money for food products or for one act of labor where each exchange is monetarily independent of any other such exchange. A contract would be any other arrangement, covering more than one instance or exchange.

A Typical Study. Our eventual goal in a typical study with VFC is to explain, step-by-step, some specific and particular economic phenomenon, say the equilibration of a purely competitive market. We believe it is not sufficient to show simply that such phenomena can happen, but rather that is necessary to detail the conditions under which the studied phenomenon happen as a result of voluntary action. With competitive markets, we would need to show the conditions which need to exist so that entrepreneurs, operating in their own best interests, act in ways consistent with competitive markets. Models which require agents to act in other than their own best interests should not be demonstrable using the VFC. We believe that economic actions modeled on autonomous agents, using agents that are smart enough to collude or engage in self-serving anti-competitive actions, provides more information than models that simply assume the ideal conditions, assume unlimited capability on the part of the agents, and thus assume their conclusion.

A Simpler Question Or Study Transaction cost economics places a great deal of weight on the behavioral and engineering assumptions on which it is built. The behavioral emphasis stresses cost minimization and self interest. The engineering assumptions stress particular production functions (in VFC, these are the technologies and the parameters which differentiate them).

In total, though, transaction cost economics deals with only simple cost minimization, and the contracts that occur are those needed to secure the necessary resources.

A simpler question to study is how the nature of the production functions, such as their “lumpiness,” leads to one market form rather than to another. Similar questions or demonstrations, such as how the introduction of a technology with increasing returns to scale affects market structure, can be studied, including the provision of a trace of how the change comes about.

A More Complicated Question Or Study. It is possible to enrich the knowledge base and behaviors of the participants (firms and workers) to more fully investigate organizational behavior. For example, explicit representations of how expectations are formed can be used to more fully demonstrate organizational evolution.

Organizational theory says that the organization behaves differently if it expects to repeat a transaction than it does if the transaction is not to be repeated. It could be the case that a transaction cannot be repeated if the firm behaves one way rather than another. There are a large number of such situations (chicken and egg questions) in organizational theory (and in transaction cost economics, too). We would like to investigate how the organization makes the determination of whether it expects to repeat a transaction. Similarly, there are organizational theories that rely on more than bounded economic self-interest, including the influence of social networks and patterns of information flow (e.g. [12]).

Bounded Rationality Economic activities occur through time. The duration between the creation and expiration of an economic opportunity affects each agent’s ability to exploit it. A model of microeconomic decisionmaking is more realistic if it can trace the making of decisions with respect to the timing of decisionmaking and the identification of the information needs of successful decisionmakers.

Knowledge-based decisionmaking is bounded by the time and limited bandwidth available for communication. It is unrealistic to expect decisionmaking to be based on full information. VFC allows explicit modeling of that boundedness, and allows the modeling to take place at the level of the individual worker or production unit. We believe models that do not adequately represent the boundedness of decisionmaking do not properly explain decisionmaking or the economic and organizational evolution that comes from such decisions.

Because we can limit agent memory and performance, and thus create agents with different memory and performance capabilities, we can create competitions between those who require more information, and therefore respond to economic stimuli more slowly, and those who require less information, and thus respond more quickly. Careful construction of those agents and

agent behaviors should allow the study of how boundedness affects microeconomic decisionmaking.

4. Overview of DECAF’s Internal Agent Architecture

DECAF (Distributed, Environment-Centered Agent Framework) is a toolkit which allows a well-defined software engineering approach to building multi-agent systems. The toolkit provides a stable platform to design, rapidly develop, and execute intelligent agents to achieve solutions in complex software systems. DECAF provides the necessary architectural services of a large-grained intelligent agent: communication, planning, scheduling, execution monitoring, and coordination. This is essentially, the internal “operating system” of a software agent, to which application programmers have strictly limited access.

The goals of the architecture are to develop a modular platform suitable for our research activities, allow for rapid development of third-party domain agents, and provide a means to quickly develop complete multi-agent solutions using combinations of domain-specific agents and standard middle-agents [1]. DECAF distinguishes itself from many other agent toolkits by shifting the focus away from the underlying components of agent building such as socket creation, message formatting, and the details of agent communication. Instead, DECAF provides an environment that allows the basic building block of agent programming to be an agent action, or a pre-specified subtask (collection of agent actions). These building blocks are then chained together by the DECAF planner. This paradigm differs from most of the well known agent toolkits, which instead use the API approach to agent construction (e.g., [8]). Functionally, DECAF is based on RETSINA [3] and TAEMS[2, 13].

Although a traditional HTN planning component is in development, currently the control or programming of DECAF agents is provided via a picture-based GUI called the *Plan-Editor*. The Plan-Editor can also be used to construct shared task network libraries for common multi-agent protocols. The GUI can also be used to visually examine automatically-constructed plans for either DECAF or TAEMS.

Figure 1 represents the high level structure of the DECAF architecture. Structures inside the heavy black line are internal to the architecture and the items outside the line are user-written or provided from some other outside source (such as incoming messages).

As shown in Figure 1, there are five internal execution modules (square boxes) in the current DECAF implementation, and seven associated data structure queues

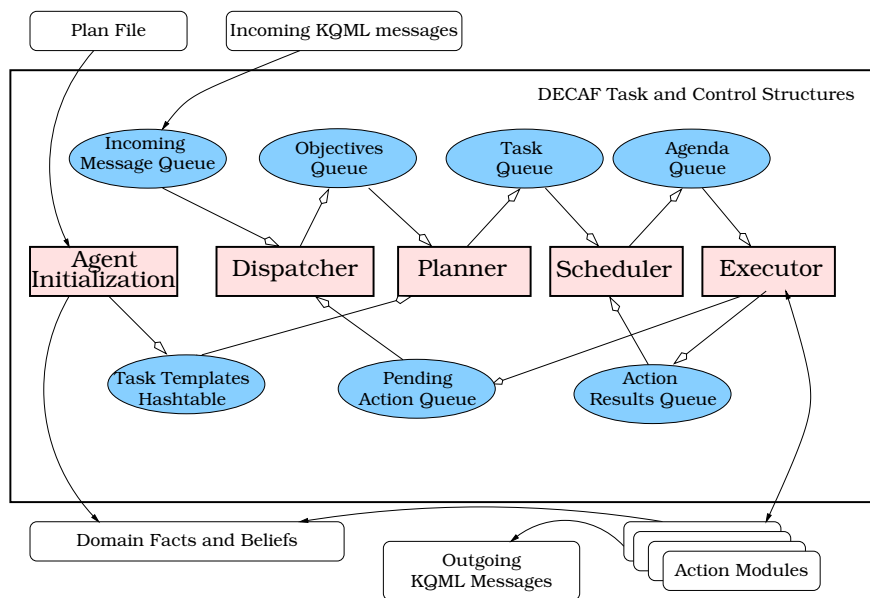


Figure 1. DECAF Architecture Overview

(rounded boxes). DECAF is multi-threaded, and thus *all modules execute concurrently, and continuously* (except for agent initialization).

Agent Initialization. When an agent is started, the *Agent Initialization* module will read a plan file containing basic action definitions and pre-defined task network reductions. Each task reduction specified in the plan file will be added to the *Task Templates Hashtable* (plan library), indexed by the particular goals that the reduction achieves.

The agent initialization process also attempts to achieve a *Startup* goal. The Startup tasks of a particular agent might, for example, build any domain data/knowledgebases needed. Startup tasks may assert certain continuous maintenance goals or other initial achievement goals for the agent. The last thing the Agent Initialization Module does is register with an Agent Name Server and set up all socket and network communication.

Dispatcher. The Dispatcher waits for incoming KQML messages which will be placed on the *Incoming Message Queue*. An incoming message contains a KQML (or FIPA) *performative* and its associated information. An incoming message can result in one of three actions by the dispatcher. First, the message may be a part of an ongoing conversation. In this case the dispatcher will find the corresponding action in the *Pending Action Queue* and provide the incoming message as an enabling provision (see “Parameters and Provisions”, below). Second, a message may indicate that it is part of a new conversation. If so, a new *objective* is created (equivalent to the BDI “desires” concept[9]) and placed

on the *Objectives Queue* for the Planner. An agent typically has many active objectives, not all of which may be achievable with an agent’s limited time and resources. The last thing the Dispatcher is responsible for is the handling of error message replies to malformed messages.

Planner. The Planner monitors the Objectives Queue and plans for new goals, based on the action and task network specifications stored in the Plan Library. A copy of the instantiated plan, in the form of an HTN corresponding to that goal is placed in the *Task Queue* area, along with a unique identifier and any provisions that were passed to the agent via the incoming message. The Task Queue at any given moment will contain the instantiated plans/task structures (including all actions and subgoals) that should be completed in response to all incoming requests and any local maintenance or achievement goals. A graphical plan-editor GUI allows the construction of static HTN planning task structures.

Scheduler. The Scheduler waits until the Task Queue is non-empty. The purpose of the Scheduler is to determine which actions *can* be executed now, which *should* be executed now, and in what order.

For DECAF, the traditional notion of BDI “intentions” as a representation of a currently chosen course of action is partitioned into three deliberative reasoning levels: planning, scheduling, and execution monitoring. This is done for the same reasons given by Rao [9]—that of balancing reconsideration of activities in a dynamic, real-time environment with taking action [10]. Rather than taking the formal BDI model literally, we develop the deliberative components based on the practical work

on robotics models [11]. Each level has a much tighter focus, and can react more quickly to external environment dynamics than the level above it. Most authors make practical arguments for this architectural construction, as opposed to the philosophical underpinnings of BDI, although roboticists often point out the multiple feedback mechanisms in natural nervous systems.

Once an action from the Task Queue has been selected and scheduled for execution, it is placed on the *Agenda Queue*. In a very simplistic Scheduler, the order might be first-come-first-served (FCFS). Recent work has resulted in the development of a sophisticated *design-to-criteria* action scheduler [13] that efficiently reasons about action duration, cost, result quality, and other utility function characteristic trade-offs. This scheduler is also available for use in DECAF agents.

Executor. The *Executor* is set into operation when the Agenda Queue is non-empty. Once an action is placed on the queue the Executor immediately places the task into execution. When the action completes it signals a specific action outcome, and the result is placed on the *Action Result Queue*. The framework waits for results and then distributes the result to downstream actions that may be waiting in the Task Queue. Once this is accomplished the Executor examines the Agenda queue to see if there is further work to be done.

5. Overview of the Virtual Food Court

VFC produces a series of arms-length voluntary exchanges of money for labor or the outputs of production. Capital products, restaurant supplies, and food products for final consumption are the outputs of production. Technologies define the combinations of factors required to produce a particular product. The factors of production are capital products, labor, and, generally, other products. The initial configuration of VFC is shown in Figure 2 as a collection of boxes and lines. Lines represent the initial communications paths built into the Java code that constitutes each agent. Arrowheads indicate the direction of the initial message. To bootstrap the connection problem, agents know of the Matchmaker to register their existence with it. Workers and Restaurants know of the existence of the Government because they report information to it.

Boxes inside the Diner and Restaurant indicate lists associated with each instantiation of a diner or a restaurant. The lists represent that particular agent’s knowledge, assets, and the values for parameter-driven behaviors. In general, no two diners have the same preferences, and no two restaurants have the same technologies, inventories, employees, or menu.

Matchmaker. Matchmaker is a DECAF facility that operates as a well-known referral service [1]. The stan-

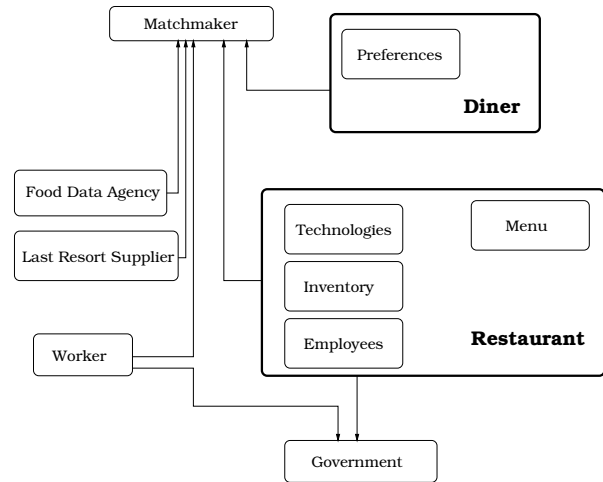


Figure 2. Virtual Food Court Architecture

dard term for this is “yellow-pages” because referrals are made on the basis of declared values submitted by agents that choose to “advertise” with the Matchmaker. Agents that wish to supply services (except the Government, which considers itself well-known) advertise with the Matchmaker. Diners do not register with the Matchmaker, as they are users of services and do not need to advertise. There is within DECAF the ability for agents to “subscribe” to advertisements as well as to advertise. Use of this feature allows Diners to receive notices of Restaurant openings.

Diner. The Diner is the key agent to triggering activity. The Diner is sent an initial message by the investigator containing the number of meals to consume. The Diner begins an eat-select loop that terminates only after the required number of meals have been consumed.¹

The Diner’s Initial Condition. Diners are instantiated with no specific knowledge of the world other than their own name and how to contact the Matchmaker. Each Diner reads a list of items that are its preferred foods and its current preference for each. Currently all Diners have the same code, but can differ in their initial preferences. As the Diner is more fully developed, the addition of various alternative methods of choosing restaurants and forming preferences can be contemplated.

How a Diner Behaves. Figure 2 shows only the initial conditions for the VFC, with the lines indicating potential message flows. The Diner asks the Matchmaker for the names of all restaurants and selects one at which to procure a meal. This action provides the Diner with the information to start a conversation with any of the Restaurants that have advertised with the Matchmaker

¹We like to call this the “Dining Economists” problem.

and which meet any other criteria of the Diner. The Diner “enters” the selected Restaurant by sending an appropriate message. It receives a Menu from the Waiter, and performs a calculation that creates a ranked list of items (based on preference and price). With its ranked preferences, the Diner then proceeds, through a series of messages and replies, to conduct a typical restaurant conversation. After a meal, the Diner updates its priorities by decreasing those of items it has received and increasing those of items it has not received. If the assigned number of meals have not been consumed, the Diner chooses a new restaurant at which to procure the next meal.

Restaurant. The Restaurant begins by reading files containing its initial inventory, its initial menu, and its initial stock of technologies. The restaurant then asks the Matchmaker for the name and address of the supplier of last resort and for the name and address of a Food Data Agency. The Restaurant then hires a waiter and advertises itself with the Matchmaker. When a Diner enters the Restaurant, the Restaurant assigns a Waiter to the Diner, and sends the Waiter the current menu, inventory, and name and address of the Diner. Once the Waiter delivers the order, the Restaurant tries to fulfill the order from inventory, and then restock inventory. If the waiter quits, the Restaurant hires a new one. If the waiter does not quit, the Restaurant waits for the next Diner to enter.

How a Restaurant Prepares a Meal. The current paradigm is one of only serving items already prepared and in inventory. If an item is on the Menu, a Diner may ask a Waiter for it. The Waiter has a copy of the inventory at the time the Diner entered, so it may accept an order for an item that has disappeared from inventory before the order is serviced. Ordered items are deleted from inventory when “sold” to the Diner. After each order is processed, a message is sent to the Waiter listing which items were available, and another is sent to the Government as Delivered items.

How a Restaurant Restocks. Restaurants produce to satisfy inventory requirements, subject to minimum order quantities. That is, for each item on the menu there is a desired number of the items to have in inventory. After each diner’s order is fulfilled, the restaurant compares its then-current inventory to the desired levels. If an item is below its desired level by at least the minimum order quantity, then the restaurant attempts to replenish its inventory either by making the item (if has the technology and the resources) or by purchasing the items from a supplier.

Initially, the only supplier is the supplier of last resort, which can supply any item in any quantity - hardly a model of reality. However, by having the supplier of last resort charge “too much” for certain items, we can

create the economic incentives needed for suppliers to compete with it. Since we want all this activity to be driven by individual agents, we add business entities that are suppliers of restaurant supplies rather than restaurants. The Restaurant at the end of the supply chain can choose from more than one supplier, or can choose to make an item itself.

How a Restaurant Learns a New Technology. The potential for many make-or-buy decisions exists within a restaurant or other supplier. It must decide what items it will offer, which it can only decide after it knows what items there are and for what items there is a market. In addition, the supplier needs some notion of how profitable an item will be. This turns out to be both simpler and more complicated than it may appear. The process is simpler in a fully functioning VFC because there is market data that allow a supplier to evaluate the decisions. However, before the market is fully developed, early participants must acquire information in a manner that is best described as speculative.

Each restaurant/supplier has a parametrically driven algorithm for producing items. A Technology consists of a list of the parameters to the algorithm and a name for the output. The supplier of last resort is the initial supplier of Technologies. Technology transfer consists in the supplier of last resort selling the technology, which is the simple act of sending the list of parameters in response to a message request. The requester then has all the knowledge it needs to produce the item, although it must still acquire the inputs to apply the knowledge. As each Technology may require further items for which the restaurant does not have the Technology or the material in stock, several rounds of make-or-buy decisions may need to take place before anything is actually produced.

Government. VFC contains two governmental functions. One is a data reporting and analysis function (the Bureau of Food Statistics, or BFS), the other is a simple fact distribution service (the Food Data Agency, or FDA). The FDA is a repository of nutritional information on food products. The BFS accepts information from reporters and provides information to requesters.

FDA is intended to be used for several purposes further along the development path. It is our intention to trace all the information used by the various agents. Therefore, we will want to trace how diners knew to ask for donuts at restaurants, and not to ask for doorknobs. The FDA is the arbiter declaring things to be foodstuffs, and diners would need to verify with the FDA that an item they had heard about was a food.

BFS accepts reports that are paired lists of food items and number of servings. Three categories of food reports are accepted: Requested, Ordered, and Delivered. Requested items are those about which diners inquired, and are without regard to whether the item was even carried

at the restaurant at which the request was made. Ordered items are those which the diner ordered at a restaurant that carries the item. Delivered items are those ordered items that the restaurant actually prepared.

Use of the data is two-fold. First, it allows the menu-design market to exploit the difference between what diners inquire about (Requested) and what is available (Ordered or Delivered) to revise menus or to add restaurants filling gaps in diner satisfaction. Second, the difference between Ordered and Delivered indicates a (market-addressable) problem with production.

Last Resort Supplier. The supplier of last resort provides whatever raw materials or manufactured goods are requested of it. Researcher control of the inputs to the diner and supplier of last resort should ensure that any product requested has a production function.

Markets for capital goods and for materials are closely related: both are manufactured goods, but materials depreciate much faster than do capital items. The essential difference is that use of a capital item can be transformed into a service material through the simple act of leasing the capital for a fraction of its useful life.

All non-labor factors of production are themselves outputs of production processes. The specification of the inputs to produce a particular output is done using a production function, which is termed a technology in this paper. With this recursive definition, of course, the process must come to some basic factors that are not themselves produced (land, air, fire, water).

Worker Labor is modeled as a set of individual workers, each of which comes to the labor market with a potentially unique set of capabilities. Each worker is constructed as a separate agent, responsible for its own welfare, principally to be able to investigate individual behavior. Also, varying levels of labor skill is not presently implemented. There is only one worker model. The only skill or function this worker model has is that of selling, wherein the worker engages in a fairly simple exchange with the customer to establish and deliver a food order. The worker passes the order, via a message, back to the employer for fulfillment. Once the employer completes the order the worker presents the finished order to the customer. Workers not under contract become part of a central labor queue. From the central queue, workers can be recruited/assigned to any task. Workers may contract to work for an individual entrepreneur, and such contracting could be a major area of study with VFC.

Diners and Brokers. In VFC, the broker function (Figure 3) is viewed as one where an agent (Broker) is hired to perform a procurement function on behalf of a client. The Broker is expected to perform the procurement operation solely in the best interest of the client. To accomplish this, the Broker is given a list of items from the client and then performs a fixed sequence of

events. First, the Broker asks the Matchmaker for a list of all “wholesale” suppliers registered there. Second, it send messages to each supplier asking for that supplier’s menu (or “catalog”). Third, it processes each received menu (requests that timeout are not processed), looking for the least costly supplier to provide each item on the client’s request list. Fourth, it orders the items from the least cost supplier; fifth it awaits delivery of the items; and sixth it sends a message to the client noting the items found and their cost.

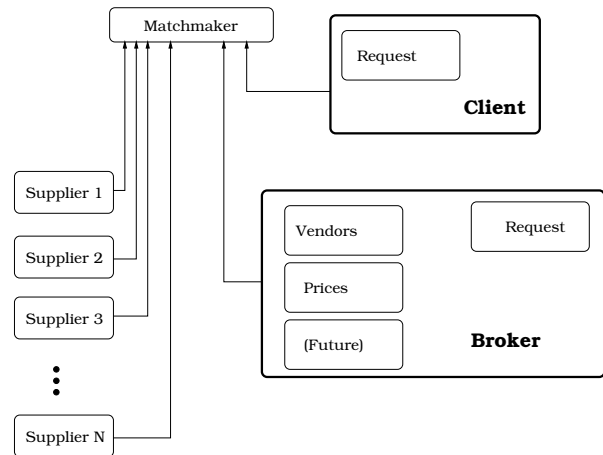


Figure 3. Virtual Food Court Broker

Further, note the similarity in how a Broker would deal with a Restaurant or a supplier of pots and pans. All the Broker needs to know is what items the supplier supplies and how much they cost. In the future, quality issues will be addressed, as well as supply contracts.

6. A Specific Partial Example

Figure 4 shows the two conversations required for a VFC restaurant to hire a waiter. The first conversation is quite simple: the prospective waiter (“Emp”) places an advertisement with the Matchmaker (“MM”) asserting that the worker agent has waiter skills.

For the restaurant to hire an employee, a more complicated conversation is required. The restaurant must discover what agents are available to function as waiters. For the conversation in Figure 4, we have the restaurant (“Rest”) issue a KQML “ask-all” message to the Matchmaker. The matchmaker responds with a list of agents that have advertised as having waiter skills. Upon receipt of the list, the restaurant decides which of the waiters it would like to hire, and makes an offer to the waiter through a KQML message sent to the prospective employee. The worker agent receives the employment offer from the restaurant, and decides whether to

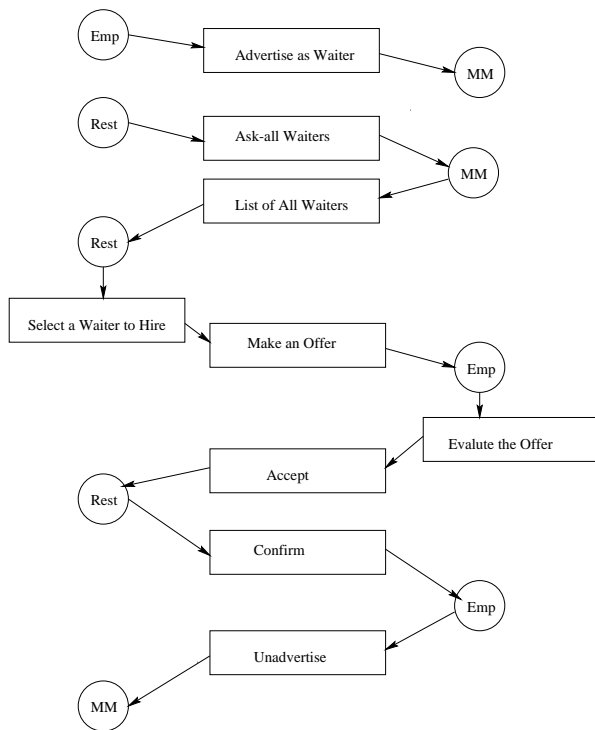


Figure 4. Hiring Conversation

accept it or not. We have shown the conversation with an acceptance, which the worker sends to the restaurant. The restaurant confirms the acceptance, and the employee sends an unadvertise message to the Matchmaker to make itself unavailable to other employers.

7. Conclusion and Future Work

Our development proceeds along two lines. In one line, we are developing the technical production functions and agent-level knowledge storage requirements and mechanisms of the agents. In the other line, we are improving the behavioral repertoire and communication skills of the individual agents. We are encouraged by the speed with which even simple collections of agents can collect and distribute knowledge, so our early emphasis will be on increasing the number of agents at work, particularly on the supply side. Our primary behavioral side issues currently are modeling how a business decides what products to produce and how to form prices. The primary technical side issues are in building a set of production functions (technologies) that reasonably express likely U-shaped short-run production functions for restaurants and manufacturers. As these issues are addressed, we expect the VFC to grow in size and in complexity of exhibited behavior.

References

- [1] K. S. Decker, K. Sycara, and M. Williamson. Middle-agents for the internet. In *Proc. 15th IJ-CAI*, pages 578–583, Japan, August 1997.
- [2] Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex computational task environments. In *Proc. 11th AAAI*, pages 217–224, Washington, July 1993.
- [3] Keith S. Decker and Katia Sycara. Intelligent adaptive information agents. *Journal of Intelligent Information Systems*, 9(3):239–260, 1997.
- [4] Elihu M. Gerson. On ‘quality of life’. *American Sociological Review*, 41:793–806, 1976.
- [5] J. Graham and K.S. Decker. Towards a distributed, environment-centered agent framework. In *Proc. ATAL-99*, pages 162–175, 1999.
- [6] H. Mintzberg. *The Structuring of Organizations*. The Free Press, New York, 1979.
- [7] Charles Perrow. *Complex Organizations*. Random House, New York, 1986.
- [8] Charles J. Petrie. Agent-based engineering, the web, and intelligence. *IEEE Expert*, December 1996.
- [9] A.S. Rao and M.P. Georgeff. BDI agents: From theory to practice. In *Proc. 1st ICMAS*, pages 312–319, San Francisco, 1995.
- [10] Stuart Russell and Eric Wefald. *Do the Right Thing: Studies in Limited Rationality*. MIT Press, Cambridge, MA, 1991.
- [11] R. Simmons, R. Goodwin, K. Haigh, S. Koenig, and J O’Sullivan. A layered architecture for office delivery robots. In *Proc. 1st Intl. Conf. on Autonomous Agents*, pages 245–252, Marina del Rey, 1997.
- [12] Arthur L. Stinchcombe. *Information and Organizations*. University of California Press, Berkeley, CA, 1990.
- [13] T. Wagner, A. Garvey, and V. Lesser. Complex goal criteria and its application in design-to-criteria scheduling. In *Proc. 14th AAAI*, Providence, 1997.
- [14] Oliver E. Williamson. *Markets and Hierarchies: Analysis and Antitrust Implications*. The Free Press, New York, 1975.
- [15] Oliver E. Williamson. *The Mechanisms of Governance*. Oxford University Press, New York, 1996.