

NASA/CP-1999-208986



Intelligent Agents and Their Potential for Future Design and Synthesis Environment

*Compiled by
Ahmed K. Noor
University of Virginia
Center for Advanced Computational Technology, Hampton, Virginia*

*John B. Malone
Langley Research Center, Hampton, Virginia*

Proceedings of a workshop sponsored by the National Aeronautics and
Space Administration and the University of Virginia Center for
Advanced Computational Technology, Hampton, VA, and held at
NASA Langley Research Center, Hampton, Virginia
September 16-17, 1998

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

February 1999

Multiagent-Oriented Programming

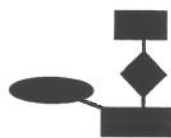
Michael N. Huhns
Center for Information Technology
University of South Carolina
Columbia, SC

Multiagent-Oriented Programming

Michael N. Huhns
Center for Information Technology
University of South Carolina
Columbia, SC

<http://www.engr.sc.edu/research/CIT/people/faculty/huhns.html>

This presentation describes a new approach to the production of robust software. The approach is motivated by explaining why the two major goals of software engineering—*correct* software and *reusable* software—are not being addressed by the current state of software practice. A methodology based on active, cooperative, and persistent software components, i.e., *agents*, and how the methodology enables robust and reusable software to be produced is described. Requirements are derived for the structure and behavior of the agents, and a methodology is described that satisfies the requirements. The presentation concludes with examples of the use of the methodology and ruminations about a new computational paradigm.



Multiagent-Oriented Programming

Michael N. Huhns
Center for Information Technology
University of South Carolina

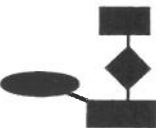
11/18/98 1:12 PM

1


Tremendous Interest in Agent Technology

There is great interest in software agent technology currently. As evidence, there were more than a dozen conferences and workshops devoted to agents held around the world during the summer. There are four major reasons for this interest:

- 1) The Internet has made vast numbers of heterogeneous resources available which software agents are needed to access and manage.
- 2) Processors are being used to control devices throughout our environment, such as automobiles, appliances, and consumer devices. These devices are much more useful if they can communicate intelligently with users and each other.
- 3) New speech understanding technology is making it feasible for people to communicate with devices in natural language, and this is more effective if the devices appear to be intelligent agents.
- 4) Software development continues to be problematic, and multiagent technology can provide a new paradigm.



Tremendous Interest in Agent Technology



Evidence:

- 400 people at Autonomous Agents 98
- 550 people at Agents World in Paris



Why?

- Vast information resources now accessible
- Ubiquitous processors
- New interface technology
- Problems in producing software

11/18/98 1:12 PMUniversity of South Carolina2

Overview

There are two primary reasons for the rise in popularity of agent technology. Each of these is fundamental to computer science, so that agent technology is likely to be important and viable for the foreseeable future.



Overview


- *The fundamental architecture for enterprise information systems is progressing beyond a client-server model*
- *The development of software is progressing beyond object-oriented techniques*

Both trends require agent technology!

11/18/98 1:12 PM University of South Carolina 3

Trends in Information Technology

Information environments have moved beyond the closed corporate environments of the past, and are now open: the resources that are available via networks are dynamic and cannot be predicted or controlled. Also, information processing tasks have moved from batch jobs to applications that combine contributions from humans and computers.



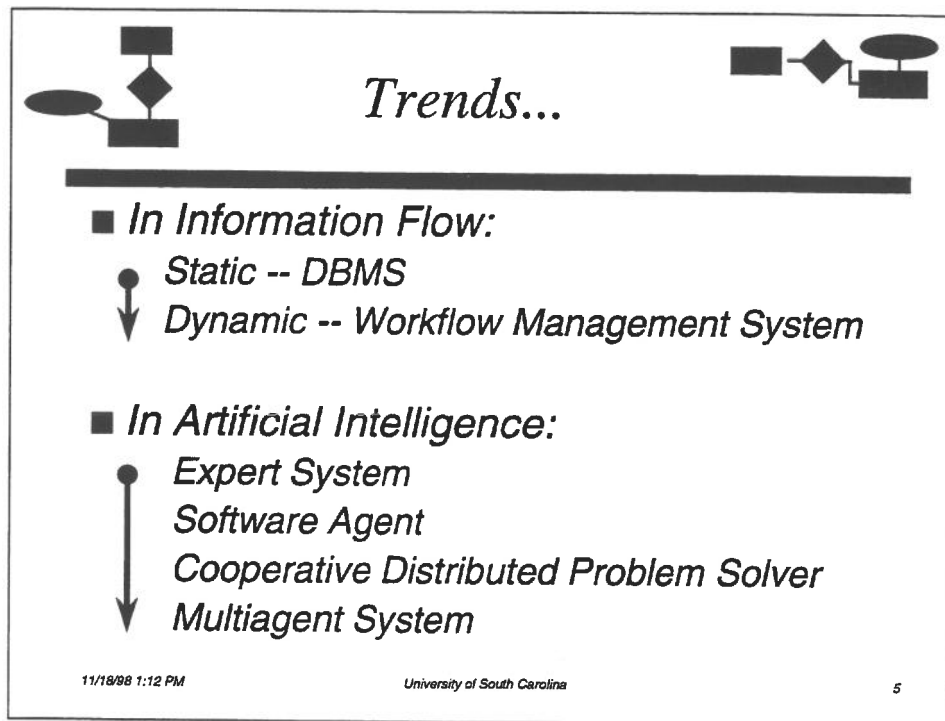
I. Trends in Information Technology

- *In Information Environments:*
 - *Closed*
 - ↓
 - *Open*
- *In Information Processing Tasks:*
 - *Automated*
 - ↓
 - *Automated + Manual*

11/18/98 1:12 PMUniversity of South Carolina4

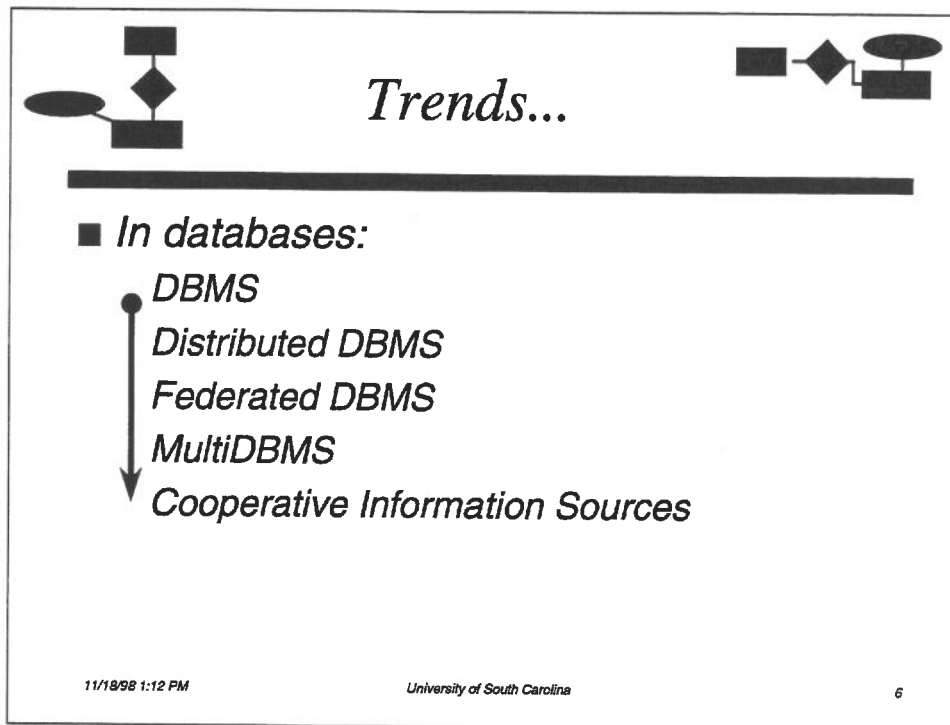
Trends...

Information is no longer treated as just the static data that is found in databases. The dynamic flow of information to and from databases must be considered as well. Artificial intelligence has progressed from expert systems to individual agents to cooperative problem-solving agents to multiagent systems.



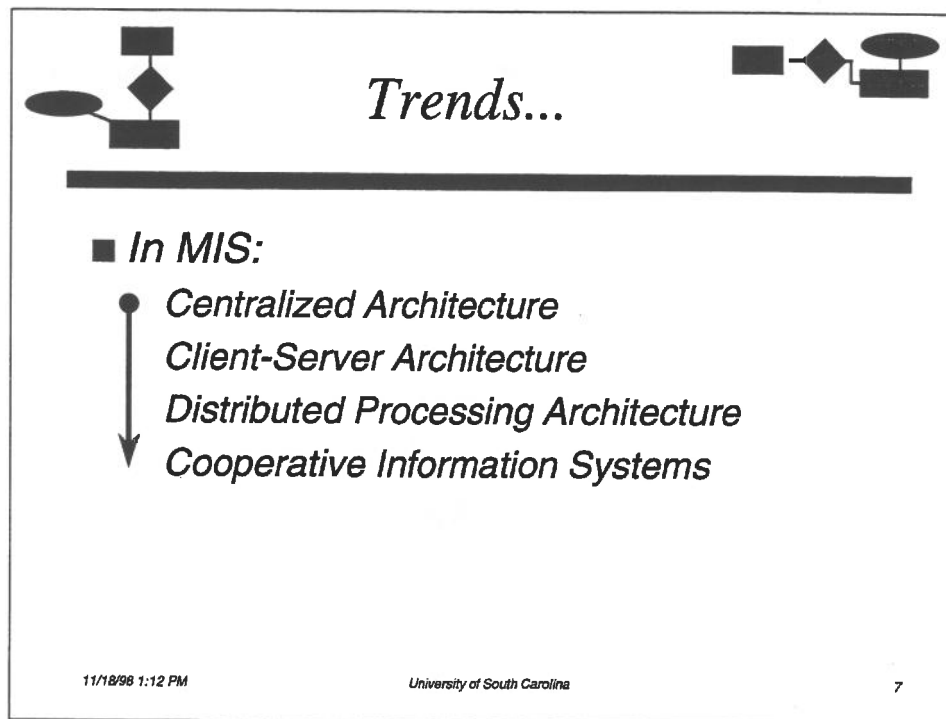
Trends...

Database technology has progressed from individual database management systems to tightly coupled, homogeneous, distributed DBMSs, to federated DBMSs with a single global schema to cooperative DBMSs that are active, autonomous, and heterogeneous.



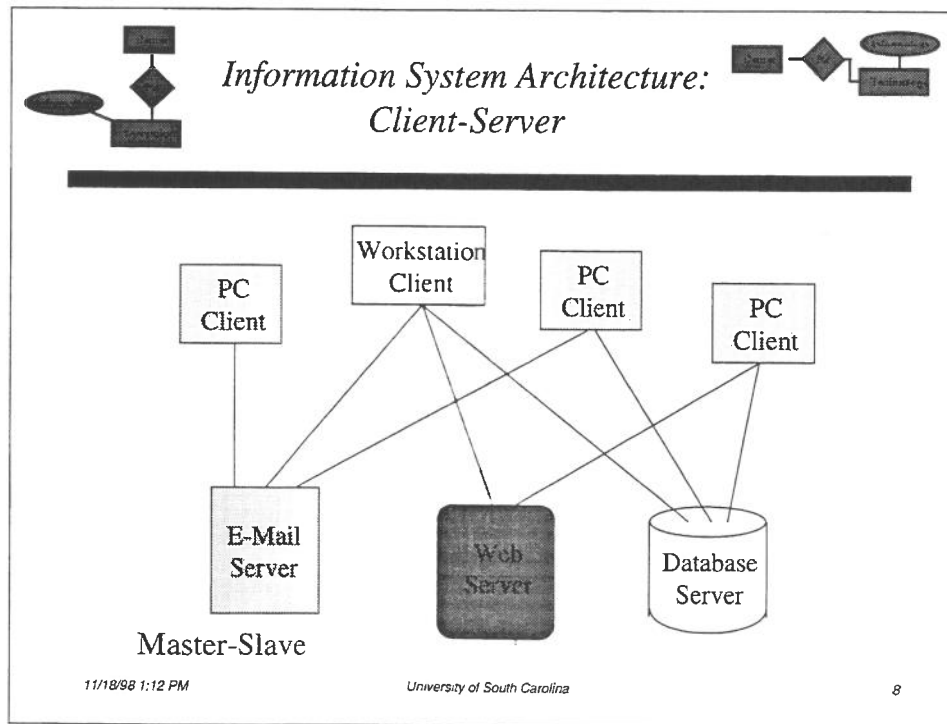
Trends...

Most corporate information systems are being converted from centralized architectures to client-server architectures. However, the trend is to move to distributed information system architectures featuring peer-to-peer interactions and, eventually, to cooperative information system architectures where the peers cooperate in processing information tasks.



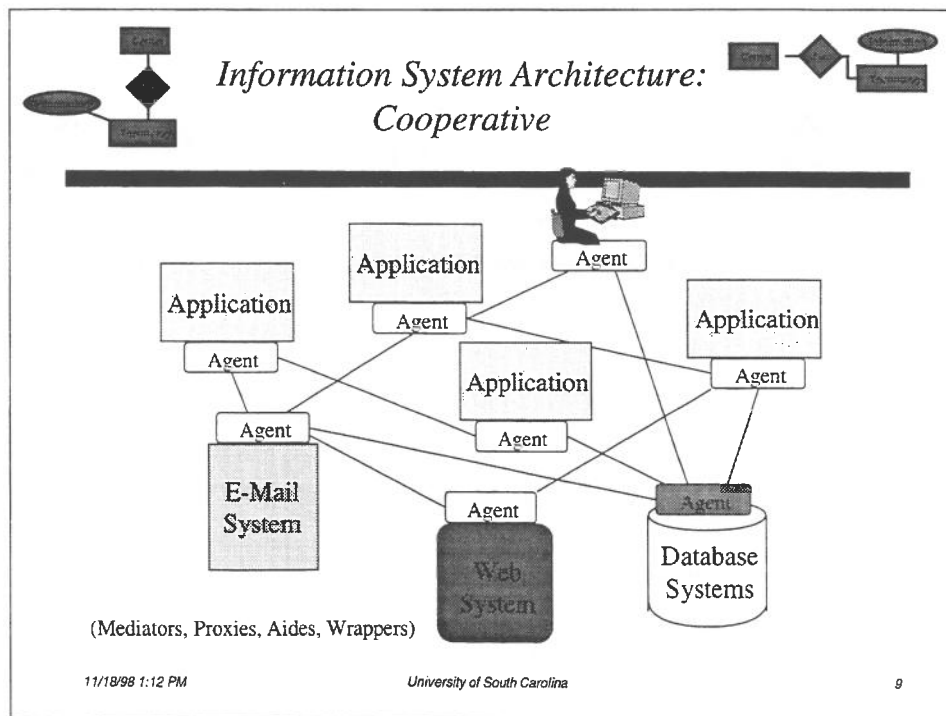
Information System Architecture: Client-Server

A client-server architecture is hierarchical, with no formal interactions among servers or among clients.




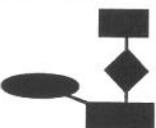
Information System Architecture: Cooperative

A cooperative architecture allows interactions among servers and among clients. The interactions can be cooperative in that the components can assist each other in solving tasks when the tasks are consistent with their own best interests.



II. Trends in Software Development

The two major goals of software engineering, correct software and efficient production of software, are not being met. Programmers currently produce approximately the same number of lines of debugged code as they ever did, in spite of many developments that were supposed to be “magic bullets,” such as structured programming, declarative specifications, object-oriented programming, formal methods, and visual languages.



II. Trends in Software Development

The two goals of software engineering

- *correct software*
- *efficient software production*

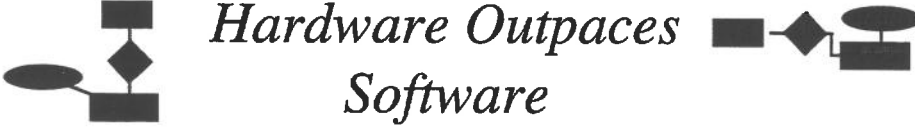
are not being met. Programmers produce ~ the same number of lines of debugged code, in spite of

- *structured programming*
- *declarative specifications*
- *object-oriented programming*
- *formal methods*
- *visual languages*

11/19/98 1:12 PM University of South Carolina 10

Hardware Outpaces Software

Over the last dozen years, processor performance and memory chip capacity have doubled every two years, in accordance with Moore's "Law." Network capacity has grown even faster, but software productivity has been almost static.


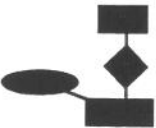


<i>Hardware Outpaces Software</i>	
	Avg. Annual Growth Rate
<i>Processor Performance</i>	48%
<i>Network Capacity</i>	78%
<i>Software Productivity</i>	5%
<i>Software Language & Tool Power</i>	11%

11/18/98 1:12 PM University of South Carolina 11

Why?

There are several reasons for the difficulty in improving the process by which software is produced. First, software is complicated, and is typically considered the most complex activity undertaken by humans. Second, software must be *perfect*, and is guaranteed to work correctly only when *all* errors have been removed. Third, the effect of an error is relatively independent of its size, in that the simple omission of a comma can render a million lines of code inoperable. Fourth, software systems are typically diverse and too often crafted afresh for each application.



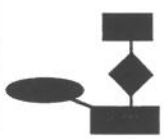
Why?

- *Software is complicated*
- *Software is guaranteed to work correctly only when all errors have been removed*
- *The effect of an error is unrelated to its size*
- *Software systems are diverse*


11/18/98 1:12 PMUniversity of South Carolina12

Programming Paradigms

There have been a number of different paradigms for the production of software since the 1950's. These paradigms have moved the basic unit of abstraction from components that model and implement computations to components that model and implement real-world objects. For example, the concept of an "employee" in a relational database is less like a real-world employee than is the class "employee" in an object-oriented database, because the object model includes the *behavior* of objects in the class.



Programming Paradigms



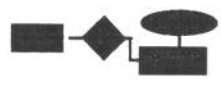
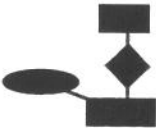
- 1950's -- *Machine and assembly language*
- 1960's -- *Procedural programming*
- 1970's -- *Structured programming*
- 1980's -- *Object-based and declarative programming*
- 1990's -- *Frameworks, design patterns, scenarios, protocols, and components (ActiveX/COM and Java Beans)*

The trend has been from elements that represent abstract computations to elements that represent the real world

11/18/98 1:12 PMUniversity of South Carolina13

A New Paradigm

It is time to consider a new paradigm for software development, a paradigm that is based on the following premises. First, it is important to recognize that errors will *always* be in complex systems, and that it is necessary to accommodate them. Second, there are circumstances where perfect, error-free code can be a *disadvantage*. Third, systems that interact with the real world can take advantage of the uncertainties inherent in the world to lead to more robust and simpler software. Fourth, the new paradigm should continue the trend toward programming constructs that are more faithful to the real-world components they are meant to model.



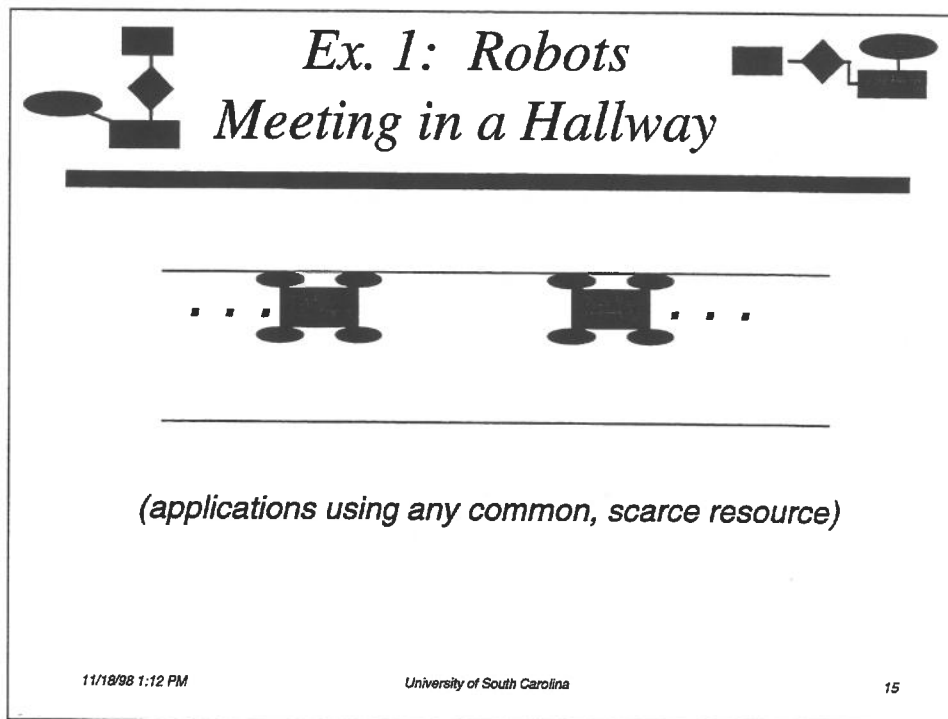
A New Paradigm

- Errors will **always** be in complex systems
- Error-free code can be a **disadvantage**
- Where systems interact with the real world, there is a **power** that can be exploited
- Continue the trend toward programming constructs that match the real world

11/18/98 1:12 PMUniversity of South Carolina14

Example 1: Robots Meeting in a Hallway

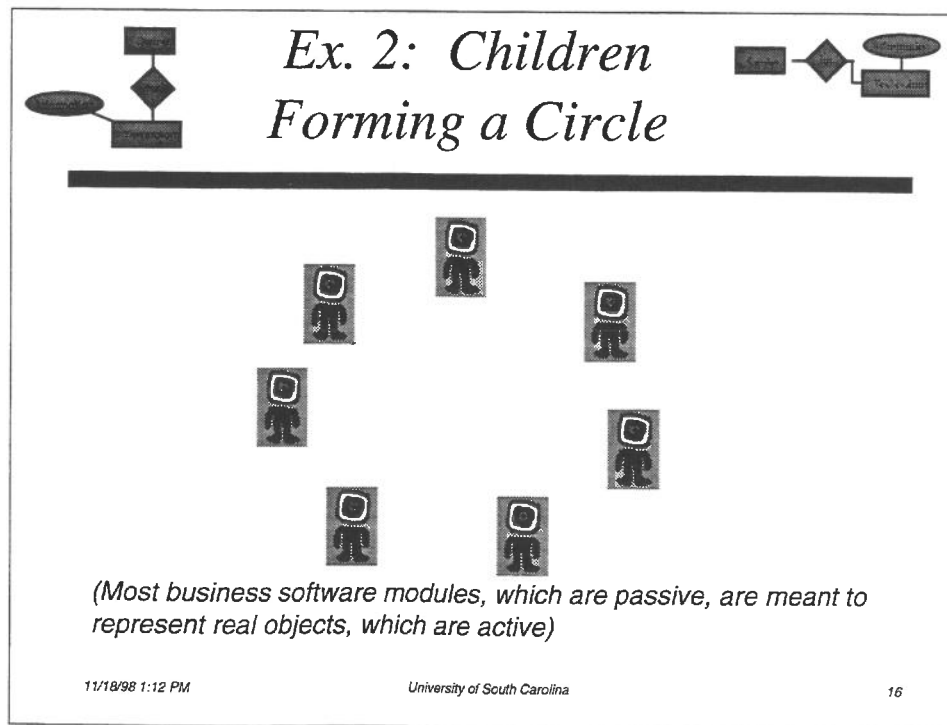
An example of how error-free code can be a disadvantage occurs when two identically and perfectly programmed robots meet in a hallway. They each will move side-to-side in synchrony and will never be able to pass. Now what if one of the robots has an error in its programming? It will then not behave the same as the other, the robots will break synchrony, and they will each be able to pass each other and continue their progress. The overall system of robots is more robust because of the presence of an error. The example is representative of any situation where there is contention for a scarce resource, such as access to a database.



Example 2: Children Forming a Circle

When a teacher tells a group of children to form a circle, they do this very robustly. They can form a circle whether there are 5 or 50 children, whether the children are large or small, and whether or not all of the children are old enough to understand the concept of a circle. Children can be added to or removed from an existing circle, and it will re-form correctly. The children implement a circle-forming algorithm that is distributed and requires no central control.

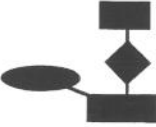
The robustness is due to the knowledge and ability of each child regarding what a circle is and how each child can contribute to its formation.




Forming a Circle

A conventional object-oriented approach to programming a circle algorithm would involve creating a class for each type of object that might be part of the circle, and then writing a control program that would use trigonometry to compute the location of each object. The addition or removal of objects would require recomputing all locations.

A multiagent or team-oriented approach would represent each child by an agent, and would give each agent the knowledge of what a circle is and the ability to position itself to be part of a circle.



Forming a Circle (cont.)

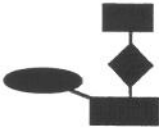


- **Conventional approach**
 - *create a C++ class for each type of object; then write a control program that uses trigonometry to compute the location of each object*
- **Team-oriented approach (based on objects having attitudes, goals, and agent models)**
 - *like children forming a circle, it is robust due to local intelligence and autonomy*


11/19/98 1:12 PMUniversity of South Carolina17

Features of Languages and Paradigms

A procedural language, an object language, and a multiagent language can be compared according to a number of criteria



Features of Languages and Paradigms

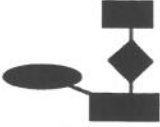


Concept	Procedural Language	Object Language	Multiagent Language
Abstraction	Type	Class	Society
Building Block	Instance, Data	Object	Agent
Computation Model	Procedure/Call	Method/Message	Perceive/Reason/Act
Design Paradigm	Tree of procedures	Interaction patterns	Cooperative interaction
Architecture	Functional decomposition	Inheritance and Polymorphism	Managers, Assistants, and Peers
Modes of Behavior	Coding	Designing and using	Enabling and enacting
Terminology	Implement	Engineer	Activate


11/18/98 1:12 PM
University of South Carolina
18

Team-Oriented Software Development

The most important characteristics of a team-oriented paradigm for software development are that the modules (1) are active, (2) are declaratively specified in terms of what behavior they should exhibit, not how they should achieve that desired behavior, (3) hold beliefs about the world, themselves, and others (whether humans or computational modules), and (4) the modules *volunteer* to be part of a software system. This last characteristic is a key to the reuse of software.



Team-Oriented Software Development


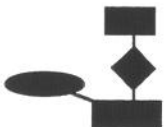


- *Modules are active*
- *Modules are declaratively specified, in terms of “what”, not “how”*
- *Modules hold beliefs about the world, especially about themselves and others*
- *Modules volunteer*

11/18/98 1:12 PMUniversity of South Carolina19

The Agent Test

Most researchers in agent technology have put forth their own definition of an agent. These definitions are usually a list of characteristics that an agent should possess, such as autonomy, persistence, reasoning ability, intelligence, communication ability, etc. Munindar Singh at NCSU and I instead propose a test for agenthood, which implies some of the above characteristics but does not require any of them. The test, to be useful, should be both necessary and sufficient, i.e., any software component that passes it should be considered generally to be an agent, and any component that fails it should be considered generally not an agent.




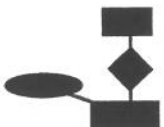
The Agent Test

- “A system containing one or more reputed agents should change substantively if another of the reputed agents is added to the system.”

11/18/98 1:12 PM University of South Carolina 20

Applications

There are many important applications of agents in a wide variety of domains that are under development at the University of South Carolina in its Center for Information Technology.




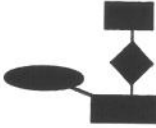
Applications

- *Sainsbury's Supermarkets (UK) simulates customers with agents*
- *Sydskraft (Sweden) controls electricity distribution*
- *HealthMagic (USA) reminds patients of prescriptions and appointments*
- *France Telecom and Deutsch Telekom diagnose circuit faults and route message traffic*
- *US Army manages logistics databases*
- *Siemens (Germany) provides personalized telecom services*
- *Amazon and Barnes & Noble help customers purchase books on-line*
- *US Postal Service includes smart-card agents on packages to track deliveries*
- *Raytheon/TI sensors cooperate in target detection*

11/18/98 1:12 PMUniversity of South Carolina21

To Probe Further...

There are many sources of information available on agent technology.



To Probe Further...

- Readings in Agents (*Huhns & Singh, eds.*), Morgan Kaufmann, 1997
http://www.mkp.com/books_catalog/1-55860-495-2.asp
- *IEEE Internet Computing*, <http://computer.org/internet>
- DAI-List-Request@ece.sc.edu
- *International Journal of Cooperative Information Systems*
- *International Conference on Multiagent Systems (ICMAS)*
- *International Joint Conference on Artificial Intelligence*
- *International Workshop on Agent Theories, Architectures, and Languages (ATAL)*
- *IFCIS Conference on Cooperative Information Systems*

11/18/98 1:12 PMUniversity of South Carolina22