CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

# A Theory of the Knowledge Environment for Construction Automation

by

Lai Heng Chua

**TECHNICAL REPORT**
**Number 22A**

January, 1990

**Stanford University**

# Acknowledgements

I wish to thank the construction engineering and management faculty, Professors Boyd C. Paulson, Raymond E. Levitt, Clyde B. Tatum and John W. Fondahl for providing me with an enriching time. I am forever in their debt for the things they taught me both in class and outside. Some of the ideas explored in this thesis have their roots organizational theory, short-interval planning and site operational analysis; for the management and control of autonomous machines, in the last analysis, revolves around the same basic issues.

Professor Paulson has been instrumental not only in the writing of this thesis but also in the understanding the overall construction automation problem and the directions for construction automation researchers. I revised my thinking more than a few times during this difficult early stage of pioneering construction automation research here. There were forays into hardware, control theory, commonsense reasoning, qualitative reasoning and learning of process operations, among others. We needed a sufficient understanding of these matters to be in the position to decide the desired overall direction of research.

I have also been privileged to know such brilliant students as Charlie Koo, Weng-Tat Chan, Nadeem Babar and David Zhu. They inspired me to take difficult courses and work on the most difficult problems. I also sounded out some of my ideas on computer science students, not a few of whom were almost too keen to argue the smallest point.

The National Science Foundation, Construction Institute, Center for Integrated Facility Engineering and Stanford University have been very kind in providing me financial support during my stay here. I benefited from NSF grants MSM-84-06852 and MSM-86-14238, grants from the Center for Integrated Facility Engineering, fellowship funds from the Construction Institute and tuition waivers from Stanford University.

# Contents

# List of Figures

# 1. Introduction

In future construction environments, there will be increased use of intelligent machines. This need is driven by demographic changes and other reasons [Skibniewski 86, Warszawski and Sangrey 85, Sangrey 85, Paulson 85]. Rapid advances in hardware technology and reductions in the cost of such systems will make it increasingly attractive to use robots in construction. These machines will work in the environment of other machines and humans, and often in conjunction with them. They must harness considerable knowledge to plan and control autonomous tasks [Paulson et al 89]. As with humans, machines are limited in knowledge and abilities and would often need help to perform various tasks. With increasing numbers of machines we believe that it will become more and more difficult for humans to manage and control every aspect of such systems. Machines need to be given more autonomy and intelligence.

This thesis presents concepts and methods that would help avoid the need prepare and prime automation systems with all the knowledge and information potentially required. The effort required to think of all contingencies and problems, work out solutions for them and input all the potentially useful information and knowledge in anticipation of possible future use is considerable. We believe that machines could eventually be given abilities to actively seek and obtain the knowledge or help whenever the need is discovered. This shifts some of the burden of determining the needs of the tasks to the automation system.

The current approach that uses human supervision recognizes the impracticality of completely preconfiguring construction automation systems. Heavy monitoring is used because adjustments have to be made continually to such systems. Unfortunately, current human supervision methodology, while suitable for this age and state of technology, leaves much to be desired. The links between the human and the machines are low-level and so the human control effort is high, and in most cases the actions of the machine are directly modified by a supervisor who is one step removed from the scene of action. Clearly such a methodology strongly constraints the complexity of automation systems that can be created and managed.

We also suggest that automation systems would need to be adapted to the conditions and structure of human work as an embedded system. There is a need for interaction between machines and humans in many cooperative, production and problem-solving tasks because so much goes on in the human world; resources are provided by human suppliers, many functional capacities and skills and even sheer workforce are contributed by humans, and there are vast stores of knowledge accessible to human cognitive abilities. The situation is not likely to change anywhere in the foreseeable future. The ability to actively seek and use human inputs would be a significant advantage for automation systems.

This thesis proposes the construction of machines that have environmental intelligence. It also proposes that machines be developed to help manage other machines to reduce the automation management load. The incorporation of environmental intelligence in field machines will better enable them to take action when faced with situations for which they have not been previously prepared. The theory is directed at making such systems a technological option. The psychological, social and political dimensions of providing the degree of capabilities and autonomy to machine systems are not dealt with here, although the eventual use or even design of automated systems depends on them.

The research required to achieve these goals is considerable. This thesis represents preliminary research only. A fraction of the work by other researchers on related problems is reviewed. We hope that eventually a body of principles, algorithms and software to enable agents to function effectively in the construction environment will be developed.

## 1.1. Terminology

### 1.1.1. Agents

For our purposes the notion of agents is rather broad and we will not try to pin it down precisely. Human beings are the prototype agents. Agents are the basic units for autonomous reasoning, decision-making and actions, but we will not provide a decision theory with which we can conclusively classify whether something is an agent or not an agent. We have no desire to have philosophical tussles over questions such as whether the human body is part of the human agent or not since the brain alone could have met our specification — we leave them for others. Machine agents are thus capable of some autonomous reasoning, intelligent decision-making and actions.

### 1.1.2. Organizations

Groups of agents related systematically or interacting in systematic ways comprise organizations. Several organizations may work together to form an even larger organization. An organization has many of the characteristics of agents, including those of autonomous decision-making and actions. So much of what we will discuss concerning agents also applies to organizations. Frequently, we will refer to both agents and organizations as just agents or as environmental entities.

## 1.2. Motivation

### 1.2.1. Agent Limitations

Agents are limited; their knowledge is limited, their reasoning capacities are limited and so are their physical capabilities. Therefore, there are situations that they cannot handle properly themselves. Besides, agents are limited in the selection of their environments; often the environment already exists before the agent comes to the scene and the agent has to begin with that existing situation.

Given a goal or a problem to solve, an agent may be unable to derive a plan to achieve the goal or obtain the solution to the problem. This may be so because it does not have the information or the reasoning capabilities required. Given a plan to achieve a goal, an agent might be unable to achieve the plan because of lack of resources on hand. Given a plan to achieve a goal and the resources required, an agent might be unable to execute it because of physical limitations. Increasing the direct capabilities of the agent itself ameliorates some of these limitations, but the environment provides numerous opportunities that can aid an agent in dealing with more complex problems.

By knowledge-limited we refer particularly to the limited extent of an agent's knowledge at any point in time. Ignorance is of several types and can result from several causes. Ignorance can result from being knowledge-limited and bounded in reasoning; these are the types of ignorance we might be able to do something about. Another reason for ignorance is that agents have bounded physical observation extent and may not observe events occuring in another place. For example, a manager in the office might not know that a machine in the field has broken down. To know about such an event, an agent has to learn about it indirectly. But if the knowledge exists, although it is in some other place, an agent might be able to do something about its ignorance. If an agent is unable to process the information itself but processing capacity exists in the environment, then the agent might be able to do something about its limitations. But ignorance can result from fundamental causes, such as randomness of nature and observation limits: Such ignorance is outside of our consideration.

We know that environmental opportunities are heavily exploited in construction and business. The human agents in construction and business organizations have considerable environmental intelligence and exploitation capabilities. We propose that automation and management systems be given capabilities to exploit some of the opportunities that exist in the environment. The next subsection discusses these environmental opportunities.

### 1.2.2. Environmental Opportunities

To deal with many tasks and problems, an agent may take advantage of opportunities in the environment. These opportunities include

i) availability of information,
ii) availability of information processing and problem-solving capabilities,
iii) availability of resources, and
iv) availability of task performance capabilities.

A suitably intelligent agent may meet some of its informational needs by obtaining the information from appropriate sources in the environment. For example, lawyers and researchers often use bibliographic databases and libraries to obtain information about particular topics of interest. There is no need to have all that information pre-encoded in an agent, provided the agent is endowed with intelligence and capabilities to seek external information.

An agent might have the information processed by other machines or agents in the environment into forms more useful to it or others. Humans have this ability and use it to a considerable extent. It is clear that human computational abilities are limited and specialized, but this does not prevent them from having the computations done. The computational abilities of individual machines are also limited and they can benefit by distributing the computational load to other agents. Distributed processing methods help share computational loads, but only among machines.

The agent might also be able to assemble resources that it does not already have from supplies of these resources in the environment. The same goes for equipment. Putting these capabilities together in machines will reduce the management and advance preparation needed to support construction automation systems.

An agent can also obtain the help of other agents in the environment to achieve its tasks. For instance, an agent can request help to obtain information, tools, equipment and other resources. It can employ several agents to help it perform tasks that exceed its abilities or complete tasks more quickly.

Without environmental opportunities we can only attempt to address the problem of agents' limitations by building more powerful and knowledgeable agents. But environmental opportunities exist, so it might be fruitful to provide means to use some of these opportunities.

Humans already exploit environmental sources to a large degree. Human managers readily take advantage of all four types of opportunities listed above. Human workers do so as well. Current machines do not have the intelligence or capabilities to take advantage of even a few of these opportunities, and this excludes the use of machines in situations requiring cooperation of many agents or unfolding dynamically in ways that are sometimes not *a priori* predictable. We explore this human ability in this thesis in order that machines, in the form of automation and computerized management systems, may also be provided with similar, although lesser, abilities to exploit the environment in numerous ways.

### 1.2.3. Robustness

There are differences in nature between human organizations and machine systems or computer programs. The former is much more robust and flexible. Human organizations can lose parts, interchange them with others, or grow to meet new demands. An integration mechanism that makes it possible for humans to work with each other appears to be constituted in most human agents .

Current machine systems and computer programs are tightly integrated internally. Strong and unique dependencies tend to exist between components of a particular machine system or a program. This type of strong integration is akin to that of the human body; each component or subroutine has specific functions and the interrelationships among these components are hard-wired. Specializations also occur in human organizations, but the range of interaction that a component in an organization can have is much greater.

Several matters suggest that increased flexibility may be valuable for construction automation systems. On the positive side, there is the likely need to interact with other environmental entities. For example, the automation system might need to obtain information from various sources, resources from suppliers, help from specialists and subcontractors. On the negative side, uncertainties and unexpected events may occur in the construction environment. The soil may differ from that expected. A window may break during installation. A machine might get stuck in waterlogged ground. A truck ferrying materials to the site may be blocking the route of an automated machine. One can prepare for some of these possibilities, but it is unlikely that one can prepare for them all. This suggests that we should incorporate some environmental intelligence into construction automation systems to obtain some of that power, robustness and flexibility.

## 1.3. Objectives

The principal objective of this thesis is to better understand the knowledge environment for construction automation systems. A theory of the knowledge environment for construction automation is constructed and described. It presents a viewpoint on the software and multi-agent interaction problems in construction automation that are intended to help researchers better understand the issues, identify approaches and thus deal with the appropriate problem. Several concepts, approaches and techniques are introduced.

## 1.4. Methodology

This thesis is descriptive and conceptual. The descriptive portions mainly describe various aspects of the environment and human organizations. The conceptual parts introduce notions of the environment and ways of thinking about the design of automation and management systems. The research is only preliminary and awaits much elaboration and extension.

## 1.4.1. Descriptive

We wish to place automation in the context of the extant environment. The thesis presents a brief description of the general background of productive work, and describes problems faced by construction automation systems. It notes the power of human management and argues for the value of incorporating human-like environmental capabilities in automation and management systems.

#### *1.4.1.1. Organization of Material*

The thesis is organized into three main portions:

I. Motivation and challenges for construction automation systems
II. A theory of the knowledge environment
III. Conceptual design of programs, machine agents and management systems in light of the theory

The discussion topics of (I) are

i) agent limitations,
ii) environmental opportunities,
iii) creation of robust systems,
iv) pre-production activities,
v) interdependencies among environmental entities,
vi) dynamic open environment,
vii) management of machines,
viii) assembling automation systems,
ix) human management of machines,
x) interaction and integration of machines, and
xi) cost of automation systems.

The topics of (II) are

i) general background,
ii) components of the knowledge environment,
iii) characteristics of the knowledge environment, and
iv) relationships and interaction of environmental entities.

The topics of (III) are

i) suggested approach to construction automation system design,

ii) architecture and design of problem solving programs,
iii) architecture and design of agents,
iv) architecture and design of management systems,
v) operational policies, and
vi) environmental mechanisms.

Chapter 11 brings the pieces of the theory together through a brief conceptual description of a hypothetical automation system embedded in the human work environment. It describes the engineering prescribed at the level of agent, site and world.

*1.4.1.2. Reader's Guide*

This thesis covers many areas. Readers may be interested in perusing only a portion of the research most relevant to their work. Although the contents would serve as the best guide, several areas are specially noted here.

Planning is a topic of wide interest to many researchers. Planning is discussed in several places such as sections 3.4.1, 4.5 and 7.5.

Those concerned with the development of intelligent automation systems may like to read discussions in sections 3.4.3, 3.4.4, 3.5, 5.1.2, 5.2, 6, 7, 8, 10.3, 10.5.2 and 11.

Those who are interested in extensibility ideas only may like to look at sections 1.2.2, 4.4, 6, 7, 8.1 and 9.1.

Those interested in the information issues may like to read sections 2.2, 2.3, 3.3.2.1, 4, 6.1, 7.8.3, 9 and 10.5.1.

Researchers interested in distributed artificial intelligence may wish to read 1.5, 2.6, 3.4.4, 3.4.5, 4, 5, 6, 8, 9 and 10.

### 1.4.2. Conceptual

The concepts mentioned or introduced include:

i) open system
ii) environmental intelligence
iii) graded reachability
iv) general core
v) planning horizons

vi) lazy planning
vii) information decay
viii) self-enhancement
ix) agent bases
x) environmental facilitation mechanism of goal transformation
xi) goal reformulations
xii) operational principle of justice

Suggestions and arguments are made for various agent and environmental operational principles, but these do not eliminate other approaches.

## 1.5. Related Work

### 1.5.1. Overview: Distributed Artificial Intelligence

The best introductory source of materials in distributed artificial research is a compendium of research articles compiled by [Bond and Gasser 88]. They described the problem of distributed artificial intelligence as involving

i) description, decomposition, distribution and allocation of tasks,
ii) interaction, language and communication,
iii) coherence and coordination,
iv) modeling other agents and organized activity, and
v) interagent disparities: uncertainty and conflict.

This thesis will touch on many of these areas but we will not provide a thorough review of the extensive materials. The majority of researchers are concerned with the creation of computational ecologies. Although we will deal with both the creation of knowledge environments and living within them, our approach focuses on the extant knowledge environment and the requirements of agents functioning within it. Other types of knowledge environments can be created and some of these might even provide better efficiencies and coherence than the existing one.

Coordination is known to be a difficult problem, whether it is matching producers and consumers or assembling, organizing and motivating a group of agents to construct a building. The mechanisms that are used in the real world to promote cooperation should be a source of inspiration to those attempting to construct better information and automation systems [Bond and Gasser 88]. There are about a dozen major classes of these mechanisms, including markets, negotiations, meetings and organizational structures. Within a class a large number of variants are possible.

To achieve coordination, data-directed and goal-directed control may be used together [Corkill et al 82]. Blackboard systems are used at Corkill's computation nodes. [Corkill and Lesser 83] proposed the use of meta-level control for coordination in distributed problem-solving networks. The use of partial global plans for coordination was proposed by [Durfee and Lesser 87]. An experimental testbed created for exploring such concepts is described in [Lesser et al 82, Lesser and Corkill 83]. This involves the monitoring of vehicle movements through acquisition and interpretation of data from sensors on a spatial grid.

Market-like mechanisms have been proposed for matching tasks to capabilities in distributed flexible manufacturing cells [Parunak 87, Shaw 87, Baker 88], air-traffic control [Steeb 81] and electronic warfare [Cammarata 83, Boettcher et al 87]. Load distribution in computer networks may also be achieved with market-like mechanisms [Malone 88]. The negotiation mechanism has been proposed for distributed multiagent planning in communication [Conry et al 86], manufacturing [Koo 86, 87, 88] and construction [Koo 86]. These research publications and reports constitute but an important fraction of the related work that ranges from social and evolutionary theory to experimental testbeds to prototype applications. A few dozen researchers are currently involved in this field of study.

### 1.5.2. Environmental Issues

Construction and business environments are open environments. Some of the problems and challenges of such environments have been discussed in [Hewitt 84, 85] primarily in the context of office systems. Many issues relevant to this research are mentioned there. Hewitt proceeded to develop and describe computation models for such open systems. The actor language is documented in [Gul Agha 86]. A method of solving conflicts in open systems has also been proposed and is under development; it is called due process reasoning [Hewitt 88]. [Kahn and Miller 88] discussed the reasons why today's programming languages and their straightforward extensions, with the exception of actor and concurrent logic programming languages, are inadequate for programming open systems. One of the reasons is, of course, the lack of fine-level concurrency.

### 1.5.3. Reasoning about Knowledge and Belief

Where several agents meet to exchange information and make decisions, there is often the need for each agent to reason about the knowledge and beliefs of the other agents. The notion of knowledge in this area of research is that an agent can only know something true whereas it might believe something that happened to be assertionally false. A substantial body of theory has been developed in the last few decades starting with the work of philosophers including [Hintikka 62]. Recent work in the area have tried to explicate the cases where common knowledge is needed [Halpern 84], when such common knowledge can be attained through communication, and the complexity of the reasoning processes [Halpern 87]. In simple terms, common knowledge of X of a group means that all agents of the group know X and all agents know that X is common knowledge. Under Byzantine conditions, where communication is not guaranteed, common knowledge cannot be attained and certain decisions cannot be made.

### 1.5.4. CONTRACT NET and Derivatives

The CONTRACT NET [Smith 79, 80, 81, Davis and Smith 83] is a powerful and general fully distributed system modelled on a system of bidding and negotiation. Bidding is the primary mechanism of distributing work and the main source of power of the system. We have not seen much use of negotiation in the problems described. All agents in the system understand the common contract language and protocol. Only those agents that need to negotiate a task have to understand the particular special task language. In this model there are *managers* and *contractors*. Any agent in the system can be a manager, contractor or both. Managers generate tasks and inform other agents of these tasks. Contractors bid on these tasks if they can perform them. Managers select the best contractor/s from those bidding, award contracts to these contractors and then manage the tasks performed by them. Managers may suspend or terminate the performance of tasks. Reports may also be requested from the contractors. In this model, award and management of the award are bound together. Other mechanisms, such as direct contract award, broadcast of idle node availability, etc., address some of the shortcomings of the primary model. Negotiation in the CONTRACT NET is not as general as it might be. Instead of exchanging information and coming to an agreement, one party states what it wants done and the other party evaluates and bids. There is no two-way exchange of information prior to contract award unless the parties state that the task is information transfer. It is assumed that the bidder is qualified to bid, which simplifies matters.

Some changes and enhancements have been proposed to the basic CONTRACT NET model by [Parunak 87, Baker 88] for use in manufacturing. Parunak proposed hierarchical negotiation layers. Baker proposed using a cost metric, and this makes it closer to a market mechanism than those proposed by others. A very ambitious program has been proposed by [Miller 88] that goes far beyond what we will consider in this proposal. He calls the systems, which build on the metaphor of current social and economic systems, Agoric Open Systems.

### 1.5.5. Distributed Planning

Planning can be done in a distributed fashion [Koo 87]. Local plans can be independently developed by agents, and the interactions among these plans can be resolved through communication and negotiation. Thus at no time does any agent or system know of the overall plan. An intelligent communication system tracks the commitments made by agents. COMTRAC-O is the language developed to allow such tracking [Koo 87]. Koo's research does not answer the problem of achieving the best construction schedule. Such a thing is dependent on the existence of a least-constrained planner. Negotiation does not force short construction sequence. Individual agents can be happy with a negotiated plan that satisfies all their wants but is not the best globally.

[Corkill 79] described the use of hierarchical planning in a distributed environment. Corkill basically took the hierarchical non-linear planner, NOAH, developed by [Sacerdoti 77] and devised a way of distributing pieces of the algorithm. Other related work in multiagent planning include [Konolige 80a, 80b, Georgeff 82, Stuart 85, Katz and Rosenschein 89]. Most of these papers deal with generation of plans to be executed by multiple agents and a few with the problem of synchronizing multiagent plans.

# 2. General Background

## 2.1. Environment

The total environment is the world — the complete context of all activities. From an agent's viewpoint, the viewpoint of interest, it is everything external to the agent. For our convenience, we conceptualize the environment as comprising the physical environment and the knowledge environment (Figure 1). Both environments have a certain order and structure. The environment important to the functioning of an agent we call the agent's *relevant* environment.

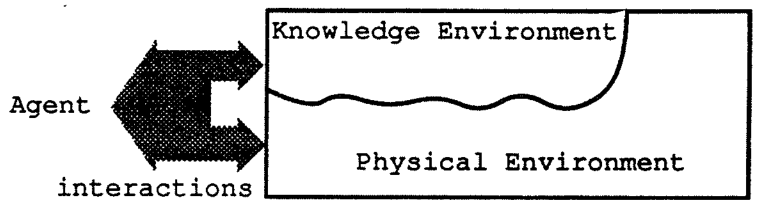


Figure 1. Interaction of Agent with Knowledge and Physical Environments

The physical environment comprises all material things. The principles that govern the physical environment are the laws of physics, which we understand at several levels of expression. For mobile robot agents, two aspects of the physical environment are especially important — the spatial aspect and the temporal aspect. All physical objects have size and location. Two physical objects cannot overlap in space. These are conclusions drawn from the physical laws. We may regard them as epiphenomenological laws. The laws of chemistry are also epiphenomenological laws. By epiphenomenological we mean that they arise from the operation of more fundamental principles.

All biological creatures have some measure of ability to function in the physical environment. They have sensory, motor and processing capabilities appropriate to their needs. By taking physical actions, the agent modifies part of the physical environment or its physical self. On the other hand, the abilities to function in the relevant physical environment have to be designed into machine agents.

But there is another aspect of the environment — the knowledge aspect that is important to the function of intelligent agents. Not only do we want to be able to move about, but also to perform tasks and make other intelligent decisions. For instance, we may want to remove obstacles from a site instead of avoiding them. The performance of some tasks depends on the physical abilities of the robot as well as the knowledge that the robot has. Part of this knowledge may be the robot's directly applicable knowledge to deal with the task, or it may be knowledge to acquire further knowledge from the knowledge environment to deal with the task.

What is the knowledge environment like? How can an agent function in it? Since many of the objects that comprise the knowledge environment have a physical manifestation, there is a great deal of structure in the knowledge environment that parallels that of the physical environment. Thus to exploit some aspects of the knowledge environment an agent may need the mediation of physical and sensory capabilities. Nevertheless, the knowledge environment is very much a human creation with human imposed order and structure. Much of it was brought into existence by human control and creative activities — things are as they are because humans made them so. No other biological creature can actively function in the knowledge environment. Just as we need to provide machines with the capabilities to function in the physical environment, we could do the same for agents with respect to the knowledge environment.

The knowledge environment has a certain amount of order and structure of its own. It has its own governing principles and temporal characteristics, even though the principles that govern the knowledge environment are not as rigid as those that govern the physical environment. The structure of the knowledge environment as it exists now is not unique but merely one of many possibilities. While the extent of influence of an agent in the knowledge environment may be greater than that in a physical environment because the knowledge environment is an artificial construct, the agent has to take many things as they exist and continue from there. An agent may not modify the laws of physics, although it might change the location of an object or the structure of the terrain. In the real-world knowledge environment, a construction agent dealing with material suppliers has to follow the operating principles of such markets if it is to be able to use them.

In manufacturing automation, a small part of the world is restructured for the robot. Basically, the only access to the world for such a robot agent is the specially structured environment and the rest of the environment remains out of reach. Construction, on the other hand, involves the task of structuring the environment itself rather than merely working in an already structured environment. An agent could profit from participating actively in the environment. To do so, the agent might need to understand the structure and operating principles of the knowledge environment as it exists.

## 2.2. Goods, Services and Consumers

Goods and services are needed to meet the needs or desires of entities in the environment — the consumers. Humans have a strong consumer orientation. However, the environment is not so bountiful that little or no effort is needed to meet consumption needs. Most goods have to be created from available components of the environment, thus requiring humans to produce if they wish to consume. In current society, it is humans and whatever helpmates they harness that contribute a great deal of this productive effort. One might mistakenly think that it behoves agents to produce what they consume, but there are several reasons why this is not the case. It is less demanding to meet needs by exploiting opportunities in the environment. Certain economies of scale can be exploited as well.

One of the striking aspects of modern human society is the division between production and consumption. Goods and services are the products, but they are consumed away from their production source. Agents no longer form the binding units of production and consumption; that is, agents rarely produce all that they consume or consume only what they produce. Even organizations do not form such binding units, although the largest ones might come closer. This separation of production and consumption creates additional interdependencies between environmental entities. While this is the extant environment, currently machines do not participate in it.

Nevertheless, there must be a link between production and consumption. This link occurs through mediation mechanisms of transaction among environmental entities. These mechanisms try to match goods and services to desires and needs within the constraints that the environment imposes. The economic system represents the expression of these environmental mechanisms — in particular, the use of tokens of exchange for transaction mediation. Other mechanisms are required to provide a sufficiently stable environment.

The producer-consumer and client-service relationships are the basic types of relationships that an agent has with other agents. In actuality there are two levels of relationships: producer-distributor and distributor-consumer. The distributor here is an environmental mediation mechanism similar in purpose to that of tokens of exchange. The notion of client-service is beginning to be incorporated into computer applications, but we would like to have both in automation systems.

## 2.3. Information Production and Consumption

Data, information and knowledge are also produced and consumed. They are secondary to the needs for goods and services. Nevertheless, they often have supportive roles and strongly affect the ability to meet those needs. Their importance should be noted accordingly. Just as there are two levels in the producer-consumer relationship, there are two levels in the production and consumption of information. The first is author-publisher and the second is publisher-subscriber. The publisher is the mediator between the source of the data, information or knowledge and the consumer of the data, information or knowledge.

Automation systems have need for information and can produce information byproducts. We will address their production and consumption of information. Where that information comes from and how it does so are subjects of inquiry.

## 2.4. Basis of Production

Goods and services are the fundamental results of production by environmental entities. They are required to meet the needs or desires of entities in the environment. However, they are rarely free for the taking. Production is the primary effort needed to create both goods and services.

The conceptual sequence of conditions and activity that eventually lead to some end product, such as a built facility, is simplistically depicted in Figure 2. The diagram is simplified because, among other things, there may be competing desires or needs; the three steps of determining a course of action, preparation and execution can occur simultaneously; and the end result may not be products. The bottom part of the figure is adapted from [Oglesby et al 89 pp. 64]. We instantiate the diagram with an example in construction and show that in Figure 3.

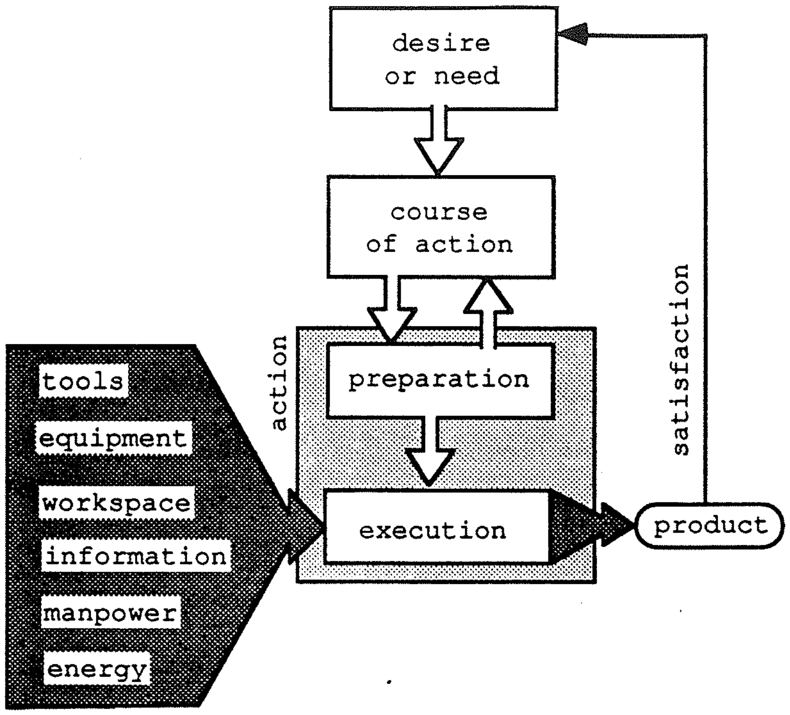


Figure 2. Basis of Production

As we can see, the driving forces of this constructive process are the desires and needs. These forces are spontaneously generated in living organisms. To fulfill these desires or needs, less intelligent organisms just do what worked in their evolutionary past. Smarter ones, given sufficient time, can analyze, plan and quickly modify their strategies with changing circumstances and accumulated experience.

Given the availability of the inputs required for production, the minimal activities that create the product are called focal productive activities. In construction parlance they are often called the work-face activities.

The place of traditional robotics in this diagram is basically that of an execution engine. Tools, workspace, information and energy are provided and products are churned out. Much effort has been devoted to the creation of this execution engine. Providing robots with the abilities to perform focal productive activities has been central in the design of machinery and preprogrammed robotics. The focus have been on kinematic and control issues. Such an automation system is managed by humans who set it up and provide the tools, workspace, information and energy.

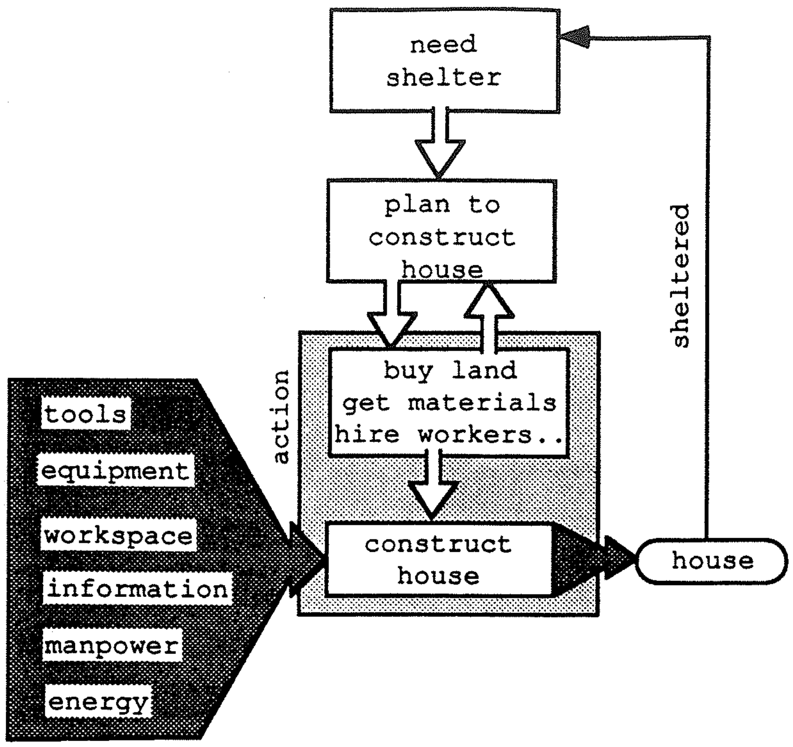


Figure 3. Motivation to Construct a House

## 2.5. Pre-production Activities

The general preconditions for an agent to make direct use of X, where X is a tool, a piece of equipment, materials or information are

i) have X,
ii) know how to use X, and sometimes,
iii) have the right to use X.

In order to have tools, equipment, workspace, information, manpower and energy for production activities, certain pre-production activities are needed. Generally, unlike desires or needs, these inputs do not come into existence spontaneously. Preparation activities include those of

i) obtaining information,
ii) obtaining or preparing the delivery of resources,
iii) obtaining tools and equipment, and
iv) hiring workers.

We distinguish preparation activities from focal production activities. If these are not already on location, someone has to obtain them from elsewhere in the environment. Agents hired to help in production activities also have to be primed with information. Priming means getting them to absorb the information for the tasks they are to perform. Much of this information will be site and project related.

Pre-production activities are commonly referred to as off-site or away-from-the-work-face support for work-face activities in construction. But since these activities are not restricted to construction but also occur in any other productive endeavor, we prefer to call them pre-production activities.

It is automation management to procure the requisite tools, workspace, information and energy. This is a suitable approach if all these requirements can be determined and met beforehand. This is the case with much of manufacturing automation, where we have seen successes of simple automation that builds on mass production approaches. The construction environment poses several challenges to this approach to automation which will be discussed in Chapter 3. In construction there is much less opportunity for mass production, but instead there are many additional problems related to environmental interactions. Automation systems can be extended to perform preparation activities.

## 2.6. Interdependencies among Environmental Entities

Dependencies fall in a spectrum. Very strong dependencies among components exist if no separation of these components may occur. Dependencies are strong if functionality is completely destroyed should components be separated. In weaker dependencies; only some functionality is lost. In other cases, some functionality remains but at a lower level; effectiveness may be damaged when components are separated.

We have seen that agents and organizations are components of the knowledge environment. These entities are capable of autonomous reasoning, decision-making and actions. The autonomy of these entities means that generally they continue to function even if separated from other entities. However, they are not strongly independent in the present environment. There are many dependencies of the producer-consumer and client-service types that influence the effectiveness and productivity of these agents. Their effectiveness and productivity can be damaged if they are isolated. Some entities consume what other entities produce, but they cannot produce what they themselves consume. If the producing entities are removed from the environment, the consumers may be unable to function productively. The same holds for service.

Many such dependencies among environmental entities occur in construction. Materials, equipment and manpower are custom assembled for each construction job. In a building job we may have marble imported by specialist suppliers. Even steel components, cement and bricks may have to be brought a long distance depending on the location of the job. The same goes for equipment. The workers used by the majority of construction companies do not come from within those companies. They may be hired from union shops or non-union labor pools.

A construction company separated from its supporting environment is not likely to be effective or productive. Much of the productive power lies in exploiting the knowledge environment.

It is not likely that many of these conditions will change as a result of automation. Materials will still have to be obtained from suppliers. Even automated machines may have to be leased from equipment centers. Workers would still continue to move about. Robotic machines will be compelled to move from site to site. The seasonality of construction work and the differences from project to project are among the major factors that cause these conditions in construction on earth. The conditions of space construction may be different.

We see that construction involves a large number of these dependencies. Managing a construction automation system might mean that a large amount of management effort would go into managing these agent interdependencies. However, there is the possibility of using computer and automation technology to assist in such management.

# 3. Challenges and Approaches for Construction Automation Systems

## 3.1. Introduction

Not all automation systems would really benefit from the kind of capabilities proposed in this research. In many manufacturing situations, preprogrammed automation would be both adequate and cost effective. The construction environment, however, has several characteristics that suggest to us that the incorporation of environmental intelligence would be useful. The dynamism of the construction environment makes it difficult to design and manage a construction automation system. The automation system needs to evolve rapidly with the evolution of the project.

i) Can we define the particular construction problem and prepare for it in detail up-front?

ii) What type of effort would be needed and how much effort does it take?

iii) How much would it cost?

We would like to have some idea of (i) because it would help tell us if we can use the well established manufacturing automation approach for construction. We would like an idea of (ii) because we have to react fairly quickly to changes in the construction environment. The effort needed to manage change may be too great for certain or all approaches to construction automation. Question (ii) is also related to (iii). Ultimately, we need to have an idea whether costs are too high and whether they can or cannot be reduced.

In relation to these questions, we discuss the reliance managers place on worker autonomy, problems in assembling and managing evolving automation systems, and the costs involved.

## 3.2. Difficulty in Problem Definition

If we do not know in advance what the problem is, then we cannot predefine the solution to the problem. Preprogrammed automation might be used if a full definition of the problem can be given. In preprogrammed automation we define the actions of the machine, and program the machine for the execution of these actions in advance. An example would be a robot preprogrammed to repeat a series of motions. For preprogrammed automation to work, we need a

i) well predefined problem,

ii) complete and computationally tractable decision procedure for the known problem, and

iii) correct execution of decisions.

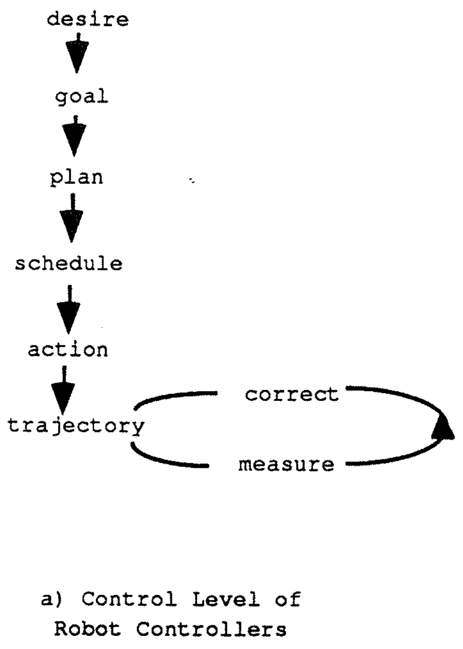
If any one of those falls short, then there is a possibility of maloperation — the robot may continue to operate but in a manner inappropriate for the circumstances. The environment of the robot is well defined in much of manufacturing automation. If the environment is also stable, then sensors may not even be needed. Otherwise, adaptive techniques would be needed. For adaptation we need to have information inputs and to be able to act on the new information. Robotic control technology offers tightly bound adaptive techniques — corrective techniques bound to actions. They are particularly useful to correct deviations from preprogrammed actions.

Is it possible and desirable to predefine the construction problem? Several characteristics of machines and the construction environment are obstacles to our ability to fully predefine the problem.

To bring the problem into perspective, let us examine the question of maintaining a model of the construction site. Such a model might, for instance, be used for planning the path of sensorless robots. Techniques for path-planning are being developed elsewhere which would enable a robot to plan the path if the layout of obstructions is given. Since the path found is executable, the robot should not need visual sensing. Nevertheless, we believe that some environmental sensing is needed for several reasons, namely

i) execution fidelity,
ii) dynamic environment,
iii) unknown environment, and
iv) involvement of black-box agents.

First, the robot might not execute the plan correctly. Errors might creep in during movement. As mentioned, robot control technology provides means to correct deviations in actions from planned actions. Control and action are bound at low levels as indicated in Figure 4(a). Unfortunately, even if we provide a sensory feedback mechanism that corrects for this deviation, it is insufficient because what we might actually need to correct are the robot's tasks. The reason is that we could have discovered that the deviations indicate that the robot has lost its precision in execution and consequently these tasks can no longer be completed. We need control activity all through the different levels of decision-making and actions, as shown in Figure 4(b).



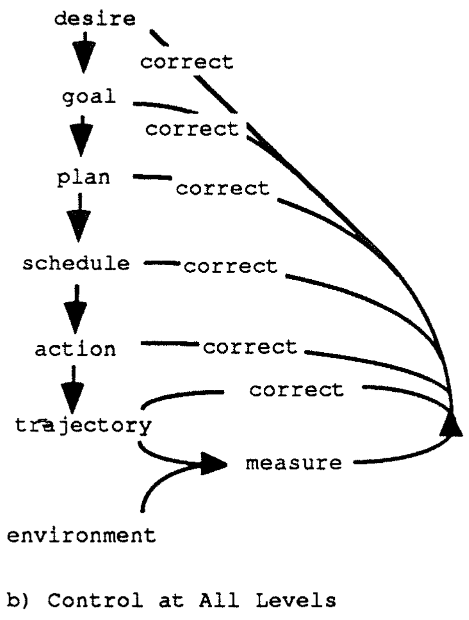


Figure 4. Control of Intelligent Robots

Second, the environment is dynamic. The word construction itself connotes that changes can be made in the physical environment. Three important causes of change in the configuration of the environment are

i) actions of agents on environmental objects,
ii) motions of the agents themselves, and
iii) natural causes of change.

Whenever agents modify the environment, information about these changes has to get to the manager of the model of the environment before the model can be modified. If there are problems with the gathering and transfer of this information, then the model might not represent the environment accurately and plans generated based on the out-dated model might not work.

Similarly, mobile agents might need to inform the manager of changes in their locations when they move about. Construction equipment might be constructed to automatically provide such information, but other means have to be found to detect the location of all humans on site.

The physical environment is not necessarily unchanging even if agents do not perform any actions on it. The physical environment is capable of changing by itself and does not inform the automation system of the changes. For example, a window transported to site breaks, a stack of bricks topple over onto the robot's path or power fails. The automation system or robot has to find out about the change. It is not much good if the robot only knows how to execute the series of actions needed to place the window but is unable to perform an appropriate action if the window has broken. A sensorless robot will not even discover the changes!

Highly dynamic and changing environments require that the temporal separation between final planning and action be small. Actions are based on the limited observable features of the environment at particular points in time and may be taken without knowing the actual conditions of other parts of the environment.

Third, the actual environment may be unknown until some actions uncover it — that is, knowing about the environment comes after some actions have been taken. A typical example is that of the below-ground environment. What lies below the ground is not readily observable until uncovered. Buried pipes, cables, boulders and even pieces of ancient civilizations may lie hidden. Excavators have to deal with such an environment. It is unlikely that one can transplant factory-type robotics for this purpose even though excavators have much in common with manipulator arms. Additional intelligence is needed to deal adaptively with problematic situations.

Fourth, there is the involvement of human agents in construction — the most obvious black-box agents. Unfortunately, although the actions of machines might be all planned in advance and expected to be executed in accordance with those plans, the precise actions of humans will only be known when they actually take place. The reasons are that the plans of humans are highly dynamic and hidden; many of the intentions and plans of human agents are not directly accessible in the way one might make machine generated plans accessible. They are black-box and free-will agents besides. We need very adaptive planning at the detailed levels for machines that work in the same environment as humans. Of course, there is also the option of excluding human workers from the active sites of automated machines. Such spatial control strategies may be sufficiently effective.

## 3.3. Cost of Automation Systems

Monetary costs of automation systems are relative measures that are highly dependent on the economic conditions, extensiveness of use of the technology, availability and costs of specialized workforce, and the actual human effort needed to create and use such systems. We focus on the actual human effort and difficulty needed to create and operate such systems. They are likely to be indicative of problems fundamental to the use of such automation systems. Strategies and methods should be found to reduce the amount of human effort needed to create and operate such systems.

### 3.3.1. Installation Effort

Installation effort includes the effort needed to predefine the problem in construction and assemble the system. The automation system needs to be provided with information. Predefination includes the need to determine possible contingencies and provide solutions.

#### *3.3.1.1. Effort to Predefine Problem*

Even if it were possible to predefine the problem in construction, the considerable effort required to think of all contingencies and potential problems, work out solutions for them and input all the potentially useful information and knowledge in anticipation of possible future use is considerable and thus militates against doing so.

Does the automation system need to be provided with information of its terrain by some human, or can the robot gather data and build a model of the terrain sufficient for its needs? If the former is the case, we are concerned with the level at which humans need to provide input. Is there the need to gather and input the data to define the terrain of the site at a very detailed level? Is there a situation in which the robot needs such detailed information and cannot obtain the information for itself? If that is the case, then the amount of human effort that has to be expended can be considerable. It is even worse if the environment can change; then, not only do we have to expend such effort once, but we have to continually expend much effort to keep the information up-to-date. There is also the need to gather knowledge and input it into the system.

For instance, to prime the mailing facility with names and addresses so that it can mail letters to people as you wish without needing to ask you for the address is probably impractical. We strongly suggest that such a facility be given some capabilities to take advantage of the knowledge of agents around it. Even a simple one, such as asking someone when it does not have the information, helps a great deal.

#### *3.3.1.2. Assembly Costs*

There is the cost needed to assemble the pieces of the automation system. The stronger the dependencies of the parts, the greater the need and effort to make them just right and the greater the costs if there is failure to attend to details.

The more diverse the hardware and software, the higher the costs to assemble an automation system. Firstly, there is a cognitive load with understanding different systems. Secondly, there is a need to make these disparate hardware and software components work together.

## 3.3.2. Operational Effort

Operating a construction automation system can be costly. Among the sources of costs are

- i) satisfying power and other physical needs,
- ii) satisfying information needs,
- iii) monitoring events and conditions,
- iv) adapting to different needs,
- v) dealing with contingencies,
- vi) regular maintenance,
- vii) correcting errors in design or implementation of software and hardware, and
- viii) trouble shooting and repair of breakdowns.

In the next subsection we discuss one aspect of (ii) and in the following section we discuss some aspects of (vii) and (viii).

#### *3.3.2.1. Effort to Keep Information Up-to-date*

The effort needed to set up sensors, program their operation, process the data received, maintain them in operating condition — all for keeping track of changes in the physical configuration of the site and the locations of all agents — is quite great. Addresses of agents change and that can be a problem if you have a large database of such addresses. If we wish to ensure that the database only provides correct information, we have to spend a lot of effort to gather information and even then it may not be possible. We propose that incorporating even a simple mechanism, such as confirming some important information with other agents in the environment, will relieve the system a great deal. There may be agents around you that have more up-to-date information. What you need is to be able to take advantage of them. We might also wish to incorporate more intelligence in the mailing service so that, when an agent changes address, it informs the mailing service instead of all the agents who know about its address (it might not even know about them all because of information diffusion).

Not only is such effort considerable but it might have a low payoff ratio. It is quite likely that only a small fraction of all potentially needed information would actually be used. But if we do not want to provide all the potentially needed information, then we should be willing to suffer possibly serious incompetencies in the automation systems or provide the system with means of correcting their deficiencies on demand.

#### *3.3.2.2. Error Correction, Trouble Shooting and Repair*

Currently, it is unrealistic to expect automation systems to be built perfectly. We must expect problems due to errors in software and hardware design. All the issues related to maintaining construction equipment in operating condition also apply to construction robots. In addition, intelligent autonomous robots will use a great deal of software. The concepts and tools to develop very large software systems are relevant. Clearly many techniques — some of which are current state-of-the-art — are required to effectively manage and operate large construction automation systems.

In addition, we might also expect problems to arise from incompatibilities between software and hardware. One of the possible sources of such incompatibilities and problems is that of dynamic incompatibilities between software and hardware; this is discussed in the next section.

Trouble shooting automation systems will be easier if most errors can be localized and contained. For example, one would like to be able to distinguish hardware faults from software faults. Furthermore, we would like to be able to trace and diagnose a system of several interacting machines. For example, a robot may generate a plan, portions of which are executed by other robots. If there are errors in the plan, problems might only surface with some other agent and there could be a cascade of failures. We would like to have some way to attribute credit or blame for the decisions and actions of autonomous intelligent decision-making robots; that is, we need a system architecture that supports accountability.

Given the time-pressured environment of construction, as in the case in manufacturing, it can be very costly if repair generally consumes an inordinate amount of time. With regard to hardware, we have to learn from practices in automated manufacturing and in the design of electronic equipment.

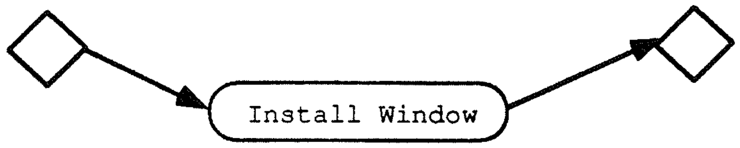
## 3.4. Management of Machines

### 3.4.1. Value of Worker Autonomy

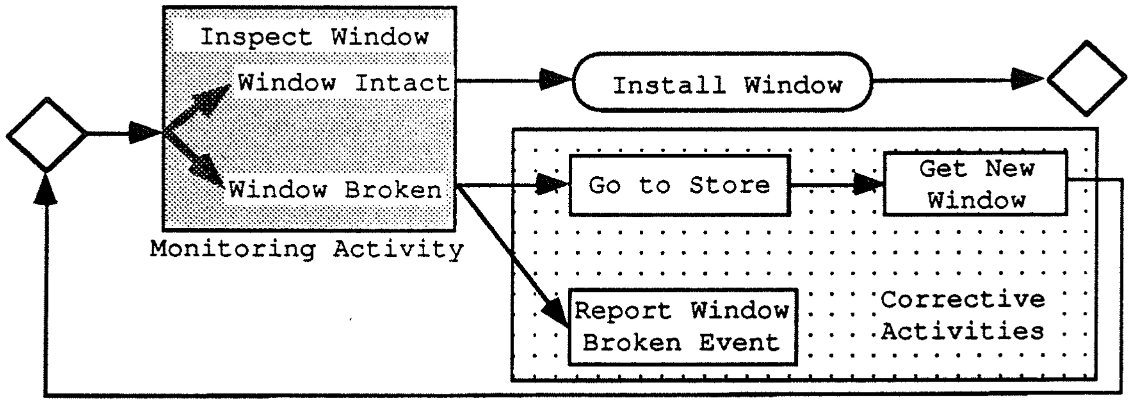
We depend on the autonomy and intelligence of the human construction worker to deal with situations on the front line — things which the construction manager does not have the time or resources to prepare for and provide instructions. With automation a machine system does some of the work of the human worker. Will the construction manager now have to devote the time and resources to prepare for these problems and provide detailed instructions because the automated machine system lacks sufficient autonomy or intelligence? If that is the case, will the savings in human labor be more than offset by the cost in human management efforts? Or is there some way to provide some of the valuable autonomy and intelligence of human workers to construction automation systems?

We will highlight some of these issues with the help of an example. The example concerns window installation, which is typical in building projects. But the discussion is not restricted in application to window installation. An agent has to face similar situations whenever anything has to be installed and whatever we discuss here has implications in those situations as well. It is basically a discussion of contingencies that might arise in construction. In this particular case the contingency is not an unexpected contingency. Yet we will show that current planning methodologies inadequately deal with such issues.

A construction manager rarely prepares a conditional construction plan which deals with a broken window. But supposing that he did, he might prepare a conditional plan in lieu of the traditional single-activity plan. We place the two plan fragments next to each other in Figure 5. If he were to do this for all activities, he would have large conditional construction plans, so contingency planning is only used for important circumstances. The construction manager does not have to develop or use these plans because the manager can and does depend on the worker to deal with less major problems, and the worker has the autonomy and intelligence to do so.



a) Fragment of Plan Developed by Construction Manager



b) Imaginary Fragment of Equivalent Contingency Plan by Construction Manager

Figure 5. Planning Window Installation

The activity "Install Window" in plan fragment (a) does not refer only to the sequences of actions directly related to the installation of the window. When we ask a worker to install a window, the request is not translated as merely be to perform the actions directly related to the installation of the window. We will be quick to fire the worker who does not use his head and installs a broken window, nor will we be pleased with a worker who approaches the construction manager to obtain a window. Surely we still cannot be happy with a worker who can deal with a broken window in the manner above but cannot extrapolate from this knowledge to deal with a broken lamp. In the general case, we would like to be able to provide machines with high-level guidance plans, a fragment of which was shown in Figure 5 (a), and rely on them to handle the contingency aspects. An interesting aspect of the contingency plan is the branch to report the broken window event. This is an activity indicating the organizational role and responsibilities of the agent in the environment of other agents.

Current CPM and PERT techniques do not tackle conditional plans of the sort indicated in (b). These plans may have cycles in them and therefore can take an arbitrary amount of time to execute. But GERT techniques and derivatives thereof [Moder et al 83] have been developed to represent and analyze conditional plans including those with cycles. The concern has been project analysis rather than for use as executable plans. The need for probability information for the conditional branches are basically pertinent to analysis and less so for execution since at the time of execution there is no more probability in that sense. Analysis of GERT networks is done with the help of simulation.

We have need of a planning technology that gives machine systems capabilities of this sort and more. A lot of the activities in the shaded boxes in Figure 5(b) are related to environmental intelligence. We shall see how we might create machines capable of dealing with this situation, not only for broken windows but also for broken lamps.

Environment exploitation knowledge is one aspect of having the ability we desire, but the architecture of the agent that provides a basis for rational operation is equally important. A plan of the sort we see in Figure 5(b) coupled with an execution algorithm that executes it are not quite adequate because we have the problem of possibly infinite cycles. A system that continually performs actions that were previously unsuccessful is non-rational. Section 7.5.2 discusses a more controlled and rational approach for the operation of an agent. In that approach there is the dampening effect of resource control. But we also need strategies for detecting problems and recovering from them.

### 3.4.2. Assembling Automation Systems

Construction automation systems have to be put together quickly and be able to adapt quickly to keep pace with the rapidly evolving construction environment. The luxury of gathering operational data over long periods of time for debugging and fine tuning that manufacturing automation affords does not exist with most construction automation. The benefits of gathering operational data are more likely to come in the next project than in the current one.

Our targets should be to

i) relate cost and human management effort with use that is approximately linear or better;

ii) minimize contamination points and isolate contamination; and

iii) make the integration of the pieces of automation nearly automatic.

The first item is the problem of scaling an automation system. An automation system does not scale well if the costs or human management effort rise rapidly to unacceptable levels with an increase in size of the system.

The second item is partly the problem of containment. A problem with one part of the automation system should not spread to large portions of the system. We do not expect that a system of this kind will ever be free of bugs, incompatibilities and other problems. Damage control is necessary but should not be consuming a disproportionate amount of resources or effort.

The third item is related to the problem of time pressure and scaling as well as to the problem of errors. In the dynamic construction environment, pieces of the automation system have to be quickly commissioned and decommissioned. If the lead time required to commission an automation system is great, then the practical utility of the automation system would be reduced.

Of course, there are cases where we would go ahead with an automation system almost regardless of the cost or lead time because the system buys otherwise unattainable safety from dangerous conditions — such as work related to hazardous materials or in hazardous conditions.

To be able to put together almost bug-free, harmoniously integrated and continually changing automation systems is a great challenge. At the present time we do not have the know-how. Nevertheless, we can learn from the techniques developed to deal with large projects and organizations and the cooperation extant in human societies comprising large numbers of agents and using large amounts of resources.

Some observations pertaining to the previous topics, which are certainly not exhaustive, may help:

i) distributed intelligence
ii) distributed control
iii) local decision-making
iv) aggregations of related knowledge
v) local truth
vi) modularity
vii) controlled interaction
viii) operational principles

Distributed intelligence means that the knowledge to deal with situations is not centralized with one or two agents. Such centralization creates not only potential bottlenecks but possibly requires large aggregations of knowledge and powerful processing engines.

Distributed control means control information does not arise from just one or two agents. Agents have great measures of autonomy to act. This is, of course, tied to distributed intelligence.

Local decision-making means that, as far as possible, decisions are to be made locally, and possibly only be known locally. Local decisions for local matters. Autonomy in decision making relies somewhat on the ability of agents to exercise distributed control, if the autonomy is to have more practical relevance; that is, we do not want distributed decision-making and fully centralized control.

Related knowledge should be aggregated in agents and knowledge objects. If there are ten knowledgebases, for example, it may be better if related knowledge resides in one knowledgebase than for the related knowledge to be scattered among all ten. The stronger the ties between the elements of knowledge, the greater the need to put them together. If there is knowledge that relates to two specializations, then it is even desirable for the knowledge to be duplicated in two knowledgebases. We want to be able to characterize these knowledgebases so that an agent needing knowledge from a particular specialization can find it more easily.

An agent, rather than expect or strive for global truth, should manage with just local truth. Knowledge in an environment can be inconsistent, and sometimes it may not be possible to determine what the actual truth is. It is enough for a small portion of knowledge to be consistent.

Modularity allows parts of the automation system to be constructed separately and combined later without substantial efforts. The ability to construct portions of the system separately reduces the complexity that has to be mastered and handled at a particular point in time.

If communication bandwidth is low or expensive, interaction should be kept low to keep these channels open for urgent matters. Interaction should be intelligently done rather than by fiat or brute force.

We can summarize these items as follows: make independent agents (i,ii,iii,iv,v,vi), give them a common language (vii) and control them by principles (viii) rather than by decree. Environment exploitation knowledge is associated with the agent and acts as a cohesive force. Agents are not interdependent for their fundamental operation (or the life imperative), only for their specialized operation. Thus agents do not become actionless because they fail in the latter. Weaker interdependencies are taken care of by environmental mechanisms. Stronger interdependencies are taken care of by organizational techniques.

If construction automation is to become widespread and cost effective, we believe that strong modularity is needed. We do not yet know if the other techniques are equally applicable to construction automation.

Modularity of automation systems allows users to purchase and use only those pieces of automation they need, put them together easily and dismantle pieces from an active automation system without dire consequences. We would like to eliminate any point where errors can enter the system. We believe this means minimal user programming. User programming under the time pressures typical of construction will be a guarantee of contamination. To achieve modularity, not only do we need to design the robots well, but also structure interaction within the community of agents well.

### 3.4.3. Issues of Human Management of Machines

In this section we look at some of the problems that could hamper the ability of humans to manage machine systems. In particular, we keep in mind the possibility of teleoperated machines and the use of other remote control techniques which are among the lower levels of human management of machines systems. Such techniques are widely used because there is less need to provide the machines with intelligence, but they do constrain what could be done.

Two aspects in human management of machines are

i) human effort, and
ii) human controllability.

Human effort relates to the amount of human support required for an automation system. Controllability relates to the extent to which a machine or an automation system can be controlled by humans.

Two types of expertise and manpower are needed to manage a construction automation system:

i) technical, and
ii) management.

Technical and automation experts are expected to be few and costly. The level of technical and managerial expertise required for construction automation should be as low as possible. We would also like the manpower requirements to be low. If one person is required to manage one machine, we would need fairly high productivity, quality, security or time gains or all of these to have practical automation.

We would also like to know how construction automation systems might scale. As the number of pieces of automated equipment increase, would the human effort to manage them increase linearly or not? This strongly depends on what humans are required to manage.

If the management is of each machine agent with little need to manage the interaction among agents, then the effort may only increase linearly. If human management is also required of interaction among agents, then a quadratic increase of effort could be required since the number of interacting pairs for $n$ agents is $n(n-1)/2$. The situation is even worse if there is need for human management for each agent's interaction with all objects on the construction site because the number of objects is bound to be much larger than the number of agents. On the other hand, if human management is only needed for key agents rather than of all agents, then a less than linear growth of human effort might be possible.

There is a question of whether humans can manage automated machines systems just as well as machines can. Human beings have certain limitations such as response time and effector and sensory resolution limits. A slower control system cannot directly control a faster changing phenomena; that is, a system which changes in a major way faster than the control signals can be applied by control system X cannot be controlled by system X. As a flexible controller, there is a limit to the control potential of the human system. Clearly a control loop that requires a control time smaller than that of the human response time is beyond human control. It has to be mediated by faster controllers.

A welding machine that needs to maintain a certain gap during the welding process is an example of this problem and automated control is necessary. A fast controller maintains the gap while the human controls at the higher level through stating the required gap. But beyond situations where we know the impossibility of direct human control, using human control in other situations is a matter of judging effectiveness and cost with respect to the available solutions. The effectiveness of human control declines with time because humans tire especially when performing repetitive control activities.

Information can be obtained using sensors that are of a different modality than those of the human sensory apparatus. In many cases this information can be transformed, such as preprocessing and making a visual display of the information. Graphic displays are extremely important because the processing of visual information is one of the strongest points of human information processing. But the number of such channels are limited since each human being can only attend to one visual item at a time.

The other problem is that of viewpoint. How effectively might a human being with two eyes, two legs and two arms control a machine with eight eyes, three wheels and four arms? We can make some concession with information processing and graphical displays. We can alternate control between the pairs of arms or have one person control one pair and another person control the other pair. That does not solve the problem that the viewpoint of such a machine is a gross mismatch with the human viewpoint. Even if we develop a machine with the same structure and proportions of a human being, the different dynamics would be an obstacle for the effective use of that machine through direct human control. One begins to see the limits of teleoperated machinery. I think what would be needed is a higher level of control by humans of machines that understand their own kinesthetics.

At the higher level, which is more removed from the direct operation of the machines, there are various demands to be fulfilled. It is necessary for management to provide the resources and information required for the productive processes. This can be done though careful and comprehensive planning. In one approach humans prepare for and directly provide the tools, equipment and information required by the machines and thus must respect the dynamics of the machine system. The faster the machines can consume resources and information and require new resources and information, the faster the humans who serve them have to provide them with resources and information, otherwise the potential of the automation is not realized.

On the other hand, this thesis presents the possibility of having humans provide sources of the required resources and information, or even just information about them, and rely on the machine to obtain the resources and information they require from these sources on demand through the use of their environmental intelligence. These machines are provided with high-level goals or tasks and go on from there. There is some similarity between this concept for controlling machines and that of exercising control by stating the size of a gap versus exercising direct control of the size of the gap for a welding machine. But we have first to provide the machine with the capabilities to fill in the lower levels.

### 3.4.4. Issues of Interaction and Integration of Machines

Interaction will occur among the agents of a construction site. When agents are involved in close cooperative work, they are essentially yoked together for the duration of that work. There are the physical aspects of interaction and the information aspects. We shall first look at the physical aspect.

Physical interaction occurs when agents meet or when they perform a joint task such as moving an object together. Whereas the physical strength, build and dynamic characteristics of human workers are about the same, there will be a disparity between humans and machines and even among the machines themselves. Difficulties might arise because of these disparities.

For example, there is a question of whether a human worker can work with an automated crane, with the human worker doing the fine positioning and other work on the ground. There is a disparity in the energies of the two agents. Force control techniques might be applied to give machines the "gentle touch" to bridge the disparity between the energies of the two systems. In the previous section we have seen that the dynamics of both systems may be insufficiently compatible. A crane that cannot respond as rapidly as a human might endanger the human worker. On the other hand, a crane that can respond much faster than a human might be underutilized if paired with a human rather than with another machine. The slower dynamical system is controlling the pace of the work as depicted in Figure 6. Otherwise, the faster dynamics introduced by the faster system may exceed the control capabilities of the slower system. Fast teleoperated robots will be limited by human dynamics. For example, the manipulator arms designed for spray painting can have faster dynamics than humans. Figure 7 depicts the adaptation of a faster dynamical system to a slower one. One system can be shielded somewhat from the very fast dynamics of another system if provision has been made to provide interaction access at different levels.

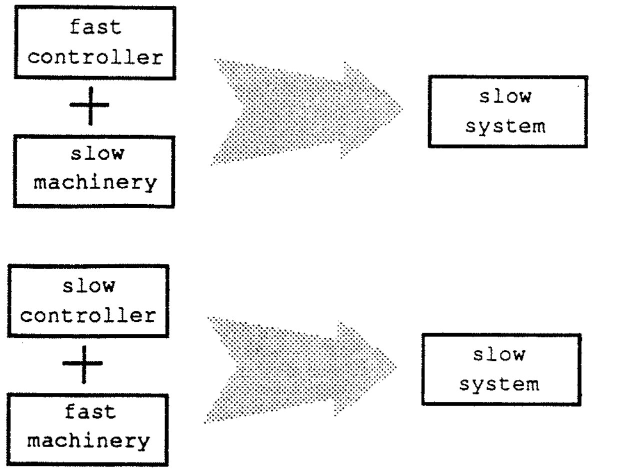


Figure 6. System with Slower Dynamics Controls Combined System Speed

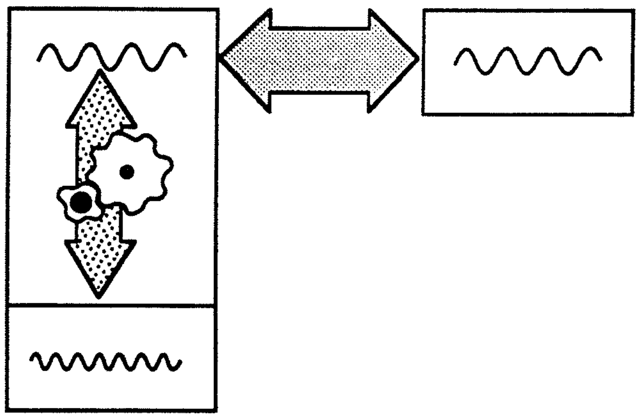


Figure 7. Meshing Systems with Different Dynamic Characteristics

Not only do these issues pertain to human-machine interaction, but also to machine-machine interaction. The dynamics of a group of machines might be determined by the dynamics of the slowest machine under conditions of interaction. For instance, if four machines jointly move an object then the slowest dynamical system controls. We may have to accept the cost of non-optimal performance, but even more importantly we have to be able to design machines that can operate over a physical dynamical range. A machine might work more slowly if it is interacting with a human worker, but faster with another machine. To obtain this, we may also have to provide the machine with an understanding of the dynamic characteristics of various agents.

Aside from the question of dynamics, there is also the problem of the interaction between these two agents at the informational level. How can these interactions be structured? How can the human agent provide positioning and other instructions to the machine? And equally important, can the machine indicate its intentions to the human worker? Again we see the disparity between the human and machine, but this time at the cognitive level. The same problem is likely to arise in interactions between machines.

Humans operate at relatively high cognitive level in the human operational sphere. We anticipate that machines will face many difficulties integrating with human systems. Some of this difficulties will arise as a result of communication incompatibilities. Using the usual QWERTY keyboard as a communication input device is restrictive to human construction workers needing to move about on the site. This does not mean that interactions between these spheres are impossible, but techniques may have to be developed to bridge certain disparities. Adults have some problems relating to children, but valuable interactions nevertheless occur. Humans can adapt by lowering expectations and through modelling the reasoning of other agents. Cognitive abilities of machines can be raised to a level that is closer to the human cognitive level. Similar techniques may be used to integrate machines with different cognitive abilities.

The integration of agents, however, will involve more than bridging the disparities between energies, dynamics and information. There is more discussion to be found on this issue in Chapters 8 and 10.

### 3.4.5. Incompatibilities

Software and hardware represents two very different activity models. Software dynamics are governed by computational characteristics and computational devices whereas hardware dynamics are governed by physical characteristics. The control problem we mentioned above, regarding the use of humans to directly control robot hardware, also applies to the use of software to some extent. Maloperation will result if the software dynamics cannot cope with the hardware dynamics.

Fortunately, we have found ways to deal with the disparity of energy levels between the systems that run the software and the effectors of machinery. The difference between the dynamics of the two systems continue to exist.

With regard to cognitive incompatibilities, we may wish to reduce some incompatibilities in knowledge representation, language and communication protocols through standardization. We live with other incompatibilities. It is debatable whether we want to eliminate all such incompatibilities.

## 3.5. Suggested Approach for Construction Automation System Design

In the last few sections we discussed some of the challenges that face construction automation systems. How would one design and operate automation systems in ways that help meet these challenges? The following subsections discuss four conceptual notions:

i) environmental participation
ii) general core intelligence
iii) modularization
iv) operational principles

We introduce the notion of environmental participation, which has two aspects. The first and more urgent matter at the moment is the incorporation of abilities to exploit the environment through knowledge about the environment and associated capabilities. Such special knowledge is termed environmental intelligence. The other aspect is that of creating agents that are useful citizens of the environment. An agent can obtain active help from the environment only if there are other entities that will possibly help. The human knowledge environment has evolved into one in which agents often help each other. This issue will become more and more important with the greater degree of automation development.

### 3.5.1. General Core and Modularization

We also posit that there is knowledge that applies across many situations and thus might be usefully incorporated into automation systems performing many different types of tasks. Thus the intelligence of an agent might be divided into a general core and specialized knowledge. Part of this general core intelligence includes environmental intelligence and self-knowledge. More discussion of the general core appears in Chapter 7.

We cannot escape from the need to provide specialized knowledge, and the development of such bodies of specialized knowledge to meet particular situations is a challenging endeavor. What should be avoided, if possible, is the effort to rediscover and structure into useable forms the general core of knowledge for every occasion. The creation of this general core of knowledge may be done once and used in many machines, rather than having to reimplement this core for each machine or even for each automation system. This will help us reduce the costs of developing, bringing up and operating automation systems.

Modularization is a technique that applies to both hardware and software. For electronic components we have chips, boards and functional components. These are meant to be easily replaceable in view of our concerns for trouble-shooting and repair. For mechanical parts, we have standardized joints and tool holders which make it possible for a machine to be configured in different ways to meet different needs.

Others have also suggested the separation of the mobile platform from the rest of the machine. This comes from the concept of component interchangeability among different machines. A particular mobile platform can be used as the mobile base of different types of process robots. Visual and power systems might be similarly modularized.

Object-oriented programming is a modularization technique used in software development. It allows substantial independence in the development of pieces of software. Software pieces are also much more reusable. For rapid replaceability, we may like to have multiple independent implementations of each object available. This will hopefully allow us to remove a faulty implementation and use another in its place. With proper management and laws, markets for software components may develop as opposed to merely for applications, as is the case today. We would like to exploit modularization as extensively as possible.

### 3.5.2. Operational Principles

We also believe that it will be useful to have certain design and operational principles. Without them, we are likely to have uncontrolled diversity of automation systems. Such uncontrolled diversity will be inimical to the development and success of automation. Certain operational principles can be used to help control the use of resources and time on the global scale. What these controlling principles of design should be is open to argument and debate, but we suggest four principles of

i) identity,
ii) commonality,
iii) cooperation, and
iv) laziness.

The principle of identity is that it should be possible for one agent to determine to some extent the identity of any agent it is interacting with. Identification of an agent might be performed to different degrees, but we believe that there is a need for agent identification in an environment with many agents and changing patterns of interaction.

The principle of commonality is that agents in an environment must have some common bases. Agents may do things differently, but it they are to interact there should be some shared ground to base it on. What these bases are will be discussed in Chapter 8.

Cooperation and not competition is fundamental to the ability for a group of agents to achieve more than the sum of what each can achieve individually. Agents may be able to interact, but we have to address the question of what would make them interact for mutual benefit. We should find principles that will help promote cooperation among agents.

Readers might be unhappy with the thought of creating lazy machines. Lazy humans are problem enough. Nevertheless, laziness is a principle for the conservation of computational and memory resources and we recommend its incorporation in machine systems. For instance, making information updates by obtaining more recent information from the environment might be postponed to times when the environment provides lower cost opportunities. Among other things, a lazy agent

i) gets information only when it needs it, and
ii) does not update information with dispatch even though the information might be incorrect, unless the accuracy of the information is very important.

Laziness is an important principle in discussions in this thesis and will be brought up several times.

These ideas are discussed in greater detail in Chapter 8. Given a fairly exploitative extant environment, it may be useful to provide protection for agents, so we seek guidelines for the design and operation of automation systems.

# 4. Components, Characteristics and Nature of the Knowledge Environment

In the next few sections we will discuss the components that make up the knowledge environment and several characteristics of the knowledge environment, including the manner in which knowledge is aggregated, distributed and propagated. We will also discuss the open bounded and multi-objective nature of the knowledge environment.

## 4.1. Components and Structure of the Knowledge Environment

### 4.1.1. Components of the Knowledge Environment

The two primary components of the knowledge environment are

i) knowledge elements, and
ii) agents.

Knowledge elements are the practical basic units that may be taken to hold any knowledge. For practical purposes we may regard words, a series of words, sentences, paragraphs, pictures or a tuple of a relational database to be knowledge elements, depending on the discussion. For example, a database name is a knowledge element that may be made up of a series of words. This is a matter of granularity convenience rather than something to split hairs over, as we know that sentences are comprised of words and words are comprised of letters and letters are comprised of lines and curves or a series of bits. We consider some encoding to be the bearer of significant knowledge at some particular point.

But it is important to distinguish the matter of a knowledge element from the way the knowledge element is coded. The knowledge element can be transported from place to place with different codings while preserving the essence of the knowledge content of the original. The name "Joe" is a knowledge element that can have encodings in sound, bits in a computer, squiggles on paper or some unknown encoding in the human brain.

Knowledge elements only have meaning in the context of some knowledge environment. For instance, the name "Joe", as we have seen, contains significant knowledge, but only with respect to the human knowledge environment; it has no meaning in the context of the Martian environment — the environment has no way to interpret or understand it.

We have already seen what agents are in Chapter 1. They are the basic units for autonomous reasoning, decision-making and actions. These descriptions are meant only to provide an intuitive understanding.

### 4.1.2. Structural Entities of the Knowledge Environment

Structural entities of the knowledge environment are entities comprised of other structural entities or of the more basic components of the knowledge environment. Among the structural entities of the knowledge environment are

i) knowledge objects,
ii) structural organizations, and
iii) systemic organizations.

Knowledge objects or knowledge repositories are encoded structural compositions of knowledge elements provided with means of access. The important thing about such objects is their knowledge content and not their physical attributes. Encoding is important. Accessibility of the knowledge in knowledge objects is dependent on encoding and the means available for access.

For example, a phone book is a knowledge object. Names and phone-numbers and the relationships between them are the major knowledge elements of a phone book. Weight is unimportant. The phone book itself has a certain knowledge structure. There are sections in which names, addresses and phone numbers are associated, other sections for emergency phone numbers and yet another called the yellow pages. The former is sorted according to lexicographic order on the names. Yellow pages, on the other hand, are structured in lexicographic order on the type of product or service. Of course, there is also the linear order of the pages. All the information in a phone book can be obtained through a linear search, but that is not efficient.

A database is another knowledge object. The database name is a knowledge element and so is the information in each field of each tuple of a relation. The information that databases contain is usually carefully structured. There may be several types of structures within a database, such as tables. Query languages provide the accessibility window to the information.

There are all sorts of ways agents can systematically relate to each other. Organizations which are partially based on authority and responsibility relationships we term structural organizations. Examples of structural organizations include armies, construction companies, manufacturing firms, banks, and at a smaller scale, teams and crews.

Organizations which develop based on systematic patterns of activity much more than authority relationships we term systemic organizations. The different markets are examples of such organizations. The professional and research community are other examples of systemic organizations.

## 4.2. Distribution of Knowledge

Knowledge is encoded in physical media or in agents and these have locality, or in carrier waves which have temporally changing locality. Therefore, knowledge at a particular time is to be found in particular places.

We will first discuss the distribution of knowledge in physical media and in agents. Not only does such knowledge have locality, but typically it is aggregated as well. In order to get to the knowledge, the location of that knowledge needs to be known. But knowledge does not only stay at fixed locations. Mechanisms exist which transport knowledge from one location to other locations.

The distribution of knowledge creates problems of duplication and consistency. These problems can exist even in a single knowledge aggregate, but they are much more likely when knowledge is to be found in numerous locations. Autonomy of knowledge management further compounds the problems. The greater the autonomy, the greater the extent of the duplication and consistency problems. These problems are not likely to go away, so the approach is not to avoid duplication and ensure full consistency, but to find ways to live with them.

### 4.2.1. Localization and Aggregation

One interesting aspect of the locality of knowledge is that knowledge is not uniformly or randomly distributed in the physical environment but rather it aggregates in huge chunks. Each chunk of knowledge may be regarded as a center of knowledge.

Human mental limits, such as the limits of human knowledge acquisitive capacity and interactive capabilities, are motivations for the development of human centers of specialized knowledge; that is, humans themselves become the centers of specialized knowledge. One would like to avoid some of the considerable costs of communication when using strongly related knowledge, knowledge which is commonly applied together in some decision-making or task. A human agent becomes a knowledge center for a particular corpus of knowledge.

Aggregation of knowledge is partly caused by practical considerations. It would be difficult to find and use knowledge if the knowledge has been randomly distributed in the environment. On the other hand, it would be difficult to bring all knowledge to a single location because it is not technologically feasible. Capturing all information centrally is not necessarily desirable because events are spatially distributed, information is generated in different locations, and the *relevance* of much of this information is localized.

The cost of communicating information was and still is high and one could keep such costs low by collecting, processing, filtering and compacting knowledge at local centers. Reducing the number of locations by creating substantial knowledge centers would reduce the amount of knowledge that agents would need to have in order to be able to find the knowledge.

### 4.2.2. Diffusion of Knowledge

Knowledge has to move from location to location. It does come into existence everywhere at once. Information may come into existence in one place in the environment but is needed elsewhere and a process of knowledge diffusion through the environment brings the knowledge there. Some of the reasons for moving information from one location to another are as follows:

i) provide users with new information
ii) correct or replace out-of-date user copies of information
iii) control the activities of other environmental entities
iv) influence the decisions and activities or other environmental entities

An example of (i) is when a manufacturing company develops a new product. One of the aspects of marketing is getting the information out to the potential customers.

An example of (ii) is the correction of address and telephone numbers that have changed because the agent has moved. There is no problem with keeping around the old copies provided that it is known that they are old copies. Database technology does not address this properly because, among other reasons, the techniques do not provide for much metalevel information and thus do not have the power to describe and characterize knowledge.

The third reason (iii) cannot be achieved easily because of the autonomy of environmental entities. However, a memo in an organization can have that effect.

An example of (iv) is a machine operator informing the site office that the machine has broken down so that the site office will send a mechanic to fix the machine. Information about the breakdown of the machine is communicated in lieu of directly communicating the desire to have the machine fixed — the decision is left to the site office. This is a case of trying to make agent `A` believe `X` by giving agent `A` information `I` so that agent `A` decides on or does `Y`. It is, of course, quite possible to use this tactic without the transferred information being correct.

We are interested in the process and mechanisms of this diffusion. Knowledge can get into portable media and be transported to another place. Knowledge can get from some media into an agent, and be re-encoded in other media by an agent. Knowledge can be broadcast in carrier waves and received elsewhere. We depict some of these mechanisms for transport in Figure 8. Knowledge can transfer from

i) knowledge object to agent,
ii) agent to agent,
iii) agent to knowledge object,
iv) knowledge object or agent to carrier, and
v) carrier to knowledge object or agent.

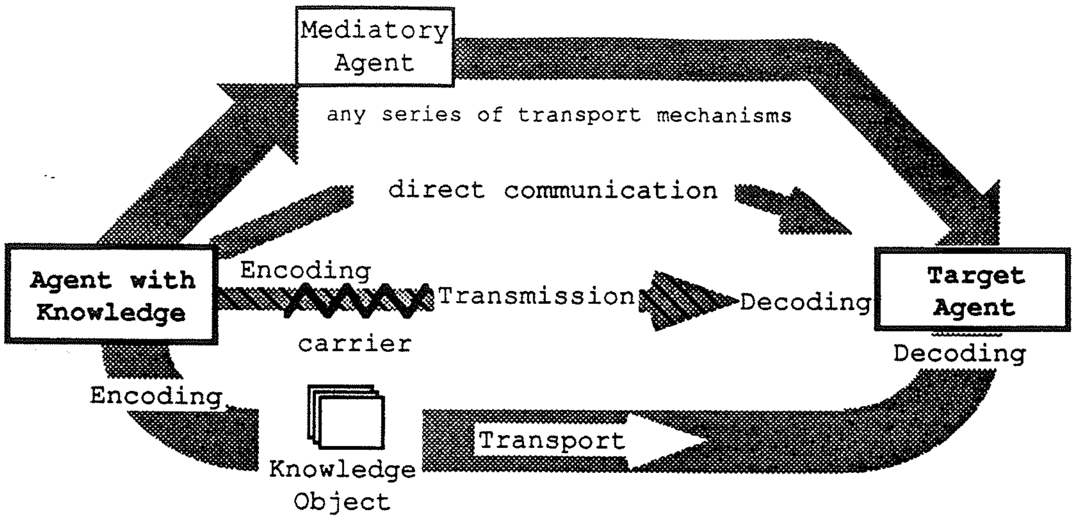


Figure 8. Knowledge Transport Mechanisms

There are two types of activities that drive the diffusion process. The activities driven by the agents who want to get the information out and the actions taken by the agents who desire to obtain the information.

To get a feel of how knowledge diffuses in the knowledge environment, we shall look at some examples. In some cases changes occur slowly and in others changes occur quickly. The demands of the latter are greater.

We will describe briefly an example involving changes of information in building codes. This change has to propagate to the engineers and other knowledge workers that need to know about it. This is an example of rather slow change since building codes do not change often. Every few years a committee is set up and revises building codes. Only a small number of agents are cognizant of the actual changes when they occur. Information about these changes is then encoded in journals and gazettes that are widely distributed to members of the structural engineering profession. Members of the profession obtain these regularly, like once a month. The building code itself is published in a new publication every few years. Once the agents needing this knowledge know about the changes in the codes, they obtain the newly published codes from the appropriate source.

We see that there are responsibilities and expectations concerning the behavior of the two groups of agents, those who formulate the new codes and those who may use them. The first group makes the information available through many recognized channels and the latter actively use those channels to keep tab on the situation and take follow-up action once they know of the changes.

Note that the first group does not swamp those channels by publishing through them the entire text of the changes, but only say that the code has changed, that the new code is available and where to obtain it, and perhaps when the new code comes into effect; they are essentially providing the minimum information and relying on the intelligence of users to follow up. Information channels have to be used circumspectly — to make effective use of limited bandwidth and not to overload users, besides, of course, to control the cost.

Such a process can take a fairly long time. Therefore, building codes are slated to become in force only some time after their development and announcement.

While providing environmental intelligence to a structural engineering design system so that it will be able to obtain the new information about changes in design codes does not buy us much, the ability of such a system to contact appropriate parties to inform them of design changes that affects them could prove valuable.

On the contrary, information about the condition of roads and highways tends to change rather rapidly in bad weather. Such information may be broadcast using a rapid channel such as the radio or television. Publishing it or using the mail process may be quite useless because the information is no longer timely. The potential user has to continually expend cognitive resources to keep track of the information — listening to the radio or watching the television. Such continual expenditure of cognitive resources was not needed in the previous example. The more rapidly such information changes, the heavier the demand on human cognitive power and the more handy it is to have automation systems with environmental intelligence that are capable of exploiting information on demand.

In the human knowledge environment, various mechanisms have evolved to transport information from source to destination.

- One mechanism is that of short-lived mass broadcasting such as the radio and television. Information does not persist very long unless a special effort is taken to capture and store it. An agent desiring to obtain the information directly has to be attentive to the broadcast during the time of broadcast. Temporally critical information benefits best with the use of such a technique.
- Another mechanism is that of medium-lived mass broadcasting such as newspapers. Usually, summary and important but not temporally critical information is distributed in this fashion. The information persists for a while but

gradually disappears from the environment except where there has been a use for it.

- Non-time-critical and massive amounts of information are usually transported through the mail system. Mail is used where information does not have a wide audience. Information is for highly selective audience, typically one on one. The development of computer-based mail has changed the picture somewhat because it is much faster; but it is dependent on mutual access to computer networks.
- Another mechanism is that of two way communication via telephone.

New and faster means of disseminating information are continually being engineered. Extensive development and deployment of local area networks and wide area networks using fiber-optics will bring more informational and computational resources in the knowledge environment within easier reach. Automation systems may be able to use these rapid network transport mechanisms in addition to other information transportation methods already available.

When information moves from location to location there are several potential problems. We do not want incorrect information to replace correct information, or old information to replace more recent information; rather, new knowledge should replace old knowledge. One way is to treat information that arrives later to be of more recent origin than information that arrived earlier. This is a useful heuristic, especially in the absence of other mechanisms. It relies implicitly on the fact that knowledge transport mechanisms of the environment normally deliver in the order in which the items were received. That is not always true, so errors can be committed if this heuristic is the only one used.

Chapter 10 mentions the use of one environmental mechanism called the clock. With the help of this environmental mechanism we can employ time tags to reduce this problem. Information that is transferred from place to place can be time tagged according to what is known about this global clock. A copy of information tagged to a later time can replace information tagged with an earlier time, but not vice-versa, even if information had been received in reverse order of its creation. This is not a fool-proof mechanism because knowledge about the status of the source is not known for certain. The clock of the source might be set back for some reason and we may have no knowledge of that event.

We are also concerned with the speed and direction of diffusion of knowledge in the knowledge environment, and with the stability of the diffusion process. How is stability achieved in the human knowledge environment? What sorts of protocols or other mechanisms are used? The speed at which new knowledge replaces old knowledge affects the functioning and strategies of agents in the knowledge environment.

Organizational structures reduce the effort needed for the diffusion of knowledge. By structuring agents into organizations, we reduce the need to diffuse knowledge. Among other reasons, this is due to the fact that agents can rely on the functioning of the organization to meet various needs; an agent playing a role in the organization needs less knowledge. Hierarchical diffusion, which is a specialized case of vertical diffusion, could occur as follows: A manager gets to know some new information. She counter checks it and thus she acts as a filter for the information. She recognizes that the information is relevant to the function of some of her subordinates. The manager might inform the chief designer (or superintendent) of the new development. The latter then informs his staff. Organizations, therefore, reduce the need for agents to monitor the external world.

Dealing with information flowing down the hierarchy is relatively simple. Since the information comes from higher up in the hierarchy, that information is likely to be both new and important. Therefore, if there is an old copy of the information, it should be replaced.

This process of diffusion up and down the hierarchy of a hierarchical organization may not be the best. A part of the organization may require large volumes of new knowledge about the state of the market. In that case, this portion of the organization could develop its own channels of knowledge to obtain information more directly. In the event of a conflict between the information from higher levels of the hierarchy and the information obtained more directly, a process of resolution is probably necessary. The reason is that, besides a need for a local decision concerning the knowledge, there is also the need to forestall the spread of the incorrect information through the organization. Indeed, that part of the organization could become a center of that particular category of knowledge for the rest of the organization, and the diffusion process could then be by request from other parts of the organization.

Other methods are also employed to facilitate the diffusion of knowledge in the knowledge environment. In organizations there may be an information center that serves to redirect queries to the relevant parts of the organization. It is a purveyor of knowledge about the organization. Some agents specialize in providing information about the organization and its members or about the agents and organizations closely related to its own. There is knowledge about who talks to whom, who has access to what and what the real responsibilities and influence of the organizational members are that may be quite apart for an organizational structure. Other members of the organization can come to such purveyors for some of their information needs. Agents outside the organization can be redirected here too.

In certain instances it is not sufficient for knowledge just to be available — the situation demands that it be *easily* available. The need for knowledge is not always equal — some knowledge is needed more urgently. Special channels tailored to the acquisitive capabilities of the agents concerned may be created to facilitate the flow of information. The hot-line is an example. In the context of developing machine systems, even more consideration should be given to the development of special channels. This is because the cost of human-machine communication is still fairly high.

Knowledge can and usually is transformed as it flows through an organization or among organizations. Special channels for knowledge may be created to facilitate the flow of knowledge to agents needing it. Means may be provided to preprocess knowledge for more efficient transmission. The processing of huge amounts of potentially useful information is costly. It is even costlier if every agent has to duplicate such processing. Several types of transformations are performed by agents to reduce the cost of transmitting knowledge in the knowledge environment. Among these are

i) sampling,
ii) filtering,
iii) characterizing,
iv) modelling,
v) theorizing,
vi) rephrasing, and
vii) compacting.

In sampling, the information is not changed in character but only some of the information is passed through. Some information might be lost. However, there are savings in the cost of communication of the remaining information. Sampling can be used not only on data streams, as the reader might imagine, but also to other knowledge elements.

Filtering is more careful than sampling. Information passes through a filtering process that has a filtering criterion. It is like a sieve that lets in certain types of information only.

Metalevel information about the information received is created with the technique of characterization. This metalevel description conveys to the reader what the information is about. For instance, an investor may be told that the stock market is unstable or the predictions of some popular quack prophet have been no better than those by chance. The characteristics of the information are provided in place of the information, and unless you question their correctness you do not have to look at the original information.

In modelling, the information is taken and fitted to particular models. The parameters of these models may then be transmitted in place of the original information. For instance, a stream of amplitude data may be modelled as a sine wave with maximum amplitude X and period T. You just need to tell that to the next agent. You can see how efficient that is.

Theorizing is more sophisticated than simple modelling. It is not quite within the ken of machine systems. Theories are much more effective than data at conveying information. For practical purposes, however, it is useful to know the limits of theories.

Rephrasing is to take one way of saying something and use a different method of conveying almost the same thing. This relies on the interpretative capabilities of the agents.

Compacting is an automatic procedure that takes the representation of the information and uses special techniques to reduce the volume of the information but not its content. Information is not lost in compacting.

Since the acquisition of knowledge from the environment is expensive, agents are selective about the channels of information they use and at demanding the level of information they desire. A top level manager will want very concise and important information only. Even the lowest level worker in a hierarchical organization is selective.

However, the channels that are used are not fixed forever. There may be reasons to switch to certain channels. For instance, important information may be about to come through a particular channel. To know that, agents rely on indirect information to trigger them if things of an unusual nature have occurred. If they occur before the occurrence of the important event, we can call them presigns or heralds, otherwise they are just called signs. When notice of the occurrence of an important event arrives, an agent's selection criteria and attention may be changed. Politicians frequently make moves or announcements that can be used as guides to things that they will do in the future. The chairman of the Federal Reserve Bank might make an announcement about an important meeting to occur the week after, and by that announcement actually influences the way people do business. He did not even have to announce a specific decision. If we do not take advantage of these heralds, we could be caught off guard. But the use of such mechanisms does not need to be thus restricted.

Sometimes the arrival of a sign may cause an agent to back up and re-examine information received previously or coded somewhere else. The agent has been alerted to the occurrence of an event that missed its attention.

### 4.2.3. Duplication

Is it desirable to have no duplication? Is it desirable for all knowledge to be consistent? On the negative side, duplication can result in the employment of out-of-date and possibly now inconsistent information, and such inconsistency can result in incorrect conclusions. However, since information resides in physical media or in agents which can be corrupted, it is necessary to have duplication to have a measure of robustness. In some database applications it is important to have consistency and duplication. Also, copies of database fragments may be kept at different sites so that read accesses can be much more efficient.

Whenever agents learn or obtain information from somewhere else or some other agent, that usually means that a new copy of that information now exists. Communication transfers information from one site to another, creating new copies along the way. If the information does not change, then there is no problem except for the cost of keeping around the duplicated information. Often, though, things change and the information which was correct previously is no longer absolutely correct.

There is not only the problem of duplicate copies of information, but the correctness of conclusions and results derived from the old information may themselves be incorrect now. The situation expands in ever widening circles. To maintain their correctness may be an expensive and unnecessary effort. We will apply the principle of laziness to this.

### 4.2.4. Conflict and Errors

Inconsistency of knowledge exists in the real world and if we wish to use knowledge in the knowledge environment extensively we have to be able to deal with it. One reason for inconsistency is the problem of knowledge distribution. The price of X from supplier Y may be stated as $4.50 by one knowledge source but $3.90 by another and the reason may be that the first source is out-of-date. One code may state that the maximum allowable load is 70% of the maximum sustainable load and another states that it is 80%. The site engineer may say that the crane rails have been painted but that is not reflected when the database is referenced. Another reason for inconsistency among the sources of information is that there may be some source that does not know the truth. One source may say P is true and another that P is false.

We do not think that agents that are to interact extensively with the knowledge environment can avoid meeting these problems. Fortunately, this does not signal a condition of impossibility, as the success of human beings testifies. We give the following reasons:

i) errors are not usually catastrophic
ii) agents have a measure of control over the influence of errors
iii) agents can take actions to ameliorate the situation when important to do so

The first reason basically says that agents can survive with a measure of errors in their decision-making and response. There is an element of payoff in this. Under certain payoff conditions, decision-making systems that commit errors occasionally can still be viable.

Second, there are methods that agents can use to control the destructiveness of errors. That way agents can even survive with inconsistent internal beliefs. This requires that such inconsistencies do not destroy the utility of the agent's knowledge. The effects of each item of knowledge are strongly localized and limited. There is no global truth, only local truth; that is, one cannot axiomatize all the knowledge that a human has and derive global conclusions from it because the human is likely to have an inconsistent body of beliefs.

Third, an agent need not be helpless when faced with problems of inconsistency in knowledge. For environmentally intelligent agents, many courses of action may be open to help deal with the situation. For example, one heuristic that an agent faced with such conflicts may apply is to check with a third source.

One would like to reduce errors because there is some benefit from doing so. We are interested in measures and techniques that can be used in the presence of inconsistency. How critical is correct information to the problem? When and how much effort should be spent counter checking one's information? External data, information and knowledge are not treated at the same level. Inconsistencies in internal knowledge may be treated in a similar fashion as inconsistencies in external knowledge.

## 4.3. Persistence and Decay

Decay is the process in which something decreases in quantity, activity or force [Webster 70]. In this section we will discuss information decay of which there are several types such as,

i) temporal decay,
ii) spatial decay, and
iii) associational decay.

Over time, pieces of information in the knowledge environment may decrease in quantity, activity, utility or force. The relevance and thus the utility of a great deal of information declines with time. We shall call this temporal decay.

The value of information frequently declines with the distance from the source of the information. The closer you are to an event, the more likely it is that information about the event is of value to you. We call this the distance decay of utility. One possible reason for this is that events have graded spatial influence. The quantity of information about an event also tends to decrease with the distance from the source of the event. The closer you are to the event, the more likely it is for you to have a more accurate account of the event. We call this the distance decay of quantity. The reason for this rests on the process of gathering, distributing and keeping information about the event.

If the distance is a spatial metric, then we call the type of decay *spatial decay*. For example, the information about a weather event is frequently most useful and relevant to the agents at or close to the site of the weather event and much less so elsewhere. A spatial metric is normally only a crude measure and there are any number of ways to calculate distance.

If the distance is that of association, then we call it associational or connectivity decay. For example, there is very little association between construction firms and garment factories, but strong associations between construction firms and architectural firms. Events relevant to the garment industry are unlikely to be of much interest to construction firms. Events relevant to architects are very frequently also relevant to construction managers, although perhaps to a lesser extent.

## 4.4. Open Systems

If we define the boundary of a system to be large enough, we could make a closed system. Nothing would flow across this boundary. However, for an environment knowable to the agent — the interesting system boundary — it appears as if the boundary is permeable (Figure 9). For a particular construction project, for instance, if we take the production system of this project as the artificial boundary, it appears as if knowledge and components flow across this boundary. Agents and knowledge enter and leave this system. Once agents leave, one may no longer care to keep track of them or maintain information about them. They fall outside the environment that one wishes to know about.

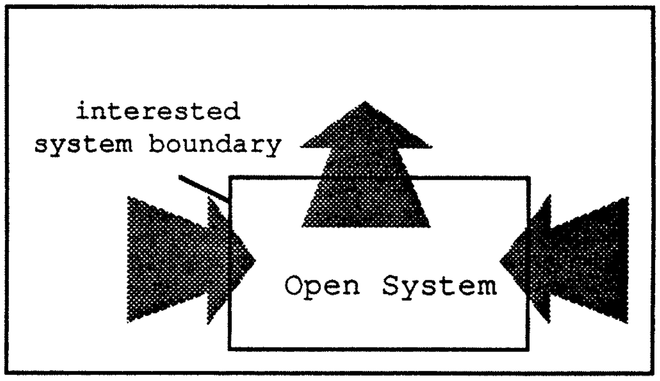


Figure 9. System of Interest is an Open System

A computer network in which a computer may be incorporated into the network and taken off of it without notifying some central control, even while the network is in full operation, is an example of the design of an open computer system.

[Hewitt 88] took the open-system perspective to office work. Office work is defined by him to exclude any manipulation of physical objects and therefore of robotics. Therefore, the focus is on information. Nevertheless, office work involves the interaction of agents which may be within or outside the organization. He is a pioneer in the idea of open systems. He characterizes open systems as having

i) concurrency,
ii) asynchrony,
iii) decentralized control,
iv) inconsistent information,
v) arms-length relationships, and
vi) continuous operation.

He noted that in open systems there are no global objects and the only thing all subsystems hold in common is the ability to communicate with each other. He also recognized many of the problems associated with such systems in saying,

> *"Coping with the conflicting, inconsistent, and partial information is one of the major challenges in office information systems."*

Therefore,

> *"Unplanned dynamic adaptation and accommodation are required in organizational information systems to meet the unplanned changing needs of coordination since the execution of any plan requires articulation, change, and adjustment."*

He also proposed some interesting ways to deal with some of these problems, in particular the problem of conflicting information and viewpoints. The reader is strongly encouraged to read about them.

Of course, the construction site is an excellent example of an open system. Not all the resources, knowledge or agents are brought in at the beginning of the project, but at intermediate times, and a large number leave before the end of the project. The constituents of the system change greatly because of these movements across the system's boundaries.

In an open system it is hard to have ready answers to questions such as:

i) How may agents are there on the construction site? or
ii) What are all the objects on the construction site?

These uncertainties are a problem for automation systems—especially preprogrammed automation — although humans seem to cope with them reasonably well. Some effort to control the boundaries or to create more controlled areas is almost always made. The problems of open systems described by Hewitt are also the problems faced by construction automation systems [Primet and Schwarz 86].

## 4.5. Multi-objective

Not all agents pursue common goals in the knowledge environment. Even if they happen to hold common goals, these goals may sometimes be in conflict. Indeed, something has to be done to mediate among disparate goals. The human knowledge environment has evolved mechanisms to mediate agents with differing goals and some of these mechanisms are discussed in Chapter 10.

The problem that agents in the construction environment do not always pursue common goals is more important than it might appear at first sight. There are reasons why agents do not start off with common goals. Construction is a major consumer of goods and services generated by a large number of industries. The goals of such suppliers and professionals are not necessarily in accord with the goals of construction agents. These agents have their own rationality for their work.

The AEC industry itself is a fragmented industry. There are architectural firms, engineering firms, specialized consultants, construction management firms, subcontractors of various sorts, labor unions, regulatory organizations, professional organizations and so on. Each environmental entity is driven by motivations of its own. Clearly, the true goals held by these entities may even be in conflict. Fortunately, they are rarely strongly adversarial goals and can often be mediated by environmental mechanisms.

One might argue that the problem is irrelevant in automated construction because, after all, one is talking about the agents of a particular construction site. That may be correct if construction agents do not to interact with agents in the external world. Such a condition might be accomplished by bringing all agents to be used under centralized control and forcing unifying goals on these agents, but such a scenario is unlikely.

Even in a single construction organization the goals of agents might differ, and this belief is supported by observations of construction organizations. There are reasons for this state of affairs. First, the motivations of a human agent may not be in full accord with the motivations for the construction project. This is depicted in Figure 10. Our building blocks for organizations may already be differently motivated! Lest one believes that this only applies to human agents and not to machine agents, consider the needs of machine agents — such as to be fueled and in proper working condition — as opposed to actual production activities.

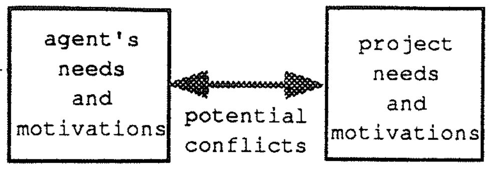


Figure 10. Conflicts Between Agent's and Project's Needs and Motivations

Secondly, although the notion of a single unifying goal is useful it cannot always be practically used as the singular basis for guiding the actions of agents. There may be a very tenuous and convoluted connection between such a goal and actual activities. It is often the case that the rationality of the original goals does not filter all the way down to the final agents. Some agents, in particular those closest to the work face, may only be cognizant of low-level subgoals. Subgoals derived from goals may conflict as shown in Figure 11. Such conflicts cannot always be resolved by returning back to the higher level goals. Holding subgoals which conflict is very similar to the problem of holding goals which conflict.

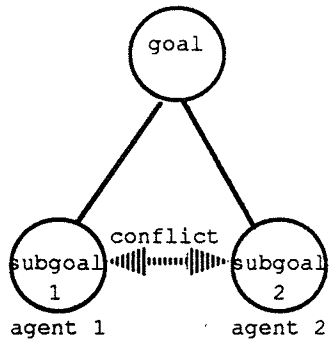


Figure 11. Conflict Between Agents Resulting from Conflict Between Subgoals of Goal

Indeed it is often useful to deliberately incorporate some conflict into construction organizations. In the matrix organization an agent may have to report to two bosses. The organization may have a mechanism to control construction costs which is in tension with the need to control construction time. This is the case in which there are two high-level goals, minimize construction costs and minimize construction time, which could be in conflict. Of course, the high-level conflict also generates conflicts all the way down through the decision-making processes. Not only is there contention among subgoals, but also among alternatives. This is depicted in Figure 12. Such contention is not easy to deal with because there is a lack of a single easily computable criteria for decision-making.

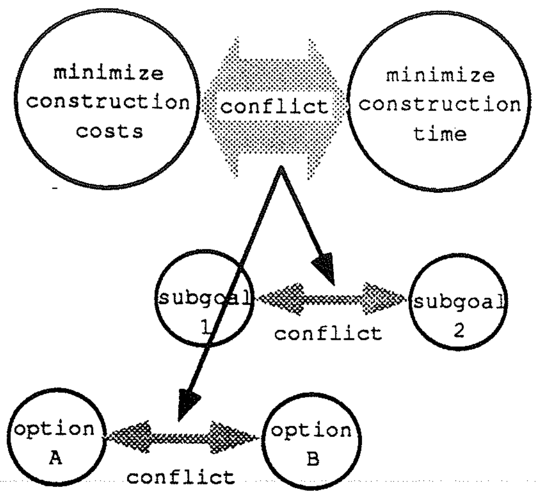


Figure 12. Conflicting High-level Goals Generate Lower-level Conflicts

It should be possible to develop machines capable of functioning in a multi-objective environment because, while the multi-objective problem exists with the current construction environment, this has not prevented human agents from working cooperatively. We have be careful, however, to understand the techniques and environmental mechanisms that have been developed to deal with the problem. For example, we might proceed in an iterative fashion, taking some effort to reduce time, then taking some effort to reduce costs, and so on. Various other mechanisms are discussed in Chapter 10.

# 5. Relationships and Interaction

## 5.1. Transactions: Structured Mechanisms of Interaction

When agents interact in a structured manner, we call that type of interaction a transaction. Examples of transaction mechanisms are negotiations and meetings .

Negotiation is a mechanism for achieving agreement between two parties. Usually one or both parties propose, then they thrash out the differences through counter-proposals, posturing or threats in an attempt to reach an agreement. When an agreement is achieved, it might result in a quick direct exchange or, if it involves some future activity, it might be 'sealed' by some means. One widely use method of sealing an agreement is for both parties to sign a contract that states approximately what they agreed upon.

The very first contact between the two parties may be established by a market mechanism, but subsequent contacts need not be. One or both the parties may learn how to contact the other directly.

Meetings are multiparty events in which each party knows about the other attenders of the meeting. In most typical meetings only one party, called the *speaker*, uses the main communication medium at a particular time, also called the *floor*, and the other parties are expected to listen. This is called the centrally directed meeting. It is widespread in the human knowledge environment because humans usually have only one intellectual focus of attention at any one time. We have two ears but we are unable to listen and understand two speakers simultaneously. Such a restriction may not apply to machines.

Other subsidiary communications may be in progress during meetings — spokesmen may be in contact with advisors or confidential written messages may be exchanged between particular parties. Parties can observe the points of view, proposals, threats etc. advanced by the other parties. Note that the parties need not necessarily be physically together.

Usually, for orderly meetings, there is a *chair* and there might be an *agenda* (both the chair and the agenda can also be determined at the beginning of the meeting itself). Both the chair and the agenda are methods to keep the focus of all the parties in the meeting on the issues of the meeting. The results of a meeting range from merely informational exchange and update, to a committed multiagent plan with all the roles of the parties determined, to the resolution of multiparty conflict which is impossible to resolve by bilateral negotiation.

### 5.1.1. Bilateral Transactions

A bilateral transaction is a transaction between two responsible entities. Usually these entities are agents or organizations, but they may even be countries. These types of transactions are the most common types of transactions. We will discuss the types of bilateral transactions, the modelling of these transactions with finite automata and a distributed planning problem.

#### *5.1.1.1. Types of Bilateral Transactions*

Some types of bilateral transaction are:

i) bilateral negotiations with immediate transaction, also called bartering,
ii) transaction mediated by tokens of exchange, also referred to as markets,
iii) bilateral negotiations with recourse mechanisms, also known as contract negotiations, and
iv) bilateral negotiations with threat, also called treaty negotiations.

Bartering is a mechanism for exchanging goods and services that is very close to the conception of negotiation propounded by researchers. In this model goods and services are directly negotiated until the parties can come to an agreement. If an agreement is achieved eventually, then an exchange is made, otherwise the parties break up. Each party tries to get the best deal that he or she can. In this system there should be a match between what each can offer the other. Bartering is unlikely to work if you are the producer of apples and want a car. This kind of disparity makes bartering unsuitable in modern society. Fortunately, in using negotiation in automation we are not too badly plagued. Machines can be made to willingly provide services for free — just for the asking. Therefore, the problem lies in trying to match tasks to capabilities. There is no exchange of value. We could also keep the quantum of the transactions roughly the same. But the problem of needing multiple bilateral transactions to reach a desired distribution of resources which can arise easily remains.

Buying and selling transactions are frequently achieved through the negotiation of the price, or of a contract for less immediate transactions. Where we have *tokens of exchange* as intermediaries we have a market model. Such tokens of exchange include gold, money and stocks. The evolutionary success of the market model versus direct bartering is clear from the preponderance of its use in the world. The beauty of this model is twofold. First, there is no need to have a match between the goods and services that each party can offer. Instead, two transactions are done instead of one. Sell your goods to get money and use the money to buy the goods or services you want from someone else. The second is you can postpone the second transaction. This allows you to obtain resources, especially perishable ones, at the time of need. It also allows you to accumulate resources and can therefore overcome the problem of disparity in transaction quantum. Pure market models have one problem, however. As with barter, you cannot get something now in exchange for something you can give in the *future*.

Contract negotiation overcomes the *credit* limitation mentioned in the previous paragraph. One party can perform a service for the other party now and receive compensation later. This is frequently the way the construction projects are negotiated — payments are made for items completed or even in progress. Indeed, there is a great variety of contract types. For contract negotiation to work we must have a decent recourse mechanism (preferably for both parties). One of the recourse mechanisms in the real world is the legal system. The aggrieved party can sue for damages.

Diplomats are also very familiar with the process of negotiation. What these specialists are concerned with is the balance of interests — more balanced their way if possible. In such situations, long standing relationships are important — vis the iterated prisoners' dilemma [Axelrod 84]. I shall not go into a discussion of these specialized negotiations. We are more concerned about a communication structure that include machines. Of course, this is neither an exhaustive list nor an in-depth discussion of these coordination methods.

#### *5.1.1.2. Modelling Bilateral Transaction Frameworks with Automata*

Let us see how a negotiation process can be modelled by automata. Figure 13 depicts an automata graph of a possible negotiation process. The circles denote states while the arrows denote transitions.

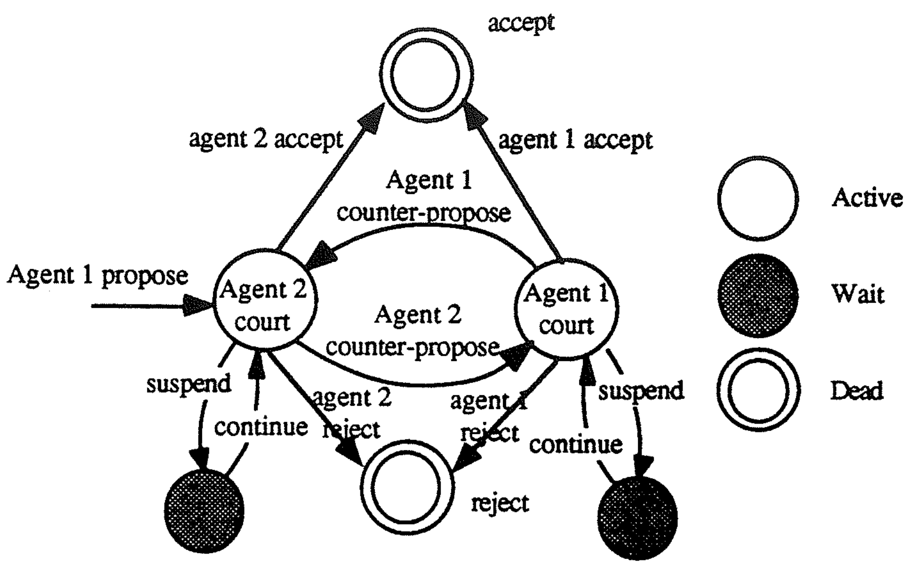


Figure 13. Automata Formulation of Framework of Negotiation

In order to be a player in the negotiation framework, each agent assumes a role and keeps track of the status of the negotiation with a transaction tracking mechanism. In order to know what is allowed or expected, each agent ought to have automata representation and update it with each change or activity. Movement between status requiring the knowledge of both parties is only made on successful completion of the communication act that transfers the information. After agent 1 proposes, the marker arrives at transactional state agent 2's court.

Agent 1 has the following expectations and rights when the marker is in agent 2's court:

- Agent 2 will be considering the proposal.
- After due consideration, agent 2 may accept, reject or counter-propose.
- Either agent may suspend but after revival we return to agent 2 court.

Agent 2's expectation and obligations are as follows:

- In this particular negotiation framework agent 1 will not change the tabled proposal although he might suspend it. Another negotiation framework might allow agent 1 to change his proposal before agent 2 decides.
- It has to make a decision on whether to accept, reject, counter-propose or suspend

In the transactional framework for negotiation described above, an agent does not have the rights to kill its own proposal or counter-proposal, although it might suspend the proposal.

Here is how it works. When a bilateral transaction begins, each agent creates and maintains an automata graph (or only the relevant parts of it) of the type shown in Figure 13. The agent initiating the transaction is called agent 1. Agent 1 initiates the transaction by proposing the terms of the transaction. After this initial proposal, the state of the transaction is denoted by the node labeled as "Agent 2 court". Both agents take note of that. The important actions in this state are to be made by agent 2. In particular, agent 2 may either accept the terms as proposed by agent 1, reject them or propose a different set of terms. The latter is called counter proposing. If agent 2 accepts, then there is an agreement between the two agents which is presumedly binding. The transaction terminates. If agent 2 rejects, then there is no agreement and the transaction terminates. On termination of the transaction, the automata graphs may be discarded.

This negotiation framework is slightly different from the ones proposed in the literature [Smith 80, Koo 87]. In particular, for this framework the transaction itself may be suspended by either agent from any non-terminating active state. Suspension and continuation has to be tracked somehow, but not necessarily by maintaining wait nodes in the automata graphs as indicated in the figure.

The active states have the following general expectations (among others):

- the agent whose proposal or counter-proposal is under consideration is expected to be in a *receptive* state for communications from the focus agent. A receptive state for communication doesn't mean that physical communication links are necessarily always there. It means that the agent will not completely ignore the communications initiated by the other party.
- the focus agent is expected to pay attention to the tabled proposal. If the agent is taking too long to come to a decision and the other party wants to do something else and cannot be in *receptive* state for communication, then the other party may unilaterally suspend the proposal.

The suspension states have the following expectations (among others):

- neither agent is obligated to give any attention to the proposal nor to be receptive to communication acts of the other party. They are therefore free to work on other matters.
- either agent may reactivate the proposal only after communication reception is reestablished.

Before anyone begins to believe that such frameworks are unique, we will modify the framework above slightly to obtain another framework, as shown below. The previous framework does not allow an agent to move from suspension to rejection directly. This could have led to some problems.

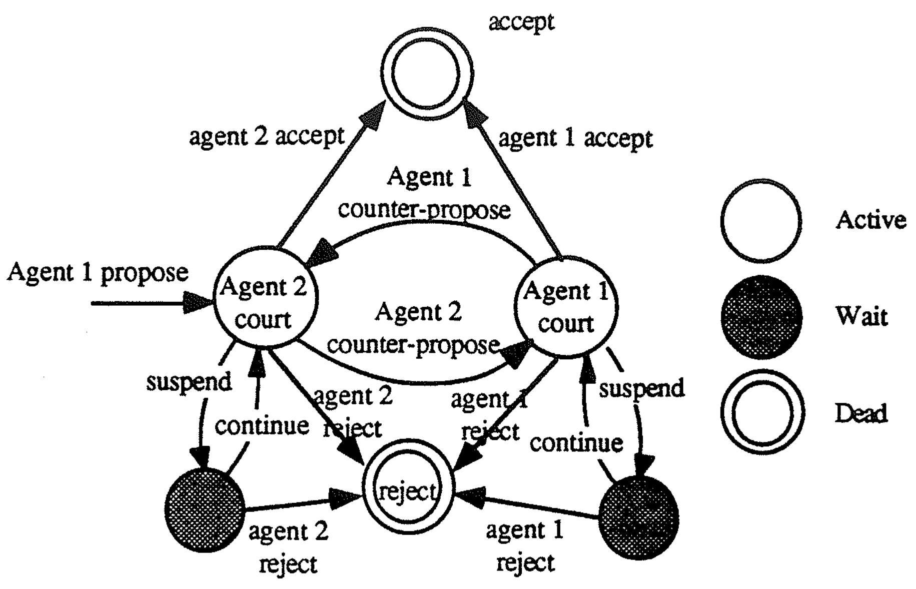


Figure 14. Modified Finite Automata Model of Negotiation Framework

A number of different transaction frameworks could exist and be used by different agents to keep track of the state of the negotiation process. In particular, in construction environments many simpler transaction frameworks may be used. For instance, there may be a limit on the counter-proposing that goes on which is not captured in the framework above. Figure 15 shows a framework that allows only one counter-proposal and no suspension of the transaction. The simpler the transaction framework, the less the overhead costs and so there is a tendency to use transaction frameworks that are very simple.

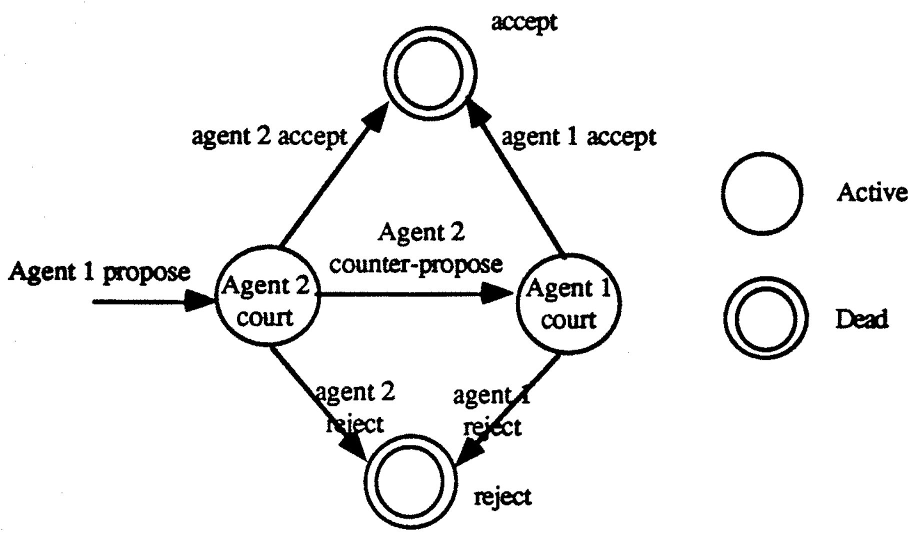


Figure 15. Finite Automate Model of Simple Negotiation Framework

Except for time, resource and capability limits, nothing prevents the agent from having several active transactions in progress simultaneously. Nor is there anything that prevents transactions being negotiated from influencing one another. The decision process that each agent uses can take into account any information that it has at that time — including the status of other negotiations. In operating system terminology this is regarded as a system with race conditions since the end result can be dependent on the temporal order of execution of each step of the different negotiations. Deadlocks can occur. Fortunately, by applying temporal decay we can force negotiations that stay around too long to be terminated. This will break the deadlocks.

There are other concerns about the theory. Can the theory be used to build components of for managing bilateral transactions among machine agents? Could it be of any use in the human knowledge environment? We give a script as an example of how two agents might use the transaction framework just described.

We will now look at an example of the use of a simplified transaction framework for material purchase. Let Joe, the construction manager, be agent 1, and Smith the marble supplier, be agent 2. Joe wants some marble for the project and proposes to buy it from Smith for an advertised price. However, Smith does not have a ready stock of the particular marble that Joe wants. Smith counter proposes with several options of marble types and their respective prices. Joe either accepts one of those offers or rejects them all.

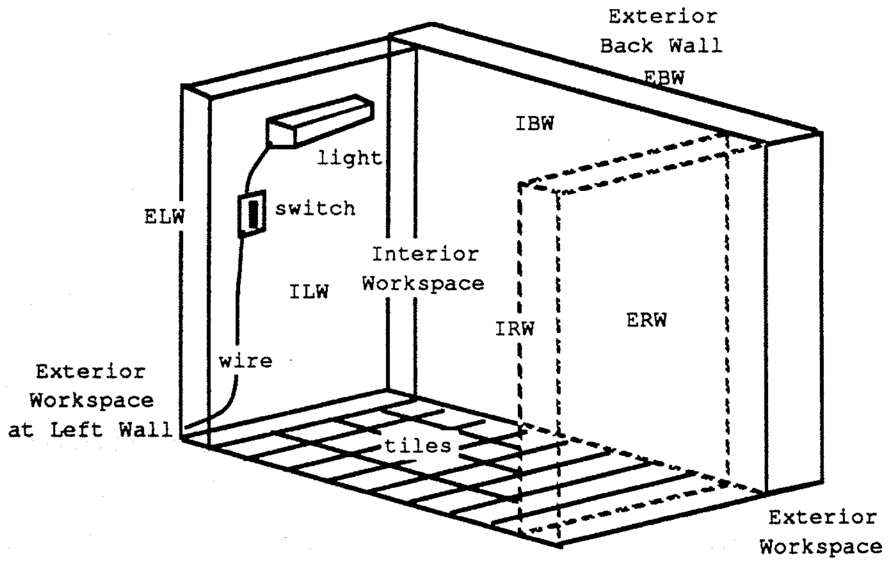


Figure 16. Bathroom

Our next example is on the use of negotiation for synchronizing plans of agents. This example is adapted from [Koo 87]. Let Mark, the electrician, be agent 1 and Charles, the framer, be agent 2. Both of them are to work on parts of a bathroom. The electrician has to install wires, switches and lights while the framer has to install the frame, three interior and exterior walls, and the floor tiles. The bathroom is shown in Figure 16. The independent plans of the two agents are shown in Figure 17.

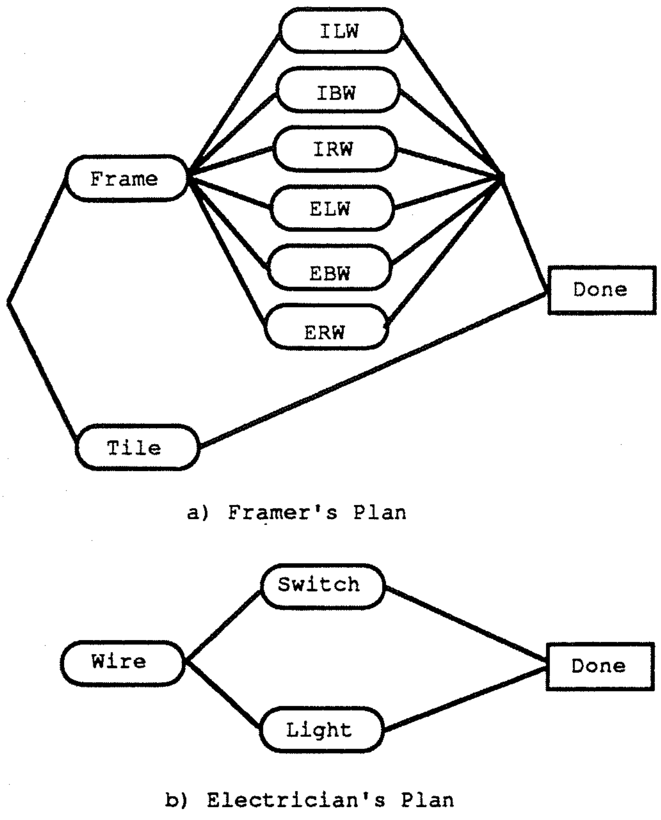


Figure 17. Framers' and Electricians' Individual Plans

There are two issues here. First, we need a correct order of installation. Each agent knows the correct order with respect to his own work. Wires run between one of the interior and the exterior partition walls. Besides, they can only be installed only if at least one of the walls have been installed first, since wires need to be supported, but it does not matter which one, which makes it even more interesting. The two options are shown in Figure 18. The electrician needs to know about these interactions but the framer does not.

Second, there is spatial interaction. While the framer works on the exterior walls with the exception of exterior wall 1, the electrician can work on the wires, switch or light. In the case of exterior wall 1, the electrician cannot work on the wires, but if the wires are already installed, he can work on the switch and light. To install the wires after the installation of interior left wall, the electrician needs the exterior left wall workspace. On the other hand, to install the wires after the installation of the exterior left wall, the electrician needs the interior workspace. When the framer works on the interior walls or on the tiles, the electrician cannot do his work. This is because it is overly crowded in the confined space of the bathroom.

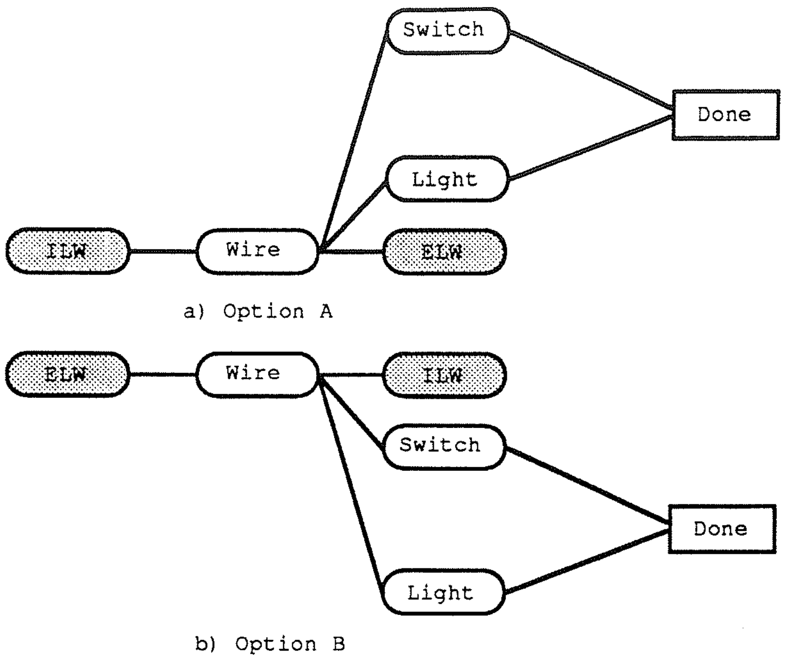


Figure 18. Alternatives in Activity Ordering Conditions

So these two agents need to sort out who does what and when. We shall assume that there are move-in and move-out costs and overall uniform time overheads for the two agents. In synchronizing their plans, the agents might like to minimize their own costs. However, the global optimum may not be arrived at as it depends on the order and progress of negotiation. The type and amount of information transferred between the participants is an important determiner. The more the electrician know about the framer's work (or vice-versa), the better the chances are of arriving at the global optimum. For instance, if he knows that the framer will require interior workspace when working on the interior left wall, then option A would be the better option, since in option B, working on the switch and light, must follow installation of the interior left wall rather than immediately follow the installation of wires. If this is not known, then the process of negotiation could help uncover this problem.

Mark may communicate immediately, or through the course of negotiation, the following information:

i) I need to install some wires in the left wall.
ii) This requires that you install either the interior or the exterior left wall first.
iii) You have to wait for me to finish with the wires before install the opposing portion.
iv) Installation of the wires will take 15 minutes.

v) During installation I shall need the workspace opposite the supporting wall.
vi) I have other work lasting 20 minutes that needs the interior workspace.

From this information Charles figures out one possibility as shown in Figure 19 (a). On the other hand, the interior back wall, interior right wall and tile can be done after the wires and before Mark's other work or after Mark's other work as shown in Figure 19 (b) and (c). Several other permutations are also possible.

The existence of alternative construction methods can create many possibilities in construction sequences. For example, there can be numerous construction sequences for enclosing a building. Equipment that has to be brought in before the building is enclosed can thus be slotted into the scheme of things in numerous ways. The ordering relationships between the activities can be different enough not to be representable in a single nonlinear plan network.

Option 2 is probably better if the framer has sufficient work with the exterior walls while the electrician uses the interior workspace as it allows the electrician to finish earlier. Notice that there are three activities that the Charles can perform while Mark performs other work. Observations of other possible plans involving the installation of the exterior wall before the installation of the wires indicate that they are not such good options. Therefore, Charles select the best schedule and informs Mark about it. Note that Charles has to make early commitments on two important things, the wall he will finish to support Mark's wires and the time at which he will do so. He can actually decide a little later on as to whether Mark's other work should proceed before or after some of his own. However, earlier commitment would be preferable because this will allow Mark to work on some other project during periods he is not committed to work on this one. Even if he made those commitments Charles has significant leeway to reorganize some of his work.

Here other work is very short so it is not so constraining on Charles's work. If other work is much longer, further negotiation might be useful to figure out if some better schedule can be constructed.

Why is establishment and breaking of communication not indicated explicitly in the transaction framework? Although we might include them explicitly, we believe these considerations belong at another level of transaction management by agents. All transaction frameworks involve communication and so it is implicit that some form of communication link needs to be established beforehand and during each communication act. Such generically applicable knowledge should be abstracted away.

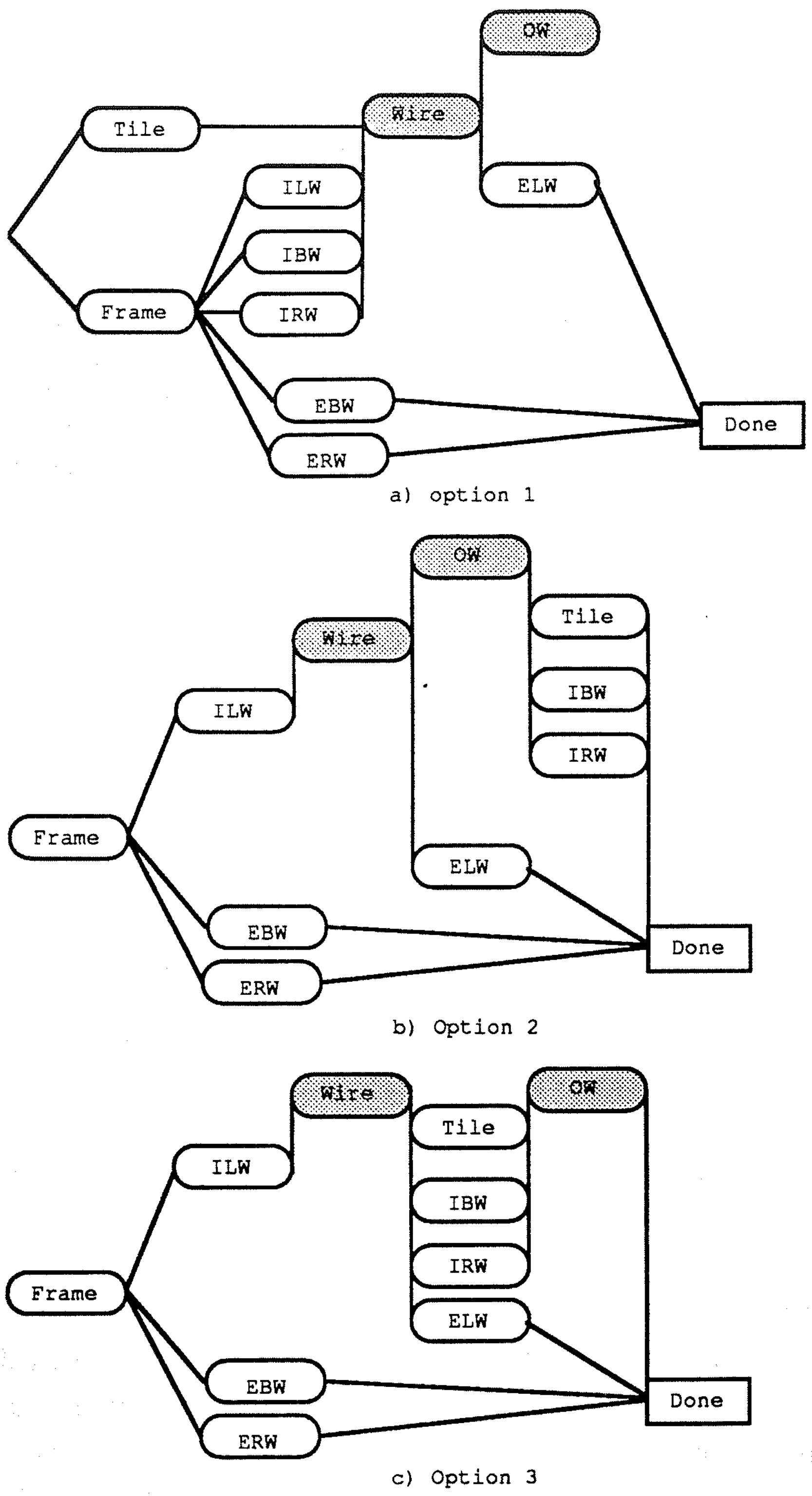


Figure 19. Example Plans Meeting Mark's Constraints

Let us try and comprehend what is going on when a transaction is being established and while it is in progress. We start with two agents as shown in Figure 20(a). Before there was any transaction between agents A and B, because of autonomy and independence assumptions, the two agents have very little to do with each other. Therefore their activities do not interact with each other. Now let us say that agent A discovers that he might be able to accomplish something by interacting with agent B and decides to do it. At this point agent B is still going about its own work independently.

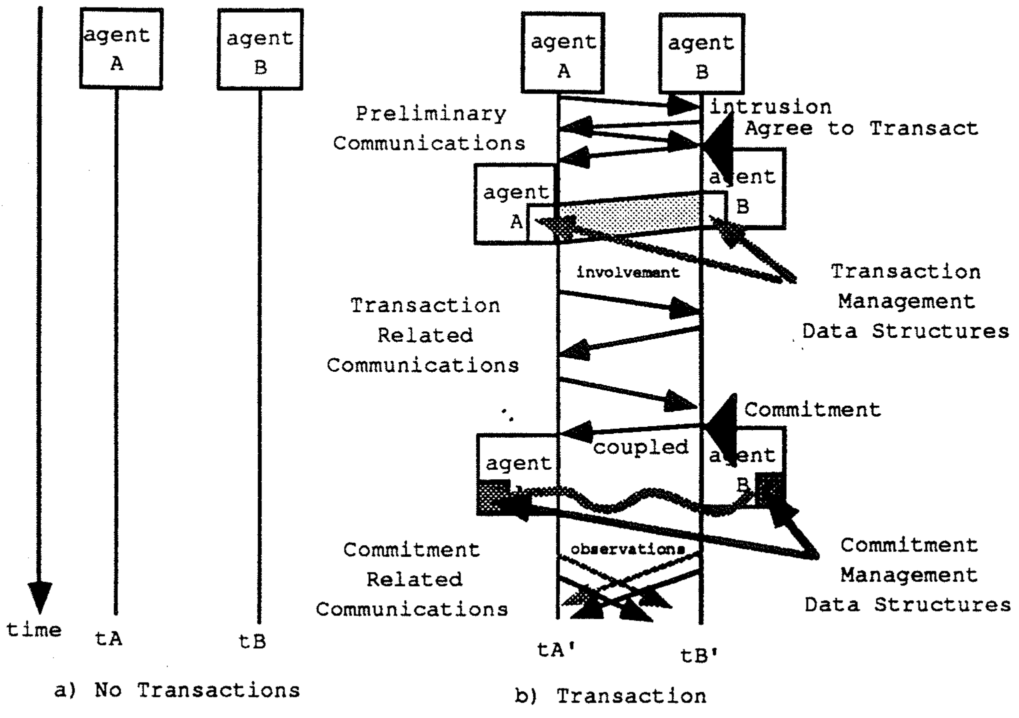

agent A
agent B
time
tA
tB
a) No Transactions
agent A
agent B
Preliminary Communications
intrusion
Agree to Transact
agent A
agent B
involvement
Transaction Management Data Structures
Transaction Related Communications
Commitment
coupled
agent
agent B
Commitment Related Communications
observations
Commitment Management Data Structures
tA'
tB'
b) Transaction

Figure 20. Temporal Progress of Transactions

Once agent A decides to make contact, we see a bootstraping process. It starts with agent A contacting agent B to establish a transaction. Agent B needs to agree to transact with A before anything more can proceed. This might involve sending a number of preliminary signals and is itself a very simple transaction not needing any further preliminaries. The first signal is an intrusion because basically agent B has been going along without any concern about agent A and then it is being accosted by agent A. Agent B may or may not agree to transact further. If agent B does not agree to transact with agent A then agent B can ignore the advances of agent A or send a *NO* — do not disturb me — reply. These preliminary communications typically follow one of a few patterns in what may be regarded as intrusion protocols. Agent B may want to know who agent A is and, in a very brief way, why it is being accosted. If agent B agrees to transact further, then further communication may take place between the agents to establish the type of transaction frameworks. Sometimes the agents may assume from a particular context what the appropriate transaction framework should be, perhaps the negotiation framework of Figure 13. For instance, when doing retail buying and selling, humans just assume a particular protocol.

Once both agents have agreed to transact, the agents have become involved with one another — they no longer live completely independent lives. There is an obligation on the part of both agents to see the transaction through to the end. In each agent there is a data structure that has no meaning and usefulness apart from accomplishing this purpose. We can view this as a single data structure which has pieces residing on two different agents and acts as a bridge between the two agents as indicated in Figure 20(b). A lot of communication may take place between the two agents subsequently and some of this communication may change the status of the transaction. Strictly speaking, it is not necessary for there to be associated changes in the two pieces of that one global transaction tracking data structure, although that is usually the case.

If the transaction results in the commitment of each agent to perform something, then the two agents are coupled as we show in Figure 20(b). The agents have moved a wee bit away from their original independence. It is an abstract coupling because we cannot easily observe it, but the agents do not have the same freedom to act as they had before.

Often the progress of these agents in fulfilling their commitments would be tracked by the other agent or by a third party. The type and intensity of such tracking depends on the nature of the transaction and contract.

Once the commitments of both have been discharged, the coupling between the two agents might be broken. For instance, if I received the goods for which I paid, then the transaction is over. Human transactions are not that simple. Even in the case just mentioned, the law assigns implied warranty of merchantability, so some slight coupling might still remain. But since we consider the agents as basically free to act after the fulfillment of their commitments, for convenience, we consider the fulfillment of contractual commitments by both agents to be the end of coupling between the agents.

Communication links can often be broken without harm in between communication acts. The reestablishment of physical communication links depends on the environmental communication support mechanism. Although physical communication links may be broken, the obligation entailed by the unfinished transaction remains, that is the communication obligation remains.

The environment may unilaterally break communication. Dealing with such spontaneous breakdown of communication is specific to each protocol but has little to do with the intent of the transaction and makes for unnecessary complexity. For instance, in the above transaction framework we might provide the following over-arcing rule: If an agent discovers that it is no longer possible to communicate with the other party in a reasonable fashion then change the status of the proposal to that of suspension (another possible consistent alternative is to go to the reject state). Since this is standardized, the proposal arrives at the same state from the viewpoint of both agents. Each agent measures the condition of the communication link and makes a decision. This works even if one side can send but not receive messages from the other party, communication cannot be interpreted or where the communication is intermittent and uncertain. Under certain conditions, it still works if an agent malfunctions.

In the negotiation framework above, each agent maintains a transaction tracking mechanism. They are supposed to be maintained in relevant synchronization. Suppose, for some reason, the markers land at different status in the different agents' transaction tracking mechanisms. In this case, expectations of the agents that rest on the frameworks do not correspond to reality. For example, agent 1 could be thinking that it is now in agent 2's court while agent two thinks that it is now in agent 1's court. If we are not careful, both agents might wait for each other indefinitely. Another possibility is one in which the agents, by mistake or misunderstanding, use two different transaction frameworks. Again expectations are not the same, potential deadlock situations could arise, and the breaking of such inter-framework deadlocks would be necessary.

A time-activated mechanism could be used to deal with these problems. When the agents in a communication are in receptive mode, we place a limit on the time allowed for some form of communication to arrive from the other party. This time might be a function of the contents of the tabled proposal — that is, an agent may decide how much time to allow for the other agent to make up its mind. If this time limit is exceeded, the agent executes a recovery procedure. It is often useful to include the time for replies in communicated messages. Recovery procedures may be associated with particular transaction frameworks or there may be general recovery procedures. The recovery procedures associated with particular transaction frameworks may prescribe the use of "compensating transactions," in the parlance of database researchers. They are additional transactions to correct the problems arising from previous transaction failure. All these are generic considerations when dealing with the complexity of the real world.

### 5.1.2. Multilateral Transactions

Several agents jointly agreeing to a transaction constitute a multilateral transaction. Multilateral transactions might include bilateral transactions. It might be necessary to activate a package of interrelated transactions for certain construction work.

#### *5.1.2.1. Conditional Transactions*

Sometimes a multilateral transaction cannot be closed at a single point in time — it has to be built in several stages. In such cases, parties may make conditional transactions such that if other agreements can be closed the earlier conditional agreements become binding automatically.

#### *5.1.2.2. Multiparty*

Multiparty transactions are transactions involving several agents. The transaction needed to bring four cranes to lift a heavy deck structure is a multiparty transaction. There is no sense in making agreements between just any two of the parties. There are other cases in construction that require agreement to be achieved among several agents for there to be meaningful agreement. Certain arms reduction agreements also require multiparty transactions to be achieved. In certain cases, multiparty transactions are not necessary but it is better to use them.

#### *5.1.2.3. Pact Negotiations*

Pacts are similar to contracts, but they are multiparty. Usually, each party has committed to make contributions to the pact and expects to receive certain benefits from the pact. The commitment mechanism in such contracts may be performance or availability driven. A discussion of these commitment types appears in the next section.

## 5.2. Commitment

Digital Webster defines "commit" to be "to pledge or assign to some particular course or use," and a "commitment" to be "an agreement to do something in the future." Here we are concerned about commitment among agents. Here commitments are made from one party to the other. The party to which the commitment is made is called the *holder* of the commitment and the party making the commitment is called the *bound* party.

Agents can agree to do anything, but meaningful commitments require, at the least, that there is some possibility that the agent who makes a commitment can meet the commitment and that there is some relationship between the commitment and the agent. There is no sense in committing to have good weather over the duration of the project if one has no control over the weather. On the other hand, one can commit to take actions to prevent accidents from happening on a site, and be held to it. To a great extent, it is possible to check if commitments are reasonable through the use of general knowledge about the limits and capabilities of agents, organizations and societies. Realistically, commitments must be fulfilled in reasonable time even if the time is unstated. Commitments do not last forever.

We have already seen that the actions of other agents can be one of the factors of change, uncertainty or unpredictability in the environment that makes it difficult for an agent to make decisions or plan its actions. *Commitments help to lessen uncertainties.* Agents have to rely on the commitments of other agents to build their own plans and, therefore, need to know about the commitments that potentially affect them. In active commitments, the agents must perform some actions in order for the commitments to be fulfilled. In passive commitments, the agents must refrain from performing actions for the commitment to be fulfilled. In the next few subsections we will discuss briefly three types of commitments, namely

i) performance-driven,
ii) availability-driven, and
iii) constraint-driven.

The first type is usually active, the second is often both active and passive and the third type is passive. In the framework of computational ecologies such as that represented by the CONTRACT-NET, we have commitments such as

i) perform an action,
ii) perform a task, or
iii) plan and take actions to fulfil a goal or subgoal,

which are essentially performance-driven commitments.

### 5.2.1. Performance-Driven

There are two subtypes of performance-driven commitments; commitment to act or action-driven commitment, and commitment to results or result-driven commitment. If you agree to dust your room then you have agreed to perform actions of dusting, but if you agree to clean your room then you have agreed to the result that after your actions the room can be judged clean. In the former case, should the room remain dirty after dusting which, after all, merely redistributes the dust, one cannot be held accountable.

There are several levels of results that an agent might commit to. It depends on how the results are to be evaluated. If the results are evaluated literally, then they only need to meet the stated criteria. For instance, if you agree to build a tower of blocks and you construct one, then you have fulfilled your commitment even if the tower happens to be unstable. On the other hand, if you agreed to build a retaining wall, then you have to build a wall strong enough to retain the earth behind it. You are evaluated according to whether you have met the functional goal implied in the result. A few examples from construction might serve to further illuminate the nature of performance-driven commitments.

An example of the performance-driven commitment is the turn-key contract. Functional turn-key contracts are at the high end of the scale since the facility must meet functional requirements as well as temporal requirements — everything that needs to be done has been done before delivery — to be considered fulfillment of the contract.

If you go to a mechanic to repair a broken tail light, you expect when he is done that, not only has the tail light being replaced with an unbroken one, but that it functions properly. The responsibility lies with the mechanic to make it functional.

Under certain circumstances, less demanding contracts might be written.

### 5.2.2. Availability-Driven

In availability-driven commitment, an agent only commits to being *available* or to make resources available over particular periods of time or until certain events occur. The former is a commitment of a chunk of time and knowledge by one agent to another to use. A typical expression of this type of commitment is the employment contract. A common type of employment contract involves monetary remuneration and a benefits package in exchange for services. Service hours are fixed by the contract.

Contracts of this kind limit the classes of activities that one agent might request under the ambit of the contract, but might not precisely specify the exact times or methods for performing them. Therefore, they preserve some flexibility for the party to which the commitment is made and constrain in some ways the scope of activities of the committed party. This gives a longer useful life to the commitment by allowing adaptation to occur at the time of need. An employment contract that stipulates the exact task for the agent becomes obsolete if conditions change and the performance of such a task is no longer needed. In construction, a wide variety of tasks need to be performed and the times for performing them tend to change. Commitments have to be obtained fairly early to obtain predictability on which plans can be based, but flexibility is important to achieve greater use of the workforce.

### 5.2.3. Constraint-Driven

In some cases agents are not committed to produce something or make its services available but are instead committed to terms that place restrictions on the agent's freedom of action or choice. For instance, a distributor may be committed to obtaining its goods from only one company. The options of the distributor have been severely restricted by the contract in exchange for certain benefits such as being the sole distributor and being given special prices.

Constraint-driven commitments are also useful in obtaining order in the construction site. At certain times, it would be useful for one agent to know that other agents are not going to be at a hazardous location. It might obtain avoidance commitments from them to that effect. A crane moving a heavy object would like all other agents to be out of or under the path of the object.

## 5.3. Power Relationships

There are many possible types of power relationships. We will only look briefly at a few. One dimension is that of command hierarchies. It is the hierarchy that deals with the cognitive level at which an agent may be instructed. A second dimension is the extent to which the agent might be expected to obey the instructions. A free agent is one with no obligations to act on the instructions of another, whereas a slave agent is compelled to do so. There is a spectrum of possibilities in between these two extremes. Another aspect of power relationships is that of assignation and sharing of power. Such relationships typically arise from delegation.

### 5.3.1. Command Hierarchies

In interactions among agents, one agent may make a request of another agent. There is a continuum of levels at which these requests may be made. We call this a command hierarchy because it is generally true that an agent with authority to make requests at higher levels of the hierarchy, and be obeyed, also has authority to make requests at one or two lower levels of the hierarchy.

At one of the lowest levels of the hierarchy, an agent might directly control the primitive actions of another agent. A teleoperated machine, for example, is a machine that takes commands at that level.

At a higher level, a machine might be controlled by commands such as move forward, backward, left or right. At still higher levels the agent may merely be asked to move from room A to room B. At even higher levels an agent may be asked to fetch a box.

The higher the up the command hierarchy that an agent is instructed, the greater the freedom of choice the agent has. Unless informed, the one who instructs the agent knows less about what would happen.

### 5.3.2. Delegation

Delegation is the assignment of tasks or goals to an independent agent with decision-making power. Another agent has been empowered to act on behalf of the delegating agent. Various types of restrictions may be placed on such transfer of decision-making power and authority.

The delegating agent typically loses some measure of control, and thus the ability to predict outcomes, as a result of delegation, as opposed to performing the tasks or making the decisions itself. The uncertainty increases with increase in the choices available to the delegate. In many cases the delegating agent is not knowledgeable in the field of expertise of the delegate and, therefore, makes an act of faith in delegation. Such ungrounded leaps of faith are sometimes termed *magic beliefs* — an agent believes something to be true without having grounds for such beliefs.

# 6. Architecture and Design of Problem-Solving Programs

## 6.1. Semantics of Knowledge Utilization

In construction, intelligent machines, like their human counterparts, will need to harness considerable knowledge to plan, execute and control autonomous field tasks. But each agent holds only a portion of the total knowledge and abilities required; it is unlikely for any human or machine agent to have all the necessary knowledge or abilities. Indeed, throughout the duration of construction there is a process of assembling and using knowledge from many sources in the *knowledge* environment. The information needs of an agent may be met from other agents at the construction site or from suppliers, databases and other information sources. Humans readily satisfy their information and knowledge needs by such means, but there is still a lack of theory that will open the way for machine agents to do so. We are developing theories that will enable us to confer machine agents with some of this useful environmental intelligence.

Taking the stand of an external observer, we can ascribe a meaning, a *semantics*, to the process by which an agent uses information. If that agent confines its use of information to the information that it started with and anything derived therefrom, we say that the agent follows closed-world semantics [Levesque 81, 84]. If the agent can also use knowledge in its external environment, we say that the agent follows open-world semantics. We hope that this notion will add to our understanding of how to construct intelligent autonomous machines. There are various degrees to which information and knowledge in the environment can be used by an agent. Ultimately, it is possible for such an agent to regard the entire world's knowledge as its knowledge base. The notion of open-world semantics has to do with the way an agent understands its own knowledge — e.g., whether its own knowledge at any particular time is to be treated as already complete or merely partial. If an agent believes its knowledge to be complete, then there is no need to look for additional knowledge elsewhere. On the other hand, an agent believing that it does not have all the knowledge might actively look for missing knowledge. [Levesque 81] describes some methods to capture this concept.

### 6.1.1. Closed-World Assumption

The closed-world assumption is implied in the design of most algorithms. Once the algorithm and data have been set up, no external new data can be incorporated that could modify the end product of the processing. The program itself does not change during the processing. The program assumes that the data provided to it is all that there is to process. We call these types of algorithms CWA algorithms.

The three portions of running a CWA algorithm are:

i) start
ii) solution time
iii) finish

At the start, both the data and the algorithm must be defined. At the finish, the results of the algorithm are available. The most interesting stage is the solution time. The solution time generally depends on the size of the problem and the type of hardware and software it runs on. During the solution time no new data may be added while the algorithm is running, and the algorithm may not be changed. If we suddenly find that we have new data to add, we stop the running of the algorithm, put in the new data and restart the algorithm. Algorithms will not even stop by themselves because they are oblivious of events occuring in the external environment. Neither can we change an algorithm midway by, say, tweaking a knob.

If we wish to obtain a sorted list of all the crane equipment companies in California, someone at least has to provide the program with the names of all such companies. There is no way for the program to discover or correct the problem where most of the crane equipment companies have been left out. The program may be able to remove equipment companies that are not in California, but not add omitted ones to its database. The program has been designed on the assumption of a closed world where its data is concerned. A human asked to do perform the same task is a more capable. He doesn't just use names he can recall from memory, which is the equivalent of what a program does. A human being follows the open-world assumption.

There is much work reported in the literature on artificial intelligence on the proper semantics to be given to closed-world databases [Reiter 78] and knowledge bases.

### 6.1.2. Open-World Assumption

Here one assumes that there may be more knowledge available that one currently has. What one knows is only a fraction of the knowledge that is available. We can even regard the knowledge of the entire world as a huge knowledge base of which our own knowledge is but a small fraction. The agent is aware of the potentialities in the environment.

We will discuss ways of creating open-world assumption algorithms or OWA algorithms in the next few sections. There are several types of such algorithms, and we list two of them below:

i) accepts data at any time but does not actively seek new data
ii) actively seeks new data and incorporates the new data

The first type can deal with the problem of, say, having a sorted list of the names of babies born in a hospital. "Bob, please maintain a sorted list of the names of babies in this hospital. Here is the initial list of names and I shall give you the names of new babies". So Bob sorts your initial list and whenever you give him a new name he adds it to this list. At any time we can put a new name in without starting the algorithm and ask for the list. What kind of an algorithm can do this?

The Bob in the previous paragraph is still not too smart. You have to find out about the new babies and which babies have left and tell Bob. What we would like is a list sorter following the open-world assumption of type (ii) — one that is aware of information availability, actively seeks information, obtains information and incorporates information.

Agents that have open-world assumptions can use closed-world assumption algorithms for part of the work. Humans, for instance, use a lot of CWA algorithms.

Such an understanding has implications on the design of intelligent agents. One simple and cheap improvement to the standard closed-world algorithm is to incorporate a guided user query and commensurate data-input facility. This takes advantage of the environmental intelligence of humans. Human beings become the information pumps for machines.

Given the existence of such a vast amount of knowledge in the knowledge environment, agents need to be circumspect when trying to exploit the knowledge in their environment. Chapters 7, 8, 9 and 10 will cover some of the related issues.

## 6.2. Self-enhancement Capabilities

Self-enhancement capabilities are those that a system can employ to augment itself. We think of a system as having particular functionalities. For instance, we can write an application that is supposed to answer queries about parents. We expect it to be able to answer queries about parents. Usually the program also has some associated data. If this data is incomplete for the functionality, then we call that partial data. If the program itself is incomplete — that is, if there exist solutions to the problem but the program cannot find them — then it is a partial program.

### 6.2.1. Partial Data

An aggregation of data, for example that in a database or knowledge base, is partial if it does not represent all the data that is potentially relevant to what the functionality of the database says. For example, a database that contains information about only half the employees of an organization but that is supposed to answer queries about the company's employees is a partial database. If we take the viewpoint of an external observer, let *W* be the totality of information available in the world about that topic and let *D* be the information captured in the knowledge base (Figure 21). *D* is a subset of *W*. If *D* is a proper subset of *W*, then there may be ways to extend *D* by obtaining from the environment information in *W* that is not already in *D*. If such extension is actively made by the knowledge base itself, then we call that process self-enhancement. The knowledge base itself recognizes and makes up for its lack.

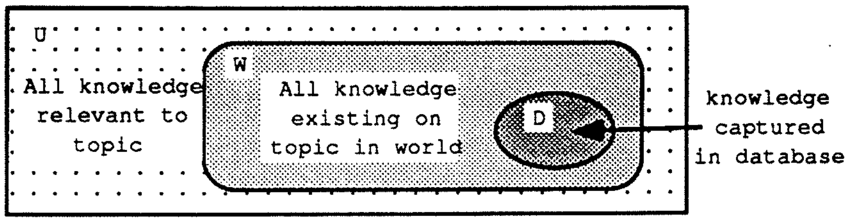

U
All knowledge relevant to topic
W
All knowledge existing on topic in world
D
knowledge captured in database

Figure 21. Partial Data and Knowledge

A simple conceptual model of the way CWA algorithms work is shown in Figure 22. The CWA logic engine basically uses only the data available from the database and the intermediate data structures. This CWA logic engine also has termination criteria which determine when it is done or when it fails. For instance, in a query of who the parents of Joe are, when we have used up all the predicates on parents it may be time to terminate. Each time through the loop we ask the database for a parent predicate that has not been given to us before and the database gives it to us or informs us that there are no more. If there are no more parent predicate, the CWA logic presents or writes out the results and terminates. Meanwhile, during the running of this logic engine we may build some intermediate data structures such as a list of the parents found so far. These intermediate data structures resulted from the CWA logic figuring out the consequences of the data given to it by the database.

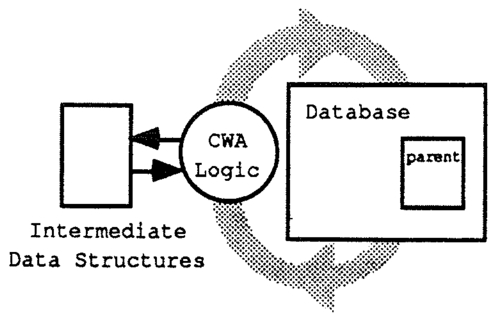


Figure 22. Using CWA Logic

Figuring out the consequences of data is the deductive step. But whether an agent should take the deductive step or not depends on what it knows about the data. There is the possibility that the data is inadequate — that is, the database represents only partial data. If the data is not adequate, then it might be better to try to gather more data before proceeding with deduction. Should this be the case, whether to base one's plans and actions on such partial data is the problem of *pragmatics*. One ŏf the ways we can have this kind of open-world semantics is to provide for data and information enhancement.

When we have a complete CWA algorithm working on a complete database, there is no need for further enhancement. Complete algorithms and complete databases, however, may not be practical because they take too much computational effort or memory resources. Typically we have partial data and partial programs. Partial programs will be further discussed in the next section. Using a CWA algorithm on partial data means that we may miss the use of information relevant to the problem that is available somewhere else.

The notion of self-enhancement is somewhat different from the notion of default reasoning. In default reasoning [Reiter 80] assumptions are made about the nature of *U -D* (Figure 21). In self-enhancement agents may take actions to obtain the knowledge in *W- D* to reduce *U - D*. Of course, it is not possible to obtain knowledge in *U - W* from the knowledge environment. If the agent cannot spare the effort, then it might fall back on default assumptions such as using default rules or circumscription.

We now discuss the concept of database enhancement. For instance, in the above case, if the database runs out of parent predicates to provide the logic engine, it can launch a search process in the environment to obtain more information about the parent predicate. For instance, the database management process can find out from environmental sources of information that the mother of Joe is Sally; from this it derives a new parent predicate stating that the parent of Joe is Sally. Once it has obtained this new information, it can provide it to the logic engine for use.

Clearly this process can be done for every predicate and rule in the database, although it may be too costly to do so. The piece of the system that makes it possible to use knowledge in the environment to extend the database is called the database extender. Such a database extender ignores knowledge that is not related to the database and the structure of the database is unchanged. The schematics of the database is unchanged. After all, a database extender merely provides known database services to the logic engine and does not pretend to be a complete learning engine. Even with these limitations, the database from the user's viewpoint is a database that follows the open-world assumption because it does not limit itself to what it knows initially.

Practically, in the design of a database enhanced with a *database extender*, we would want to control up front what is extensible and what is not. We might state that the parent predicate is extensible, which means that it is partial data, potentially incomplete. Thus if all the parent predicates in the database have been used, then some attempts might be invoked to obtain more parent predicates.

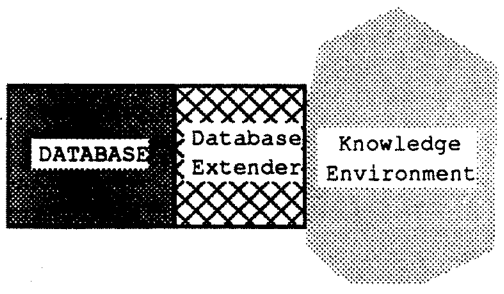


Figure 23. Self-Enhancing Database

The architecture that makes this possible involves using open-world semantics in the database engine and environmental knowledge to exploit the knowledge environment. We explore a little how this might be done using pseudo-Prolog code.

```
% Meta-interpreter

solve(true).                                                              (1)
solve((GoalA,GoalB)) :- solve(GoalA),solve(GoalB).                        (2)
solve(Goal) :- clause(Goal,SubGoals),solve(SubGoals).                     (3)
solve(Goal) :- selfEnhance(Goal),solve(Goal).                             (4)

% For selectivity use instead
% solve(Goal) :- extensible(Goal),selfEnhance(Goal),solve(Goal).          (4)
% Basically, it says that to solve a goal (prove it is true):
% (1) if it is true that is it, return the bindings
% (2) if it comprises a conjunction of goals solve them one by one
% (3) if it cannot be proved directly
%     find its subgoals and prove its subgoals,
% (4) if no more subgoals exist then try enchancing the database with
%     new information about the Goal, and
% if enhancement is successful continue solving for the goal.

% The implementation of selfEnhance is up to the designer.
% We just show a simple approach, but
% what we really want cannot be done in Prolog.

selfEnhance(Goal) :- enhanceTechnique(Goal,Technique),Technique.          (5)
selfEnhance(Goal) :- !,fail.                                              (6)

% To enhance the database for the Goal,
% (5) first find the specific technique for doing so.
%     If technique is found, execute technique.
% If technique runs successfully then return to solve the goal.
% (6) is added for safety reasons to prevent backtrack (this is a hack)

% Technique is a prolog program that attempts to gather
% more information related to Goal.
% The new information is asserted into the database.

% Knowledge about the knowledge environment
% Suppose there are known remote query ports, then
% a very simple program can use all these by brute force.
% Merely run some kind of query related to the Goal on all these ports
% and gather data to add to the database.

knownPorts(humanPort,[portOnMachineA,portOnMachineB,....]).
knownPorts(relDBPorts,[portDB1,portDB2,....]).

% "knownPorts" provides lists of ports of a particular type
queryAllPorts(Query) :- setof(PortList,(
      knownPorts(PortType,PortList),
      generateQuery(Query,PortType,QueryInstance),
      runQueries(PortType,QueryInstance,PortList)
      ),Dummy).

% this does not implement what we really want
% what we really want is to place queries and
% put a hold on this branch of the logic
% and while waiting for results to arrive from the remote queries
% we would like to proceed with other reasoning tasks such as on
% other branches of the search tree
% "generateQuery" creates an appropriate query for the port type.
% This involves a synthesis process which is not given here.
% It uses metalevel information on which arguments are extensible.
% "runQueries" may send a fill-in-the-blanks mail template to an
% address.  It might say "machine Z needs to know who the mother
% of Joe is ______"
% If any runQueries succeed, then queryAllPorts succeed, for example.

enchanceTechnique(parent(X,Y),
      (queryAllPorts((mother(X,Y)),assert(parent(X,Y)))
      ,queryAllPorts((father(X,Y)),assert(parent(X,Y))) )) :-!.

% This says that one method of enhancing the parent predicate is to post
% queries about mothers and fathers.
% If you use this already do not use other enhancement techniques.

% Sometimes we can even generate these queries automatically
% from rules such as
% parent(X,Y) :- mother(X,Y).
% parent(X,Y) :- father(X,Y).

enhanceTechnique(Goal,Technique) :- generalize(Goal,GGoal),
      set_of(SubGoals,clause(GGoal,SubGoals),ListSubGoals),
      eachQueryAllPorts(ListSubGoals).

% With generalize the query becomes less focused so we may or
% may not want to generalize. Subgoal queries may be posted together
% or separately. We have to be careful to recombine all the subgoals to
% obtain the proper predicate for assertion.

% example database

parent(bob,sally).                                                    (p1)
parent(joe,jane).                                                     (p2)
parent(mark,smith).                                                   (p3)
currentPrimeMinister(india,gandhi).                                  (pm1)
currentPrimeMinister(britain,thatcher).                              (pm2)

% rules
parent(C,M) :- mother(C,M).                                           (r1)
parent(C,F) :- father(C,F).                                           (r2)

% metalevel information about predicates
extensible(parent(X,Y)).
extensibleOn(parent(name,name)).

% information about the environment
knownPorts(humanPort,[portX]).
knownPorts(relDBPort,[portY,portZ]).

% we shall assume that currentPrimeMinister is non-extensible
% although in reality it is extensible on the first argument
% ie. extensibleOn(currentPrimeMinister(country,name)).

% query

?- solve((parent(joe,X),currentPrimeMinister(britain,X))).          (q1)
```

Of course, this does not work as we wish in the usual Prolog; that is, it is not particularly satisfactory. What we want is to use all the channels at our disposal to make our queries more or less simultaneously, gather the results incrementally, and assert the results in the database for use in solving the goals as new information arrives. While these queries are running off-line, we want to be in full control of the logic engine so it can process some other queries or branches of the search tree while waiting. Furthermore, anytime we obtain some new information, we can proceed with solving the goal. If we are not careful how we do this in this particular program, we can loop indefinitely.

For relatively sequential execution, the above query is meant to proceed approximately as shown in Figure 24.

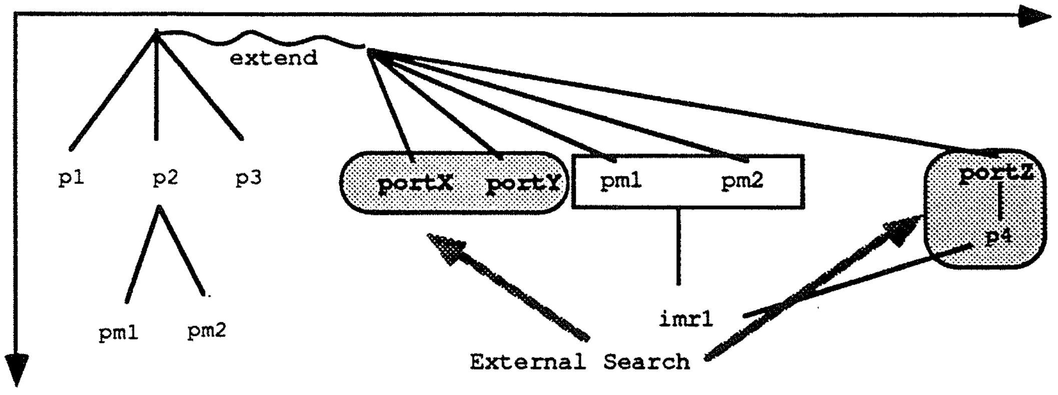


Figure 24. Progress of Query

One possible sequence of execution is as follows,

*p1 → p2 → pm1 → pm2 → p3 → no more predicates, generate queries and post queries to portY and portZ possibly using mother and father → use pm1 and pm2 to derive intermediate result imr1 → if reply has not been received yet post query to portX → if still no reply suspend query and work on something else → reply from portX revives query → asserts new predicate parent(joe,tom) ... (p4)→ use imr1 → timeout on query, indicate findings "has determined so far that parents of Joe are Jane and Tom and neither one is the current prime minister of britain, search on databases at portX and portY in progress, do you whish to continue?" and terminate if the user does not wish to continue as in this case. Indicate to databases that it no longer needs the information.*

Note that, since databases at *portY* and *portZ* took too long to reply, information in these databases is not used. Another interesting point to observe is that the system does not wait for the reply before working on the determination of the prime minister. Generally, query processing or other activities should continue while the database awaits extension.

It is possible to obtain more concurrency in the execution of such queries. The system might immediately recognize the parent predicate is extensible and immediately place queries before proceeding with the internal database search. This can be done by moving statement (4) to the head of the program. It might make sense to do it this way sometimes because of the time needed to obtain replies for such external queries. This is a preemptive system in which search in the environment is not triggered by database failure but by mere knowledge that something is extensible.

The blackboard architecture tends to support these desired capabilities and program behavior better. In the parallel blackboard architecture, each remote query can send its results to the originating blackboard where the results are incorporated into the blackboard. The posting of new information on the blackboard triggers further computation, a particular result of which can add information to a list of parents of Joe found so far.

We have not exactly addressed when these queries end. How do we determine when we should stop the extension of a database? Perhaps the way to set it up is that once the user indicates that he is no longer interested in the answer to the query, the query ends. A simpler way is to terminate the queries when a partial answer has been delivered to the user. The database might inform the user that the partial solution to his query is that Jane is one of Joe's parents. If it is even more intelligent, it will know that there is another parent to be expected, that is there should normally be two solutions to this particular query, and that one is insufficient. An even more intelligent system will check that the parents found are comprised of one male and one female based on knowledge about the conditions for completeness of the parent predicate.

The remote queries may be terminated by temporal decay — that is, if they are not answered after a while, they are destroyed. Interest in the results of these remote queries can also decline and die with time and results that come too late will be ignored.

The technique of database extension can be applied to relational databases as well. Suppose we want to make a join between two relations, one of which is extensible. Barring a few difficulties, the method of extending such a relation can be very similar to what we have seen so far concerning Prolog queries (Figure 25). Meanwhile, the join can proceed while the extension is being done. If new information arrives after the initial join operation is already complete, an incremental join can occur and the new results will be propagated through the system (assuming the join is part of a larger query).

There are various issues involved in this. One might need to keep around some of the intermediate joins in anticipation of new information in order to save recomputation.

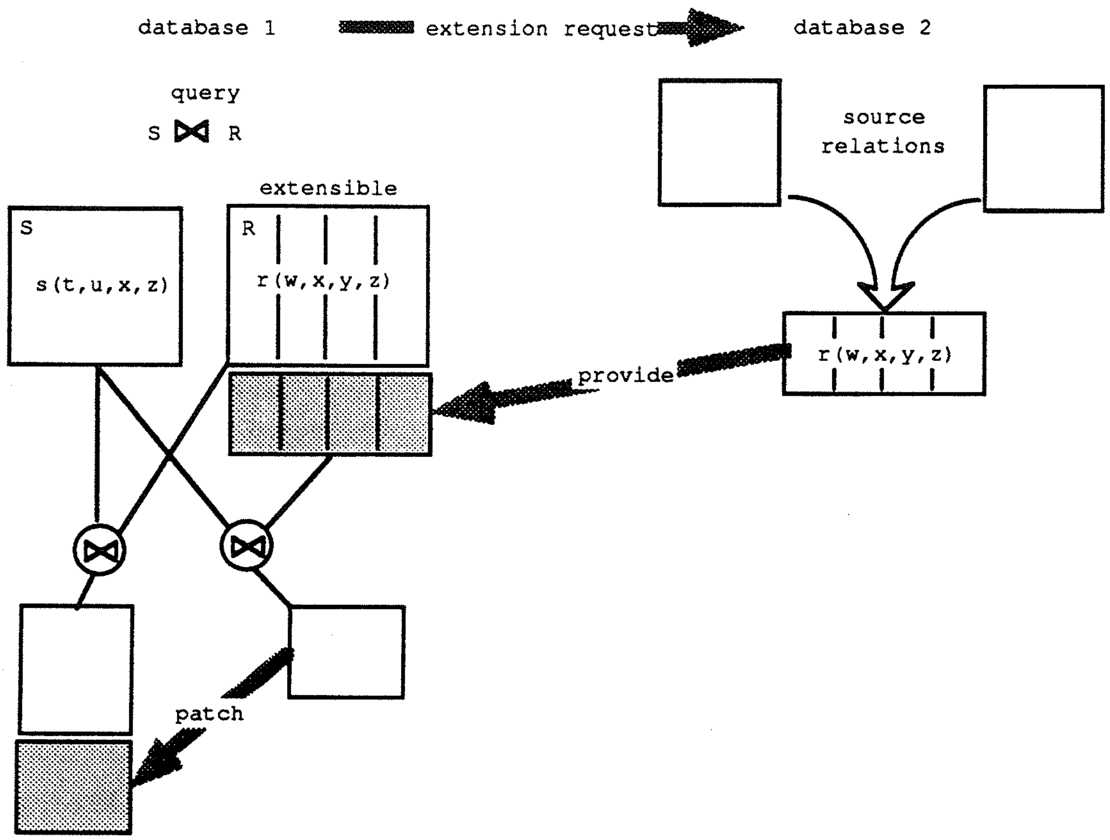


Figure 25. Example of Database Extension

In our conceptual model, various degrees of intelligence in the database extender are possible. A full-blown environmentally intelligent agent may be used to encapsulate the database. Such an agent is described in other parts of this thesis.

On the other hand, if the database exists within an agent, then the database extender could use the environmental intelligence of the agent and only have to worry about implementing simpler open-world semantics. All those portions about query generation and customizing and managing transactions with entities in the environment can be delegated to the more intelligent capabilities of an agent.

### 6.2.2. Partial Program

We may regard a program as a decision-procedure. If the decision logic of a program cannot find a solution to a problem that exists, then it is a partial program. For instance, if the parent query program always ignores alternate parent predicates, then it might miss a solution for a parents query even if that solution exists. Such a program is only a partial program and can be improved to take into account the other predicates.

We specify the program and specify the data on which the program operates. It is often the case that the data may change while the program remains the same. On the other hand, it is possible to conceive of decision-procedures that get smarter by obtaining and absorbing other decision-procedures from the environment. The programs that operate over the data change. It is not easy to see how to build such programs.

We can draw on the research on learning. We will distinguish different types of learning by what happens during learning.

i) New data, program does not change, representation of data does not change
ii) Representation of data changes, program changes to match
iii) New capabilities added, old capabilities remain unchanged
iv) Old capabilities changed

In the previous section we have already seen how new data from the environment can be actively brought into play in a program. This is a very controlled procedure since no new predicates, rules or schemas are added. However, the total coverage of particular extended predicates with respect to the global database have been improved through the absorption of information from the environment — the system knows more about parents after database extension than it did before. It is demand-driven in that the extension procedures are activated only when the database runs out of the required predicate. It is possible to develop techniques to generate queries intelligently to obtain information from the environment. We know of some techniques to assimilate the new information into a database framework that already exist. Relational databases can be extended in a similar fashion to have controlled self-enhancement capabilities.

Is it possible to extend the concept to knowledge-base extension as depicted in Figure 26? This is not a straightforward extension of the techniques employed earlier. For example, how can rules be learned from sources of knowledge in the environment? How can we generate queries to agents to help us learn the rules we would like to learn? How can these rules taken from the environment be assimilated into the knowledge base? These questions cannot be easily answered.

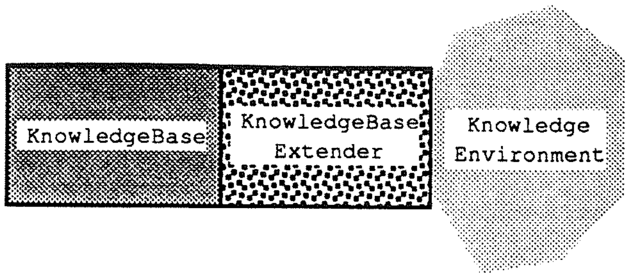


Figure 26. Self-Enhancing Knowledge-Base

In order to explore these concepts further, let us work on the parents example. The program as we have presented it before is a partial program. This is because it limits itself to the ports and databases that it knows about initially, which is merely a portion of the environmental search control logic. Its environmental search procedure is restrictive and not smart enough to take advantage of other available ports and databases. How can such a program self-enhance? The parents program can self-enhance by seeking and absorbing more knowledge about ports and databases and using the new knowledge to help it answer the query. By doing this, the program is actually extending its search procedures.

We will look at another problem. When does a program know when to self-enhance? How would a program that uses search know when to try to bring in and incorporate more sophisticated and efficient search procedures? We suggest two situations:

i) procedure timed out on task
ii) poor performance on similar task relative to other agents

This is not so much about learning classification rules from examples — which is creative learning — as it is about absorbing knowledge processing techniques and rules from other agents and incorporating them into one's own reasoning. By having both the ability to enhance the data that it processes and also to enhance the techniques of decision-making, an agent that can actually move from one specialization to other specializations.

Knowledge-base extension is much more difficult to achieve. Fortunately, we can already obtain a lot of power and flexibility with just database extension. By such self-enhancement techniques and using environmental intelligence, a machine agent can tap into some of the knowledge in the environment. We will continue with the discussion on environmental intelligence in the next chapter.

# 7. Architecture and Design of Agents

## 7.1. Environmental Intelligence

Knowledge about the environment and how to reason about, manipulate and use it — being intelligent about the environment (as opposed to just performing some specialized task) — is knowledge that has great potential power. Future automation agents will not work in isolation but as productive entities in an environment of other entities. By providing machines with environmental intelligence we bring them to a cognitive and dynamic level closer to the human operational sphere (see Section 3.4 for a discussion). This reduces both the difficulty and the amount of human effort needed to assemble and operate automation systems.

There are two aspects to environmental intelligence (refer to Section 2.1). The first aspect is the intelligence to function in the physical environment. The second is intelligence to work in the knowledge environment. Both are important, although this thesis does not deal with the first aspect.

It is useful to think of them separately even though they are not independent because many of the items in the knowledge environment have physical structure. Animals such as lions, antelopes and monkeys have a great deal of intelligence to function in the physical environment but little intelligence to function in the knowledge environment, which is mostly a human creation.

Researchers have been working on developing reasoning systems that have some environmental intelligence. Motion planning is one such area. Visual sensing and interpretation is another. Control of effectors in the physical environment is yet another. Agents need all these types of intelligence to function effectively in the physical environment.

What we propose here is for automation systems to be given substantial intelligence to exploit the knowledge environment as well. To to that we first conceptualize what this environment is, how it operates, what is important about this environment, what are some of the challenges that agents working in it face, and how they might deal with these challenges. Chapters 4 and 5 mainly conceptualize the knowledge environment. Chapters 6, 7, 8, 9 and 10 present some ideas and concepts to help realize environmental intelligence. But they represent only a preliminary treatment. Many of these ideas and concepts await more extensive investigations in future research. Chapter 11 discusses application of some of the theory to construction automation systems.

## 7.2. Triggering Use of Environmental Knowledge

One of the problems related to the use of environmental knowledge is how does the agent *know* when to use it. One way is to provide the proper semantics and mechanisms to the internal database, knowledge bases and programs that the agent uses.

### 7.2.1. Conceptual Scheme

We saw in a preliminary way how to extend data in Chapter 6. We employed the technique of characterizing the data. First, data is classified by topic. We say more or less "all this data is on this topic." Then the particular body of knowledge about each topic or group of topics can be characterized. For example, we can say that the data about this topic is *extensible*. It means that we only have partial data. Assertional information is usually extensible. This is a meta-level description of the knowledge that we have on the topic and it is explicit. It is a type of self-knowledge since we are essentially describing our own knowledge of the topic. By making this self-knowledge explicit, we can reason about the knowledge that we have, and thus better devise strategies on how to use our knowledge. This type of meta-level description can be applied to databases and knowledge bases as well, but more research is required.

Characterizing our knowledge is not enough; we also need operational strategies. Numerous strategies are possible and each has its advantages, so we would not like to be locked into any particular one. [Genesereth and Lenat 80, Konolige 85] presents some formalisms for representing and reasoning about self-knowledge. If we are sophisticated we might wish to explicitly describe each strategy and the techniques of evaluating it. But we will be simple-minded for the time being.

One approach we might take would be to immediately start the process of obtaining new information from the environment on the rationale that the environment might take a long time to provide the information relative to the time scale of our reasoning processes. While the environment is busy trying to fulfil our requests, we proceed with using our knowledge. This is not too smart because it will tend to produce environmental search queries for which information was not needed.

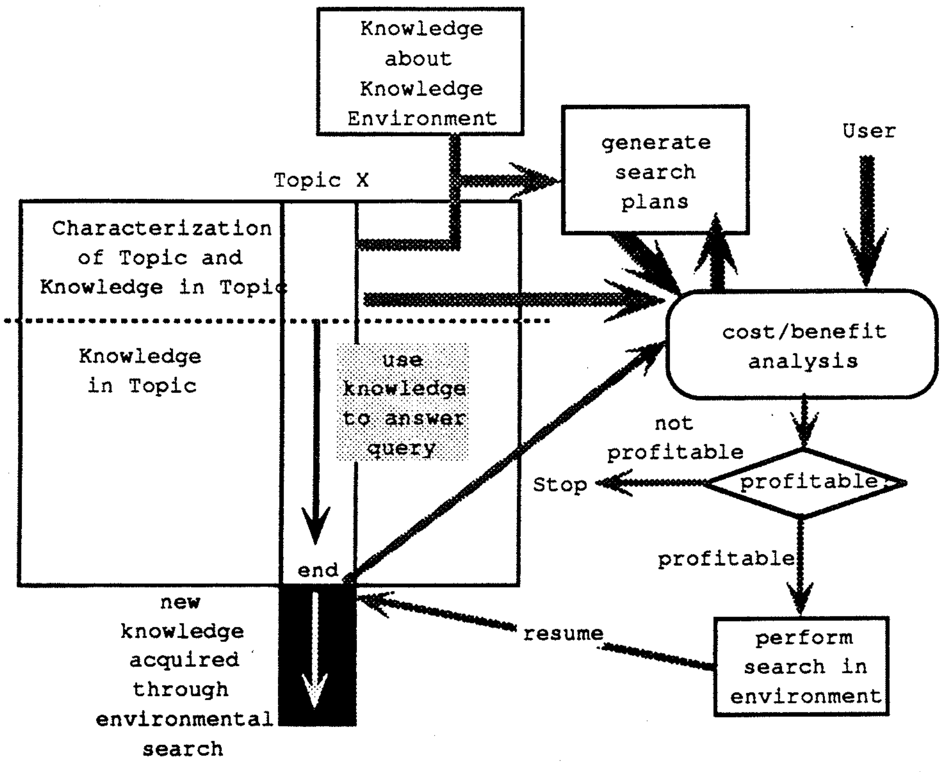

Knowledge about Knowledge Environment
generate search plans
User
Topic X
Characterization of Topic and Knowledge in Topic
Knowledge in Topic
use knowledge to answer query
cost/benefit analysis
not profitable
Stop
profitable
profitable
end
new knowledge acquired through environmental search
resume
perform search in environment

Figure 27. Conceptual Scheme of Failure-driven Environmental Information Search

Another approach would be to keep track of how much of the data have been used; when we have used all that we have, we start looking for new information. This is failure-driven use of environmental intelligence. An extension of this scheme for reasoning in machine agents is depicted in Figure 27. The rationale for the use of failure-driven schemes is that we should use the information that we already have first because it costs the least to use, while environmental search consumes much effort, time and ultimately costs. This approach is smarter than the first because we have already exhausted our knowledge and we surely know we need information from external sources. However, if we proceed in a straightforward manner we will tend to incur high environmental search costs which may well be disproportionate to the benefit derived from having a better answer.

We would like to be circumspect about where in the environment we look for information, if we decide to do so at all. Section 7.3 discusses the notion of graded reachability. If the agent has knowledge about the knowledge environment, and knowledge about the topic of query, it can be circumspect in generating search plans which can then be evaluated and screened before execution. Execution of knowledge environmental search results in the augmentation of the knowledge on the topic which can then be used to answer the query on the topic. Note that we do not do much analysis when using knowledge we already have because such analysis is likely to consume disproportionate amounts of computational resources relative to benefits derived. But, where very large internal knowledge bases exist, such analysis might be beneficial.

Cost/benefit analysis is an important part of the conceptual scheme and is one of the points that distinguishes this scheme from work with distributed databases and knowledge bases, especially transparent distributed databases and knowledge bases. It is anathema for the latter that the same query may result in different answers. The most they will do about costs is to either solve the query completely or do not work on the query at all, which is appropriate because they treat databases and knowledge bases as tools. In developing machine agents, however, we should proceed differently since they are resource-limited.

### 7.2.2. Conditions and Mechanisms for the Use of Environmental Knowledge

In the previous section we have mentioned failure-driven use of environmental knowledge. When an agent has exhausted use of its own knowledge, it seeks knowledge in its environment. What we have seen so far is rather limited. In this section we will discuss a few other conditions where one may like to use environmental knowledge. Some mechanisms to support the use of environmental knowledge are also proposed.

One condition that springs to mind is the lack of quality of the information you have on a topic. If you know that the quality of the information you have on a topic is poor and there are more reliable sources of knowledge on that topic available, then the smart thing to do is to use those sources. If the source is a knowledge object, then one would probably try to obtain knowledge from it. In that case it might even be best to discard one's previous knowledge! If the source is an agent, then there are other options, such as ask it to solve the problem.

How might you know or guess that a particular body of knowledge is poor? One way of knowing that would be if some other party has characterized the knowledge base as poor. There is also the likelihood that if your knowledge base on the topic is small relative to other knowledge bases, then your knowledge base is poor. One might also learn gradually from feedback that a knowledge base is poor because the performance of the knowledge base has been poor. Failure to derive some conclusion might also be indicative of inadequacy of a knowledge base.

Lack of knowledge for reasoning might also occur. For instance, it might only have been possible during the limited time available to implement part of the knowledge base of an agent. In behoves the implementor to state the condition of the implementation and some way for the agent to realize such conditions and take the appropriate actions.

To be alerted to the poor quality of your knowledge on a topic, one might employ an internal complainer. A complainer does two màin things. During idle times it analyses the use and performance of the knowledge of the agent, generating meta-level information about the knowledge that the agent has. When the agent uses knowledge that is adequate, the complainer does nothing. If there is some potential problem with the knowledge, such as the poor quality of the knowledge, then the complainer interrupts the reasoning processes so that deliberation about the knowledge can proceed first.

Of course, complainers are rather sophisticated mechanisms and other simpler approaches can be used. One such mechanism is the trip-wire. Trip-wires are essentially code embedded in the knowledge base that takes the process of reasoning out of its normal progression. The *trip* mechanism is very similar to the *cut* mechanism of Prolog. The *cut* mechanism causes termination of various branches in the search tree. *Trip-wires* do not necessary cause termination of branches of the search tree, but rather pause the base-level reasoning process for meta-level deliberations. Different types of trip-wires can be devised and used. Note that trip-wires are not really a part of the logic of the problem-solving process in which they are embedded whereas the cut is. Some psuedo-Prolog code with embedded trip-wires might look like the following:

```
% Example use of trip-wires
plan(Goal,Plan) :- subgoals(Goal,Subgoals),
                     trip(checkTopicKnowledge(Subgoals)),
                     trip(checkExecutionTimes),planAll(Subgoals,Plan).
```

```
% After determining subgoals of goal but before continuing with the
% planning it might be wise to deliberate on the knowledge available
% to solve the problem and execution time needed for subsequent
% planning.  This might take into account all the present
% branches of the search search tree that is to be investigated.
% The decision might be to completely stop the planning.
```

Other simple mechanisms are timers and counters. One way of using timers is to set the amount of time for some operation or task. If time-out occurs, then some review may have to be made to decide whether to commit further computational resources.

It is not always the case that one wishes to check the quality of the knowledge that one uses since the process consumes computational resources. However, you want to double check whether it is critical to have correct information. The agent might recognize some categories of problems to be critical ones. Alternatively, a user might state that the given problem is critical. In an economically-based reasoning system, the amount of benefit to be derived for the solution of the problem or performance of the task could be a good indication of the importance thereof.

Other mechanisms that might be incorporated include partial-state observers, monitors, watchers and scribes. Partial-state observers examine intermediate solutions. Monitors may be used to track the use of resources. A dead-end watcher might trigger when conditions indicate that the reasoning process has reached a dead-end. A cyclic watcher might trigger when conditions indicate that there may be some problems with the reasoning which causes it to cycle. Special watchers might be used to keep tabs on unusual conditions. If the machine appears to be malfunctioning, then such a watcher might alert the agent. Parameters that might indicate malfunction in a mobile machine might include cyclicity in visual data, unusual vibrations and high temperatures. Scribes record some salient features of the reasoning process and intermediate data structures for use in deliberations.

Special knowledge and procedures may cannibalize the partial solution to determine what knowledge to search for and special knowledge about the environment would be used to help determine how the search could be done.

## 7.3. Graded Reachability

There is a gradation in the difficulty and effort — and consequently the cost — to reach information in the environment. Therefore, it is sensible to use more easily obtained information first and, if the situation warrants, to look for more inaccessible information. This is depicted in Figure 28. The lines indicate information traffic from knowledge sources in the environment — the larger the line the greater the amount of traffic. There is, for practical considerations, almost an unlimited amount of information that exists in the knowledge environment. Much of it the agent might not even know about directly. Nevertheless, with environmental intelligence and help from environmental entities, it is not impossible to reach a very vast amount of information and knowledge.

Note that we are not only talking about the tangible monetary costs involved in extracting information, but also of other dimensions of costs such as the resources and time that are incurred. Quite often, even if the costs are relatively high, it pays to obtain the information.

For example, a structural designer who has discovered that the structural design code has changed would make the special effort to obtain and incorporate the new structural design code. This is not information that is locally available or readily accessible, but there are methods that an environmentally intelligent agent can employ to help it obtain that information.

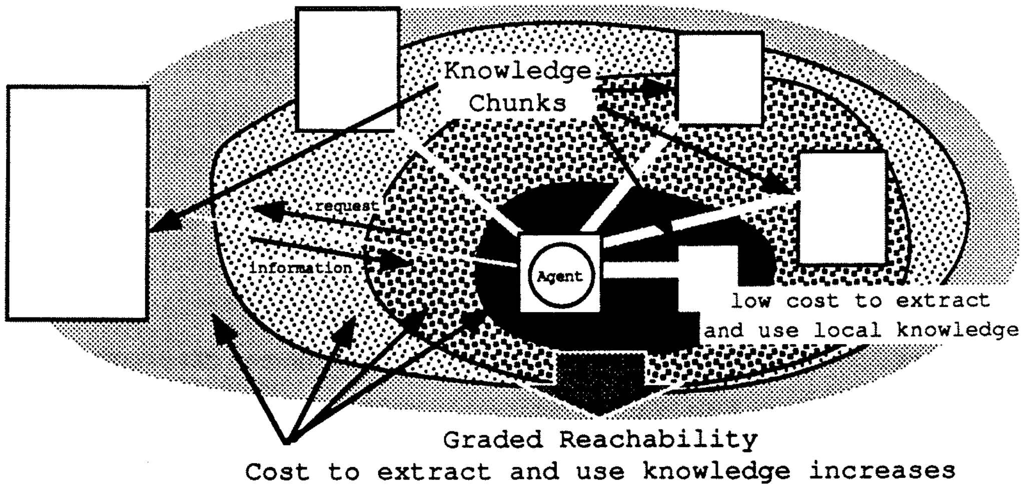


Figure 28. Graded Reachability of Information

The concept of graded reachability applies not only to knowledge but also to resources. So if an agent needs to obtain a tool not in its immediate possession, it would search the most readily accessible possible source first. If it also has likelihood or probability information available, it can use that to develop an even more effective search strategy.

One of the reasons to incorporate graded reachability in our design is that of spatial distribution. But there are other reasons. For example, the barriers caused by different representations may make information hard to obtain — not impossible — even if they reside in memory on the same machine.

Computing the cost of a search is not always an easy thing to do. Fortunately, it is not necessary to be very accurate about such computations. It is often sufficient to be able to get a rough estimate and use such estimates to quickly rank accessibility of sources relative to one another. Since the ability to effectively employ a limited agent's resources for search in the environment depends on reachability estimations, the agent ought to be able to refine such knowledge. It might keep track of the effort, resources and time used in search occasionally and update its knowledge. The strategies of environmental search might also need to be revised.

Suppose an agent reaches a point where it would like to obtain particular information. It develops plans to obtain the information from, say, three sources. Such plans may be lazy plans such as the ones described in this thesis (Section 7.5.2). It evaluates these courses for the costs, resources and time involved. Normally they are only very rough estimates unless there has been extensive experience and the environment is slow to change. The agent weeds out any untenable option. It might consolidate the estimates on costs, resources and time into a single effectiveness estimate. Then it might pick one or more such plans for execution. A simple selection algorithm is to pick those that rank the highest in effectiveness. A serial step by step approach for this might look like this:

Task to get information on X.

- formulate plans to obtain information on X → Set of plans
- evaluate plans for costs, resources and time requirements → Table of costs resources and time
- filter choices → Set of acceptable plans
- consolidate different requirements into an effectiveness metric → Rank of plans by effectiveness
- pick most effective plan → Chosen plan
- execute → New information
- inform of degree of success or failure

A partial pseudo-Lisp implementation of graded reachability aspects is as follows

```
;;; Graded reachability
;;;   getInformation returns information
(defun getInformation (Topic)
   (execute (select (formulatePlans (Topic)) ))
)
```

```
;;;   select returns a single plan
(defun select (Plans)
   (best (filter (evaluate (Plans)) Plans))
)

;;;   evaluate returns an association list of plans and estimates
;;; How evaluation is done depends on the type of plans generated,
;;; what kinds of knowledge can be used and what the implementor wishes.
;;;   filter throws away unacceptable plans
;;;   best returns single plan
```

Whereas in traditional programming we more or less treat all knowledge on an equal basis, environmentally intelligent machines might have to treat knowledge from different sources unequally. We have seen some measure of this in distributed database operation. In that case the database has been fragmented to different sites and techniques for selecting the type and order of database operations have been developed to reduce the costs of moving data about to answer database queries.

## 7.4. General Core for Environmentally Intelligent Machines

The idea here is that there is a significant amount of knowledge that can be used by different agents and we may be able to capture and modularize such knowledge. Specialized abilities and knowledge can then be added to this general core to enable the agent to perform specialized tasks. This way we might be able to reduce the cost of putting together automation systems. Figure 29 is a simplified conceptual figure to depict the process of assembling the knowledge of an agent.

We shall explain this figure using the analogy of a carpenter. The general core of knowledge, $G$, is the knowledge that the carpenter has but is also shared by the majority of the human race. Therefore, a banker or an architect has this knowledge also. We know that a carpenter has knowledge that many other agents do not have which has to do with carpentry. This specialized knowledge ranges from skills, $Sa$, to expertise, $Sd$. Before the carpenter is involved with the project he has the knowledge depicted by $S*$ and $G$. Indeed $G$ includes some knowledge that gets him on to the job site. Each job is different, so the carpenter has to add to his knowledge job-specific knowledge denoted by $Sp$ that allows him to apply $S*$ and environment specific knowledge $Ed$ so that he can use $G$ for that environment. Now agents do not work alone. The hexagon indicates that the agents are bound together by organizational structures while their interactions, indicated by arrows, are made possible by the general environmental knowledge.

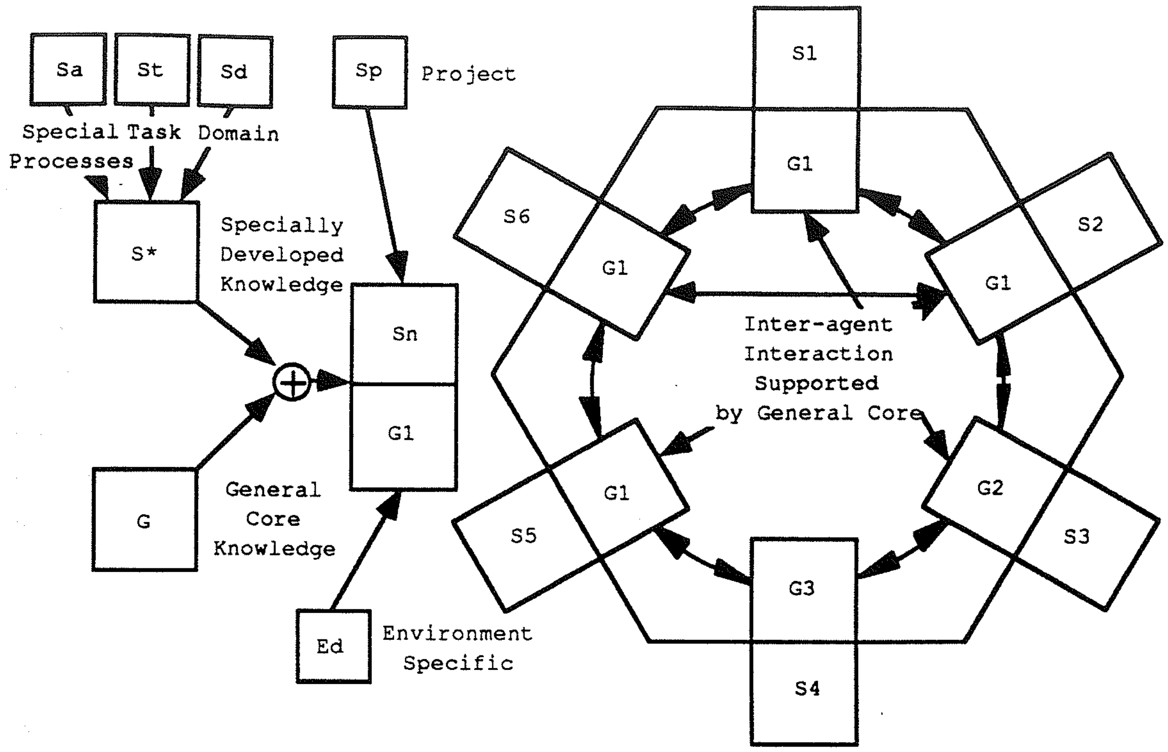


Figure 29. Assembling the Knowledge of an Agent and a System of Agents

There is more to the notion of a general core. It is called a core because we believe that the specialized knowledge of an agent is to be added subsequent to the provision of the general core of knowledge. We enhance the agent's general core with specialized knowledge rather than enhance the agent's specialized knowledge with general core knowledge. We believe that this way of assembling an agent's knowledge would prove the easier, but the extent to which this belief is applicable still not known. We also do not think that the two types of knowledge have an equal standing — that is, they are not peer level, general knowledge is more fundamental to an agent, whereas specialized knowledge does not affect the "survival" of the agent. We take our guidance from the way nature does it and from the way many operating systems are built, but this issue is open to debate.

The next pressing question is: what constitutes this general core knowledge that is so important to all agents? This is also very much an open question and there may be disagreements. We think that there is some knowledge central to the need for agents to deal with their environment and with themselves. Therefore, we should bring this knowledge into the general core knowledge. Most if not all internal resources ought to be under the control of the agent, so in many ways this central core constitutes a very powerful active operating system (Section 7.8).

One part of the general core, we argue, is self-knowledge. Self-knowledge is useful for automatic planning of the activities of the robot as well as for self-diagnostics. Self-knowledge is also needed for the agent to explicitly determine whether it can undertake various tasks. Self-knowledge is useful to determine the type and extent of the agents own knowledge and thus whether to obtain more knowledge from the knowledge environment. Such knowledge is also useful for internally managing an agent's own knowledge and computations. Agents also need to know about themselves in order to communicate the information about their knowledge and capabilities to other agents needing their services. Although it might appear that each agent is different regarding the specific content of self-knowledge, we believe there is some systematic way to describe and reason about self-knowledge that should be common to many machine agents. We demonstrated the explicit characterization of some of an agent's knowledge in Section 6.2.1 and Section 7.2.

A second part of the general core, we believe, is environment exploitation knowledge. With respect to environmental exploitation knowledge, we have argued in Chapter 2 that there are two aspects, at least conceptually, namely, the general knowledge related to dealing with the physical environment and the knowledge related to functioning of the agent in the knowledge environment.

Agents need the ability to interpret and navigate in both environments. We would include in the knowledge environment exploitation knowledge, knowledge for communication and for managing multi-agent transactions and relationships. We have not dealt much with either purely communication issues or multi-agent relationships. There is some discussion of multi-agent transactions in Chapter 5.

## 7.5. Implicit and Explicit Resource Control

[Horvitz 87] discussed the use of meta-level reasoning to deal with the problem of computational resource constraints. First, we have to break away from the requirement for complete results. Complete results require algorithms to run to completion. But we want to be able to interrupt the running of the program on a problem and obtain results useful enough to be the basis of decisions or actions. It would be desirable to have a system that generates results whose values increase monotonically with the amount of computation time expended. Two current approaches to deal with the problem of bounded computational resources are, according to Horvitz (Section 7),

i) *the development and characterization of intrinsically flexible inference strategies, and*

ii) *the mastery of techniques and computational architectures for efficient decision-theoretic control.*

Among the intrinsically flexible inference strategies that are under development are probabilistic inference approaches which include stochastic simulation, bound calculation and completeness modulation. Other approaches include abstraction modulation, local reformulation, default reasoning and pre-compilation [Horvitz 87].

There has not been much progress in developing techniques and computational architectures for efficient decision-theoretic control. We think that it may be possible to tradeoff the generality of problem-solving for incremental problem-solving. Various techniques fall into a spectrum of possibilities. One point in the spectrum that has high potential for incremental problem-solving but needs very specific problems is represented by the parallel distributed processing or neural network computation paradigm. These problem-solving techniques are based on propagation and relaxation techniques.

In the following sections we will discuss some aspects of planning, resource use and execution. We shall look at the horizon phenomena in planning and reasoning in the next section.

### 7.5.1. Planning Horizons

In planning, human agents limit the consideration of activities to a certain range of time. They also limit the number of activities considered, the detail to which activities or information are considered, and the extent to which uncertainty is taken into account. We list them below:

i) time
ii) number of activities
iii) granularity
iv) contingency

An implicit restraint is thus placed on the resources that an agent applies for its deliberations. Doing so, however, sacrifices guarantees concerning the result.

Agents that function in a dynamic environment do not plan far into the future because changes in the environment might invalidate the plan thus waste the planning effort. The further into the future the execution time of the planned activities are, the more likely it is for changes to occur that might affect it. The planning done by human agents has elements of all four types of planning horizons.

The horizon effect can be created by explicitly stating the horizon threshold. In reality, just as the actual horizon continues moving away even as we approach it, there may not be fixed horizon threshold. It appears that there is a horizon because agents expend their limited resources on things that are usually the most relevant to them, such as the things in the future closest in time to what happened to be the current time. For the purposes of this thesis we shall take a simple view and think of the agent as recognizing particular thresholds within which it confines its planning activities.

We introduce the notion of time binding for single time-stream agent. If an agent is free to use its time as it wishes, then we say that the time of the agent is unbound. If the time is committed to the performance of some activities, then the time is bound. Commitment of time that is not fixed to a particular block of time is bound by amount only. An agent should not commit more time that it has available. It can only add new time commitments if there is sufficient unbound time.

In this model an agent is willing to commit to activities of a plan if they fall within the agent's time planning horizon, but not beyond. As there is no commitment to plans made beyond the time horizon, these tentative plans tend to carry little weight and get little attention. However, there may not be one universal time horizon on all activities, but typically different time horizons for different classes of activities. We are familiar with long, medium and short range planning, each of which have different horizons. Of course, such horizons can be and are occasionally revised. When tentative plans fall within the new horizons they may be seriously reconsidered.

Planning with a time horizon is quite typical of human agents and organizations. Governments may have a ten-year planning horizon for almost all activities, while small commercial firms have one to five-year horizons on their planned commitments. A human might have only a few seconds planning horizon for the movement of his arm during driving. It does not seem appropriate to plan the activities of the arm for even a minute. Commitments will not be made on matters beyond this horizon, although discussions may be held and tentative plans may be made. However, time marches forward and horizons are frequently revised, bringing new time blocks into consideration.

Figure 30 shows simple time allocation for an agent with a time planning horizon. Such an agent will only consider activities for execution if the activity falls within the planning horizon. The amount of time for the new commitments should not exceed free blocks of time available within this horizon.

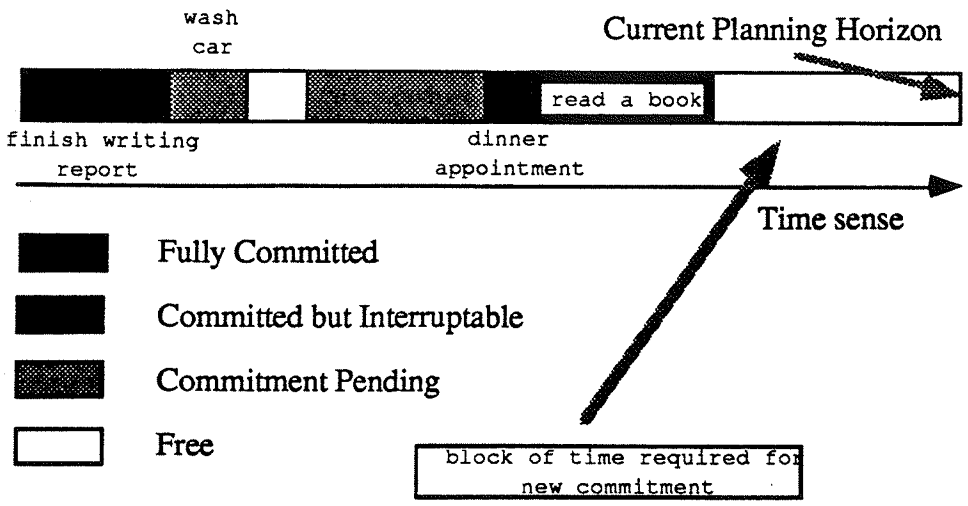


Figure 30. Simple Time Allocation for Single-Time-Stream Agent

A subcontractor who can only perform one job at a time would have a similar method of allocating time. Various jobs requiring various amounts of time are fitted accordingly. If the job exceeds the time available, then it is either rejected or the time horizon is extended to accommodate it. This is one way a subcontractor can allocate time for jobs at various sites. By fixing the actual times, both the subcontractor and the contractor know for sure when the work is to be done. The commitments are meaningful because of the time horizon acceptable to both parties.

Not surprisingly, we also see such allocation practices with use of other resources besides agents. Booking places such as convention halls and rooms for various periods of time before use is common practice. People never book very far into the future, although the time horizons may stretch to a year or more for popular places.

Of course, the notion of time horizons does not dictate the way the time is to be used within it. The type of allocation practices we see above are rather simple and quite typical of the real world. The full power of allocation will only be realized with the use of general constraints of which time binding is but a specialized constraint.

An agent will not plan too many activities, producing an activity horizon in planning. Of course, the granularity of the planned activities is important. Even when there are activities of different granularity, the total number of activities "on hold" is limited. Humans generally limit the number of high-level activities on hold to fewer than two dozen. Perhaps several meetings, tasks, or shopping activities might be planned for a day. There are many other activities that the agent would perform during the course of the day, but planning for them is more or less on demand. Machine agents might have a higher limit but would still be subject to an activity horizon because of resource limits.

An agent will not generate very detailed plans at an early stage. The detailed plans are developed later. There is a horizon effect on the granularity of planning. The further into the future the planning concerns one, the larger the granularity of the planning is likely to be. Long-term plans contain high-level goals. In medium-term plans we begin to see concrete targets and activities. Very short-term plans are full of activities.

An agent does not develop plans to meet every contingency. Agents limit the number of contingencies they consider during planning. It is not necessarily the case that they do not know about possible contingencies, but they do not usually plan for them. Out of ten possible worlds, the agent might only plan for one or two. Clearly, then, they may not have planned in advance for some actual event. Such situations are dealt with as they arise. The major problem with planning for too many contingencies is the combinatorial explosion and its consequent expenditure of effort, most of which will be unprofitable.

### 7.5.2. Lazy Planning

Basically, the idea of lazy planning is to plan later rather than earlier, and it is not as foolish as it might sound to the uninitiated.

In a dynamic environment, where even the direction of change may be uncertain, events may occur between the time of planning and execution that could influence the plan or even invalidate it. This interposition of events between planning and execution militates against early planning. This is depicted in Figure 31. We indicate several concerns:

i) The plan is not valid at time of execution — it no longer achieves the desired goals
ii) More information has become available to construct a better plan
iii) The goals have changed

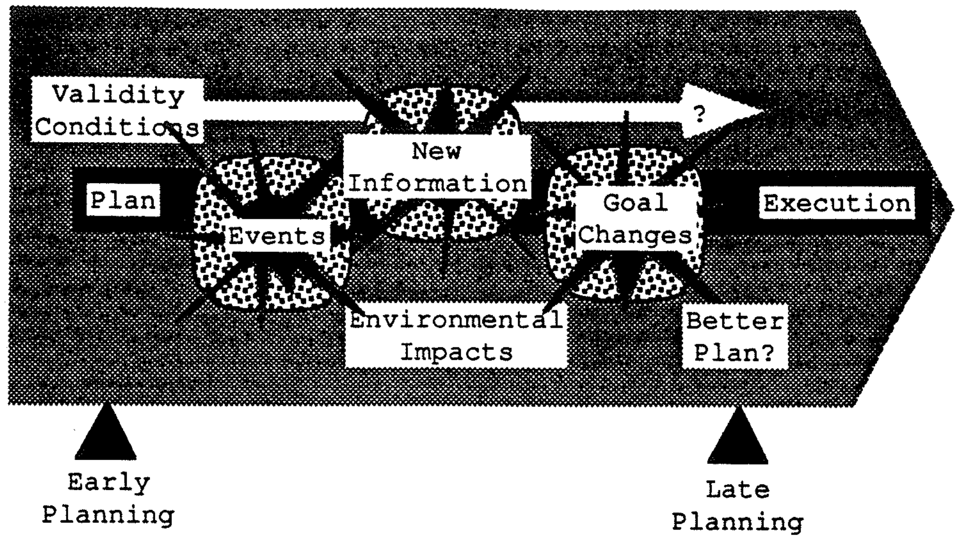


Figure 31. Early vs Late Planning in a Dynamic Environment

The first one is a serious problem, but we may be able to live with the second. The results may be undesirable if a plan constructed earlier, but which is no longer valid, is executed. As we have seen, one approach that attempts to deal with these problems is to build conditional plans. The agents execute different branches of the plan depending on which conditions have been observed. It might be necessary for the agent to keep track of the state of the world as the plans are executed.

The third is a dependency condition. The desirability of executing a plan depends on the goals that the agent has at the time of execution. If the purpose for which the plan was constructed no longer holds, then the plan should be shelved.

We see that the problem here is cost. Developing plans requires effort and conditional plans require even more effort. If the plans are useful, then the planning effort is justified. But it seems that in dynamic circumstances plans have to be discarded or modified and so the utility of early plans declines. There is a depreciation in the value of plans the greater the temporal separation between the time of plan construction and its execution.

Agents deal with this problem by stretching out planning temporally, in particular, postponing to a later time some of the planning. Pieces of plans are built and modified over time. To understand what planning is done early and what later, we ought to know why agents have to develop plans before they are executed. Why does it not work for agents to plan only the things they would do immediately?

Temporally stretching out planning does not mean that planning is just spread out over time in equal intervals or some other pattern. Indeed, plan expansion is frequently linked to the time of expected execution. Let us say that on August 1st you are introduced to Joe and he invited you to visit him on August 10th. His place is in the suburb of the city. It seems reasonable that on August 1st that you plan to drive to his place. It is not likely that you have planned the route then. Even though we may not have much change in the environment, human planning still tends towards lazy planning. Possibly on August 9th you start planning the route. Even then you do not have the most detailed route possible. You leave the most detailed planning and decision-making to the last moment — when you are actually driving and can observe or obtain information about the conditions of the freeways. If the freeway you planned to use on August 9th has heavy traffic or is blocked because of a traffic accident, you plan another route. The marvelous thing is that if on August 8th Joe changed his mind and cancelled his invitation, you have not spent any effort planning a route to his place or worrying about the conditions of the freeways!

We believe that the nature of early plans and later ones differs. A plan such as "drive to place X" has much temporal stability but, of course, the conditions that allows for its execution can change. For example, the car might be needed by your wife on August 10th. Several reasons might be given as to why you constructed such a plan on August 1st:

i) The likelihood of events occuring invalidating the plan is low.
ii) It is cheap to construct such a plan.
iii) It is easy to keep track of such a plan.

We see that all these reasons are economically motivated. Therefore, lazy planning can be regarded as an implicit method of controlling the reasoning and memory resources of an agent. A possible lazy plan for installation of a window is depicted in Figure 32.

In this plan we have embedded a construct (shaded oval) which is a meta-planning action. Unlike the conditional plan in Figure 5 in Section 3.4.1, the planning action has not been done yet. It awaits the *need* to do so which arises when the robot discovers that the window is broken. At the time of generating this plan, the planner realizes that there may be a need to plan further but does not do so; instead, the planner makes a note to the effect that further planning is needed in the plan itself. If, during execution, the execution process for this particular plan stumbles across such a "note", then actual planning activities are instituted and patched into the existing plan. Run-time planning capabilities are required.

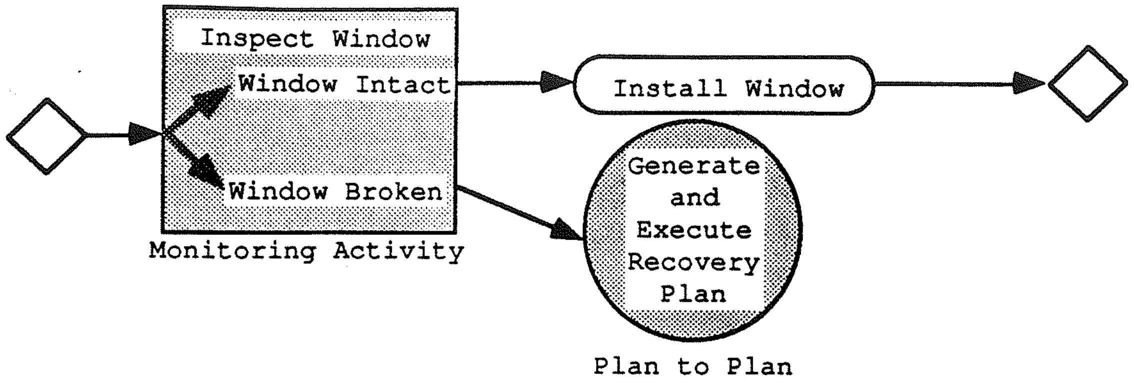


Figure 32. Simple Lazy Robot Plan to Install Window

A more flexible approach is depicted in Figure 33. In this figure there are two types of plans, the basic plan and the monitoring plan. The basic plan comprises executable actions or, in the parlance of construction management, those activities that create value. Monitoring plans mainly comprises tokens which can be traded in for further work or value. The two plans may have some bindings between them. In certain cases, with the appropriate bindings, the monitoring plan can serve as the control plan for the basic plan.

Before the installation of a window can be done, a "watchdog" discovers that there is a binding to the monitoring plan token to "inspect window". The token is traded in for further work or value. This is like an agent having a voucher to spend at that point in time. It says, "This is what I would like done now as part of a continuing plan of action and please figure out what to do." The install window activities are held off until the matter is settled. In effect, the token is used to launch the agent's general inspection capability.

When activated, this process may generate an inspection plan and the meta-control action "generate and execute recovery plan." Note that the inspection plan did not exist when the basic production and monitoring plans were created. Only at a time much closer to the time of expected installation of the window was the plan to inspect generated. The type of plan generated can depend on the current sensory and resource availability and criticality requirements. In this particular case, the inspection plan may be to run a daemon process that does an inspection every 10 seconds for the condition of the window while the install window activities are being executed. This daemon process terminates after all install window activities finish. The time of planning is at the time of need. The monitoring action "inspect window" is a token that holds the *promise* of an inspection plan but is not the inspection plan itself.

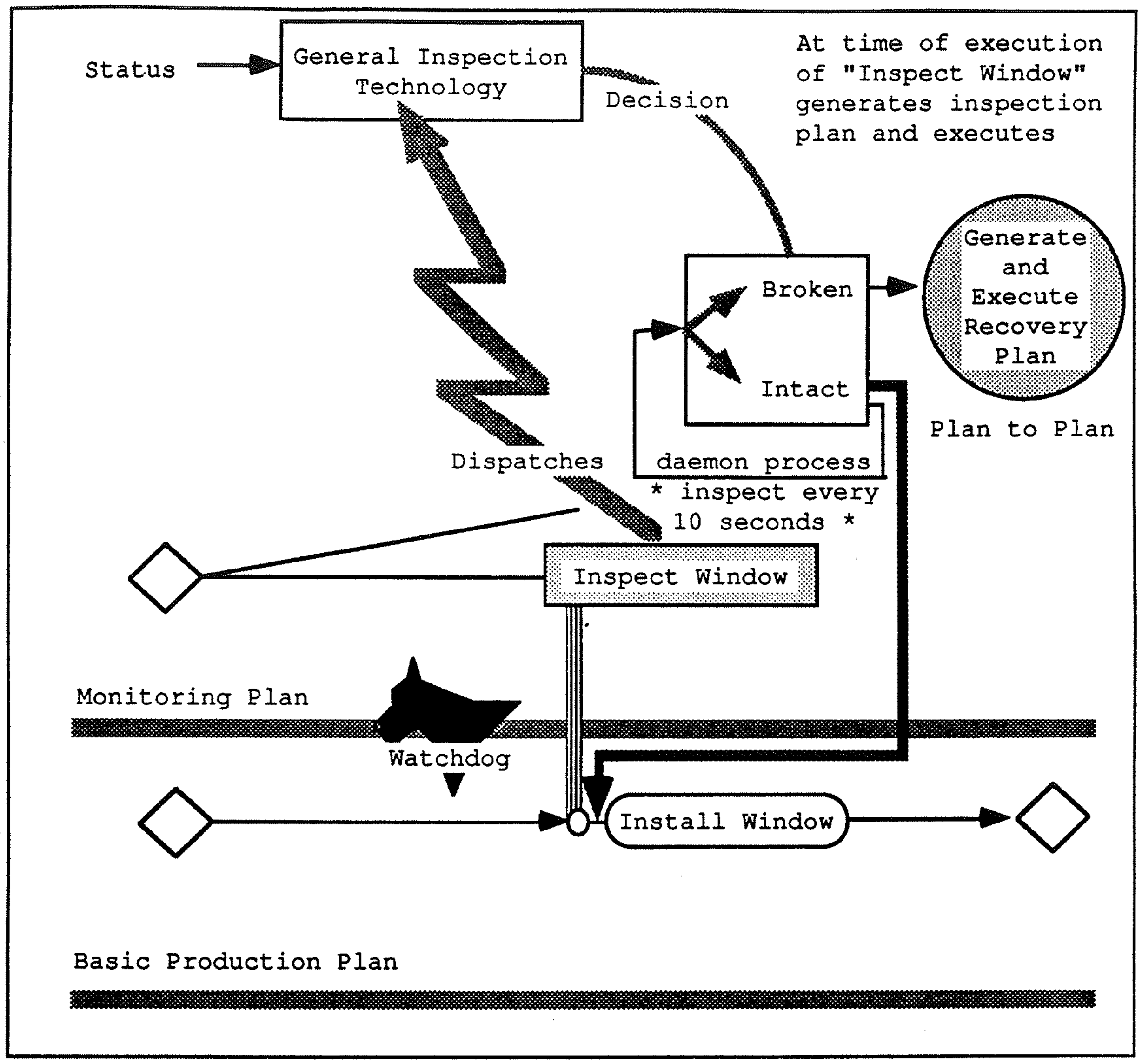

Status
General Inspection Technology
Decision
At time of execution of "Inspect Window" generates inspection plan and executes
Generate and Execute Recovery Plan
Broken
Intact
Plan to Plan
Dispatches
daemon process
* inspect every 10 seconds *
Inspect Window
Monitoring Plan
Watchdog
Install Window
Basic Production Plan

Figure 33. A More General Lazy Robot Plan for Window Installation

Note that this approach has potential strengths that the earlier lazy plan of Figure 32 might lack. The sensory resources of the agent might be limited and a contention for these resources is possible. The earlier lazy plan arrogates to itself these resources whereas in the latter approach decision-logic can be used to select the allocation of such resources at run-time.

The generality of such monitoring plans is greater than it appears from this example. Complex monitoring structures and meta-plans can exist while the basic production plans remain relatively simple. Indeed, multiple production plans might be in various stages of execution at the same time. It is possible to extend the scheme to deal with goal changes which can invalidate parts of or even entire production plans. It is possible to build monitoring plans lazily just as it is possible to build any other plans lazily.

We suggest that in human planning much of the tokens of abstract planning are really anchors or guide posts rather than goals. The are referential constructs that are combined with a methodology that permits agents to focus their attentions. The tokens are semantically meaningful but they do not merely expand to actions. An agent executing a referential plan changes his focus in a controlled fashion. The token in effect nudges the agent into a frame of mind. The referential plan is meant to put the agent on a path that hopefully leads to the attainment of the agent's goals. This also means that there can be many ways of abstraction and several abstractions can be used simultaneously — that is, an agent can hold and consider several abstract plans about the same potential set of activities for the same goal. These anchoring points or tokens have goals and reasons behind them. These goals and reasons constitute some of the rationality conditions of the abstract activities. If these goals no longer apply to the situation, then actions related to it may be dropped. Several such plans may compete for the agent's limited resources and time at any point.

The notion that we can embed meta-level actions into a plan at the time of planning is an interesting one. We provide a mechanism to tell ourselves that we should do something additional — we are planning some of our future planning activities and linking it to other activities. But we have to investigate the semantics of such meta-level actions since they do not have the semantics of ordinary actions. One should be aware that there are difficulties arising from not knowing what the actual actions would be and consequently the effects that these actions have. Since it is merely a token or reference point, we do not know for certain what it actually achieves. For instance, what would be the duration of an activity that has such an action embedded in it? What would be the resources required? What are the results of such an activity? Because of this, the planner is not free to plan arbitrarily as in AI planning.

### 7.5.3. Decay

We have already mentioned in Chapter 4 that the relevance and thus the utility of a great deal of information declines with time. We have also mentioned that the value of information frequently declines with the distance from the source of the information. We propose some implicit methods of resource control based on these notions of decay. We will discuss the use of decay as an implicit resource control mechanism in an agent.

With time planning horizons we essentially constrain the distance into the future that an agent considers. Temporal decay, on the other hand, constrains the distance into the past that information, knowledge, processes and so on remain in attention, memory or consideration. It is possible to conceive of many different types of decay mechanisms. There are two somewhat orthogonal dimensions. The tracking mechanism is one dimension. The method of decay and the application schedule constitute the other dimension.

Along the tracking mechanism dimension some possibilities for tracking are

i) absolute time-stamp,
ii) relative time-tag,
iii) association or reference count,
iv) use count,
vi) priority measures,
vi) saturation measures or determinants, and
vii) positional.

Possible positional mechanisms include

i) finite stacks,
ii) queue based, and
iii) partially ordered trees.

Some of the application schedules are

i) regular sweep,
ii) programmed sweep,
iii) triggered sweep,
iv) regular incremental, and
vi) triggered incremental.

The methods of decay might be more involved and can be roughly classified into elementary and non-elementary procedures. Elementary procedures do not use much knowledge in reclamation and decay decisions or use extensive processing of information. Non-elementary procedures might use information processing techniques and complex knowledge-based techniques in conjunction with decision techniques. It is beyond the intent and scope of this thesis to go into these decay methods.

In the theory of multi-user operating systems there are concepts directly related to the ideas in this thesis. There is the notion of active, waiting, suspended and dead jobs constituting a spectrum of processing status. The operating system regularly makes decisions on where the job is — fast memory, slow memory, high-speed disks or long-term storage — and which jobs to run at the moment. For a running job, only portions of the code are brought into memory in chunks called pages. The use of these pages is monitored using reference counts or time stamping. With time stamping the oldest pages are discarded after memory fills up. The procedure is a little more sophisticated with reference counts. Clearly, current operating system technology has employed a variety of decay techniques to manage the limited memory resources of a sequential computer.

In the blackboard architecture of artificial intelligence, a scheduler makes the control decisions. Such a system is particularly amenable for modification to run decay mechanisms.

Several mechanisms can co-exist in an agent. For instance, time-stamp may be used for keeping track of communication waits. A designated amount of time may be allocated for waiting and when the time is up the communication might be terminated. Depending on the transaction framework, the transaction associated with this terminated communication might also be terminated. This is a very useful decay mechanism to ensure that limited communication channels are not tied up as a result of abnormal conditions. One of these abnormal conditions is that of deadlock. The same agent might employ *use* counts as the method of choice in controlling the use of long-term memory. Information that has not proved useful is discarded.

Several plans to achieve several goals may have been made by an agent. The passage of time may, however, make some of them out-of-date. A method employing relative time-stamps might be a useful method to use here. Time starts to run at some point when the plan is expected to prove useful. If our expectations regarding the plan are incorrect, then the plan may be on hold for a while but not forever because it might become invalid and totally useless after awhile. In that case we should discard the plan or portions thereof to reclaim memory resources and at some latter time replan if that is warranted.

While several mechanisms can co-exist in an agent, it is not clear how they will interact and how effective they are for various needs. These issues require extensive future research. We conceive of eventually being able to manage the all the important resources of an agent with the help of appropriate mechanisms.

The use of decay mechanisms has consequences on the interaction among agents and on the expectations that one agent may have regarding the information or knowledge of another agent. Since information might be unilaterally removed from memory by an agent unless stated otherwise, transaction approaches might have to take into account the possibility of information missing for this reason. With decay agents no longer have perfect memory.

### 7.5.4. Budget Management

The two methods discussed above do not directly relate the agent's management of its computation and capacity resources to the goals or tasks that it wishes to accomplish. They are implicit resource control methods. It is because resources and time are limited that we employ such methods — they are heuristic strategies — but we do not necessarily take explicit account of the limited resources or time.

To relate strategies explicitly with resources or time we may wish to have something called budget management. On one side there are the resources that the agent can employ and on the other side the goals and tasks in the agent's wish list. Here we employ some explicit control. In budget management the agent tries to match to the best of its ability the resources that it can employ to the goals that it has. This is generally a difficult problem, but there are techniques from operations research that might be used.

Knowledge as well as computational abilities are needed to do a good job. An agent might not have enough of one or the other.

### 7.5.5. External Feedback

Sometimes it is the case that the agent does not know the cost of executing particular courses of actions. For example, the agent might not know how much effort it would require to find information related to a particular building code. In that case simple budget management cannot be performed since in direct budget management the agent needs to know the resources required to decide. Feedback about the effort required would be useful in such a situation.

### 7.5.6. Reasoning

What we have said about planning also applies to many types of reasoning. In particular, in some cases we can use lazy reasoning. In lazy reasoning reasoning actions are postponed to some future time or await some future development. Sometimes we want to do that even when the reasoning can be done now because we would like to take advantage of new information available at a latter time. Or we might feel that the reasoning efforts might be a waste because we believe that the environment will change, perhaps even in unpredictable ways, and invalidate the results of our reasoning. Reasoning can be planned as well. Planning reasoning is a way of setting up a series of stepping stones into the future. A very lazy machine might even put off most of its reasoning until it believes that it has to reason. Preliminary system analysis of limited reasoning is given in [Fagin and Halpern 85].

## 7.6. Local Truth

The basic idea of local truth is that an agent works with truth that can be determined locally with relatively small amounts of information. Local truth is relative truth. An agent accepts as the basis of decision-making and actions locally determined truth. Consequently, it accepts the problem of relying on incorrect local conclusions.

### 7.6.1. Use of Internal Knowledge

We have argued that it is difficult to create and maintain a large error-free body of knowledge. If the body of knowledge is big enough, we will almost certainly find inconsistencies, incompatibilities and even downright errors. It is quite likely that automation systems will be using a large amount of knowledge. Based on these needs, we can see that some rationality architectures will not be adequate without modifications.

For example, an agent which derives its conclusions based on a single database of facts and rules, large indeed, and a resolution engine would be prone to the problem of inconsistency. One way of dealing with this problem is to classify knowledge.

Knowledge within a class cannot easily affect knowledge outside a class. Another method is to use knowledge incrementally.

### 7.6.2. Use of External Knowledge

An agent should be careful in using external knowledge. If it was difficult to create and maintain a consistent and error-free internal knowledge base, how much more likely then that such a situation does not exist amongst external sources of information. An agent cannot take everything told to it as the pure truth.

We have seen that classifying knowledge can be used to partition the knowledge base of an agent so that the influence of the knowledge can be contained. An additional approach is for the agent to have knowledge filters. Just as perceptual filters screen incoming visual information, knowledge filters screen incoming knowledge. Is something that I have been told new? How does what has been told measure up against what I already know? Is the knowledge reliable?

Tied to this approach is the need for resolution techniques. The techniques for resolution are essentially dynamic resolution methods. The idea is to try to resolve conflicts only when discovered — on demand. If a conflict has been discovered, what should the agent do about it? A simple approach would be to ignore the incoming piece of knowledge. On the other hand, that may not be a smart thing to do since you may be mistaken. If the knowledge comes from a reliable source, the agent might choose instead to use the incoming knowledge and note that what it knew earlier was incorrect. There is a possibility that other conclusions it had derived may be based on the incorrect knowledge.

[Hewitt 83, 88] discusses some methods to deal with conflicting information. He introduced the concept of *microtheories*,

> *"A microtheory is a relatively small, idealized, mathematical theory that embodies a model of some physical system. Prescriptively, a microtheory should be internally consistent and clearly demarcated."*

and *metamicrotheories*, which are microtheories of microtheories.

Clearly, the concept of microtheories is related to the notion of local truth, although they are not equivalent concepts and the concept of metamicrotheories is related to the ideas of metaknowledge. But to deal with contradictory information, he introduces the concept which comes from the field of law, called *due process*,

> *"Due process is the organizational activity of humans and computers for generating sound, relevant, and reliable information as the basis for decision and action within the constraints of allowable resources."*

He states that

> *"Due process uses debate and negotiation to deal with conflicts and inconsistencies."*

Exactly how these mechanisms work still remains to be seen.

## 7.7. Parallelism

### 7.7.1. Multithreaded Planning

In section 7.5.2 we have described an example of multi-threaded planning. In that case a robot generates two plans. One is a basic plan and the other a monitoring plan. We have split off monitoring from the basic plan, but provide for mechanisms to link these plans. Indeed, we can generate several plans of different types and link them together in such a fashion. This linking may even be done dynamically.

To get a better idea of what multithreaded planning means, we first look at how we usually view goals, planning and execution. An agent having a goal generates a plan to achieve the goal and executes the plan. It then proceeds to the next goal and the cycle continues. This is called the plan-act cycle. Viewed temporally, this is somewhat like a series of pulses.

It does not reflect the way human agents work. Agents keep going all the time. New goals are continually created. Plans or plan fragments for the chosen goals are made. Production as well as monitoring plans may be generated. These plans are *assimilated* into the overall activity schedule of the agent — which is in continuous flux. There are also activity threads that function for the survival of the agent and they continue to grow into the future. New daemon processes may also be introduced that generate activity threads over particular periods of time. Environmental events may also invalidate goals and parts of the activity schedule. It is quite hard to depict what is going on, but Figure 34 makes an attempt. Execution of pieces of a plan may be, and generally are, expected to proceed in parallel.

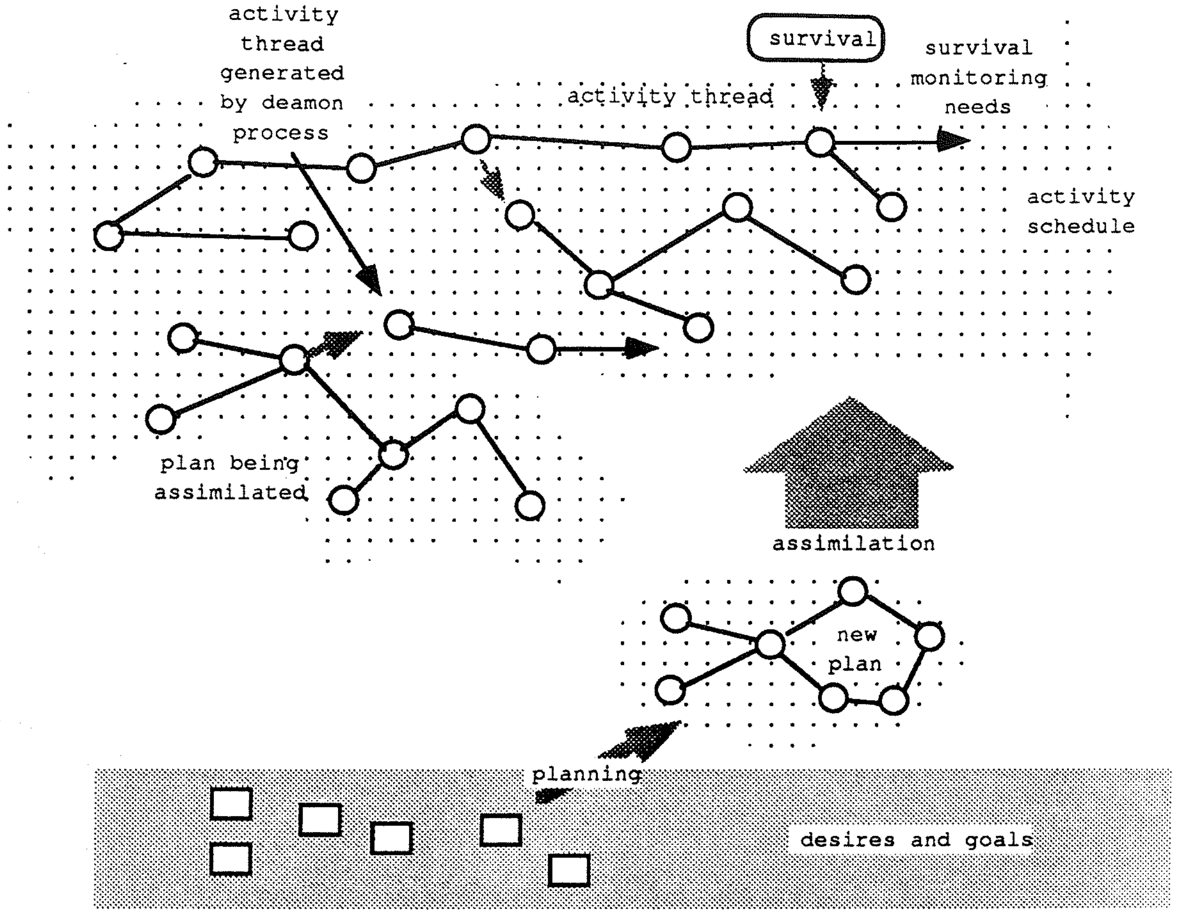


Figure 34. Assimilation of Plans into Agent's Mutlthreaded Activity Schedule

The activity schedule is constrained by the capabilities and resources of the agent. Activity threads which are related might be bunched together into bigger threads. If explicit budget management is used, we might wish to delay the assimilation of plans. Instead, we place them in a holding area for evaluation. The best plans are then assimilated. Plans for which the conditions or original goals no longer hold are dropped. Dropped plans may be discarded or placed in long-term memory.

How is this concept different from the generation of one master plan? In the concept of one master plan, basically we have a single plan that controls all the activities of the agent. There is little notion of contention of plans. Here we are assimilating plans continually generated from goals that the agent has or accepts from other agents into the activity schedule of the agent — contention is always present. The activity schedule contains actions for monitoring for survival, portions of previous plans and other monitoring activities. The former may be called normal background activity of the agent. It is not as if planning is done against a blank slate. When we simulate an agent, we might want to simulate it long enough to provide a stream of normal background activity before we introduce goals for the agent.

During the process of assimilation, competition for resources may occur between the plans to achieve the new goals and the scheduled background activities or activities for previously assimilated plans. Some of these activities might have to be sacrificed or reduced in frequency and scope as a result. Alternatively, if it is not possible to accommodate the new plan, scheduling of the new plan has to be postponed.

Two aspects of assimilation are the generation of monitoring plans and predictions of sensory signals. For example, if you turn the steering wheel to the left, you expect the field of view to move to the right. You monitor for such movement and if it does not occur you realize immediately that something is amiss.

Using lazy planning in combination with multithreaded planning, a machine ought to allow us to generate low-level plans on the fly within the framework of all the other activities of the agent. A concurrent language is needed to represent such plans.

## 7.8. Architecture of a Machine Agent

There are a large number of possibilities for putting together an agent. The debate is not settled on what would be the best architecture for an agent. Our approach uses the analog of an operating system but incorporates in the core an active element and further expands on the concept.

### 7.8.1. Overview

An agent needs to be able to control its computational, memory, effector, sensory and other resources. Thus it is a kind of operating system — an expanded notion of an operating system. An agent will not function well as an application in a multi-tasking system because of lack of control over the timing of activities. However, it can acquire and use computation resources from such systems to perform particular computations. Some researchers, quite appropriately, refer to an agent as an *engine* [Gassner et al 87].

We would like agents to be able to generate and execute concurrent plans of actions. Agents also need to monitor their environments and opportunistically respond to the events they detect. In the multi-agent environment of construction, an agent ought to be able to interact with other agents both synchronously and asynchronously as the case may be. All these needs are extremely demanding. What programming paradigms and platforms might meet these needs?

"Opportunistically respond" immediately brings to mind blackboard architectures [Hayes-Roth et al 85]. There is a global data structure called the *blackboard* where information and partial solutions are posted. Knowledge sources "observe" the blackboard for information relevant to them. Triggered knowledge sources produce knowledge source activation records which are queued. A scheduler selects the best knowledge source to execute. Such systems have been expanded to include control blackboards. Therefore, control actions can be intermingled with domain problem-solving.

While strongly opportunistic, blackboard systems have a number of weaknesses as platforms for creating agents. The usual blackboard system cannot deal with simultaneous sensory input and concurrent execution of actions, because they are strongly serialized. There is a move towards creating parallel blackboard architectures that preserve much of the original opportunistic nature of blackboards.

Another type of opportunistic framework is the CONTRACT-NET architecture. In this architecture agents advertise tasks and other agents bid for them. There is no central scheduler or global data structure. This permits distributed implementation of the system. There is a high degree of concurrency since each node works on its own problems and communicates asynchronously with other nodes. Each sensor subsystem and effector subsystem can be managed by one or more computational nodes. One problem with such a framework, which it shares with blackboards, is that the problem-solving behavior might lack focus. There is a tension between focus and opportunism.

This brings to mind hierarchical architectures. In such architectures there are several levels of control. Higher levels direct the activities of lower levels. They can be used to provide focus for the activities of the agent such as concentrate computational and other resources on particular tasks or problems instead of spreading them out among numerous tasks or problems. Some work in this area include [Corkill 79]. Hierarchical systems have also been popular in robot control systems.

Actors [Agha 86] is a programming paradigm specially created to exploit massive parallelism. Actors are computational agents which map each incoming communication to a finite set of communications sent to other actions, a new behavior to govern the next communication and a finite set of new actors created. They are thus useful as a base language for computational algorithms because they permit the flexible use of available computational resources; actors can be distributed and they interact with each other only through communication which can be performed over computer networks.

We propose an agent architecture that can be conceptualized as containing four main functional components, the nexus, knowledge manager, determiner and action manager. The latter three may also be called the satellites. Under certain circumstances we may want each to have some dedicated processing power and memory resources. All parts of the system are heavily compartmentalized and modularized, with redundancy in some portions of the system. Parts of many systems may fail without destroying completely the functionality of the system. The separation of duties makes such an architecture differ somewhat from uniform centrally managed architecture, such as those using central schedulers and a single inference engine.

### 7.8.2. Nexus

This is the kernel of the agent. It might be described as an active distributed operating system. It is active because it does not necessarily wait for input before acting. It is distributed because it should be able to run on several processors. It performs general resource management and contains active elements that drive the agent. It monitors subsystems and controls conflicts and misbehavior.

It generates high-level plans for activities from desires and goals. With the help of the knowledge manager, it assimilates these plans into its activity schedule.

This system is clearly the most critical system of the agent, so it has to be developed carefully. The design of the nexus would be debatable, but we suggest different levels of operational reliability. At the heart would be a micro-kernel that should contain the most essential functionalities and must be able to run when everything else is non-operational or even inimical. For instance, it should continue to run in the presence of virus agents and never be completely wiped out. The activity scheduler is outside the micro-kernel and activities of the micro-kernel are not managed by it. The micro-kernel may include security services, basic memory and resource management, essential safety services and so on. A simple view of the nexus is given in Figure 35.

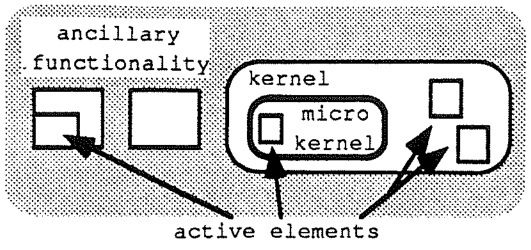


Figure 35. Suggested Components of Nexus

Ancillary functionality include various systems for keeping track of resource usage, quality of subsystem performance, evaluation of plans generated, priority schemes, resolution of subsystem conflicts and so on. They also form the intellectual core of the system. Active elements drive the processing done in the nexus and outside. The nexus need not be a single computational thread but can use any number of computational elements.

### 7.8.3. Knowledge Manager

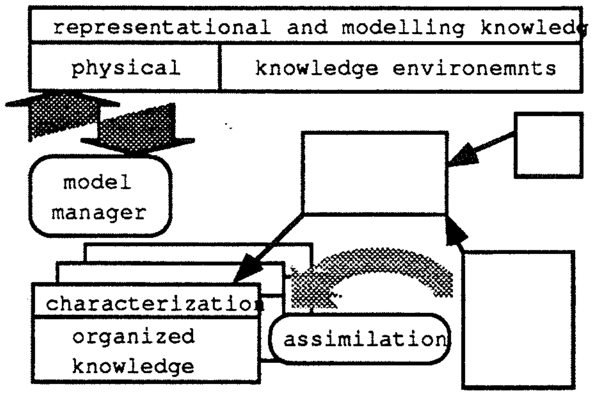


Figure 36. Partial View of Knowledge Manager in Operation

The knowledge manager deals with the organization and use of an agent's knowledge and serves some of the knowledge needs of other components. The knowledge it manages includes self-knowledge, environmental knowledge, specialized knowledge, project knowledge and so forth. It employs several types and levels of metaknowledge to perform its knowledge management task. It controls knowledge acquisition, assimilation and forgetting. It does database management, knowledge base management and some knowledge processing.

Figure 36 shows a few of the components of the knowledge manager and the processes of managing knowledge. It may have an active component that uses idle time of the agent to archive and reorganize its internal knowledge base such as shifting information from short-term to long-term memory and vice-versa.

### 7.8.4. Determiner

The determiner deals with sensory information from its environment and from itself, and, therefore, comprises of most of the subsystems related to sensors and communication. The determiner is in a sense the environmental information interface of the agent. Among the functionalities of the determiner are

- the processing, classifying and interpretation of sensory information from the environment and from self-monitoring sensors,
- receipt and management of communication signals and messages from databases and other agents, and
- the deciphering of structured information in knowledge objects.

It determines the condition of the agent and the state of the environment. It does so with the help of a lot of knowledge from the knowledge manager.

Between it and the knowledge manager, they devise and maintain models of the physical and knowledge environments. As the determiner's computational tasks are computationally demanding, it would usually have dedicated hardware for many aspects of its processing. It obtains additional computational resources from the nexus in competition with the knowledge manager, the action manager and other processing required by the nexus.

Many types of computational architecture may be used, but the more computations that are done in parallel, the better. Parallel dataflow computer architectures, connection machines and array processors are among the types of hardware, allowing a great deal of computational parallelism, that can be used. Interpretation techniques based on neural networks or even parallelized computer vision algorithms can be implemented on such architectures.

### 7.8.5. Action Manager

The action manager manages the externally related activities of an agent — communication acts and effector motions. It takes fairly high-level actions and produces detailed and executable actions at the environmental interface of the agent. This is frequently referred to as compilation, as depicted in Figure 37. It may also be used to manage some aspects of transaction protocols and low-level control of activity. These tasks may be helped with the maintenance and use of models of the physical and communication characteristics and resources of the machine. It would be useful if some of these descriptions can be modularized.

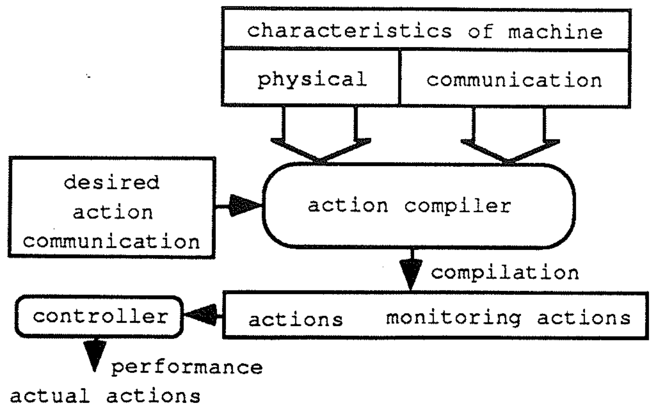


Figure 37. Compilation by Action Manager

Compilation is not monolithic, as one might mistakenly deduce from the figure. As in other parts of the system, the design should be modular. Even if parts of the action manager fail, most of the system should still continue to function. For instance, if the hydraulic system fails the agent should still be able to communicate.

### 7.8.6. How these Pieces Function Together

The nexus runs continually and actively drives almost everything. When needed it creates special processes to perform particular tasks and allocates computational resources to them. The nexus interacts heavily with the knowledge manager, determiner and action manager. The knowledge manager, determiner and action manager also interact with each other. Most of the interaction occurs through internal communications and, in certain cases, through transferred or shared blocks of memory. These subsystems are not passive. They have a large measure of autonomy and intelligence and are capable of active activities.

One way to get the different parts to work together is to use a market-like mechanism for bidding for resources and services. The satellites bid for these resources from the nexus. They also bid for services from each other. However, there are portions of the nexus which can override these mechanisms and acquire immediate services.

An agent is not equivalent to a computational node. Several computational devices might be used by an agent. An agent can migrate from one computational device to another. Special hardware may be used by the agent for signal processing, floating-point computations, inferencing, device and effector management and so on. Computational devices are resources for the agent and the processes constituting the agent may be distributed across several processors. The agent has to be able to withstand a certain degree of computational device and communication system failure — this is one of the major differences with human agents. It may regularly back up fragments of itself at various nodes. Should failure of a computational node or memory device occur, the agent might be a bit confused for awhile but will frequently survive the failure.

It should be expected that unrecognized failure of parts of the determiner can induce strange behavior in the agent. For instance, if the agent thinks that there is an obstacle in front of it as a result of sensory failures, it might go in circles. Failure of effector systems are more easily detected. Sensory inputs can be compared against predicted expectations and, if they do not concur within acceptable limits, then there is some problem with the system.

The architecture as we conceive it has some similarities with those suggested by others in the literature [Georgeff and Lansky 87]. Comparisons with these systems will indicate differences as well. Many systems advocate either uniform or strongly hierarchical architectures, whereas the system we suggest has both elements of peer-level interaction and hierarchical interaction of components. Most of these systems are not internally partitioned, whereas we advocate both internal partitioning and redundancy. Most systems are built on top of regular operating systems, using the services provided by them, but we advocate bringing the operating system into the agent itself so that the agent can better manage its resources.

It is difficult to create systems with fixed resources that are generally optimal. The agent may have an excess of resources or inadequate resources. Since there is likely to be a mix of tasks that agents are required to handle, there would be a mismatch between what should be optimal resources and what is available. In this case, we may want to construct an agent that is optimal over a range of tasks but possibly suboptimal for many of the tasks. We also need to know what constitutes minimal competence at different levels of decision-making and action. The agent's capabilities should not fall below these minimal levels.

It is sometimes possible to improve the efficiency though more flexible inter-system resource management. A flexible agent uses a range of resources and, therefore, can be mixed and matched for particular tasks as they arise. Also, non-time-critical computations can be contracted out to remote computation sites, and non-essential knowledge and data can be shifted to remote storage sites. We will discuss this system support for agents in Chapter 10.

We have argued for creation of a general core of knowledge which can subsequently be enhanced with specialized knowledge. Specialized knowledge and abilities may involve the entire spectrum of processing and thus we should allow such access as far as possible. If the security of the agent might be compromised, we might develop almost independent systems as tools to be used by the agent instead of incorporating the knowledge into the agent. For example, we do not have to incorporate all the welding knowledge into a welding agent. We still have the general capabilities of the agent to fall on when needed. Self-knowledge of the agent may help reduce the effort of merging specialized knowledge with the general core. For instance, it might be possible to feed project knowledge and knowledge about the physical and knowledge environments to the system, if the system knows how to represent and organize the knowledge internally.

# 8. Operational Policies

This chapter discusses some of the principles and policies that one might use as guidance in the creation of multi-agent automation systems. It is not meant to provide a thorough coverage or to proceed in great depth. Only a very limited extent of the work by other researchers will be mentioned. It has two main emphases. The first three sections emphasize multi-agent issues which stem from the need for common ground, identity and cooperation. The last two sections cover some policies for the agent itself that are directed to its use of resources and time.

## 8.1. Commonality Principles

In computer programming methodology we have seen a move away from monolithic programs to procedural programs and more recently to object-oriented programs. In monolithic programs control and data are all intermingled. One of the early conceptual breakthroughs has been the decision to separate decision-logic from data. Data is much more variable than programs. Therefore, we can have a program that can use different sets of data to produce useful results. We added functional decomposition since pieces of the logic of a program can be reused in several places.

A new conceptualization began with the realization that one can break potential dependencies by associating data manipulation code with the data. Behavioral engineering began in earnest by restricting the ways data can be manipulated through encapsulation. The notion of datatypes in programming methodology is a way to associate a particular set of operations with the "type" of the data. Ontological engineering began with need to map the encapsulated data with concepts that are readily understood by humans. One of the important ontological concepts is the notion of an object. The notion of an object is a strong intuitive notion that had its original basis in experiences in the physical world. It says that we can conceptualize an imaginary boundary so that whatever is inside this boundary is part of the object and whatever is outside in not a part of the object. Our organizational power expanded further with the notion of abstraction. Things that share characteristics in common can be grouped together and the group assigned a name. Groups that share the same characteristics together can be further grouped. Obviously there will always be different ways to group things together and so different abstractions are possible. The notion of abstraction has existed for a long time in human knowledge engineering in forms such as taxonomies. These taxonomies have inheritance of attributes and behavior.

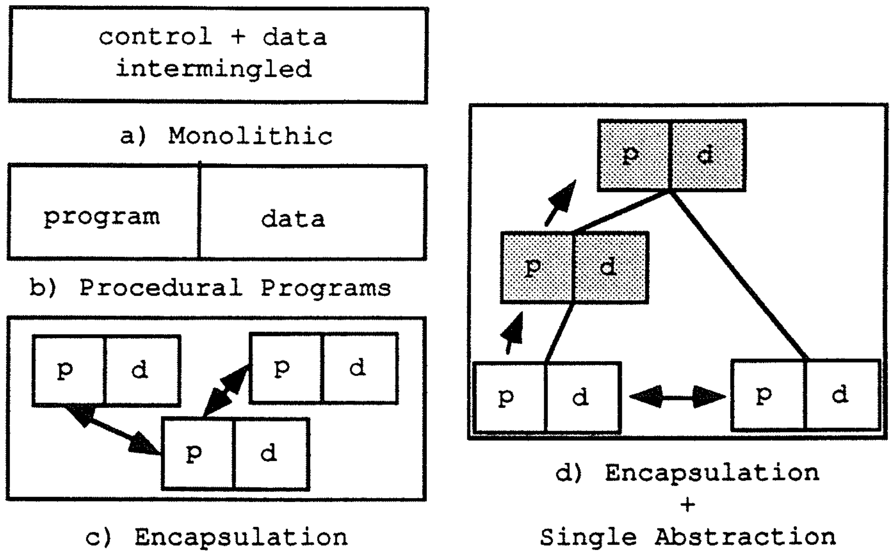

control + data intermingled
a) Monolithic
program
data
b) Procedural Programs
p
d
c) Encapsulation
d) Encapsulation + Single Abstraction

Figure 38. Changes in Programming Methodologies

In object-oriented paradigms we have both encapsulation and abstraction. Encapsulation is a matter of behavioral engineering, which will be discussed in Section 8.1.2. Some of these changes in programming methodologies are depicted in Figure 38.

### 8.1.1. Ontological Engineering

In many cases the abstraction takes the form of a single abstraction hierarchy. Recently, multiple-inheritance (Figure 39) has become available but is largely unused for most usual programming. Where such techniques may prove valuable is in knowledge engineering. There are many ways to build abstract conceptualizations of the world.

Take as an example biological taxonomy. It is certainly one of the most elegant and useful way to conceptualize the living things of the world. Nevertheless, it is not the only way. Laymen often do not understand why porpoises are not fish but mammals.

One of the major problems for the interaction or cooperation of several agents is that they may not have the same ontology — they do not view the world or concepts the same way. CYC is an effort to address this by proposing the development of a common ontology referred to as *consensus reality* [Lenat, D. and Guha, R. V. 88]. Much effort is spent on deciding the meaning of each abstraction category and their relation to others.

We might also keep several common abstractions available and use some method to select abstractions to be used in a particular circumstance.

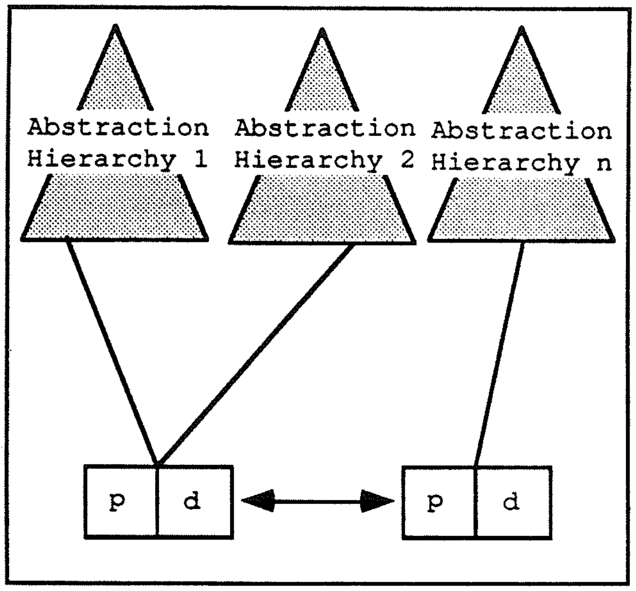


Figure 39. Artificial Intelligence-Like Ontological Abstractions

A minor problem with the pure object-oriented paradigm which derives its power from encapsulation and abstraction is that it is difficult to take into account relationships among objects. In frames and its derivatives, one of the approaches is to have slots that hold these relations in the frames themselves. If we apply this then the relations will have to be scattered among the frames. This introduces a dependency among the frames.

For example, suppose Mark is a teacher and Joe is one of his students. In Mark's frame we have a slot ***students*** which holds a set and in this set we have Joe. In Joe's frame we have a slot ***teachers*** which holds a set and in this set we have Mark. The single teacher-student relation between the two objects Mark and Joe has essentially been broken apart and placed in these two frames.

KEE and other frame-based languages have recognized this problem. Most of them provide a trigger mechanism (called active values). A trigger can be attached to the slot and will trigger when the content of a slot changes. Once it triggers it can execute an arbitrary program. In particular, it can execute a program that maintains or deletes relationships among objects. For example, if Joe is no longer Mark's student, when the system removes Mark from the teachers slot of Joe, it triggers a program that also removes Joe from the students slot of Mark. It is difficult if not impossible to use this approach to deal with general constraints and relationships among objects.

Another possible approach is to regard objects as basically capturing local information only. Inter-object information such as relationships and constraints among objects is not local information and is represented separately. A separate mechanism is used to maintain their consistency. We conceptualize the distinct basic conceptual elements as objects and relationships among objects.

### 8.1.2. Behavioral Engineering

One aspect of behavioral engineering is the encapsulation of state. Encapsulation in objects generally means that the state of the object cannot be directly modified. Specific messages have to be sent to the object to effect such modifications. By restricting the way interaction can proceed, we provide a measure of error containment. We also reduce the complexity of interacting with the data since each message sent to an object results in several actions being taken. We control and thus reduce what we need to know about the object in order to use it

Although we have been becoming increasingly successful with ontological engineering, researchers have not gone quite as far with behavioral engineering. We have restricted the ways by which data can be manipulated through object-oriented programming. The object-oriented paradigm, however, resulted in the creation of a multiplicity of micro-languages. In order to integrate these encapsulated objects together the programmer has to understand these micro-languages. Engineering has to be done at the detailed level of object-object interactions. Fortunately, this is considerably helped by the abstraction mechanism as well. As long as these remain small and easily understood, we do not have too much difficulty working with them.

However, with the creation of increasingly complex systems such as automation systems, we should reexamine the question. An encapsulated object with a hundred-message interface, several of which are long, begins to stretch human cognitive abilities. Also, although we have controlled and reduced what we need to know about each object class, we have not been so successful in controlling and reducing the total amount of what we need to know about the classes; there is an almost linear increase in the number of micro-languages with respect to the number of object classes. The communications that need to be proceed between complex agents such as humans and automated machines are one order above simple object-oriented message passing. In order to reduce this complexity, it might be a good idea to make it possible at the human-machine interaction level, at least, to be able to use a simple language to communicate to the objects. We do not allow the agent programmer complete freedom to set up his own micro-languages. If the intuitive meaning of two messages on two different objects happens to be similar, then a similar way of communicating should be used. We now move on to engineering the behavior of agents.

The hundred-message interface for an agent is not so bad if all agents respond to the same messages in intuitive ways. For instance, machine agents should have some understanding of the word "stop". It gets a little more difficult when we may have to remember that a few agents do not respond to certain messages while the rest do.

This suggest that we should not allow the proliferation of micro-languages when designing agents such as in object-oriented programming, but strive to provide a simple, intuitive, powerful and slightly extensible base language for all machine agents.

Whereas in ontological engineering our concern is the meaning attributed by agents to the knowledge they have, in behavioral engineering our concern is the way agents should behave with respect to other agents. Protocol's and patterns of behavior serve to provide common ground for the interaction of agents that may know very little about one another.

Much as message passing encapsulates local state and descriptions, the use of tokens of exchange simplify dealing with great diversity in transactions. Transaction frameworks themselves simplifies dealing with great diversity in decision making and the heterogeneity of pursuits. We have in fact a great number of different objectives being pursued by various agents and organizations in the environment. What ties agents and organization together now are a few mechanisms that give considerable reach and potential power to entities who know how to use them, but still retaining an innocent simplicity.

The description of transaction frameworks and their operation in Chapter 5 hints at the structure and processes that are going on in these mechanisms. The majority of negotiations and meetings tend to be short lived and 'focused' utility mechanisms while markets and organizations are longer lived and serve more abstract purposes. However, systems involving machines would not be using mechanisms as complex as real-world ones anytime in the near future. Our research will study the use of simpler forms of these mechanisms and combinations thereof.

## 8.2. Independence and Identity

One of the problems when working in an environment of agents is the degree to which an agent would need to identify the agents with which it is interacting. How do agents identify one another and to what extent? Does the identification need to be unique? Under what circumstances is it enough to know that it is interacting with a human but doesn't know the particular human? For example, consider the problem of a seller is interacting with a buyer. Does he need to know the buyer's full name? Or is knowing the buyer's first name and address sufficient? If it is sufficient, why is it sufficient; if not, why not?

The above example is one of interaction over a short time. Difficulties are greater for longer interactions. The agent's knowledge and address may change. How do you know it is the same agent? It may not be possible to assume that the agent now interacting with you knows the things that you told it before because it might have changed into a different agent. Security would be compromised.

We can still learn from the way societies deal with the problem of identification. Societies use multiple identification features, such as pictures, social security numbers, finger prints, addresses and background. The extent these are used depends on the security needs. One possible method we might adapt to machine systems is the use of trusted third-party identification confirmation. For example, we can have a seed and time-dependent scheme. When asked, the agent provides the time, nominal name and the identifying number which has been computed based on the time-dependent scheme. The asker confirms with the secure third-party by providing the time, name and identifying number and receives from it confirmation or rejection after a time quantum. The identifying number is only valid for the time of asking, so the asker cannot use it himself to impersonate the agent. The time quantum negates the need for complete synchronization among all agents. More than one secure third-party can be used, but that might require modification of the scheme. The scheme is weak in the sense that the third-party needs to be secure.

## 8.3. Cooperative Principles

What are the principles that promote cooperation among agents? Some have claimed that no more is needed than self-interest [Durfee et al 1987]. That is true in essence if we can define self-interest and believe that an agent, in its limited perspective of time, can compute it. Immediate payoff is not a sufficient measure of self-interest because it cannot explain the behavior of existing agents in the environment. The next few subsections present the following principles for cooperation:

i) tit for tat
ii) win-win
iii) justice
iv) service

### 8.3.1. Tit for Tat

[Axelrod 84] describes the results of two experiments in an attempt to answer the question,

> *"When should a person cooperate, and when should a person be selfish, in an ongoing interaction with another person?"*

The welfare of an agent is dependent on the principles it follows in its interaction with other agents. The game chosen was that of the prisoner's dilemma[1]. Strategies developed by various people were pitted against each other. In the first tournament there were fifteen entries and in the second tournament there were sixty-three. The strategy called tit for tat proved to be the most robust strategy in both tournaments. The strategy is basically

i) to cooperate on the first move, and
ii) on subsequent moves, do what the opponent did on the previous move.

The moral lessons of these experiments were:

> *1. Don't be envious.*
> *2. Don't be the first to defect.*
> *3. Reciprocate both defection and cooperation.*
> *4. Don't be too clever.*

It is relatively easy to understand why the first three are useful. The fourth item is related to the difficulty of agents to discern overly clever strategies. If an agent is dealing with another agent, it is useful to be able to discern the strategy of the other party in order to take that into account in developing your own strategy. We want to rationalize the other party. We often simulate the thinking process of the other party in *lieu* of directly asking the other party for its decisions or conclusions. But such rationalization is, of course,

[1] Prisoner's dilemma is a situation in which what is best for each person individually leads to mutual defection, whereas everyone would have been better off with mutual cooperation [Axelrod 84 pg. 9].

bounded by our own intelligence and subject to error. Mistakes in discernment lead to non-cooperation where it might have been otherwise. If the other party uses a simple method of decision-making, then we can simulate it more accurately and therefore be able to cooperate better should we desire to do so. In this game, tit for tat is a very easily discernable strategy. Clearly, in an adversarial situation, if strategies are discernable, then it is important that they are not easily taken advantage of. Tit for tat cannot be easily taken advantage of.

There has been a lot of research as to what should constitute the rational behavior of agents. [Rosenschein 85] discusses many of the problems of coordination and cooperation in cases where communication is not allowed, and also in cases where agents misrepresent themselves during communication. Detailed analysis is given in many cases. Application of probability theory to the analysis of communication-free interaction is given in [Rosenschein and Breese 89].

## 8.3.2. Win-win ($W^2$ principle)

The idea of win-win in interaction among agents is that whenever possible all parties gain. They should be better off than before. Win-win thus represents distributed hill-climbing. Win-win implicitly recognizes that one of the objectives of the other party is to profit in some way, whether the profit is maximal or not is a second-order consideration. This is in accord with the experience that many human decisions are satisfying decisions rather than optimizing decisions. Win-win is not the superordinate environmental strategy but should be subordinate to tit for tat. Win-win gives the rationale of the first decision in tit for tat. It also provides the rationale in a finite iterated prisoner's dilemma.

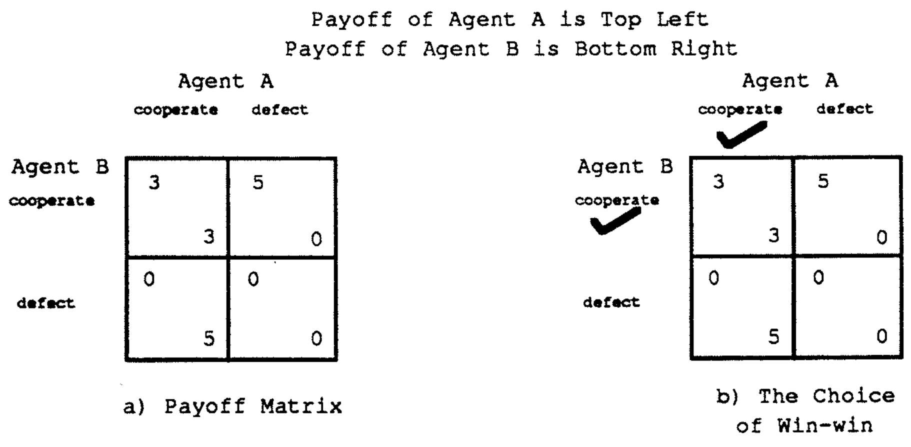


Figure 40. The Solution of Win-win to Prisoners' Dilemma

We shall show how two rational agents subscribing to the $W^2$ cooperative principle deal with the real prisoner's dilemma. Figure 40(a) shows the payoff matrices as believed by the two agents as an assessment of their circumstances. Figure 40(b) shows the decisions chosen based on win-win. Using win-win both parties come out winners. We see that both agents A and B discover that defection results in only win-lose or lose-lose. Win-win can only be achieved by cooperation. If both parties have the same assessment of the situation, then one party's win-win must be the other party's win-win. If there are several exactly equivalent win-win options, then communication may be needed to resolve the choice.

It is a robust technique and not very sensitive to the actual perceived payoff functions of the two parties. The method will function adequately as long as the methods of evaluating payoffs are roughly commensurate. Therefore, it is in accordance with the principle of local truth enunciated in Section 7.6.

$W^2$ also works in multi-agent and group interactions and therefore is of potentially wider applicability than the concept of tit for tat.

The philosophy of win-win can be applied to the construction of agent proposals in negotiations. In many transactions, agents have much more room to maneuver than in the case of agents' dilemma. Proposals can be put together in many ways. When a proposal is put together, there must be benefit for both agents according to the win-win philosophy. It does not prescribe completely the distribution of benefits. For that we suggest another cooperative principle called *justice*, which we will discuss in the next section.

One of the problems of $W^2$ is the semantics of win and lose. It is a relative concept. An agent wins if the situation of the agent improves. An agent loses if its situation deteriorates or does not improve — the latter is included because there is no benefit that the agent concerned derived from the transaction. If we raise the payoff matrix by 5 uniformly, then the problem is no longer one of win or lose but one of relative gain. With relative gain, agents feel freer to compete more vigorously.

Another of the problems (and perhaps the strength) of win-win is that it is hard-wired by evolution, a magic belief the rationale of which lies in the evolutionary past and possibly unknown to the agent now. It is a solution dependent on the way things developed before. It works for the group of agents that exists now and not necessarily for any arbitrary group.

Win-win solutions are not necessarily globally optimal. The overall benefit might be smaller than that possible. For instance, if we changed the payoffs for the two agents in the above example from 3 to 2, the agents will continue to make choice 1 but the overall payoff is only 4, which is smaller than the maximum combined total of 5. In order to accommodate some agent, we might have to make some sacrifices. Win-win also does not prescribe the best distribution of benefits. The next section describes the notion called justice.

## 8.3.3. Justice

This is a notion that is strongly prevalent in human society. As a principle, it is at the same fundamental level as tit for tat, since tit for tat may be thought of as an implicit justice mechanism, but in human use it might be relegated to a secondary position to win-win. A decision is normally filtered through tit for tat and win-win before its justice aspect is analyzed.

But we should not be deceived about its fundamental importance. It is the rationale behind the rule of law that is going to be discussed in Chapter 10.

One offshoot of it is the belief that agents should derive reasonably *fair benefit* from transactions. To be able to use this concept, each agent has to come up with an estimate of the payoffs for the other agent. What goes for fair benefit is derived locally in keeping with our strong decentralization orientation. So what one agent figures to be fair benefit can differ from what another agent figures to be fair benefit.

Payoff of Agent A is Top Left
Payoff of Agent B is Bottom Righ

Agent A

| Agent B | choice 1 | choice 2 |
|---|---|---|
| choice 1 | 4 fair 4 | 3 6 |
| choice 2 | 3 6 | 2 8 |

a) Payoff Matrix

Figure 41. Suboptimal Solution Using Principle of Justice

Another problem is that it might result in suboptimal solutions just as in the case of win-win. Figure 41 is a payoff matrix that helps illustrate this. Choice 2 does not involve any fair solution so agent A chooses 1. The globally optimum solution is for agents A and B to make choice 2. A selfish solution would result in agent A choosing 2 and agent B choosing 1.

But we should not forget the original self-centered motivations of the agent as described in Chapter 2 — this drive to maximize its benefits. There is a tension between justice and self-interest. This will provide the rationality for the desired solution in the iterated interaction case. We have to incorporate that drive to maximize its benefits. The iterated case where agent B can decide to proceed with further transactions with agent A or go elsewhere, is interesting. The desired solution is for agent A to build a relationship with agent B by making some choice 1's. Otherwise, agent B would try its luck elsewhere.

The principle of justice can be followed to various degrees by an agent. It is often the case that an agent will try to make a deal "fairer" to him than to others, but overall the principle of justice still applies. An agent is said to resort to criminal behavior (even if it did not break specific laws) when other agents believe it has deviated too far from the principle of justice. In human societies, criminal behavior will possibly result in the withdrawal of cooperation and support. Punitive measures may also be taken by other agents. The consequences make it costly for an agent to violate the principle of justice.

### 8.3.4. Service

#### *8.3.4.1. Neighborliness*

We can sum up this operational policy with the words of sages, *"do for others what you would have others do for you."* The principles that we have already enunciated does not answer some questions about cooperation that occurs in the human knowledge environment. What is the rationality in the following interaction?

For example, Joe asks Bob to tell him the time. Bob does. What benefit did Bob derive? There is no agreement for Joe to tell Bob the time either. Reciprocity is missing.

Sometimes the agent following such a principle might be even worse off. For instance, a person who dives into the water to save a drowning stranger endangers himself.

We can begin to understand such phenomena more if we look at the big picture. The human society is not going to function well if there is no drive to the tune of this principle. Since we have the intention of creating machine agents, we should pay attention to this principle as a possible rationale for engendering cooperation and not only with respect to machine agents but also humans. In the next section we discuss an interaction in which one peer agent is asked to take on the task of another peer agent.

### *8.3.4.2. Expanded Interest*

Next, we introduce the notion of *expanded interest*. We have already seen that in a distributed environment of autonomous agents it is rarely possible to compute or know the globally optimal decision. This is because each agent has only limited information about the environment. The other extreme, then, is to compute based on self-interest. Clearly, local information is much more readily available and it is easier to determine what the optimal decision is to oneself. Nevertheless, it is often the case that agent may have some information about other agents with which they closely interact. They can develop some simple models of the needs of these agents and the circumstances of these agents. With the additional information it is possible to optimize over the needs of several agents as opposed to just for one's own needs. Figure 42 depicts the idea of expanded interest as an intermediate point between global consideration and self-consideration.

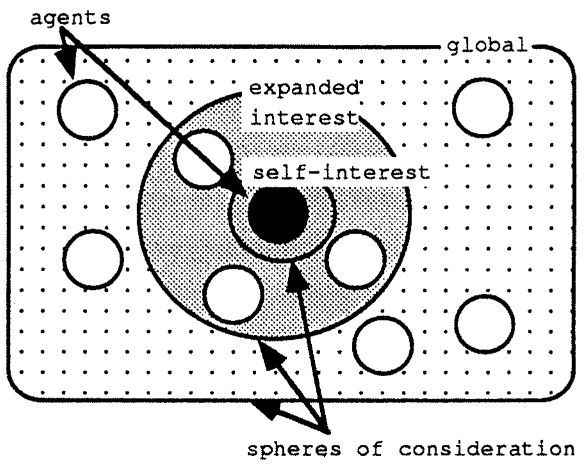


Figure 42. Expanded Interest

For example, Joe asks Bob to bring him a wrench. Under certain circumstances, Bob does so. What benefit did Bob derive? Bob not only did not derive any benefit directly but instead might have to employ his resources and time on the mission on behalf of Joe, perhaps to the detriment of himself.

In this particular case, we believe that the agent Bob is not only applying the words of sages but is actually considering the payoff for the enterprise, that is, for the two of them rather than his own personal payoff. The agent recognized that it is only a part of a larger system. By applying the principle of economy to a group of agents (himself and Joe); perhaps Bob figures that some effort could be saved if he performs the task on behalf of Joe and therefore is for the overall good even though his own effort has increased without due compensation.

This is not so altruistic as it might first appear. There is a link between the success of the enterprise and the success of Bob, but that is an unknown and non-computable payoff function, at least, as far as Bob is concerned.

But since there is a need to integrate all these principles and there is no actual payoff in many cases of applying this principle, evolution has dictated that the agent generates some kind of phantom payoff internally. This dummy payoff is a useful technique to integrate the various operational policies of an agent.

We will look at fairness and deadlock in the case of the dining philosophers' problem from the viewpoint of expanded interest. In this particular problem, we have five philosophers around a round table and they each need two utensils in order to eat. The utensils are, however, of shared availability with neighbors. If they base their decisions only on the information they have locally, that is whether they are hungry or not, and make requests for the utensils in order, then deadlock can occur. There are ways of breaking deadlock, such as time-out, surrender of resources and random retry. These methods ensure that eventually deadlock will be broken. In practice, deadlock will be broken quickly. Nevertheless, the method does not assure fairness and some poor philosopher may be starved. On the other hand, if these philosophers were not so self-centered but have information about the philosophers adjacent to them, both the problems of deadlock and fairness go away. If the philosopher observes that one of his neighbors already has the shared utensil (between the two of them), then it is foolish for him to pick up one of the utensils. If he also keeps track of the last time his neighbors have eaten, then starvation can be avoided. To obtain such a situation, the philosophers either have to communicate or have to observe. The deadlock problem does not go away in the general case.

Most researchers in the arena of distributed artificial intelligence have expanded interest principle at the back of their minds when they say they want *friendly* agents. They incorporate the principle of expanded interest implicitly. Friendly agents are agents which will help another agent if they are in the position to help. We merely explicate the principle of expanded interest so that people can think about the issue more deeply and realize that it is one of the principles of cooperation. As with all the other principles, an agent can follow such a principle to various degrees or not.

Environmental intelligence confers an agent with the ability to exploit its environment. Exploitation of the environment enables the agent to achieve more that it can accomplish by itself. But, just as in human societies, a purely exploitative morality may no longer be the best approach. By exploitative morality we mean the use of environmental intelligence and capabilities to serve only an agent's internally generated goals. Any benefit accruing to other agents is purely incidental.

Cooperation among human beings is a hard-won aspect of human society. The principles that engender cooperation are very often powerful enough to override the maximization of local benefit for an agent. They are environmental control mechanisms — distributed mechanisms that allow groups of agents to reach a level of achievement beyond the sum of what they could achieve individually, and even beyond the sum of what they can achieve if they cooperate, but only on the basis of their own self-interest.

An aspect of service is the service of goals that may be generated from other members of a group of agents. This might demand occasionally placing aside local maximization of payoff. An action requested of an agent that has no payoff or negative payoff for that agent may have significant payoff for the group of agents. For instance, an agent may be requested to take on the task of another agent. Taking on the goals or tasks of another agent may not benefit the agent directly but could result in an improvement of the group of agents.

Another aspect of service is the active contribution of an agent to the needs of those around it. Contribution means contributions to the functionality of the environment. Agents working in the midst of other agents should contribute something to facilitate the overall well being of other agents. An agent might act as a secondary, but a more easily reachable, source of information for other agents. We have already seen the simple question about the time. The agent might also have acquired some knowledge about the state of affairs at its previous locations. For instance, it might know that some routes are currently blocked and impassable to traffic. It will actively warn agents who are going in the direction it came from of the difficulties it encountered.

A very useful environmental service is for an agent to report the breakdown of a machine agent. There is a chance that a machine agent is so badly broken that it is unable to perform that important task, or perhaps the machine is a simple machine without such abilities.

The examples we have seen so far just give an idea of what the policy of service means practically. It is by no means an exhaustive list. We explicate this principle because there is a danger of basing the design of machines purely on self-interest.

## 8.4. Economy

There are various aspects of economy that may be used to guide the actions of agents who have the power to exploit the environment's apparent infinite resources. Examples of principles of economy are "minimize computable costs", "use well", "need only", "conserve" and "reuse".

Some of these principles may conflict. The last four items tend to reduce cost implicitly but often in ways that are not easily computable. It is up to the automation designer to choose or merge these operational principles to create the automation system with the desired properties.

### 8.4.1. Minimize Computable Costs

If costs can be computed and there are alternatives and you have a choice, then choose the cheap way. Since costs to a degree integrate information about the effort involved in making resources available or energies required it is a good guideline for an agent to try to minimize costs. By doing so the overall environment conserves the use of its less available resources and concentrates its efforts in the direction of need. This effect is not easily visible to an agent that has only a limited view of the world.

### 8.4.2. Use Well

This principle merely says that an agent should use the resources it has as effectively as possible. This should be the case whether costs can be computed or not. Associated with this is the need to discover effective methods. Discovering effective methods could involve the use of environmental intelligence. For instance, the machine might seek out and check with an expert whether a particular construction technique is best.

### 8.4.3. Need Only

This principle says that agents ought not be too greedy. An agent that has the tendency to grab all the resources for itself and its work might have an undesirable effect on the functioning of other agents in the environment. It might be able to achieve its goals very well but this has the tendency to result in a local optimum. Other agents might need these resources and if they can use them the total benefit accrued would be greater.

Since agents are no longer isolated but should work with other agents in the same environment, the resources should be shared effectively among the agents. Therefore it might be better for the resources to be requested and controlled by an agent on the basis of need. Associated with this is the injunction not to hoard resources. With self-interested agents and without this principle we have the danger that resources may be removed from general availability.

### 8.4.4. Conserve

If there is a choice between a method that results in irreversible change of a resource into an unusable product and one that permits some recovery, choose the latter. This principle might override the principle to use the cheap way or even the wish to use well.

The reason for this is that resources in the environment are limited — great though they might appear from the viewpoint of the agent. By taking a resource and using it so that it becomes no longer usable results in a reduction in the amount of available resources in the environment — an increase in entropy.

### 8.4.5. Reuse

Whenever possible, reuse. Resources in the environment are limited. By reusing resources the overall availability of resources will be greater. Humans in the past and present manage the use of resources, but in creating agents that manage resources autonomously we need to provide them with some sense of social responsibility with regard to the use of these resources. For instance, if you have a worker Mary on site, then try and keep her there and use her instead of hiring a new person. This might be in conflict with other principles.

## 8.5. Work Ethic

This is the last of the operational policies for an agent that this thesis shall look at. So far we have seen no policy that clearly says that a machine agent should even work. Surely one would not be talking about creating machines that have no wish to work. All machines should at least have the implicit ethic to work. Actually, work ethics are operational policies for the resource time. Although the earlier policies such as *use well* may have some applicability to time, time has the unique characteristic that it must be spent or wasted. To use time well means to spend it doing useful things rather than waste time.

Associated with this drive to work it the drive to avoid idleness. Therefore, if a machine has run out of work it should seek useful work from the environment — perhaps it could help another machine perform some task better. With increased autonomy and independence comes responsibility.

We should note the difference between laziness, used to conserve computational and memory resources, and idleness, which results in lost opportunities to get work done. Laziness conserves resources so that they can be used for other more beneficial and urgent tasks.

The drive to work may be non-beneficial if the cost of resources consumed for work exceeds the benefits of working. For example, if two machines were adequate to move an object but a third machine chips in to help, the additional benefit derived from the faster work may not compensate for the running cost of the third machine. Therefore, work ethic is a subordinate policy. However, it is very useful in many cases, such as in the use of computational resources of the agent. Unused computational resources do not result in accumulated computational power. When computational resources are not required for urgent matters, other less urgent computations can be done such as archiving, evaluation of experiences and learning, preventive diagnostics, long-range planning, information acquisition and so forth.

# 9. Architecture and Design of Management Systems

This chapter will discuss some of the implications of environmental knowledge on management requirements, the use of software management agents, the management of a large number of software agents, and the management of the knowledge environment. Incorporating environmental intelligence in management systems will make management systems more powerful and flexible.

## 9.1. Implications of Environmental Knowledge

### 9.1.1. Bringing More Knowledge to Bear on Problems

Let us consider the problem of finding the best architect. In the environment there are many architects, most of whom you know very little about. How do you find the best architect or the best lawyer? You might say that you have checked ten persons and this one is the best. What about the eleventh person, and so forth? You can keep going; you can always bring more knowledge to bear on an problem like this. Finding the best architect is an open-ended problem and is indeterminable until all agents have been evaluated.

In practical cases, faced with such a situation, an agent may first determine the amount of time or resources that can be spent on it and does its best. Doing its best means spending the designated amount of time or resources to bring more knowledge to bear on the problem until time runs out or the resources are exhausted. The search is done with the help of environmental intelligence.

In the above problem, one might learn about the architects one by one, first their identity, then their reputation, etc. Evaluate the new information each time. The solution potentially gets better and better.

Such a problem can be cast as distributed sorting. But the algorithms of distributed sorting cannot be implemented in the human knowledge environment. Instead, we can use concentration procedures where knowledge is sought by and brought to an agent.

A management information system or strategic business management system that can use environmental intelligence would be more powerful and effective. Both, but especially the latter, have a lot to do with events in the environment. It has to gather and process knowledge of pertinent events so that well informed decisions may be made. The next section discusses the use of more recent information in decision-making.

### 9.1.2. More Up-to-date Information

One typical example drawn from construction is the need to locate the supplier with the lowest price for some materials. The prices of materials are often very changeable. You can keep a list of suppliers and the prices they charged, but that is likely to be out-of-date. What you would like to be able to do is to obtain the information on demand. Environmental intelligence would help provide such a capability.

Not only would it be possible to poll suppliers for information, but it is also possible to discover suppliers you did not know about. Machine polling may be done by sending fax messages or by communicating directly with supplier computer systems. Information can be double checked with non-supplier agents and from historical data as well.

Locating suppliers and obtaining price information from them could be useful as adjuncts to both estimating and purchasing systems. Subcontractors would also find it helpful to have a system capable of locating potential clients and workers by tapping into various knowledge sources and databases.

### 9.1.3. Dealing with Unexpected Situations

One of the banes of computer and automation systems is their general inability to deal with situations for which they have not been explicitly prepared beforehand. The same holds true of many management information systems. This is because almost all systems are designed to deal with specific identified problems and needs. We can easily see the utility of such specific knowledge, and directly associate derived value and the costs expended to provide such knowledge. On the other hand, general abilities and knowledge appear to be quite useless. What problem is it for? What does it mean when one says that the knowledge or abilities to be provided are not for any specific problem?

As we conceive it, general abilities such as environmental intelligence and capabilities alone cannot make for a very productive agent. For example, the ability to move about does not seem to be very useful unless we add special abilities to perform tasks such as load and unload materials. The ability to communicate with other agents does not seem to have any direct utility — of what use is an agent that can only communicate but that does nothing useful? — unless the agent happens to need information to perform its task which it can obtain through communication with other agents.

Although intelligence and capabilities to use the knowledge environment do not seem to compare with abilities to do welding or solve other specific problems, such intelligence does create avenues that an agent can use to deal with various situations. An agent that has been ill prepared for particular tasks can fix its deficiencies. An agent that cannot perform a task alone can obtain help from other agents. Besides, if something totally unexpected occured, the agent has general fall-back positions.

For example, a machine has been designed for clearing the ground. The designer of this machine overlooked the problem of the machine getting stuck and, therefore, made no provision at all for solving this problem. The machine will remain stuck until someone becomes aware of its condition. We are quite aware of the possibility of unexpected events occuring and so the general management practice is to provide human supervision. However, human supervision, aside from being costly, can only deal with observable deficiencies. An agent who cannot perform a task by itself may avoid the task instead, and only sometime much later would the problems at the site be discovered.

If there are going to be some totally unexpected problems, then is it not impossible for an agent to identify them? The notion that an agent can identify the problem suggests that the designer of the machine system must have identified the problem beforehand and thus contradicts our notion of unexpected problems. We provide abilities to deal with the unexpected problems, not singly but as a group. In that sense, there might be no unexpected problems, but one does not need to identify specific problems. One way to do this is to concentrate on potential symptoms of problems regardless of their cause. Observing symptoms tells an agent that something has gone wrong, but it does not necessarily enable the agent to discover exactly what has gone wrong.

For example, we have suggested in Section 7.7 that agents not only generate and execute plans of actions but also generate predictive maps of the future and monitor them. If the predictions and the actual conditions do not concur, then something has gone wrong. Either the agent does not know how to generate the correct predictions — that is, its knowledge is insufficient or incorrect — or something is wrong with it. It may be that there is a failure in its sensory system or in its effector system. Depending on the sophistication of the agent, it might eliminate some possibilities and take actions according to the remaining possibilities. It might make some noises to indicate its distress and warn away humans in the vicinity, call for help from the repair station or other nearby agents if its remote communication systems have been disabled as well. The ability of the machine to initiate such actions would reduce the need for close human supervision.

We do not advocate eliminating the use of human supervision: rather, supervision efforts can be reduced and made more effective through the incorporation of environmental intelligence in machine systems. In addition, some degree of machine supervision can be used. We can create a machine supervision and management system that deals with both the observable and many non-observable deficiencies.

So far we have only dealt with the application of intelligence in field agents. Can management systems also face unexpected developments? Suppose we create a management system to handle a certain expected situation but the situation has changed as a result of uncontrollable environmental forces which makes part of the management system either inadequate or obsolete. Can the management system itself adapt? Human management systems can adapt to inadequacy by bringing in more management expertise, perhaps by calling the home office for help or hiring experts. A machine system lacking computational power might subcontract out computations to external computational contractors.

Since the construction site evolves, the management system for it ought to evolve to better meet the needs of the situation. The evolution and adaptation of the management system can be done purely through human effort or some of that may be achieved by self-driving characteristics of the management system aided by environmental intelligence. Environmental intelligence is not only useful to field agents but also to management systems since management tasks and problems have some similarity to field tasks and problems.

## 9.2. Agent bases

Over the years we have seen progression of software and information management technology. We had files to keep data sequentially and then databases to provide some structure and organization for the data. Problem-solving knowledge and algorithms are kept in subroutine libraries. Program packages which are problem-solving tools are created. Data and their processing algorithms are recombined with objectbases. What might the future bring? We have discussed the creation of software agents, so the next logical step would be to manage these software agents. One suggestion would be to create libraries of such agents for hire to those who need them to run automation systems.

Conceptually, agentbases are pools of software agents, analogous in some ways to manpower and subcontractor pools. They may be active or inactive. Active agents may be working or idling. Inactive agents may be suspended or dead. When the need arises, dead agents may be revived and inactive agents activated, then primed for the task at hand. Under certain circumstances idling agents may be decommissioned. One difference with the work by distributed artificial intelligence research so far is that these agents are grouped logically and physically according to various criteria. Each agent can be characterized in some fashion and each group or agents characterized as well.

Heavy equipment operators may, for instance, constitute such a pool of software agents. The agents may be characterized by the type of computer hardware on which they can execute and the level of expertise that they have. Summary information of their operational characteristics and past performance may also be available from agentbase managers or other environmental sources. This way humans and other machine agents can make intelligent decisions about these software agents through a higher-level understanding without the need to understand their inner workings.

For example, a manager might discover that he needs a software agent to run a piece of equipment, say a scraper or a truck. Working from what he knows about the requirements of the job, he tries to find the best match from agents in readily available agentbases — maybe the equipment agentbase owned by Smylar Inc. — hires that agent from the agentbase, and downloads it or has it downloaded onto the machine using a remote downloading facility. The same download facility may be used to download another agent for another piece of equipment. The agent is fed information from the site database and from management agents. After finishing the job, the agent is decommissioned and uploaded back to the agentbase. As we shall see, there is a reason for uploading an agent because agents are more sophisticated than programs which do not change.

The concept of agentbases has numerous similarities with that of federated databases. The principal difference is that agents are more powerful components than data or even objects. These environmentally intelligent agents contain not only data and data manipulation routines with reactive responses, but are autonomous entities capable of taking independent action. For instance, an agent might hire several other agents to help it. This goes on recursively. It may have the knowledge and abilities to obtain information that it might need from environmental sources of information such as databases. It can seek additional computational power and other resources from the user.

Agents are very powerful and the level of human-machine interaction should be at a higher level than that permitted by databases or objectbases (databases of objects). An agent may also learn and improve after its stint on the work site and return to the agentbase more capable than when it was hired. We can start off with a generic software agent for handling a wide class of construction equipment and end up with a specialized one for handling scrapers or trucks. We keep a work history of the agent — such as the type of jobs it has worked on, its performance on these jobs and so on — which can be used in the selection process. A higher price may be charged for hiring more experienced agents. It is also interesting that software agents can be hired by other software agents, leading to self-determination of machine organizations to achieve particular goals.

To make it all possible and practical, especially with regard to the use of data by these software agents, the techniques developed for federated databases are essential. Data mapping, representation transformation methods and conflict resolution methods may be applicable. But because agents do not need to treat each piece of data in isolation but may be able to associate them to what they know already, they can derive greater benefits and reason and act with more intelligence.

## 9.3. Management Agents

Management is not the sole purview of human managers; it is quite possible and in some cases even desirable to develop software agents to relieve the burden of managing automation systems. As with human management systems, we can have line management, functional management and various hybrids thereof.

In line management we might have a machine charged with supervision of other agents. Proper management typically requires the manager to provide management services and management control. Management services may include some types of conflict resolution and the provision of various information. Line management control may include planning, delegation, resource allocation and performance monitoring.

In functional management we might have an agent provide special services which may be called upon by line agents. There are also both service and control aspects of functional management. Management services may include temporal synchronization, traffic management and data management. It is often useful for functional managers to control shared resources such as inventory and equipment. Functional managers can also exercise purchasing control and personnel control.

It is clear that there is a great diversity of management agents that can be created. The choice and design of these management agents would depend on the task and the environment. One might use "foremachines" to supervise the work of crews composed of other machines, and include a human with overriding power to supervise the foremachines in one scenario. In addition, one might provide fuel and power-supply machines and their managers.

## 9.4. Knowledge Environment Management

Whatever can be said for autonomous intelligent agents, we still believe that many benefits will be unattainable just with unstructured purely distributed systems having self-serving agents. Some kind of structure makes the environment more secure and productive. Chapter 10 discusses some of the environmental mechanisms that are in the existing knowledge human knowledge environment. Chapter 11 shows how some of them might be used in managing automation at a construction project. This section only briefly mentions some of the possible management techniques.

When we create a management system, we are creating a working environment for agents as well. As managers we would like to put certain services within easy reach of the agents and thereby improve their productivity, and provide certain controls to discourage unhealthy exploitation and uncooperative behavior. This is what knowledge environment management is about.

### 9.4.1. Clearinghouse

One mechanism that has been proposed by other researchers [Oppen and Dalai 83] in the context of office work is called the *clearinghouse*. Oppen proposed a decentralized agent to act as the clearinghouse to help other agents locate office objects. It is one example of services that managers of the knowledge environment can provide.

In view of our ideas of environmental intelligence it is not necessary for such clearinghouses to have all the information they may need. They can be given the ability to obtain the information and thus satisfy information needs on demand.

### 9.4.2. Contact Management

In managing the work of several interacting agents, we would need to provide facilities for agents to contact and meet with each other. This includes various communication systems and devices. Conference facilities can be provided to support meetings.

Machine agents might be supported by computer networks and radio communication facilities. Blackboard systems can be used for meetings of machine agents. To a degree, such systems might be extended to accommodate human agents as well.

The interfaces between humans and machines have to be provided as well. Machine-to-human communication can be supported by voice and display technologies. Human-to-machine communication can be supported by voice, pointing devices, scanning devices and active image interpretation devices.

### 9.4.3. Distributed Control and Mediatory Mechanisms

Another useful mechanism is the use of market-like systems for distributed control. Resource allocation can be controlled in this fashion. Some type of task distribution can also employ such methods.

We have already seen that the needs of agents can be mediated through tokens of exchange. We should understand the difference between mediation and control. In mediation we want to bridge two entities whereas in control we influence the decisions of the agents. A single token system may have both functions!

There are two major classes of systems that can be created. In one system we have sources and sinks (Figure 43), whereas in another we have a cyclic flow of tokens (Figure 44). The distribution of such tokens can be placed under the control of high-level managers, especially in the case of former. If we use tokens of exchange, as in the latter, then the system becomes an economic system, and the amount of such tokens is held relatively stable.

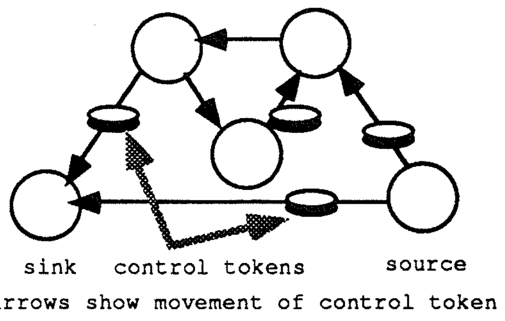


Figure 43. Sources and Sinks of Control Tokens

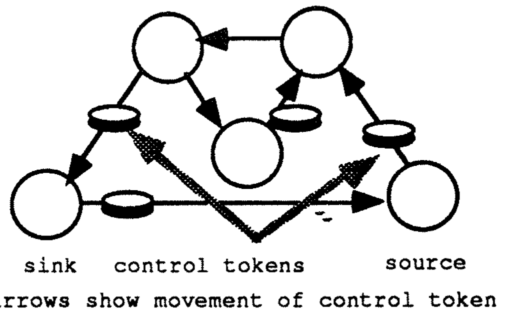


Figure 44. Cyclic Flow of Mediatory Tokens

Many different types of tokens can be used in a single system to provide more control. The use of different types of tokens is especially useful when the dynamics of the subsystems are different. A subsystem that is meant to adapt and evolve slowly should not be subject to the wild swings of some other subsystem, which could happen if we couple the two systems together through the use of a single token. However, we should be aware that using different types of tokens will tend to increase management complexity.

Tokens of exchange are not the only mediatory mechanism that can be used to facilitate management. It is possible to develop systems with agent intermediaries. However, such systems tend to have bottlenecks.

### 9.4.4. Commitment Management

On a larger scale we see the use of courts as mechanisms to resolve conflicts among agents and confer meaning to contractual obligations. Without such control mechanisms, use of commitment would be nearly impractical. It is partially the lack of such mechanisms that makes the attainment of meaningful agreement among nations difficult. Cooperation would then degenerate. There are many other types of knowledge environment management techniques and mechanisms which might be adapted from the human knowledge environment to human-machine knowledge environments.

Since it is often too easy to forge and erase electronic data, hardcopy trails of commitments have to be laid. Provision should also be made for recourse if commitments are not met. How could machines be responsible? Basically, the same question might be asked of human agents. One is to provide the proper evolutionary environment to weed out irresponsible agents. Machines that are not responsible will suffer demerit costs that tie back to their designer and organization and eventually such machines and designers will be forced out of the system. On the other hand, responsible machines and their designers will have a reputation for reliability and continue in the knowledge environment. The other is to provide the proper remedying mechanisms. Just as in contract negotiations among organizations, various bonds may be required. In non-monopoly systems, these measures will drive toward better designs and reliability of machine systems.

### 9.4.5. Security Management

Some provision of security at the site has to be made. Methods of identifying, characterizing and tracking agents are useful for both operational management and security management. Section 8.2 discussed identification and suggest a method for implementing identification in a distributed manner. One of the problems was that agents might impersonate other agents after obtaining identification information from the other agent. A pure direct password system, for instance, may not work for this reason. A secure third party is needed for stranger-to-stranger interaction. For the initially known party to known party interaction, some sort of password system can be devised.

The identification problem is not easily solved. We also have to devise means to prevent the impersonation of secure third parties! It is very difficult if we have an open environment because we cannot assume that new agents entering the environment will have the knowledge about secure sources of information and services.

Fortunately, unique and certain identification is not always necessary for agents to interact. If transactions involve immediate exchange, then monitoring of the exchange could be sufficient without either party knowing the true identity of the other.

Some redundancy that makes cross-checking possible can be employed to promote system security. We have seen this used extensively in accounting practices and it makes sense to use it for other security needs as well.

Separation of powers might also be used to provide additional security. We have seen the successful use of this technique with governments and accounting.

# 10. Environmental Mechanisms

[Werner 89] provide an excellent analysis of value of social structure and communication pragmatics as a framework for multi-agent interaction with emphasis on attaining social goals. In this chapter, we will look at some of the mechanisms in the human knowledge environment that promote useful multi-agent interaction. Many of these techniques are well developed in human systems, but we are interested in using them in machine systems. The following discussions will provide us with guidance on how multi-agent machine systems might be constructed.

## 10.1. Service Centers

The knowledge environment has several types of service centers, the existence of which indicates the non-homogenity of agents in the knowledge environment. Service centers may be generally passive or may actively market their services. The level of knowledge they purvey can range from raw data to recommendations. The use of service centers makes the flow of information in the environment more efficient and reduces the amount of knowledge that an agent needs about the knowledge environment, the former through the reduction of duplication in processing information and the latter through reduction in the number of environmental components that an agent needs to know about.

The relatively long life of service centers as opposed to the duration of agents' activities gives us another advantage, which is the ability to obtain good characterizations of these centers. Thus service centers can develop reputations in accuracy, timeliness and scope which may be relied upon by other agents. Although agents operating these centers may change, the reputation of these centers persists.

Some mechanism is needed to set up and maintain these service centers. Service centers may be supported by funds from governmental agencies, from donations or from the sale of services.

### 10.1.1. Data and Information Purveyors

Sometimes unprocessed or lightly processed information is what agents need. Historical information is one such type of information. For historical information, archiving is often used. Processing and use of the information could take place at an indeterminate point in the future. The value of historical data may grow with the passage of time. Archives are thus data centers and purveyors.

Another type of data which are stored in their lightly processed form is accounting data. Unlike historical information, accounting information is frequently processed to obtain information for strategic decisions. The value of the data for control is short-lived. The data must, however, be kept for a longer time for auditing.

Raw data is frequently kept in the case of scientific experiments and is provided to other researchers. This is because the interpretation of the data may be debatable and the data has to be available for checking alternative theories.

### 10.1.2. Knowledge Purveyors

In most cases, information is processed before being provided to other agents. Knowledge purveyors typically screen information. Information compression or elaboration are frequently done as well.

Among the types of information supplied by knowledge purveyors are rankings, statistics, trends, financial summaries, evaluations, commentaries and theories. The *Consumer Reports* magazine typically provides evaluations of consumer goods and rankings of consumer items from different consumers according to meaningful criteria. Textbooks provide an agglomeration of well investigated techniques on a subject of wide interest. Think-tanks produce high-quality assessments of situations which are disseminated to policy makers. The better business bureau provides information about business organizations. Educational institutions prime agents with coherent bodies of knowledge.

There are also knowledge purveyors who provide information of general interest and for entertainment. Weather information comes from analysis of data from various metrological devices and radar scanning — the raw information would be of not much use to anyone. But knowing the weather at their location is of general interest to agents.

## 10.2. Information Channels

In Section 4.2.2 we discussed some of the environmental mechanisms used for transporting information in the knowledge environment. These include mail, radio, television, telephone, courier and voice.

They are means for affecting the transfer of knowledge from agent to agent. Not only do we have such means for communication, but we deliberately set up regular systems whereby information moves to us and from us, and these mechanisms are called information channels. Information channels may use any of the transportation mechanisms. If an agent desires a regular stream of information based on some source, it can set up a channel of information from that source. This is more effective than looking for such information sources each time it wishes to look for information. Information is thus more easily available.

For example, we can read particular journals in our field of research in the local library. But if the field is important to us we often subscribe to the journal directly. By subscribing we set up a reliable and timely means by which information important important to us is delivered to us.

### 10.2.1. Publication and Subscription

How are information channels set up? One of the environmental mechanisms is subscription. Basically, a subscription is an agreement consummated with a knowledge purveyor for information which the knowledge purveyor has compiled to be sent to the agent desiring the information. Such compilations are usually done regularly. This creates a controlled flow of information from a knowledge source to an agent.

But how does the knowledge purveyor obtain his information? There are several ways. One important mechanism is publication. Agents who have information to share seeks to have it published in appropriate sources. The knowledge purveyor will screen the information so that only information relevant to the focus of his publication will be published. Some policy mechanisms are also present to reduce the redundancy of publication; that is, the same information is published by too many knowledge purveyors. Too much redundancy will make the publication system inefficient and expensive.

### 10.2.2. Advertisement and Mass Broadcast

Publication and subscription serve specific information needs of agents. For general information needs, mass channels are better. Several types of mass channels exist in the human knowledge environment and they have different characteristics.

One situation where mass broadcast may be appropriate is when an agent does not know the audience for the goods or information and the potential audience does not know about the goods or information either. If the former does not hold, then direct mailing might be used to link up the two parties. If the latter does not hold, we might rely on the consumer to take the initiative to contact the purveyor. There are, of course, situations in which the two approaches mentioned are not as cost effective as mass broadcast. Mass broadcast channels of information might then be used, but the practice should be weighed against the cost to non-consumers of the information.

One way for the knowledge purveyor to obtain mass-broadcast information is called advertisement. It is similar to publishing but frequently differs in two respects. First, only very light screening on the information is done by the broadcasting agency. Cost rather than content controls information that is to be distributed this way. Second, the information is frequently repeated. This is because mass broadcast information typically has a short persistence. Mass broadcast information needs to have short persistence; otherwise, the environment will soon be awash with information.

## 10.3. Transformation and Modification of Goals

### 10.3.1. Facilitation Mechanisms: Tokens of Exchange

The original goals of an agent might not be easily met directly. It is the elegance and beauty of the human-evolved environment that a great number of these goals can be transformed or modified into other goals that permit transactions to occur with other environmental entities that would eventually lead to the satisfaction of the original goals. One of the mechanisms the environment provides is that of tokens of exchange. Currency is the most well known of these tokens of exchange, but other tokens also exist. One of the important aspect of currency is the persistence of their value. Perishable goods, for instance, would be a very weak choice for use as tokens of exchange.

Let us see how the transformation of one type of goal into another is made through the mechanism of tokens of exchange provided in the environment, and how that could lead to the satisfaction of the original goals.

For example, take a human agent, say Mary. Her desire is to have shelter. One type of shelter is a house. There is a change from the abstract desire to have shelter to the concrete desire to have a house. Let us say Mary owns a piece of land large enough to build a house and wishes to have her house there. So Mary now goes from the desire goal to have a house to the intent goal of building a house on her land.

Let us assume that Mary knows all the activities and resources needed to build a house. Therefore, she can generate a production plan for building her house. She is still unable to fulfil her desire because she does not have the materials nor the ability to build the house.

To see what the existence of facilitation mechanisms in the environment does, let us suppose there were none. Some of these materials may be lying around in the environment and Mary can gather them. Still there are likely to be agents in the environment with materials and other agents with abilities. In that case, Mary will have to barter with specific agents for the materials and with other agents for their help in building her house. For that she will have to have something to exchange directly. Mary talks to them and finds that some of these other agents need to eat and one of them needs a pair of shoes. Therefore, Mary might hunt some animals or gather some food for exchange. Unfortunately, Mary does not know how to make shoes. One possible way out is for Mary to find someone who knows how to make shoes. This goes on. We see that the intent goal of building a house has expanded into intent goals for hunting animals and finding a person who knows how to make shoes: Two or more different intent goals. This makes Mary's life difficult.

Let us see what Mary needs to do if the environment were more favorable. In this favorable environment one can obtain materials for money and hire other agents for wages. In this case, Mary needs to find out how much the materials and help would cost her in terms of money. Mary now has to figure out how to obtain that amount of money. We see that Mary's intent goal of building a house becomes the intent goal of obtaining a certain sum of money.

Thanks to the favorable environmental situation, there are plenty of ways one can obtain money and Mary knows of a stock of such methods. So Mary finds the one which most suits her. Maybe that is hunting or maybe that is working in a factory. If it is hunting, Mary sells her game for money. With the money she can then buy materials from other agents and get the help of other agents to build her house. It is so much more convenient for Mary and not only for Mary but also for the person needing the shoes. The latter can work for Mary for wages and use the wages to get his shoes. Mary does not need to find the agent who makes shoes, that is the task of the one needing the shoes. We see that the ordering of the different agents' activities has been loosened a little and the responsibilities have changed somewhat. Obtaining money is the universal mediating goal for most agents in the environment.

We will realize how much easier things become as a result of the existence of tokens of exchange in the environment when a planner that plans for Mary is developed. This is especially so when Mary has many different goals. For example, she may want a house, a car, a television set, shirts, education and TV dinners. Suppose we want to develop a planner that can plan for the achievement of these goals. The agent's planner takes as input all these desires and transforms them to the goal of making money additively. The planner then expands the goal of making money selecting from and using the stock of methods of making money that the agent knows about. Finally, on having the money Mary uses some of her environmental intelligence to plan and capabilities to carry out activities to obtain goods and services from sources thereof in environment. The binding between the agent's needs and desires with particular methods have been loosened. This is graphically depicted in the figure below.

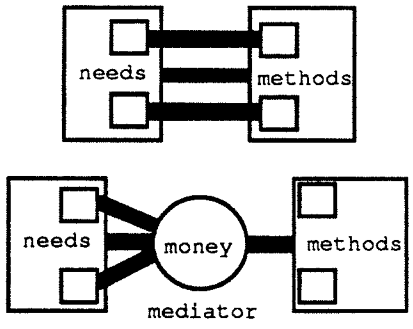


Figure 45. Break in Binding Between Needs and Methods

Tokens of exchange is a remarkably useful and elegant mechanism. Among other important benefits, this mechanism contributes to the reduction in the amount of things that environmental agents such as Mary need to know. For example, Mary does not need to know about IC chips just in case there is someone with IC chips to trade against something Mary needs.

There is also an enhanced temporal separation between needs and desires and obtaining the means of satisfying them. The chronological order is no longer that the needs and desires come before action can be taken to satisfy them. Not only can Mary earn money after her desires or needs arise, but she can prepare by earning money in anticipation of future desires or needs. She knows that money can be used to satisfy them.

What we are doing here is to introduce mechanisms for transforming a goal into a totally different one. In particular, we want local goals, arising from local desires and needs, to be transformed into goals that can be shared among many agents. The goal is not global. The environmental mechanism works because a large portion or all agents want it to work and benefit from its function in bridging them. We highly advocate its use in machine systems.

An aside, the problem is that it works so well that what is merely an environmental mechanism has been elevated to fundamental status in the beliefs of many human beings. To support this thesis, I would point to the assignation of deity to the sun by ancient human beings. The sun was critical for agricultural pursuits and life in general and so the ancients cast it into a god.

### 10.3.2. Packaging

The technique here is a little more subtle. Several items of possibly individual transactions are combined together in a single transaction. For example, instead of selling only the items that potential customers want, you sell them items which they want together with items which they may not need. Perhaps the latter items have not been very popular. These items are packaged together and sold as a single item. These additional items or deals have been piggy-backed. If the consumer is sufficiently motivated, he will buy the items he does not need together with those he needs, because he cannot obtain only those items he needs.

Packaging is extremely common in contracts. Contractual terms are not all favorable to both parties. In order to match the needs of both parties to a contract some terms undesirable to one party may be packaged with others. Let us see how packaging works with an example.

For example, Joe might present the following contractual package to Mark. The package says that Joe will give Mark $500 if Mark agrees not to bid on the project. Mark would like to bid on the project, but he also likes the $500. If the overall package is attractive to him, Mark will accept. He cannot accept any of the items of the package separately.

Clearly, packaging is a technique that is useful when performing negotiations in the face of self-centered and self-serving agents. It is a mechanism that allows deals to be made, and thus cooperation to proceed, that are otherwise difficult or impossible to achieve. Packaging is especially useful where tasks or goods are of different value to the different participants. Bringing them together in a package helps satisfy all parties.

This technique, properly used, has the effect of balancing the benefits among agents in addition to its role in facilitating transactions.

### 10.3.3. Cloak and Manipulate

This is a troubling technique although it is observed in use by humans. Basically an agent camouflages its own goals so well they cannot be easily determined by the other party. Instead, the other party is made to believe that the agents' desires or goals are something else.

This technique is frequently used by humans because many believe that agents will not transact with them unless these agents believe that the transaction is in their best interest and only from that viewpoint. Unfortunately, many transactions cannot both be to the other party's best interest and the agent's own best interest in the short run, so resort is made to such a tactic as a facilitation mechanism.

It is debatable whether human agents are purely self-interested. The rationality and desirability of this mechanism is questionable.

### 10.3.4. Reformulation

In reformulation of goals the original goals are preserved but presented in a different light. The same thing is accomplished with the transaction but the agents feel better because it is less crude or offensive. When it is done in speaking or writing it is called euphemism. The original idea or thing is preserved through the reformulation. This is done in relationships among humans. This is not a cloak because the original desires and goals behind them are easily understood by the parties.

### 10.3.5. Yes+

We introduce a notion called yes+. This is preliminary to the next section on reformulation+. In many transactions an agent has the ability of presenting choices to the other party. A very simple choice is yes or no, take it or leave it. This is a simple choice for the other agent to make but it does not fully exploit the possibilities of transaction. This is the case even if we permit negotiation. Take, for instance, the bilateral negotiation paradigm described in Section 5.1.1.2. When a proposal is made by one party, the other party basically has three options — two basic and one to further the negotiation. When a proposal is accepted the entire proposal is accepted, and if rejected the entire proposal is rejected.

We believe that in many transactions we do not need to be so harsh. We can leave it much more open ended. Not only can an agent say yes or no to a proposal but it can say yes+ and the deal is closed. Let us see this in a more concrete way.

Agent A and agent B are in a transaction. An agent A may have the following open ended proposal,

*"Agent B, take any number of items you wish, this is my price schedule."*

Agent B has a yes+ decision to make. In accepting the proposal agent B can say yes and also specify the number of items he wishes. This results in a commitment of both agents. Agent A no longer has to say yes to agent B's yes.

### 10.3.6. Reformulation+

When I started writing about this concept as a generalization of the notion illustrated by $W^2$ in Section 8.3.2, I could not find a word that accurately describes the phenomena to put as the subtitle. The main difference between this concept and the one stated in Section 10.3.4 is that in this approach modifications of the original goals may take place.

Packing differs from reformulation and reformulation+ in that the items of desire did not change but we have hooked something to the transaction. We bind a deal that is less desirable for the other party to a deal that is very desirable for the other agent. In facilitation we introduce mechanisms for transforming a goal into a different one. In particular, we want local goals to be transformed into goals that can be shared among many agents and therefore permit a level of interaction much better than before. Now that we have cleared the air, let us see what this concept is about.

In reformulation+ we take one way of presenting choice to the other party and formulate another way to present choice to the other party. What can be achieved for the purposes of transaction by one set and another set of choices typically differ.

The $W^2$ philosophy is the philosophy of making and presenting choices that results in benefit to both parties to a transaction whenever possible. "Whenever possible" means that the agent is willing to forgo even greater gains in favor of win-win — win-win is not a principle of pure self-interest. Fair benefit is not fundamental to $W^2$ but is a compatible notion. We have seen in Section 8.3.2 that this is a powerful notion because even without iteration the $W^2$ principle solves the prisoner's dilemma.

## 10.4. Public Institutions

Typically, public institutions are agencies for maintaining environmental order that is difficult or impossible to maintain by lack of authority. Examples of such institutions are governments and their agencies.

We have discussed the use of commitments among agents in Section 5.2 and elsewhere, but commitments among agents have no practical value if there is nothing to back them up. If the environment does not provide public institutions for this purpose, then an agent has to enforce the commitments made to it by other agents, and this is normally difficult or impossible. For example, it might require the application of force, but a single agent might not have the necessary force against, say, a firm. Public institutions make it possible for meaningful commitments to be made between disparate environmental entities and remove the requirement of force.

Public institutions can enforce standards, conventions and law. Although we do not expect agents to be generally exploitative and not keep their commitments on a single construction site, it is quite possible for commitments to be broken by agents such as suppliers and subcontractors. If we do not have some means to enforce commitments made by these agents to agents at the construction site and vice-versa, then such interactions cannot be meaningful. Since neither of these parties can enforce these commitments, enforcement must revert to an independent third party.

## 10.5. Framework for Interaction

We have discussed the operational policies that we might wish to incorporate into autonomous machine agents. That does not fix all the problems of multiple agents working in an environment. There are matters to take care of to create an environment conducive to the functioning of multiple agents. In the human knowledge environment we have the following framework that helps to provide an adequate context for agents to interact:

i) standards
ii) conventions
iii) rule of law
iv) social principles

In particular, we should not underestimate the importance of standards and conventions.

### 10.5.1. Standards

Often there are many ways to do things. One way may be better than another for a particular purpose. There may not be one best way to do everything. This might result in agents choosing to do things differently. This is one of the reasons for diversity. However, in many cases doing things differently is merely a matter of inertia. The world may have changed to take advantage of improved methods of doing things but old methods of doing things remain.

For example, take the case of imperial or metric units. Each represents a different way of measurement so fundamental to communication of information. If agents do not standardize on one or the other, then the communication workload must increase because it becomes necessary to know which of the two standards the agent you are communicating with follows before the numerical information can be interpreted correctly.

Standardization is the prior agreement of a community of agents to use particular ways of representation, communication or rationalization. Agents who subsequently enter the community do well to learn and observe the standards.

Standardization makes the most sense in an environment with multiple agents. If there is only one agent, then choosing the optimum way to do things is the most appropriate and, if any way is equally suitable, then an arbitrary choice can be made. This is not the case when an agent has to give due consideration to interaction with other agents in its environment. Standardization helps to make interaction between agents easier and less costly. This is reflected in human society. The areas receiving strong push for standardization are

i) measurement of time
ii) language,
iii) units of measure,
iv) currency,
v) codes, and
vi) laws.

Standardization of the first item is so universal that it is almost unnoticed. This standardization has profound and widespread effects on the function of agents in the knowledge environment. Among other things, it permits easier coordination and synchronization of activities of agents. Coordination could become much more difficult if agents follow different ways of measuring time. Before we had such standardization, we could coordinate through the use of the "heavenly" clock as the *de facto* standard. The coordination of two separated generals and their soldiers to attack at sunrise is an example.

The next two items are strongly tied to the need for agents to communicate information. With respect to language, there is usually a need for a language that everybody can use for general-purpose communication. Other languages can be used as preferred or convenient, but there should be at least one language that any agent can use when meeting a complete stranger.

Standardization of currency is important for smooth transactions between agents. Its role as a mediatory mechanism between agents will be diminished if there are several currencies which are not inter-convertible. Standardization of codes and laws reduce the amount of information that agents need to know and thus increase the predictability of the environment. We already saw that standardization will also reduce communication load.

### 10.5.2. Conventions

Conventions prescribe behavior; that is, the use of conventions is a way to standardize certain types of behavior. That agents follow conventions is especially important where there are incompatible alternatives, such as

i) in the sharing of space,
ii) the sharing of communication medium,
iii) communication, and
iv) transactions between agents.

In the next few sections we shall describe and discuss a few conventions. We should be reminded of these conventions when we try to develop multi-agent automation systems.

### *10.5.2.1. Right of Way*

Right of way on the road is a very important convention. For instance, motorists in America drive on the right side of the road while motorists in Britain drive on the left side of the road. Clearly there is nothing fundamental about which side of the road one can drive on, but total chaos would result if there is no convention that clearly states which one prevails. Traffic lights and the order with which cars cross a four-way stop all have their conventions, and these permit greater and safer traffic flow.

Suppose we have two machines moving toward each other in a narrow corridor that is wide enough for only one machine. Which machine should yield? Without prescribing some convention to this situation it is not clear why either machine should yield. Perhaps the machine with the lower priority task should yield or perhaps the faster moving machine should yield to the slower one. There is rationale for both approaches but without prescribing the one behavior that should apply we can get into a bind.

### *10.5.2.2. Communication Protocols*

When we call someone up over a telephone, there is a pattern in the initial communications. First we have to establish some minimal information about the other person (or machine) at the end of the line. We say "hi", "hello" or any of a number of greetings and expect the person to reply. The reply and pause following the reply indicate that someone is at the end of the line and listening.

The other aspect of using a telephone is that for the most part only one party should speak. If the other party wants to speak badly, he interrupts by speaking at the same time or during a pause in the conversation and the first party, by convention, should stop speaking and let him speak. Conventions of this sort makes for more orderly conversations.

### 10.5.3. The Rule of Law

The law both prescribe and proscribes behavior. The rule of law provides a known framework under which agents interact. Rules are needed in both cooperative and competitive situations, but more so in the latter.

One of the most important is the rule of law in transactions also known as contract law. Clearly, it is best if agents can and will fulfil agreements and commitments they have made. However, if such agreements or commitments are broken with damages to one or both parties, then there should be some recourse for the disaffected parties. The specification of this recourse is contract law. Without the rule of law, agreements might not be made in the first instance.

Why might this be relevant in construction automation? There are many agents on the construction site, such as subcontractors, or off the construction site, such as suppliers, all of whom have their own motivations. Without the rule of law, agreements made with them are only paper agreements. Violations of their agreements or commitments are bound to have harmful consequences on the project. We cannot allow such violations to occur freely and without recourse. Just as we need the rule of law with human agents, we also need such a framework of interaction when machines interact with human agents.

### 10.5.4. Social Principles

Neither standardization, use of conventions or the rule of law fully determine the behavior of an agent. We do not necessarily want to standardize everything and overly constrain the behavior of agents in the environment. As far as possible, we want agents to willingly subscribe to standards, conventions and the rule of law, because if such standardization or laws does not have a good benefit to costs ratio, then agents will not support them and they will eventually fail in an environment with decentralized power. Nevertheless, standardization, conventions and law have to be enforced.

In cases which do not need the enforcement of standards, conventions or law, the interactions among agents could be improved with the application of social principles. These are principles to which agents who wish to use them subscribe and are not enforced. An agent cannot rely on other agents to follow these principles. As a result, social principles are weaker and will not work as a method to control traffic behavior, for example.

What might constitute social principles in human society? For example, chivalry had been around for a long time although it is now defunct. Others, such as giving way to the handicapped and elderly, still exist as social principles of behavior. There is a whole host of other human interactions governed by social principles.

What kinds of principles might we like to have for automation systems? principles to which agents are free to subscribe but which are not necessarily enforced? For instance, it might be useful if agents regularly informed other agents of their addresses or location changes. Such a principle is not generally enforced. Nevertheless, it is beneficial to both the agent subscribing to it and to other agents and so we would like to build such a social principle into our machines. We might also wish to have new agents introduce themselves and be introduced to other agents. Such things do not need to be governed by law or convention, but would nevertheless be useful operational principles in multi-agent environments.

# 11. Preliminary Conceptualization of Construction Automation System

Many aspects of the knowledge environment and some techniques for creating machines that work in such an environment have been proposed in this thesis. It is not necessary or even desirable to deal with all the issues or with all the techniques. It is up to the implementors of construction automation systems to choose the level of functionality to implement. They should understand the value of each facet of the technology and take into account the costs involved. It is possible, for instance, to implement some aspects of lazy planning and some small aspect of environmental intelligence and leave it at that.

Others may be more ambitious and attempt to implicitly control reasoning and memory resources with the help of planning horizons and explicitly with the help of budget management techniques. Others may be even more ambitious and try to build systems that have a significant amount of interaction with human workers and managers rather than just with automation specialists and monitors.

We do not think that it is either necessary or desirable for a single structure for all automation needs. There is no need to have a centralized database nor is it necessarily a bad thing to have one. There is no need for a single language to communicate everything nor is that necessarily a bad thing to have. There is no need for a consensus reality but we might benefit a lot if we do employ a single reality.

This does not mean that this thesis does not make important proposals. In particular, we advocate that a substantial level of environmental intelligence be provided to some machines. We advocate the development of a general core of intelligence. We think there is a need to define and enforce standards and establish certain conventions. We should use policies that encourage cooperation among agents and conserve resources. Agents should be identifiable. There should be mechanisms in place to provide proper remedies if commitments among agents are not kept. There are different approaches to provide these features. Modularization and reduction of dependencies among systems and agents will provide us with flexibility in creating automation systems and allow such systems to evolve with growing or diminishing need.

In this chapter we attempt to bring some of the pieces together for a overall view of how construction automation systems might be installed, operated and decommissioned. It represents one of the possible ways we can design an automation system. We shall look at three levels of engineering, those of agent, site and world. We shall clarify though the partial description of an automation system for large building construction.

## 11.1. Agent Characteristics

### 11.1.1. Agent Types

We might roughly categorize agents at a building construction site as belonging to production workforce, support workforce and interface workforce. A production workforce would include agents capable of a variety of tasks at the workface. A support workforce would provide for many of the needs of the production workforce. An interface workforce would provide some of the access to the world as well as buffering against it. These types represent decomposition along the functional dimension. It is quite possible to have an agent act in all three capacities.

In building construction, the production workforce would include agents or systems of agents capable of excavation work, concrete work, steel reinforcement work, welding, wood work, surface material movement, vertical material movement, pipe handling, plumbing work, electrical work, fixture installation, concrete surface finishing, painting and so on. Some agents might perform more than one task.

The support workforce would include agents providing or setting up sources of information, materials and energy. The support workforce would also provide diagnostics, coordination, general planning, resource allocation and so on.

The interface workforce would provide permit, material and equipment acquisition from the external environment. This workforce also interfaces with other agents that provide goals, plans and resources.

### 11.1.2. Agent Intelligence

As we saw, there are many types of agents. Some of these agents on the construction site may be machines while others may be humans. Machine agents at the workface would have to have specialized knowledge and abilities to enable them to perform their tasks. In addition, they may be provided more general abilities such as mobility and interfaces with information systems and other agents. Their environmental orientation should include abilities to avoid collision with other agents, which are among the things needed to be good citizens of the environment. Some of them may also perform process planning and self-diagnostics.

Management agents, which could be either humans or machine, would need abilities to plan and coordinate other agents. Work would be scheduled so that unproductive interaction at the workface is reduced or eliminated. These agents would also actively obtain the information they need from sources in the knowledge environment.

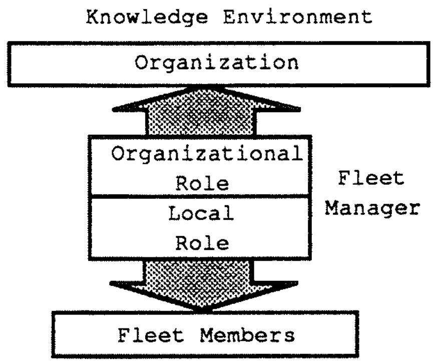


Figure 46. Functional and Organization Roles of Fleet Manager

Let us consider a carrier fleet manager. Does it only need fleet management expertise? It needs fleet management expertise but also needs abilities to interact with the knowledge environment. For instance, such a manager should have the ability to negotiate routes with other fleet managers, obtain information needed for planning fleet operations and deal with unexpected problems with the fleet. Thus we see that the fleet manager has an organizational role in addition to its primary functional role as depicted in Figure 46.

### 11.1.3. Agent Interfaces

An agent may have to interface with machines, humans and the world. The method of communicating between machine and machine is likely to differ from that between human and machine. How information is presented, received and interpreted differs in each case.

By structuring agents in organizations, we may be able to buffer certain agents from direct interactions with the world. To a lesser extent it is also possible to buffer some machines from human agents.

Several types of communication devices may be used. Simple languages can be created that can be easily learned by humans and understood by machines but still be very useful for general agent interactions.

### 11.1.4. Agent Structure

Machine agents would be highly modular in structure. The physical structure of the agent might comprise the mobile platform, power system, sensor system, communication system and specialized manipulators and effectors (Figure 47). It might be wise to provide a separate backup power system for the communication and reasoning subsystem. Note the choice of modularization along mainly functional lines. However, the mobile platform might be a self-contained system, inclusive of power supplies and basic sensor systems.

Agents preforming different tasks may use the same mobile platform. The characteristics of the platform constitute the self-knowledge that the agent has concerning this subsystem. Part of this knowledge will include the terrain, speed, acceleration, weight, carrying capacity and power requirements. This knowledge is then integrated with knowledge that it has about other subsystems. Thus, it is able to reason about these characteristics and plan its movement in conjunction with other activities. The dynamics of the entire machine, such as the braking time and tilt-over moments, would depend on the way these subsystems are assembled. The sensor and communication subsystem might also be common for machines performing different tasks. Additional special sensor systems might be needed for machines performing specialized tasks such as welding.

In order for it to be possible to assemble modular hardware components together, special attention has to be given to the design of each piece to provide for standardized interfacing fixtures. In particular, thought should be given to the problems of supplying power and information to the various portions. Otherwise, considerable effort would be required to design and build a number of machines, because every machine would almost have to be designed and built from the ground up. There is still a lot of research needed on the techniques of hardware modularization before extensive modularization becomes reality.

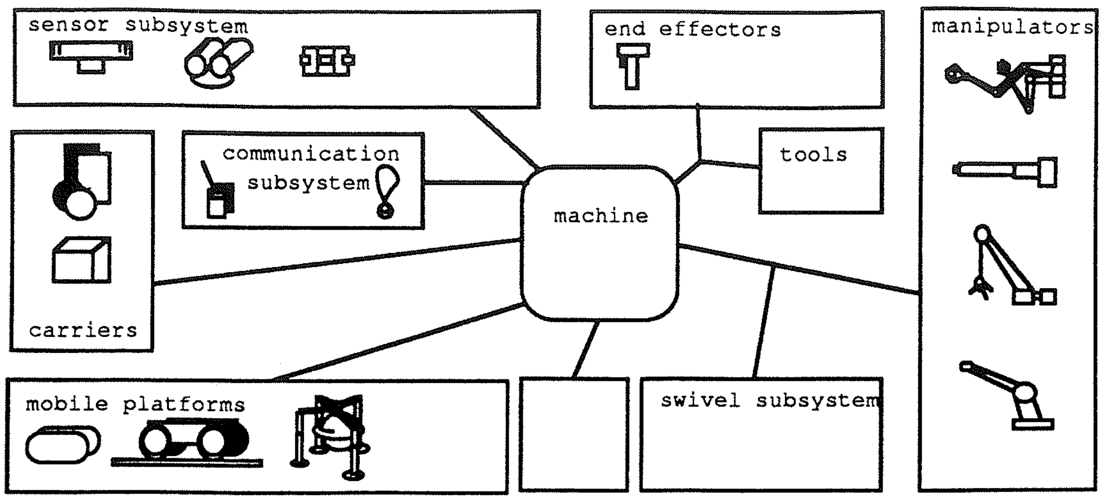


Figure 47. Components of Machine

The software structure of an agent should also be highly modular. Self-knowledge can be built from utility knowledge about each subsystem and how they interact with the other subsystems. Special knowledge and particular diagnostic capabilities may remain separate. Knowledge management, agent interfaces, activity management are all subcomponents of the agents software architecture. They have to work in harmony, but failure of any one part should not completely destroy the functionality of the machine.

One method we mentioned is through partitioning of functionality. For example, the portion managing the communication subsystem would be partitioned from the portion managing the effector subsystem. Within the communication subsystem, the portion dealing with network communication would be partitioned from the portion dealing with voice communication so that some software problems with one can be contained there. A software bug that causes the crash of the entire machine is very bad. For critical portions we may also want to provide some redundancy.

## 11.2. Site System

### 11.2.1. Information Support

The amount of information support for a construction site would vary. We may want to use systems to support the needs of the office, field or both. We will mention several types of information support that one can provide. Communication networks could be used for fast and reliable communication. However, we should note that such a network needs to be designed as an open system that permits much flexibility in growth and dismantling. The system should continue to function in the case of failure of communication lines and some processing nodes. Robustness can be improved through the use of a backup wireless system. Attached to the connectivity network could be databases, management agents, monitoring terminals, phone links, hardcopy centers and special information support stations. Wireless systems can be attached to provide communication with mobile platforms. Interaction through the system is monitored and controlled by intelligent network managers which also have the abilities to reconfigure the network. Particular interaction protocols might be enforced by the system.

Special human interfaces, such as voice links which process voice communications, can be attached to enhance the reachable scope of the system. Gateways may be provided for communication with the world. Indeed, wireless information can also be obtained from external sources. Figure 48 depicts a portion of a possible information support system for construction automation.

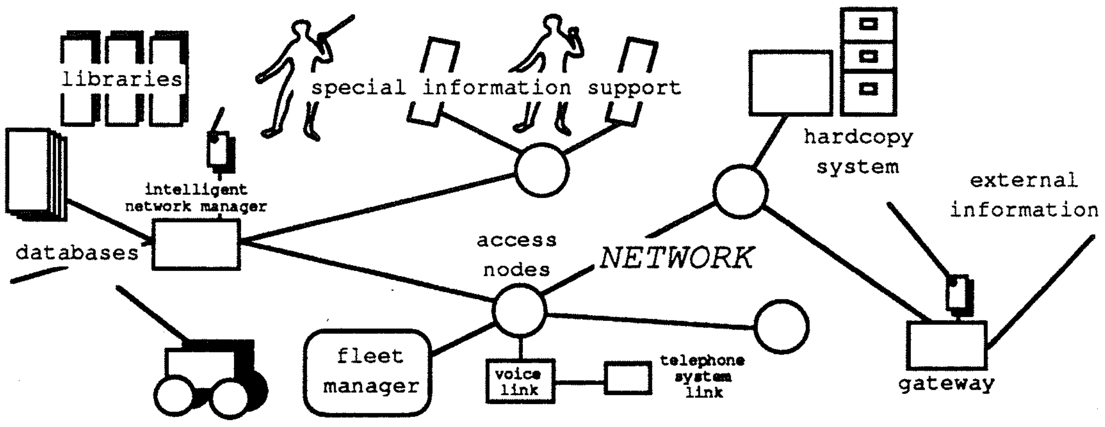


Figure 48. Portion of Information Support Systems

If the system is run as a purely passive information routing system as we do today, we might run into some trouble. We would like to manage information so that we can obtain and use more up-to-date information. Environmental and information management intelligence can be incorporated into the network managers as well as various agents so that information diffuses in the proper manner.

## 11.2.2. Problem-Solving and Decision Support

There are many decision-making agents in a distributed field construction automation system. Some decisions can be made by a single agent without consultation with others, but other decisions might require inputs from several agents.

One of the ways to control decision-making and problem-solving is to provide an organizational framework for all the agents. This framework provides some standard problem-solving approaches and defines levels of decision-making authority. Certain resource allocation decisions might be placed with an area resource allocation manager or a fleet manager. Other resource allocation decisions might be made by a site manager.

In addition to the use of an organizational framework, various problem-solving and decision support frameworks can be made available. When a problem arises that needs several agents, a blackboard might be set up for that purpose. Relevant agents are informed and they contribute to the solution of the problem through interaction via the blackboard. When everything is over and the blackboard has no further use, it is dismantled. Several blackboards can be dynamically created and used in this fashion.

## 11.2.3. Support for Agents

Both human and machine workers at a construction site need support for materials, information and contingency handling. A market-like resource allocation system may be developed for handling material movements. This avoids the need of pre-programming material movements. Market-like systems can also flexibly adapt to growing and shrinking needs. If we also use tokens of exchange, then we can obtain a substantial degree of distributed control. The system is depicted in Figure 49 and uses a source-sink model. Each worker agent knows of platforms that can provide it with materials, or learns about them, and can send in requests for supplies as the need for such materials is about to arise. In addition to indicating the type, amount, place and time required of each material or resource item, the worker bids a certain amount for such services depending on need. Managers arrange for workers with the more critical tasks to have greater bidding resources. Materials can be supplied to places having the greatest need for them. We can avoid the need of a central controller. The overall system will also have the property of graceful degradation to failure.

If bidding resources provided appear to be insufficient, workers can negotiate with area or site managers for additional resources. Such negotiation would involve some information flow, such as providing justifications and conditions at the workface. Another aspect of the scheme depicted in Figure 49 that one might note is that agents might bid for a regular stream of resources instead of for each resource. This is in keeping with the notion of regular work rate at the workface. Such resource stream bidding keeps down the amount of bidding communication traffic.

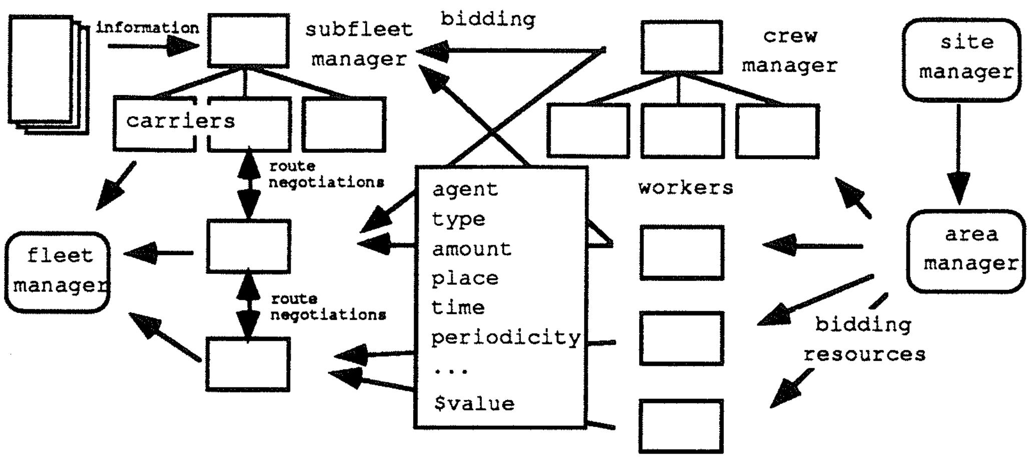


Figure 49. Market-like Mechanism for Distributed Control of Material Handling

Agentbases constitute another type of support that may be provided. Since software agents can acquire knowledge, they are different from application programs. It would be useful to be able to retain some of the learning of these agents. Just as we can have labor pools, software agents may reside in various agentbases. These agents can be hired, downloaded and activated when needed and uploaded when their duties are done.

In constructing automation systems that evolve over time, we would also need to provide support for programming, testing and diagnostics. Information gathering is required for both testing and the evaluation of system performance.

## 11.2.4. Application of Operational Principles.

We have argued for the need of identification and security. In addition, we believe it is also useful for the site system to have commitment tracking and processes for remedying broken commitments. This is because we will have both human and machine agents working together at the site as well as interaction with the external world.

In a construction automation system, we might provide a gatekeeper, agent characterizer and agent tracking mechanisms. This is depicted in Figure 50.

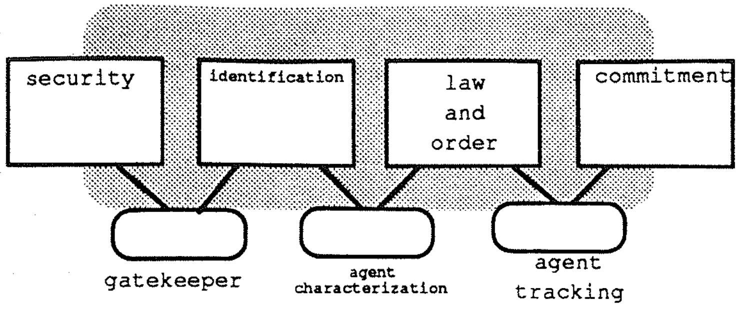


Figure 50. Support Devices for Identification, Security and Commitment Needs

We have to select and use various policies that promote cooperation among agents. One choice might be to use *expanded interest* in the material transportation system. In addition, agents should be designed to work well and use resources effectively.

## 11.3. Environmental Support

Support for automation systems at the level of the global knowledge environment is also needed. Global environmental support systems can provide important standardization, information about external environmental events and enforcement of commitments with external agents. Standardization can be provided through institutions for standardization. This includes provision of single time framework for all agents on and off site. Other types of standardization, such as standardization in software and hardware interfaces, data formats and knowledge representation, can also be effected though standardization institutions.

Conflicts among agents could be resolved through some kind of public court system. The need of enforcing commitments among environmental agents means a need for non-forgeable paper trail. Electronic trails are too easily destroyed. This requires standard contract formats for the paper trail. We would also benefit from standardization of basic communication protocols and inter-agent intrusion protocols. Serious inimical behavior ought to be suppressed by public law and evolutionary mechanisms. Surety and bonding mechanisms can be used to promote reliability.

Brokerage and bank-like services can also be provided to machine systems. For example, excess funds can be banked or loans obtained, and blocks of computational time or mass storage can be sold in advance. Unused equipment and agents can be proffered. This will not only help reduce idling and wastage but also levels resource usage.

We have also seen the need to tie in current information sources and systems to enable machine systems to access them better. Logistic planning would require information about current traffic patterns and routes. Scheduling of tasks might be dependent on both traffic and weather information. Weather and other types of generally useful information for machines can be provided by radio. Of course, such information is useful not only to construction automation systems but to partially computerized military and business systems.

# 12. Conclusions

## 12.1. Summary of Contributions

This thesis presents a preliminary theory of the knowledge environment and its relationship with construction automation. We have argued that agents involved in construction work are interdependent. They are limited and can benefit from exploiting various environmental opportunities and thus should have some environmental intelligence. The rationality for this applies not only to human agents but also to future machine agents.

Researchers in distributed artificial intelligence have often looked to the real world for insights into the functioning of multi-agent systems. Mechanisms such as bidding, contract negotiations, commitment, money, markets, courts and organizations were invented by humans long ago and have evolved ever since. Limited aspects of bidding, negotiations, commitment and markets have been adapted for use in machine systems by distributed artificial intelligence researchers, but a lot remains to be done. Our research also takes its inspiration from the human knowledge environment and the way human agents work.

While it is not a requirement for an Engineer Degree thesis to provide original contributions to knowledge, some useful new concepts have been developed in this research. To the best of the author's knowledge, the following notions are among those newly introduced:

- knowledge environment
  — an environment of knowledge with structure and behavior
- open world assumptions
  — the semantics of knowledge use by agents that regard knowledge of an agent as merely capturing a fraction of the knowledge of the world
- graded reachability
  — a concept which underscores rather than ignores the cost of using information in the environment
- laziness

— expressed in such forms as lazy planning and lazy reasoning, as resource control techniques

- agentbases

  — as pools of software agents

- environmental facilitation mechanisms of goal transformation

  — a group of techniques for making wants of agents fit better.

Other useful concepts we have mentioned, but which are not new, include the concept of an open system [Hewitt 82], knowledge-limited agents and the idea that there is a general core of intelligence across many machine systems.

Every concept and notion has a place in providing us with insight about the knowledge environment and how we might create machines capable of functioning within it. Pointers into some of the extensive literature have been provided. Each idea is interesting in its own right and will require full development to realize its potential; however, the development of these concepts ought to done within an over-arching framework such as the one constructed here, so we will know better why we are doing it.

For example, the construction of lazy planners (or reasoners) never had any attention in artificial intelligence research or in more traditional planning research. People reflexively reject the idea of building lazy machines, which the concept mistakenly invokes. They are more concerned about building complete and correct planners and reasoners — if they can. However, such systems, by ignoring knowledge and resource limitations, and environmental dynamics, will continue to miss the real target of creating practical and realistic systems — ones for the real world. Examples of such practical systems are evolutionarily fashioned biological life-forms, especially human beings. Such theoretically complete and correct systems do have their roles in the real world, but mainly as tools to be used by real agents much more apt at working within resource limits and more rational in coping with an environment that changes. Besides, an understanding of the human knowledge environment is also needed if we are to design automation systems that work extensively with humans.

## 12.2. Future Research

The research in this thesis represents an attempt to build a framework for some of the software aspects of construction automation. It represents only preliminary research, and much more is still required to investigate many of the issues raised and create solutions to the problems described. One dozen suggestions are given below:

i) To further investigate the way transaction management can be combined with other aspects of the agent's activities.

ii) To combine various aspects of planning into one coherent system.

iii) To combine various operational principles governing the decision-processes of an agent.

iv) To investigate the implications of various operational principles.

v) To implement methods of goal transformation.

vi) To combine planning with decision-making.

vii) To develop models for agents and the knowledge environment that can be incorporated into machine systems.

viii) To specify the general knowledge an agent has about the knowledge environment and the capabilities and reasoning techniques needed to use this knowledge.

ix) To investigate the use of self-knowledge, which is part of the core knowledge of an agent.

x) To investigate communication processes beyond that of transaction management.

xi) To investigate control aspects of machine agent architecture.

xii) To further investigate the use of environmental intelligence in management information systems.

Research is currently underway to more deeply understand the nature of environmental intelligence and devise applicable representations of such knowledge in machine systems. This represents merely one item of research and, clearly, there is need for other researchers to help develop a cohesive body of theory and techniques to support future automation systems. We have pointed out the existence of dynamic and cognitive disparities among systems, but how do we analyze the dynamics of a reasoning system or measure the cognitive disparity between two systems? It is hoped that this thesis will serve as an introduction and overview, as it contains ideas to prepare and stimulate those research efforts.

# 13. Bibliography

Agha, G. A. 1986. *Actors*, MIT Press, Cambridge, Massachusetts.

Axelrod, R. 1984. *The Evolution of Cooperation*, Basic Books, Inc. Publishers, NY.

Baker, A. D. 1988. "Complete Manufacturing Control Using a Contract Net: A Simulation Study," *Proceedings of the International Conference on Computer Integrated Manufacturing*, Rensselaer Polytechnic Institute, May, pp. 100-109.

Bond, A. H., and Gasser, L. 1988. *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., San Mateo, CA.

Boettcher, K., Perschbacher, D., and Wessel, C. 1987. "Coordination of Distributed Agents in Tactical Situations," *Proceedings of the National Aerospace and Electronics Conference*, IEEE, Dayton, May, Vol. 4, pp. 1421-1425.

Cammarata, S., McArthur, D., and Steeb, R. 1983. "Strategies of Cooperation in Distributed Problem Solving," *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, pp. 767-770.

Conry, S. E. Meyer, R. A., and Lesser, V. R. 1986. "Multistage Negotiation in Distributed Planning," *Readings in Distributed Artificial Intelligence*, Bond, A. H. and Gasser, L., (eds), Morgan Kaufmann Publishers, Inc., San Mateo, CA., pp. 367-384.

Corkill, D. D. 1979. "Hierarchical Planning in a Distributed Environment," *Proc. 6th International Joint Conference on Artificial Intelligence*, Vol. 1, Tokyo, pp. 168-175.

Corkill, D. D., and Lesser, V. R. 1983. "The Use of Meta-level Control for Coordination in a Distributed Problem Solving Network," *Proc. 8th International Joint Conference on Artificial Intelligence*, Vol. 2, pp. 748-756.

Corkill, D. D., Lesser, V. R., and Hudlicka, A. E. 1982. "Unifying Data-directed and Goal-directed Control: An Example and Experiments," *Proceedings of the 2nd National Conference on Artificial Intelligence*, AAAI-82, pp. 143-147.

Davis, R., and Smith, R. G. 1983. "Negotiation as a Metaphor for Distributed Problem Solving," *Artificial Intelligence Journal*, Vol. 20, pp. 63-109.

Durfee, E. H, Lesser, V. R., and Corkill, D. D. 1987. "Cooperation Through Communication in a Distributed Problem Solving Network," *Distributed Artificial Intelligence*, Michael N. Huhns (ed), Morgan Kaufmann Publishers, Inc., Los Altos, California, pp. 29-58.

Fagin, R., and Halpern, J. Y. 1985. "Belief, Awareness, and Limited Reasoning: Preliminary Report," *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, Vol. 1, pp. 491-501.

Gassner, L., Braganza, C., and Herman, N. 1987. "MACE: A Flexible Testbed for Distributed AI Research," *Distributed Artificial Intelligence*, Michael N. Huhns, (ed), Morgan Kaufmann Publishers, Inc., Los Altos, California, pp. 119-152.

Genesereth, M. R., and Lenat, D. B. 1980. "Self-Description and Self-Modification in a Knowledge Representation System," *HPP-880-10*, Computer Science Department, Stanford University.

Georgeff, M. P. and Lansky, A. L. 1987. "Reactive Reasoning and Planning," *Proceedings of the 6th National Conference on Artificial Intelligence*, AAAI-87, July 13-17, pp. 677-682.

Georgeff, M. 1982. "A Theory of Action for MultiAgent Planning," *Proceedings of the 2nd National Conference on Artificial Intelligence*, AAAI-82, pp. 121-125.

Ginzberg, M. L. 1987. "Decision Procedures," *Distributed Artificial Intelligence*, Michael N. Huhns, (ed), Morgan Kaufmann Publishers, Inc., Los Altos, California, pp. 3-28.

Halpern, J. Y. 1986. "Reasoning about Knowledge: An Overview," *Proceedings of the Conference on Theoretical Aspects of Reasoning About Knowledge*, Halpern, J. Y., (ed), Monterey, CA., (Mar. 19-22), Morgan Kaufmann Publishers, Los-Altos, CA., pp. 1-17.

Halpern, J. Y., Moses, Y. 1984. "Knowledge and Common Knowledge in a Distributed Environment," *Proceedings of the 3rd ACM Conference on Principles of Distributed Computing*.

Halpern, J. Y., and Vardi, M. Y. 1987. "The Complexity of Reasoning about Knowledge and Time, I: Lower Bounds," *Research Report RJ 5764 - 58103*, IBM Almaden Research Center, San Jose, CA.

Hayes-Roth, B. et al. 1985. "Blackboard Architecture for Control," *Journal of Artificial Intelligence*, Vol. 26, pp. 251-321.

Hewitt, C and De Jong, P. 1983. "Analyzing the Roles of Descriptions and Actions in Open Systems," *Proceedings of 3rd National Conference on Artificial Intelligence*, AAAI-83, Aug., pp. 162-167.

Hewitt, C. 1985. "The Challenge of Open Systems," *BYTE*, Apr., pp. 223-242.

Hewitt, C. and de Jong, P. 1984. "Open Systems," *Conceptual Modelling*, Topics in Information Systems, Perspectives from AI, Databases and Programming Languages, Brodie, M. L., Mylopoulos, J. and Schmidt, J. W., (eds), pp. 147-163.

Hewitt, C. 1988. "Offices are Open Systems," *The Ecology of Computation*, Studies in Computer Science and Artificial Intelligence, Vol. 2, Huberman, B. A., (ed), North-Holland, pp. 5-23.

Hintikka, J., 1962. *Knowledge and Belief*, Cornell University Press, Ithaca, NY.

Horvitz, E. J. 1987. "Reasoning About Beliefs and Actions Under Computational Resource Constraints," *KSL-87-29*, Knowledge Systems Laboratory, March.

Kahn, K. M. and Miller, M. S. 1988. "Language Design and Open Systems," *The Ecology of Computation*, Studies in Computer Science and Artificial Intelligence, Vol. 2, Huberman, B. A. ed., North-Holland, pp. 291-313.

Katz, M. J. and Rosenschein, J. S. 1989. "Plans for Multiple Agents,"*Distributed Artificial Intelligence*, Vol. 2, Gasser, L. and Huhns, M. N., (eds), Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 197-228.

Konolige, K. and Nilsson, N. J. 1980a. "Multiple-Agent Planning Systems," *Proceedings of the 1st National Conference on Artificial Intelligence*, AAAI-80, pp. 138-141.

Konolige, K. 1980b. "A First-Order Formalization of Knowledge and Action for a Multi-Agent Planning System," *Technical Note 232*, SRI International, Menlo Park, California.

Konolige, K. 1985. "A Computational Theory of Belief Introspection," *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, Vol. 1, pp. 502-508.

Koo, C. C.-L 1987. *Synchronizing Plans Among Intelligent Agents via Communications*, Ph.D. Thesis, Department of Civil Engineering, Stanford University, Stanford, CA.

Koo, C. C.-L., and Cashman, P. 1986. "A Communication Model for Distributed Manufacturing," *Proceedings of the ASME Symposium on Knowledge-Based Expert Systems for Manufacturing*, ASME, pp. 155-166.

Koo, C. C.-L., and Cashman, P. 1988. "A Commitment-based Communication Language for Distributed Manufacturing," *Intelligent Manufacturing*, Proceedings from the First International Conference on Expert Systems and the Leading Edge in Production Planning and Control, Michael D. Oliff, pp. 155-166.

Lenat, D. and Guha, R. V. 1988. "The World According to CYC," *MCC Technical Report No, ACA-AI-300-88*, Sep.

Lesser, V. R., and Corkill, D. D. 1983. "The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks," *AI Magazine*, Vol. 4., pp. 15-33.

Lesser, V., et al 1982. "A High-level Simulation Testbed for Cooperative Distributed Problem Solving," *The 2nd International Conference on Distributed Computing Systems*, Florida, (Oct. 18-22), pp. 341-349.

Levesque, H. J. 1984. "The Logic of Incomplete Knowledge Bases," *Conceptual Modelling*, Topics in Information Systems, Perspectives from AI, Databases and Programming Languages, Brodie, M. L., Mylopoulos, J. and Schmidt, J. W. (eds.), pp. 165-189.

Levesque, H. J. 1981. "The Interaction with Incomplete Knowledge Bases: A Formal Treatment," *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, Vol. 1, pp. 240-245.

Malone, T. W. 1987. "Modelling Coordination in Organizations and Markets," *Journal of Management Science*, Vol. 33, No. 10., pp. 1317-1332.

Malone, T. W., Fikes, R. E., Grant, K. R. and Howard, M. T. 1988. "Enterprise: A Market-like Task Scheduler for Distributed Computing Environments," *The Ecology of Computation*, Studies in Computer Science and Artificial Intelligence, Vol. 2, Huberman, B. A. ed., North-Holland, pp. 177-205.

Miller, M.S and Drexler, K. E. 1988. "Markets and Computation: Agoric Open Systems," *The Ecology of Computation*, Studies in Computer Science and Artificial Intelligence, Vol. 2, Huberman, B. A. ed., North-Holland, pp. 133-176.

Moder, J. J., Phillips, C. R. and Davis, E. W. 1983. *Project Management with CPM, PERT and Precedence Diagramming*, Van Nostrand Reinhold Company Inc., New York, NY.

Oglesby, C., Parker, H. and Howell, G. 1989. *Productivity Improvement in Construction*, McGraw-Hill, Inc., NY.

Oppen, D. C., and Dalai, Y. K. 1983. "The Clearinghouse: A Decentralized Agent for Locating Objects in a Distributed Environment," *ACM Transactions on Office Information Systems*, Vol. 1, No. 3, July, pp. 230-253.

Parunak, H. V. D. 1987. "Manufacturing Experience with the Contract Net," *Distributed Artificial Intelligence*, Michael N. Huhns (ed), Pitman Publishing/Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 285-310.

Paulson, B. C., Jr., and Lai-Heng Chua 1989a. "Software Environments for Construction Automation and Robotics," *Proceedings of the First IES Information Technology Conference*, Singapore, Vol. 1., May 25-27, 1989, pp. 68-76.

Paulson, B. C., Jr., Thomas Froese, and Lai-Heng Chua 1989b. "Simulating the Knowledge Environment for Autonomous Construction Robot Agents," *Proceedings of the 6th International Symposium on Automation and Robotics in Construction*, Burlingame, California, June 6-8, 1989, pp. 475-482.

Paulson, B. C., Jr., Nadeem Babar, Lai-Heng Chua, and Thomas Froese 1989c. "Simulating Construction Robot Agents and their Knowledge Environment," *Journal of Computing in Civil Engineering,* ASCE, Vol. 3, No. 4, pp. 303-319.

Paulson, Boyd C., Jr. 1985. "Automation and Robotics for Construction," Journal of Construction Engineering and Management, ASCE, Vol. 111, No. 3, September, pp. 190-207.

Primet, P., and Schwarz, J. J. 1986. "The Open System Concept in Robotics," *Proceedings of the 16th International Symposium on Industrial Robots, 8th International Conference on Industrial Robot Technology*, Brussel, H. V., (ed), (30 Sep. - 20 Oct.), Belgium, pp. 863-873.

Reiter, R. 1980. "A Logic for Default Reasoning," *Journal of Artificial Intelligence*, Vol. 13, pp. 81-132.

Reiter, R. 1978. "On Closed World Data Bases," *Logic and Data Bases*, Gallaire, R. and Minker, J. (eds), Plenum Press, pp. 55-76.

Rosenschein, J. S. 1985. "Rational Interaction: Cooperation Among Intelligent Agents," *KSL-85-40*, Knowledge Systems Laboratory, Stanford, CA.

Rosenschein, J. S. and Breese, J. S. 1989. "Communication-Free Interactions Among Rational Agents: A Probabilistic Approach," *Distributed Artificial Intelligence,* Vol. 2, Gasser, L. and Huhns, M. N., (eds), Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 99-118.

Sacerdoti, E. D. 1977. *A Structure for Plans and Behavior,* American Elsevier, New York.

Sangrey, Dwight A. 1985. "Quality and Reliability as Motivation for Construction Robotics," *Proceedings of the Second Conference on Robotics in Construction,* Carnegie-Mellon University, Pittsburgh, PA, (Jun. 24-26), pp. 31-40.

Shaw, M. J-P. 1987. "A Distributed Scheduline Method for Computer Integrated Manufacturing: the use of local area networks in cellular systems," *International Journal of Production Research*, Vol. 25, No. 9, Sep., pp. 1285-1303.

Skibniewski, M. J. 1986. *Engineering and Economic Analysis of Robotics Application Potential in Selected Construction Operations,* Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA., (Mar.).

Smith, R. G. 1979. "A Framework for Distributed Problem Solving," *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, Vol. 2, Tokyo, pp. 836-841.

Smith, R. G. 1980. "The Contract Net Protocol: High-level Communication and Control in a Distributed Problem Solver," *IEEE Transactions on Computers*, C-29, No 12, (Dec.), pp. 1104-1113.

Smith, R. G. 1981. *A Framework for Distributed Problem Solving*, UMI Research Press, Ann Arbor, Michigan.

Steeb, R., et al 1988. "Archictures for Distributed Air-Traffic Control," *Readings in Distributed Artificial Intelligence*, Bond, A. H. and Gasser, L., (eds), Morgan Kaufmann Publishers, Inc., San Mateo, CA., pp. 90-101.

Struat, C. 1985. "An Implementation of a Multi-agent Plan Synchronizer," *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, Vol. 2, pp. 1031-1033.

Warzawski, A., and Sangrey, D. 1985. "Robotics in Building Construction," *Journal of Construction Engineering and Management,* ASCE, Vol. 111, No. 3, (Sep.).

Webster, M. 1970. *Webster's Seventh New Collegiate Dictionary*, G & C Merriam Co. Publishers, Springfield, Massachusetts.

Werner, E. 1989. "Cooperating Agents: A Unified Theory of Communication and Social Structure," *Distributed Artificial Intelligence,* Vol. 2, Gasser, L. and Huhns, M. N., (eds), Morgan Kaufmann Publishers, Inc., San Mateo, California, pp. 3-36.

CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

# Toward a More Intelligent and Cooperative Generation of Construction Robots

by

Boyd C. Paulson

**TECHNICAL REPORT**
**Number 22B**

January, 1992

**Stanford University**

Copyright © 1992 by

Center for Integrated Facility Engineering

If you would like to contact the authors please write to:

*c/o CIFE, Civil Engineering,*
*Stanford University,*
*Terman Engineering Center*
*Mail Code: 4020*
*Stanford, CA 95305-4020*

# Toward a More Intelligent and Cooperative Generation of Construction Robots[1]

Boyd C. Paulson
Charles H. Leavell Professor of Civil Engineering
Department of Civil Engineering
Stanford University

Stanford, California 94305-4020
USA

## ABSTRACT

In future construction environments, intelligent machines must harness considerable knowledge to plan and control their tasks in spite of being limited in their own pre-defined knowledge. This paper first briefly describes the achievements and limitations of today's construction robots, then addresses the more complex environment in which their successors will work. To better integrate robots coming from various sources, they will need a unifying core of intelligent software and a framework to communicate knowledge about designs and field operations. Tomorrow's machines will need better abilities to access knowledge sources in their environment and to work cooperatively with other agents. While the development of such capabilities is among the most challenging tasks facing robotics researchers, it is not too soon to start coming to a consensus about the general nature of the intelligence robots will need to integrate into their environment and become more productive under complex conditions.

[1]Note. This paper first appeared as: Paulson, Boyd C., "Toward a More Intelligent and Cooperative Generation of Construction Robots," Keynote Paper, *Proceedings of the First Symposium on Construction Robotics*, Tokyo, Japan, June 25-27, 1990, pp. 1-10.

## INTRODUCTION

There can be little doubt that, while construction was once considered a rather backward industry, it is increasingly being recognized for providing some of the most complex and interesting challenges to researchers working in computer science, robotics, artificial intelligence, sensors, communications and equipment engineering. For example, last March on the Stanford University campus, the annual meeting of the American Association for Artificial Intelligence (AAAI) was held almost simultaneously with the annual symposium of the Center for Integrated Facility Engineering (CIFE). The latter is a new research center, with substantial support from the Japanese construction industry among others, that is focused on the needs for better integration across the engineering design and construction process. Although the AAAI and CIFE meetings were not organized as a joint conference—indeed, they met in separate locations on campus—it was interesting to see that many of the AI and robotics researchers, including pioneers of AI and expert systems such as Edward Feigenbaum,[2] spent much of their time at the CIFE symposium, and some of the leading construction researchers such as Kiyoshi Niwa[11] of Hitachi attended the AAAI meeting. There was a genuine intellectual exchange among these people.

Although construction technology may indeed have evolved slowly for many decades until the early 1980's, the last few years mark the beginning of a renaissance in this field, and there is reason for optimism about the future. The pioneering work on construction robotics that has been emanating mostly from Japan is one of the best examples of this renaissance,[5] and it is thus all the more of a privilege to be invited to address this audience.

Even as today's first generation of construction robots begins moving from the research laboratories out onto field construction projects, it already is time to be thinking about the future. Most of the successes thus far have been with very specialized machines working almost in isolation from other machines and workers around them, but this is not typical of most of the work that goes on in construction. My main focus in this paper will be on the need to better integrate future construction robots into the total construction environment, and with the engineering, design and administrative functions that support construction.

## TODAY'S ROBOTS: ACCOMPLISHMENTS AND LIMITATIONS

Given the difficult and complex environment of construction, it is remarkable that robots are already performing routine tasks on some jobsites. The first construction robots have either been derived by adding sensors and computer-based controls to existing construction equipment (e.g., to control the cutting edges or screeds on various types of earthmoving and paving equipment), by adapting the comparatively rigid factory-type robots to construction (e.g., for spraying fireproofing material or painting), or by developing hybrids of the two (e.g., robot arms mounted on tunnel machines). While the sophistication of their mechanisms and sensors has often been quite high, these robots have had only the most rudimentary forms of programmed "intelligence," and some machines that have been called robots are really just teleoperated devices without any programmed automation at all.

Most of the construction robots developed to date are stand-alone devices designed to perform narrowly defined tasks without the need to communicate or cooperate with other machines. The concept of a construction "crew" does not really apply yet to construction robots. However, coordinated teams of robots quite commonly perform sequential operations on factory assembly lines, and there are some formal communication mechanisms linking them together; it is only a matter of time before similar technology also moves to construction.

The challenges of the construction jobsite are much greater, however, than those of most factories.[12] To begin, the products of construction are much more complex and vastly larger. Furthermore, in contrast to the repetitive products that flow down production lines, the design of the construction product and the process to build it are usually uniquely adapted in each case. While the manufacturing process is largely steady-state once the production stage starts, that in construction is ever changing. The physical environment of construction is often much more hostile to machines as well as people, so machine design must account for extremes of weather, dust and unexpected forces. While these differences may seem obvious to researchers coming from a construction background, it is sometimes surprising that their implications are not at first obvious to researchers coming from other fields. The increasing cooperation among researchers from many disciplines is improving understanding at all levels, and more and more useful results can be expected.

While there are many challenges facing the advancement of construction automation and the development of more capable construction robots, perhaps the most difficult is that of developing the intelligent software to integrate future machines into the complex environment where they will work. Let us first take a closer look at the environment of construction from the intellectual perspective of a future construction robot.

## THE FUTURE ENVIRONMENT FOR CONSTRUCTION ROBOTICS

The knowledge environment in which future construction robots are to be embedded is potentially vast. The environment is relatively unstructured and fast-changing, so there may not be accurate correspondence of an agent's knowledge about the environment to its real state at any time. In future construction field environments, intelligent machines, like their human counterparts, will thus need to harness considerable knowledge to plan and control autonomous tasks in spite of the fact that they will be limited in their own knowledge and abilities. They will need a unifying core of intelligent software and a framework for defining and communicating knowledge about designs and field operations in a way that can effectively be utilized for their production tasks. At Stanford and elsewhere, researchers are working toward such a core of software in order to support the work of others on practical applications of robots in field conditions.[7,13,14] Some of this research focuses on cognitive aspects of future machines to endow them with common modes of "thinking" and communicating—in effect a common *culture*—so that they can work together and with humans in groups.

The scope of research needed to build theories and core software to support construction robotics is also vast. Each step in this research should lead toward a general architecture handling the knowledge an agent needs to function productively in a knowledge environment. The resulting software could then be extended by developers of applications-oriented robots to handle particular areas of expertise, whether in managing other machines or in doing specific physical tasks. The ultimate objective should be to design and develop the general theory and software core for machine agents—the rudimentary "brains" of the beasts—which can then be embedded in agents specialized for particular tasks.

Figure 1 shows a broad conceptual view of the construction knowledge environment, and some ways in which the core software of an intelligent robot might interact with the environment. It illustrates the organizational context in which the robots might be working, the interfaces to computer-aided design (CAD) databases and reasoning, interactions with other field agents—both human and machine—and interfaces to knowledge sources in the world beyond the field. To establish such an environment with robotic *hardware* is not feasible now. Even if one could afford them, robots with sufficient flexibility and computing power for diverse operations do not yet exist. It is not too early, however, to begin coming to some form of consensus—perhaps even to be thinking about standards—so that such core software can evolve along with the sensors, control systems and mechanisms for these machines. I will outline some general needs and directions in the next section.

## COMPONENTS OF ROBOTIC INTELLIGENCE

Before considering what should go into the core of construction robot software, it is important to think about some bounds on this software. For the most part we are looking at the intelligence to support the successful execution of construction tasks, and this probably will not require much in the way of, for example, aesthetic appreciation, the ability to sing or dance, or innumerable other facets that characterize human existence. Relative to the intelligence and human dimensions of a typical construction worker, we are still looking at a most rudimentary kind of "intelligence" to form the core of a construction robot's software.

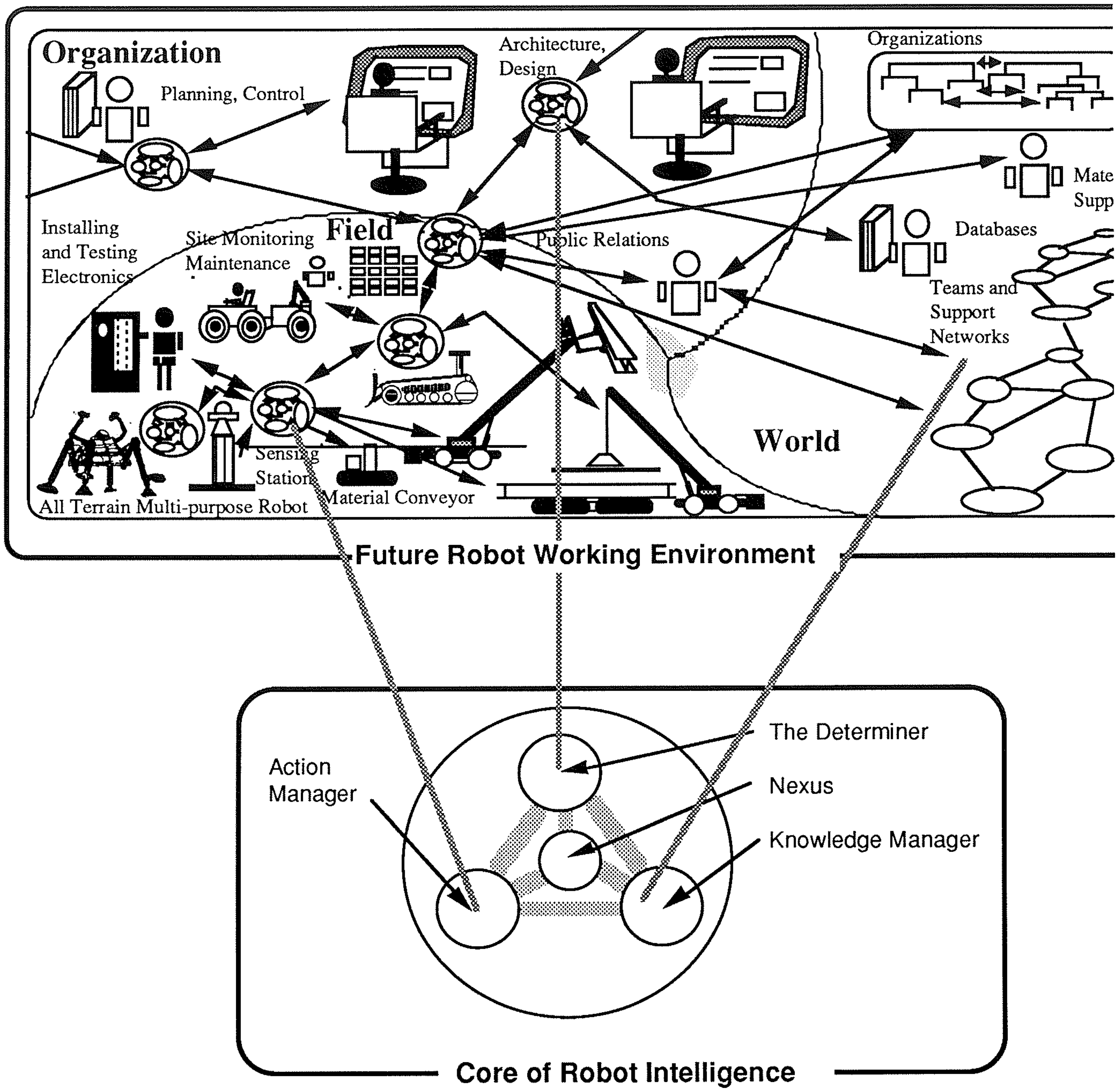


Figure 1—Conceptualization of Agent Core Software Interactions with the Knowledge Environment for Construction Automation

In general, what is needed is some way of modelling *within* robot agents some "understanding" of their environment, such as key characteristics of objects and other agents, in ways useful for reasoning. Among other things, researchers should seek to reduce the knowledge that needs to be encoded in machine systems *a priori* by enabling them to tap the vast knowledge sources in their environment when needed. This is extensibility, which some might call a simple form of learning. Automatons should be able to assemble knowledge and enlist other agents needed to perform a task and respond dynamically to change. For example, robot reasoning and control software should deal with unexpected obstacles, road conditions, failure of a machine positioning system, damaged material, improper tools, or imprecise instructions.

Examples of some basic types of "cultural knowledge" that would be common to and not task-specific for such robots could include the following:

- Knowledge and abilities to deal with space and time, such as:
  - interpretation of and reasoning about geometric 3-D space
  - motion planning
  - Newtonian mechanics (velocity, forces, time, etc.)
  - the ability monitor the location and status of other objects within the environment
- General abilities to receive, analyze and respond to input from sensors
- Communications abilities to access knowledge and data in the environment:
  - access to design data and project information data bases
  - knowledge of organizational structure and access to other agents within it
  - ability to communicate and work co-operatively with other agents
- Task planning abilities
  - understand and use design specifications for task planning
  - seek and interpret project administrative specifications for schedules, etc.
  - be able to select and locate the appropriate tools and materials
  - do the low-level planning for the task based on high-level inputs
- Extensibility and learning, the ability to assemble information from sources beyond

  the robot, and to retain the information for the future
- Self-knowledge, to know the robot's own limits and capabilities and relate these to

  the demands of its tasks and to requests from other agents in the environment

The sections below will examine some of these in more detail and, where appropriate, indicate current research activities that may contribute to this overall body of knowledge and abilities.

## Space, Time and Mechanics

The logical starting point for a robot's model of 3-D construction could be some of the more advanced work on 3-D computer-aided design (CAD) systems now becoming the basis for graphical communications. For example, the Bechtel-Hitachi effort that has led to the software called *Construction CAE*[4] has many of the components that enable construction processes to be simulated graphically and in scaled time against the background of realistic 3-D images of the facility being designed. It takes only a small leap of imagination to substitute real automated machines for the graphical images of construction equipment moving on the CRT screen.

The core software should also implement basic physical mechanics to monitor agent motion and location. Rapid progress in this area is being made by researchers under Professor Jean-Claude Latombe, Director of the Computer Science Robotics Laboratory at Stanford. In the past year or so they have improved motion planning algorithms to run over 100 times faster than the best previously available, and to handle over 30 degrees of freedom in three dimensions where others have struggled to handle 5 or 6 in two dimensions.[10,19] In path planning and vehicle simulations they can manage operations like backing semi-trailer trucks through complex mazes of objects, parallel parking a car, etc., all with the automated path planner.[3,9,20] There seems to be a great need for applying such core software work in evolving machines like the pipe-manipulator at the University of Texas,[1] whose 8 degrees of freedom seem to baffle even some of the best human operators.

## Sensors

While specific sets of sensors on a given robot will be oriented toward a given application or class of applications, there are some general capabilities related to receiving and analyzing the data from the sensors that can be developed in a generic fashion and thus be included in the core intelligence. For example, machine vision and pattern recognition are fundamental problems that currently are receiving the attention of many researchers. Today this particular technology lacks fast enough processing capability to handle the types of complex images found in general construction work, but as breakthroughs in theory are made they will find their way to this field. Numerous types of simpler sensors for force, distance, temperature, etc. are already being included in construction robots. In almost all cases, however, custom programming is done to integrate them with the needs of a particular machine. As more and more machines become available, the repetition of this custom effort should be supplanted by generic package software that can be adapted to particular needs with little or no programming. Already this has happened for laboratory data acquisition and processing (e.g., *LabView*™ for the Macintosh™), and similar approaches could be taken for construction robots.

## Language and Communication

To support this new breed of intelligent machines, issues of language and communication need to be addressed soon. Machines must evolve from isolated to cooperative states, either laterally or hierarchically. Researchers may begin by better understanding *what* information is necessary for communication—both in type and content (e.g., bitmaps, graphics, standards, knowledge, design data, task specifications, etc.)—and *how much* information to pass. Then they should address the theoretical core of languages: syntax, semantics, and pragmatics. The objectives will be to design the communication system for robotic agents working in a construction knowledge environment. Computer Science Professors Terry Winograd[17,18] and Yoav Shoham[15] are presently focusing advanced research on language issues for construction robots under two separate CIFE projects at Stanford.

One complexity in language design is that different communication tasks require different levels of knowledge representation. As an example, two robots working together might need to communicate with each other about the geometry and the exact positioning of an object. A *syntactic* representation with built-in words for geometric shapes and different types of motion may be more efficient for this robot-to-robot communication. To update a central computer's database about the status of their task, a *functional language* with descriptive codes and numeric quantities may be enough. Finally, to inform their human supervisor that a part is broken, they do not need to convey the graphical representation or the precise coordinates, just a general idea of what has happened, perhaps in a form more like *natural language*.

Another issue related to the design of a language is the level of implementation. Should a robot have enough background knowledge to infer missing parts of a discourse commonly left out in human conversation? Or should it be given complete and detailed instruction at every step? For example, if a human tells a robot driving a truck that a tire is flat, should the robot know about flat tires and thus know to stop, or should it require explicit instructions? The choice of a high-level implementation with background-knowledge may seem obvious, but, considering the dynamic nature of field project sites, providing robots with even a fraction of this knowledge would be a major undertaking, and supplying all the knowledge is impossible.

Implied in communication is some means to provide a robot with knowledge of a type of organizational hierarchy (e.g., so the robot will know how to reach a central supervisor agent—whether human or perhaps an expert system—to which task agents can turn for help). While I have listed it here under communication, it is actually part of a major research area of adapting the vast knowledge of human and organizational behavior to the more limited needs and capabilities of robot agents.

**Task Planning**

There are several efforts to apply advanced artificial intelligence techniques to the rudimentary planning of construction operations. Examples include site layout planning,[16] multi-agent coordination,[8] and some very interesting work to deduce plans and schedules directly from CAD drawings in their electronic form. For example, Shimizu's Mr. Kenji Ito, working for the last two years as a Visiting Fellow at CIFE, has developed CIFE CAD,[6] which can take AutoCAD data as input and produce a schedule that recognizes the sequence of operations implied by gravity, adjacency, enclosure, etc. Such research may well lead to the ability to convert design specifications into incremental task objectives for robots.

**Extensibility**

Figure 2 represents a machine seeking external knowledge in spheres of increasingly difficult reachability. The nucleus of the figure shows the knowledge and data that might be pre-programmed within the robot itself and thus be immediately accessible. The next layer out could be the knowledge and data available nearby from supervisors or other agents (human or robot), or from computer databases or expert systems on-site. The third layer might still be within the robot's organization—perhaps an expert at the home office. Beyond that could be the world at large—material suppliers, equipment manufacturers, consultants and public databases. The concept of *graded reachability* in the figure, or extensibility in general, implies that, while it becomes more and more difficult and time-consuming to tap data and knowledge further and further from the robot agent's knowledge core, the agent still should have enough general knowledge and communications abilities to (a) recognize the need for knowledge beyond its own sphere, and (b) to know at least where to begin, whom or what to ask, or how to at least start to obtain the knowledge it needs to perform or continue its tasks.

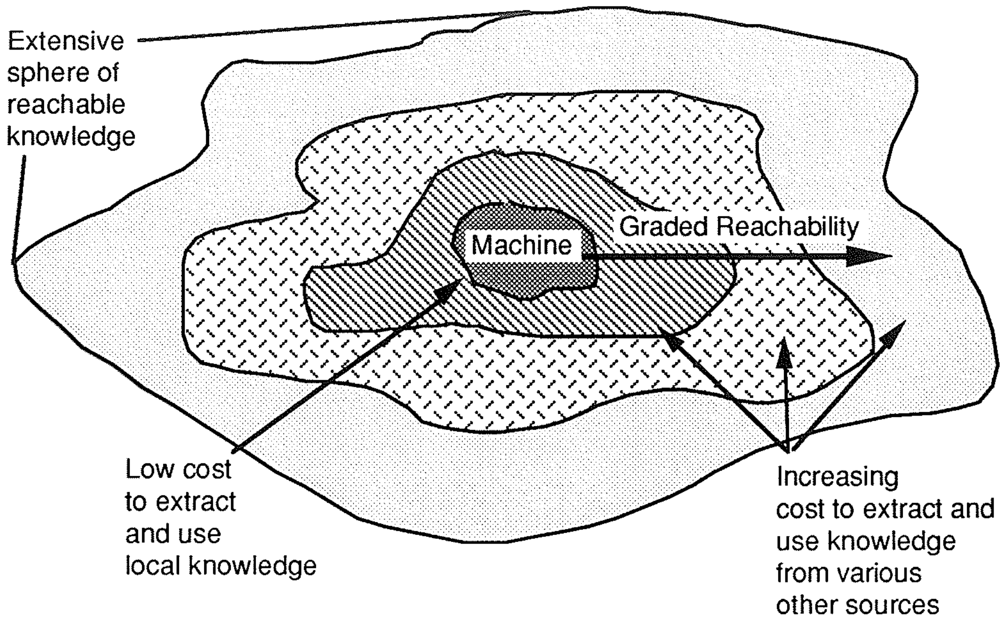

Extensive sphere of reachable knowledge
Machine
Graded Reachability
Low cost to extract and use local knowledge
Increasing cost to extract and use knowledge from various other sources

Figure 2—Accessibility of Knowledge in the Environment for Construction Automation

## Self-Knowledge

To respond intelligently to unexpected situations, robots must know their own capacity and the limits thereon, if only to seek help from a human. *Self-knowledge* can help a robot determine its capability to perform tasks safely and efficiently. Self-knowledge is also essential if a robot works together with other machines or humans. Robot self-knowledge should be accessible and interpretable not only by the robot itself, but also by a central knowledge manager. The former will enable the robot to evaluate its own capability, while the latter is important when the robot needs help from the manager or wants to notify other robots about its state. Research should focus on designing a knowledge base for self-knowledge which is flexible, efficient and compatible with the general language design and communication requirements presented earlier in this paper.

Additional research is needed to define the self-knowledge needs and the environmental-knowledge interpretations of the robotic agents. The method of investigation for the environment could involve formulation of knowledge about the tasks and processes in general and about suitable tasks in particular. The self-knowledge might be divided into two parts: machine-specific self-knowledge like self-diagnostics and error-recovery; and environment-specific self-knowledge like machine weight, physical dimensions, lifting capacity and available power. The next step would be to match suitable language constructs to the representational needs.

## CONCLUSION

This has been a brief overview of a complex fabric of interwoven research needs. Machine agents in complex field environments will have to deal with many difficulties. The only tractable approach in the short term is to limit the use of machines to environments that are—or can be—carefully structured, or to leave most of the control and sensing with human agents. Researchers must look further to develop the underpinnings for more capable machines. We must discover and formulate the general computer-based reasoning and communications core for machines that would provide them with the ability to deal with unexpected events to a greater extent and exploit the opportunities provided by the knowledge environment. In particular, the kind of machines envisaged will be able to deal with uncertainty, adapt to a dynamically changing environment, be able to seek knowledge beyond their spheres, and work together in teams to perform complex tasks. Future research should provide a more robust basis for integration of machines with human organizations, with other machines and with a multitude of tools of production.

## ACKNOWLEDGEMENT

For this research we are fortunate to be part of the Center for Integrated Facility Engineering (CIFE) at Stanford, which is operated jointly by the Departments of Civil Engineering and Computer Science and has corporate members from the design, construction and computer industries and representatives from owners and government agencies. We are also indebted to The National Science Foundation, which, through a series of grants (including MSM 84-06852, MSM 86-14238 and MSM 89-05573) has provided the intellectual freedom to explore the basic research that led to the formulation of the research agenda described herein.

## REFERENCES

1. Alciatore, D. G., P. J. Hughes, A. E. Traver, and J. T. O'Connor, "Development of an Ergonomic Control System for Large Construction Manipulators,' *Proceedings of the Sixth International Symposium on Automation and Robotics for Construction,* San Francisco, California, June 6-8, pp. 183-188.

2. Barr, Avron, and Edward A. Feigenbaum, 1981-2. *The Handbook of Artificial Intelligence,* HeruisTech Press, Stanford California, William Kaufmann, Inc., Los Altos, California, Volume 1, 1981, Volume 2, 1982, Volume 3 (by Paul R. Cohen and Edward A. Feigenbaum), 1982.

3. Barraquand, Jerome, and Jean-Claude Latombe, 1989. *On Nonholonomic Mobile Robots and Optimal Maneuvering,* Technical Report No. 015, Center for Integrated Facility Engineering, Stanford University, Stanford, California, July.

4. Ditlinger, Stephen, and Kermit Gates, 1989. "Animation/Simulation for Construction Planning and Scheduling," *Proceedings of the Sixth International Symposium on Automation and Robotics for Construction,* San Francisco, California, June 6-8, pp. 491-498.

5. Hasegawa, Yukio, 1989. "'Cutting Edge' of the State of the Art of Construction Robotization in Japan," *Proceedings of the Sixth International Symposium on Automation and Robotics for Construction,* San Francisco, California, June 6-8, pp. 27-32.

6. Ito, Kenji, Yasumasa Ueno, Raymond E. Levitt, and Adnan Darwiche, 1989. *Linking Knowledge-Based Systems to CAD Design Data with an Object-Oriented Building Product Model,* Technical Report No. 017, Center for Integrated Facility Engineering, Stanford University, Stanford, California, August.

7. Keirouz, W., Rehak, D., and Oppenheim, I., 1988. "Issues in Domain Modelling of Constructed Facilities," *Fifth International Symposium on Robotics in Construction,* Tokyo, Japan, June 6-8, pp. 341-350.

8. Koo, Charles C., 1987. *Synchronizing Plans among Intelligent Agents via Communication,* thesis submitted to Stanford University in partial fulfillment of the requirements for the degree of Ph.D., August.

9. Latombe, Jean-Claude, 1988. *Global Path Planning Approaches for Material Movements in a Worksite,* Technical Report No. 003, Center for Integrated Facility Engineering, Stanford University, Stanford, California, May.

10. Latombe, Jean-Claude, 1988. *Generation of Sensor-Based Motion Strategies in the Presence of Uncertainty,* Technical Report No. 006, Center for Integrated Facility Engineering, Stanford University, Stanford, California, June.

11. Niwa, Kiyoshi, 1990. "Human-Computer Cooperative Systems in Project Risk Management," *Proceedings of the Symposium of the Center for Integrated Facility Engineering,* Stanford University, Stanford, California, March 26-27.

12. Paulson, Boyd C., Jr., 1985. "Automated Control and Robotics for Construction," *Journal of Construction Engineering and Management,* ASCE, Vol. 111, No. 3, September, pp. 190-207.

13. Paulson, Boyd C., Jr., Thomas Froese, and Lai-Heng Chua, 1989. "Simulating the Knowledge Environment for Autonomous Construction Robot Agents," *Proceedings of the Sixth International Symposium on Automation and Robotics for Construction*, San Francisco, California, June 6-8, pp. 475-482.

14. Paulson, Boyd C., Jr., Nadeem Babar, Lai-Heng Chua and Thomas Froese, 1989. "Simulating Construction Robot Agents and their Knowledge Environment," *Journal of Computing in Civil Engineering*, ASCE, Vol. 3, No. 4, October, pp. 303-319.

15. Shoham, Yoav, 1990. *Agent-Oriented Programming*, Technical Report, Department of Computer Science, Stanford University, Stanford, California, forthcoming.

16. Tommelein, Iris D., Raymond E. Levitt, and Barbara Hayes-Roth, 1989. "SIGHTPLAN: An Artificial Intelligence Tool to Assist Construction Managers with Site Layout," *Proceedings of the Sixth International Symposium on Automation and Robotics for Construction*, San Francisco, California, June 6-8, pp. 340-347.

17. Winograd, Terry, 1983. *Language as a Cognitive Process*, Addison-Wesley, Menlo Park, California.

18. Winograd, Terry, 1989. *Language/Action Approach to the Computer Integration of Construction Work*, Seed Research Proposal to the Center for Integrated Facility Engineering, Stanford University, Stanford, California, December.

19. Zhu, David, and Jean-Claude Latombe, 1989. *Contingency-Tolerant Robot Motion Planning and Control*, Technical Report No. 014, Center for Integrated Facility Engineering, Stanford University, Stanford, California, July.

20. Zhu, David, and Jean-Claude Latombe, 1989. *New Heuristic Algorithms for Efficient Hierarchical Path Planning*, Technical Report No. 018, Center for Integrated Facility Engineering, Stanford University, Stanford, California, August.