DESIGNING SUPER-AGENTS

Sarosh Talukdar
Engineering Design Research Center
Carnegie Mellon University
Pittsburgh, PA 15213

INTRODUCTION
A super-agent is an aggregation of lesser agents, just as a person is an aggregation of cells, a corporation is an aggregation of people, a flock is an aggregation of birds, and a complex program is an aggregation of routines. The aggregation schemes (organizations) of synthetic super-agents often use mechanisms borrowed from other systems. For instance, the hierarchical structures of modern corporations can be traced back to those of ancient tribes, the alternating cycles of creation and destruction in genetic algorithms have been adapted from the theory of evolution, and the arrangements of artificial neural nets have been borrowed from models of the human brain. Where might the designer of future super-agents look for mechanisms to borrow and adapt? How should such mechanisms be combined? More generally, how should the synthetic super-agents of the future be designed?

There is a vast body of literature on organizational design. It goes back to Adam Smith's work on the division of labor ("An Inquiry into the Nature and Causes of the Wealth of Nations," 1776), and stretches even further, to Sun Tzu's the Art of War (circa 500 BC) and the Book of Exodus (circa 1500 BC). Scattered through this literature are organizational structures that have, in certain circumstances, been found to work well. Hierarchies, free markets, task forces, assembly lines, genetic algorithms, simulated annealing and artificial neural nets are some examples. But there is no uniform treatment of such structures, nor any indicators of where new and better structures may be found. In other words, there is no comprehensive space of organizational structures, nor any way to explore such a space, were it to be assembled. In the succeeding material, I will attempt to make good a part of this deficit. Specifically, I will describe a fairly comprehensive space of organizations and outline a procedure, based on this space, for designing organizations for software agents.

SKILLS AS COMMODITIES
Think of a task (problem) as a demand for skills and an agent (problem-solver) as a supply of skills. Then the successfully completion of a task is contingent on meeting its demand for skills from one or more supplies (agents).

TERMINOLOGY
Certain terms that are essential to the succeeding discussions are defined below.
Skill: an ability to transform a given (initial) object or condition into a desired (goal) object or condition.
Operator: a computational model of a skill. More specifically, an operator is a transformation from one set of data-objects (called an in-set) to another (called an out-set). These sets may be arbitrarily large and the mapping arbitrarily complex. In other words, an operator can be used to model arbitrarily complex skills.
Task: a demand for skills. As such, a task can be modeled as a need to map an initial state (such as a set of specifications for a house) to a goal state (such as a set of blueprints that meet the specifications).
Agent: a bundle of skills together with some means for managing their deployment. As such, an agent can be modeled as a set of controlled operators. The controls can be divided into two categories: selectors and schedulers. The selectors pick objects for the operators to transform. The schedulers decide when these transformations will occur.
Autonomous agent: one that makes all its selection and scheduling decisions internally. An agent is non-autonomous (supervised) if it receives some selection or scheduling instructions from other agents.
Super-agent: an aggregation of agents. (Super-agents exist because the skills of any single agent are limited and because there are important tasks whose demands far exceed these limits.)

Organization: a structural model of a super-agent, just as circuit diagram is a structural model of a microelectronic chip and a blueprint is a structural model of a building. More specifically, an organization identifies a set agents and describes how they interact.

Cooperation: any exchange of data among agents, whether productive or not.

Openness: a super-agent is open if it can be assembled and disassembled easily. More specifically, a super-agent is "open to a set of agents, A," if members of A can easily be added to, or removed from, the super-agent.

Task-cover: the net skills of a super-agent. In other words, the set of skills it can synthesize from those provided by its agents. The extent of the task-cover depends on how this synthesis is done and is often less than the union of the agents' skills, but can, occasionally, be greater.

Effectiveness: a super-agent is effective if it performs its assigned tasks well. More specifically, a super-agent is effective over a task if its cover includes the skills demanded by the task.

TASK-SPECIFIC ORGANIZATIONS: THE DESIGN PROBLEM

The effectiveness and openness of a super-agent depend on its organization, that is, on its agents and how they interact. The same agents, interacting in different ways can have very different levels of effectiveness and openness.

Many types of organizations sacrifice openness for effectiveness, or vice-versa. The problem considered here is how to design organizations that make such sacrifices unnecessary. In other words:

Given: T, a set of tasks,
A, a space (set) of existing and anticipated agents, and
C, a means by which the agents in A can communicate;

Design: k*, an organization that is effective over T and open to A.

The intent is to design super-agents that are specialized for the tasks they are to perform. Having completed these tasks, the super-agents could be disassembled, making their agents available for reuse in other super-agents. Or they could be expanded with new agents to become effective over even more difficult tasks.

Note that the extent of A is limited only by the means available for its members to communicate. If the forecasts for growth in communication networks prove to be right, organization-designers could, in the near future, extend A to include virtually every person, robot and other computer-based agent in the world.

REPRESENTING ORGANIZATIONS

This section argues that any super-agent can be thought of as a net of memories and agents. This net can be represented by a pair of directed graphs, one to show the paths by which control flows in the super-agent, the other to show the paths taken by data. The focus is on organizations for computer-based agents. Other sorts of agents, such as insects, fish and even people, are included provided they communicate in ways no more subtle than those used by computers.

Modeling Assumptions

1. The communication processes used by agents can be simulated by shared memory processes, that is, by agents reading from, and writing to, memories they share. (All the communication processes used by computer-based agents satisfy this assumption. I do not know the extent, if any, to which other agents may violate it.)

2. The contents of each shared memory are stored in the form of data-objects, that is, encapsulated packages of data. (This assumption has no effect on model fidelity. Its only purpose is to establish a convenient visualization device, called a state space, for each memory. This space is the set of all the data-objects that can be stored in the memory.)

Data And Control Flows

The definition of an agent together with the modeling assumptions allow one think of an organization as a net k, of memories and
agents, and further, to represent this net by a double:

\[ k(a) = \{Df(a), Cf(a)\} \]

where \( a \) is the set of agents in the organization, and \( Df \) and \( Cf \) are graphs. More specifically, \( Cf \) is a directed graph, called a control flow, whose nodes denote agents, and whose arcs denote the supervisory relations among these agents. \( Df \) is a special type of directed graph, called a data flow, whose nodes are Venn diagrams. Each of these diagrams shows the intersections between the state space of a memory and the in- and out-spaces of the agents that share (can read from or write to) that memory. Arcs in \( Df \) identify the in- and out-spaces of each agent.

Note that:
- If the control flow of an organization is null (a graph with no edges), then its agents are autonomous.
- Agents in disconnected parts of a data flow can cooperate (exchange information) in only very limited ways--by issuing commands that must travel over the routes, if any, provided by the control flow.
- Most organizations have a few memories to which agents write infrequently, if at all. Henceforth, such memories will be called reference memories to distinguish them from the other memories which will be called working memories. The purpose of a reference memory is to store information of long-term value, such as reusable knowledge, learned knowledge, and the invariant parts of task descriptions. The purpose of the working memories is store the transient results of an organization's activity. As we shall see, working memories reflect the task decomposition scheme used by the organization. In fact, each working memory is dedicated to some sub-task.

**Organization Spaces**

Let \( A \) be the given set of agents and \( S(A) \), called the organization space of \( A \), be the set of all the different ways in which the members of \( A \) can be organized. In other words, \( S(A) \) is the set of all \( k(a) \) for all \( a \), where \( a \) is a subset of \( A \).

**A TAXONOMY OF ORGANIZATIONS**

Even when \( A \) is small and homogeneous, \( S(A) \) can be large and surprisingly diverse. In the following material we will partition \( S(A) \) into smaller more homogeneous regions.

**Feedback and Autonomy**

Suppose that \( S(A) \) is partitioned along its data-flow-axis into organizations with strongly cyclic data flows on one side and organizations with essentially acyclic data flows on the other; suppose also that \( S(A) \) is partitioned along its control-flow-axis into organizations with mostly autonomous agents on one side and mostly non-autonomous agents on the other (precise membership functions for these fuzzy partitions are not important here). As a result, \( S(A) \) is cut into four qualitatively different regions (Fig. 1).

Organizations from the two upper regions of Fig. 1 rely on the use of supervisors, and usually, hierarchical control flows. Consequently, they tend to be more efficient in their use of agents but less robust. (Agents that are told what to do by their supervisors are less at risk of working at cross purposes than autonomous agents. On the down side, supervisory errors are amplified by hierarchical control flows, and the loss of a key supervisor can be crippling to the organization.)

Organizations from the two regions on the right of Fig. 1 use acyclic data flows. This makes them well suited to repetitive and routine tasks, but ill suited to tasks that are uncertain or have conflicting requirements, because such tasks invariably require feedback and iteration, that is, cyclic data flows.

**Differential, Integral and Hybrid Task-Covers**

To be effective over a task, the cover (set of net skills) of a super-agent must include the skills demanded by the task. This cover can have three forms: differential, integral and hybrid.

In a differential cover, the overall task is decomposed into sub tasks so each can be assigned to a single agent. (Note: these sub tasks do not have to be disjoint. some may overlap in the skills they demand, and some may combine or otherwise use the results of other tasks.) An integral organization does not decompose the task; rather, it integrates the skills of its agents to meet the demands of the
overall task. A hybrid organization is part
differential, part integral.

Processes by which differential covers are
formed have been carefully studied for many
centuries (see, for instance, the writings of Adam
Smith, 1776) under titles such as "the division of
labor" and "the delegation of responsibility." They
are now well understood and intuitive.
Integral covers are much less familiar. By way of
an illustration, consider the social insects (ants,
termites, certain bees and wasps). For them, the
construction of a nest is a massive engineering
project requiring the cooperation of many
workers. Invariably, the nest is sophisticated in
architecture and customized for its surroundings:
features that are remarkable for four reasons.
First, there are no leaders to decide what is to be
done, nor any centralized coordination of effort.
Second, the workers have access only to local
information (in space as well as in time). Third, the
individual insect has only modest capabilities for
learning, reasoning, and abstracting knowledge.
Fourth, there are no blueprints to show what the
final result should be nor any centralized
planning. In other words, there are no provisions
for decomposing the overall task into sub tasks,
nor any assignment of sub tasks to individual
workers. Rather, the insects act as autonomous
agents and attack the overall task en masse.
Plans, designs and other products of task
decomposition, if they exist at all, are
unobservable. In still other words, the insects
use integral covers: their organizations integrate
their skills to cover the demands of large tasks
and in the process, allow them to do so much
with so little (in the way of individual skills).

Note that the form of a super-agent's task-cover
is determined by the form of its data flow: if this
flow has only one working memory, the cover is
integral, if it has several working memories but
only one operator terminating at each of these,
then the cover is differential, otherwise, the
cover is a hybrid. Also, in a differential or hybrid
cover, each working memory is dedicated to one
task.

It follows that each of the regions of Fig. 1 can be
divided into three sub-regions by partitioning it
along the data-flow-axis corresponding to
differential, integral and hybrid covers.

Organizations that use differential covers are the
most familiar, and therefore, the easiest to
design, but hybrid and integral covers have
some very desirable properties that we that we
do not have the space to discuss here.

A DESIGN PROTOCOL
The taxonomy above suggests a design protocol with three stages that may be performed
once, but more often, require iteration:
1. Based on the characteristics of the tasks to be
performed (whether they are routine or not,
whether centralized planning and control
would seem to be indicated or not, and
whether the tasks are decomposable or not),
select one of the 12 sub regions of the
organization space. This selection determines
the gross characteristics of the organization's
data and control flows.
2. Complete the data flow. The steps are:
   • Decompose the tasks into sub tasks. Design a
     working memory for each sub task (choose
     how many estimates to the sub task-solution
     will be held by the memory and how these
     estimates are to be represented).
   • Design and stock reference memories for each
     sub task.
   • Choose the mix of agents to write to
     each
     working memory paying particular attention to
     the balance between their construction and
     destruction skills.
3. Complete the control flow. The steps are:
   • Design the selection controls for each agent.
   • Design the scheduling controls for each
     agent.
   • Distribute these controls, that is, design the
     supervisory links among the agents.

ASYNCHRONOUS TEAMS
One of the least explored and most interesting
parts of organization space (Fig. 1) is in its South-
West corner. Specifically, it is the subsubregion
of strongly cyclic data flows, virtually null control
flows (most, if not all, the agents are
autonomous), hybrid covers and asynchronous
cooperation. (Recall that agents are said to
collaborate if they exchange data. We will say that
they cooperate asynchronously if the exchanges
occur so no agent is required to pause or wait for
data being produced by another. In other words,
asynchronous cooperation allows all the agents
to work in parallel all the time.) We will call
members of this subsubregion asynchronous teams (A-Teams, for short).

**Advantages and Disadvantages**

Autonomous agents that cooperate asynchronously have considerable advantages. They yield super-agents that are extremely open and can mount massively parallel efforts without the need for centralized schedulers or any fear of deadlocks. But how is all this computational effort to be put to good use? Any agent, able to decide for itself what, if anything, to do could choose to sit idle or even undo the work of its fellow agents. When computer costs were high, such individual freedoms and the resultant wastage of computer cycles, were unthinkable. But now, with costs decreasing by

the month, large networks of computers, each dedicated to one agent, can be assembled. Individual productivity is of little consequence. Some agents may be unproductive or even counterproductive as long as the collection of agents, as a whole, is effective in performing its assigned tasks. How may this overall effectiveness be achieved? We have developed a set of case studies that provide some guidance. Space constraints allow me to only partially list these case studies here. They include: traveling salesmen problems, high-rise building design, task-specific robot design, network diagnosis, contingency planning for electric networks, train scheduling, job-shop scheduling and steel mill scheduling.

![Fig. 1. A taxonomy of super-agents obtained by partitioning S(k), the space of all super-agents, into four fuzzy regions. All the regions contain natural and synthetic super-agents but the distribution is far from uniform. Specifically, there are few examples of synthetic super-agents in the lower left region. In contrast, many natural super-agents have found it advantageous to reside there. Each of the above four regions can be divided into three sub regions by the form of its task-cover (differential, integral or hybrid).](image)