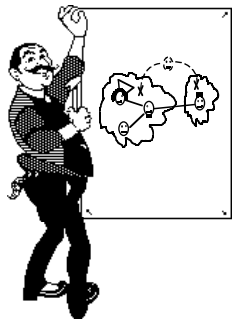
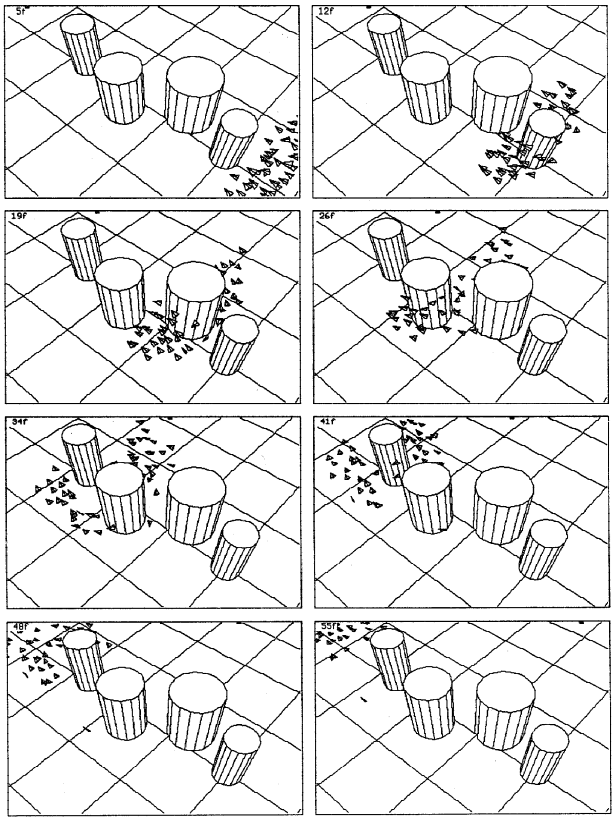
Introduction to Agents



A FLOCK IS NOT A BIRD



A FLOCK IS NOT A BIRD

Most bird flocks have no leader(s). Such flocks are examples of "self organization."

Each bird basically reacts to birds nearby by following a set of simple rules.

In the "boids" simulation of Craig Reynolds, each bird had *three simple rules of behavior*:

1) Maintain a minimum distance from other objects, including other boids.

2) Try to match velocities with the other boids.

3) Try to move towards the perceived center of the mass of boids in its neighborhood.

Craig Reynolds, DreamWorks SKG, Los Angeles (formerly from Symbolics Corp.); http://hmt.com/cwr/boids.html

COORDINATED WITHOUT A COORDINATOR

Such as,

- ❏ the photorealistic imagery of bat swarms used in *Batman Returns* and *Cliffhanger*, and
- ❏ the wildebeest stampede in *The Lion King*,
- ❏ also, ant colonies, stadium crowds, highway traffic, market economies, and immune systems.

StarLogo is descibed in *Turtles, Termites, and Traffic Jams* by Mitchel Resnick. Software can be downloaded from http://www.media.mit.edu/~startlogo.

COMPLEX ADAPTIVE SYSTEMS

A way of thinking about multiagent systems and their collective behavior

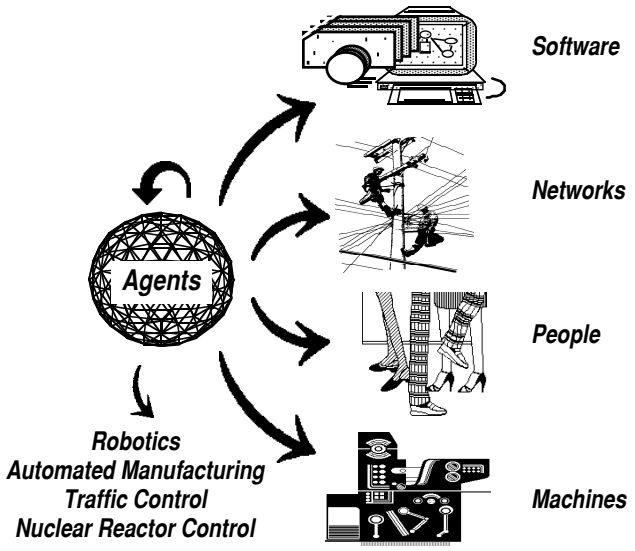
It is the study of the behavior of collections of simple units—
or agents, e.g., atoms, molecules, neurons, people,
that have the potential to adapt and
can give rise to coherent collective behavior.

Also,

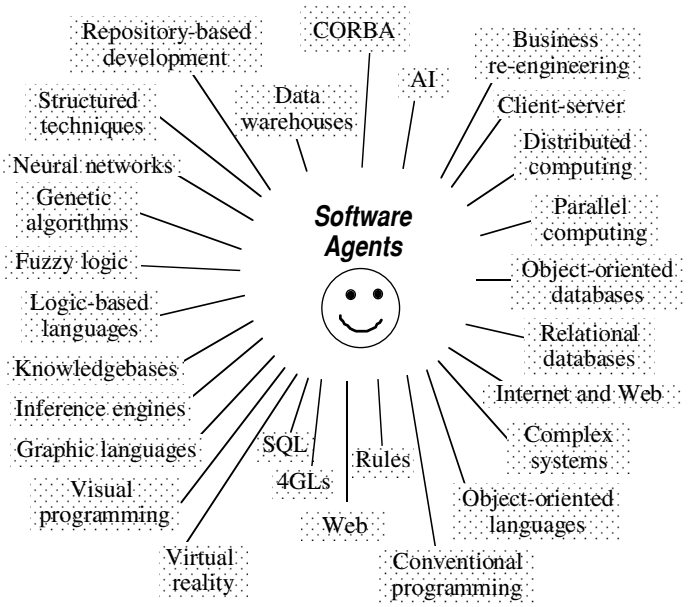
System (science)	Typical Mechanisms
Nucleus (physics)	Quarks, gluons
Atom (physics)	Protons, neutrons, electrons
Molecule (chemistry)	Bonds, active sites, mass action
Organelle (microbiology)	Enzymes, membranes, transport
Cell (biology)	Mitosis, meiosis, genetic operators
Multicellular organism (biology)	Morphogenesis, reproduction
Social group (biology)	Individuals, social relationships
Ecosystem (ecology)	Symbiosis, predation, mimicry

And, it can provide scaleability!

AGENTS CAN BE IMPLEMENTED IN MANY WAYS



SOFTWARE AGENTS CAN BE USED WITH OTHER TECHNOLOGIES



AGENTS STANDARDIZATION

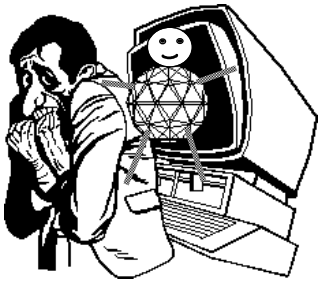
- **OMG Agents Working Group** recommends standards for agent technology where appropriate—particularly the OMG's Object Management Architecture (OMA). (www.omg.org)
- **FIPA (Federated Intelligent Physical Agents)** has been working to develop and promote standardization in the area of agent interoperability since 1996. It has an on-going work program, meeting around the globe on a quarterly basis, with excess of 50 member organizations. (www.fipa.org)
- **US DARPA (Defense Advanced Research Projects Agency)** has several research programs that address aspects of agent technology. These include:
  - Control of Agent-based Systems
  - Advanced Logistics Project
  - DARPA Agent Markup Language
- **KQML (Knowledge Query and Manipulation Language)** is a language and protocol for exchanging information and knowledge. It is part of a larger effort, the ARPA Knowledge Sharing Effort. (www.cs.umbc.edu/kqml/)

AGENTS STANDARDIZATION

- **AgentLink** is Europe's ESPRIT-funded Network of Excellence for agent-based computing. It is a coordinating organisation for research and development activities in the area of agent-based computer systems aimed at raising the profile, quality, and industrial relevance of agent systems in Europe. AgentLink (www.agentlink.org) divides its activities into four main areas:
  - Industrial action
  - Research coordination
  - Teaching and training
  - Infrastructure and management

- **CLIMATE (Cluster for Intelligent Mobile Agents for Telecommunication Environments)** represents a pool of agent-related projects within the European Union. CLIMATE was formed in Spring 1998 and currently comprises 14 core projects, and investigates various application areas, such as service control in fixed and mobile networks, telecommunications management, electronic commerce, and multimedia applications. For more information, see www.fokus.gmd.de/research/cc/ecco/climate/climate.html.

Agents: What are they?



AGENT

Something that acts—Webster

Some common properties of agents:

- **Autonomous -** is capable acting without direct external intervention.
- **Interactive -** communicates with the environment and other agents.
- **Adaptive -** capable of responding to other agents and/or its environment
- **Sociable -** interaction that is marked by friendliness or pleasantness
- **Mobile -** able to transport itself from one environment to another.
- **Proxy -** may act on behalf of someone or something.
- **Proactive -** goal-oriented, purposeful; does not simply react..
- **Intelligent -** state is formalized by knowledge (i.e., beliefs, goals, plans, assumptions) and interacts with other agents using symbolic language.
- **Rational -** able to act based on internal goals and knowledge.
- **Unpredictable -** able to act in ways that are not fully predictable.
- **Temporally continuous -** is a continuously running process.
- **Credible** - believable personality and emotional state.
- **Transparent and accountable -** must be transparent when required, yet must provide a log of its activities upon demand.
- **Coordinative -** able to perform some activity in a shared environment with other agents, via a plans, workflows, or some other process mechanism.
- **Cooperative -** able to coordinate with other agents to achieve a common purpose. (*Collaboration* is a synonymous term)
- **Competitive -** able to coordinate with other agents where the success of one agent implies the failure of others (the opposite of cooperative).
- **Rugged -** able to deal with errors and incomplete data robustly.
- **Trustworthy -** adheres to Laws of Robotics and is truthful.

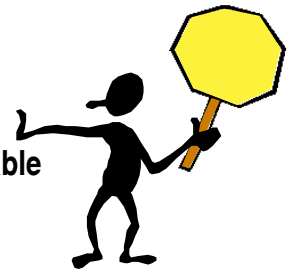
TWO ASPECTS OF AUTONOMY

Dynamic Autonomy

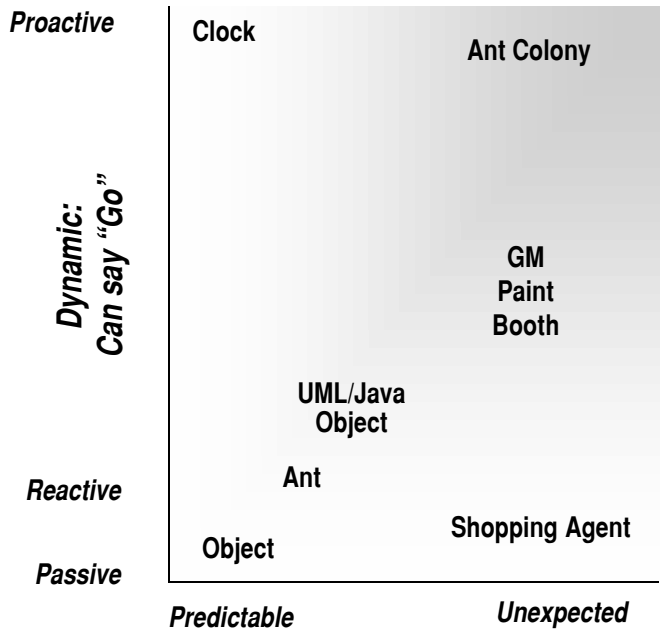
- **Can say "Go"**
- **Reactive vs. proactive**
- **Determined by agent's internal structure**

Unpredictable Autonomy

- **Can say "No"**
- **Predictable vs. Unpredictable**
- **Determined by external observer**



TWO ASPECTS OF AUTONOMY



ASPECTS OF ADAPTION AND INTERACTION

Adaption

- ❑ Reaction } Reactive
- ❑ Reasoning
- ❑ Learning } Deliberative
- ❑ Evolution

Interaction

- ❑ Communication
- ❑ Coordination
- ❑ Cooperation
- ❑ Competition

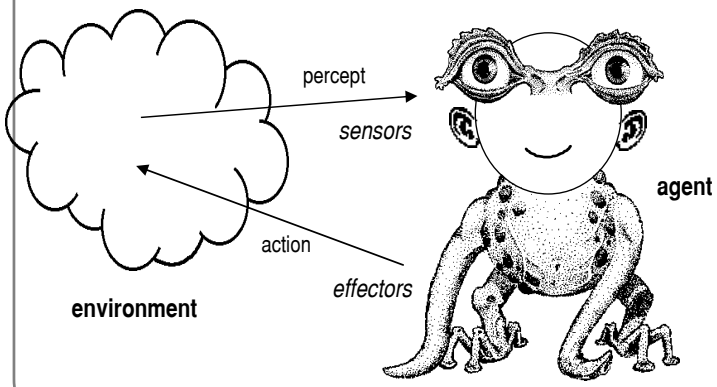
AGENT

Basic Working Definition

An autonomous entity that can interact with its environment.

In other words, it is anything that can be viewed as:

- perceiving its environment though sensors, and
- acting upon an environment with effectors.



To be useful, agents should —to *some* extent— be autonomous, adaptive, and communicative.

SOFTWARE AGENT

An autonomous software entity that can interact with its environment.

- ❑ In other words, they are agents implemented using software that
  - perceives its environment through its sensors, and
  - acts upon an environment with effectors.
- ❑ They can interact with other kinds of entities—including humans, machines, and other software agents in various environments and platforms.
- ❑ Function CALLs and object messages *invoke* operations—they do not involve *perception*.
- ❑ Software perception involves receiving external messages (e.g., requests, events, queries) on which the agent can choose to act.

SOFTWARE TECHNOLOGY AND AGENTS

	Monolithic Programming	Modular Programming	Object-Oriented Programming	Agent Programming
Unit Behavior	Nonmodular	Modular	Modular	Modular
Unit State	External	External	Internal	Internal
Unit Invocation	External	External (CALLed)	External (message)	Internal (rules, goals)

Software history is one of increasing localization and encapsulation.

Parunak, H. Van Dyke, *Autonomous Agent Architectures: A Non-technical Introduction*, Industrial Technology Institute, 10/13/95.

OO AND AGENTS

Conventional object orientation:

- ❑ is biased toward class-message-state.
- ❑ is centrally organized; yet some situations require a decentralized and self-organized approach (e.g., flocks of birds and paint stations).
- ❑ depends on external activation of objects as opposed to the continuous and concurrent activity of agents.
- ❑ does not express some business concepts (such as rules, constraints, goals, and roles and responsibilities).
- ❑ OO can be used to enable agent technology.

AGENT SOFTWARE

with examples of vendors*

Languages
- ❑ Microsoft, Inc.
- ❑ General Magic, Inc.
- ❑ Sun, Inc.
- ❑ Vertel, Inc.
- ❑ Agentsoft, Inc.
- ❑ First Virtual Holdings, Inc.

Development environments
- ❑ Agentsoft
- ❑ Autonomy
- ❑ Crystaliz
- ❑ Firefly Network
- ❑ FTP Software
- ❑ Fujitsu
- ❑ IBM
- ❑ KYMA Software
- ❑ MCC
- ❑ Microsoft
- ❑ Mitsubishi
- ❑ ObjectSpace
- ❑ Oracle
- ❑ Reticular Systems
- ❑ Toshiba
- ❑ Blackboard Technology
- ❑ Neuron Data, Inc.

Personalization
- ❑ Broadvision, Inc.
- ❑ Guideware, Inc.
- ❑ Agnetsoft, Inc.
- ❑ Wisewire, Inc.
- ❑ Aptex, Inc.
- ❑ Vignette, Inc.
- ❑ Firefly, Inc.

Research and development
- ❑ British Telecom
- ❑ Oracle, Inc.
- ❑ Digital Equipment Corp
- ❑ AT&T
- ❑ Apple Computer, Inc.
- ❑ Logica, Ltd.
- ❑ Siemens

Class libraries
- ❑ Agentsoft, Inc.
- ❑ IBM, Inc.
- ❑ General Magic, Inc.
- ❑ Objectspace, Inc.
- ❑ FTP Software, Inc.
- ❑ Crystaliz, Inc.

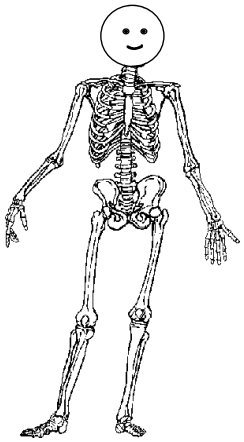
***See Appendix for a more complete list.**

HOW WILL WE USE AGENTS?

In general:
- ❑ **Personal use**
- ❑ **Interest matching**
- ❑ **Network and system management-** load balancing, failure anticipation, problem analysis, and information synthesis.
- ❑ **Information-** synthesis, decision, and logistics support
- ❑ **Process control-** ensure activities are carried out
- ❑ **Business process management-** capable of providing a variety of services, e.g., multiagent supply chain systems
- ❑ **E-Commerce-** with buying, brokering, bidding, and selling agents
- ❑ **Product design-** designing components and subsystems of a complex product

Some application areas:
- ❑ Manufacturing
- ❑ Utilities (electric, gas, telephone)
- ❑ Information
- ❑ Retail
- ❑ Finance (banks, insurance, stock market)
- ❑ Health
- ❑ Government (military, NASA)
- ❑ Air traffic control
- ❑ Media
- ❑ E-Mail
- ❑ Web personalization
- ❑ . . .

Agent Anatomy



AN AGENT SYSTEM

Agent System = <Agents, Environment, Coupling>

Agent = <State, Input, Output, Process>

State is the set of properties (values, true propositions) that completely describes the agent.

Input and **Output** are subsets of State whose variables are coupled to the environment.

Process is an autonomously executing process that changes the agent's State.

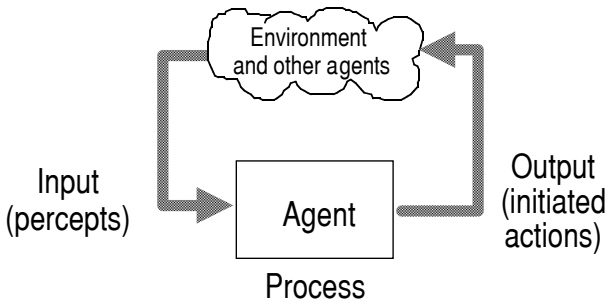
Environment = <State, Process>

The Environment has its own Process that can change its state, independent of the actions of its embedded agents.

Coupling is a mapping of an agent's Input and Output from/to the Environment's State.

Parunak, H. Van Dyke, *Go to the Ant: Engineering Principles from Natural Multi-Agent Systems*, Industrial Technology Institute, 4/4/97.

AGENT AS A BLACK BOX

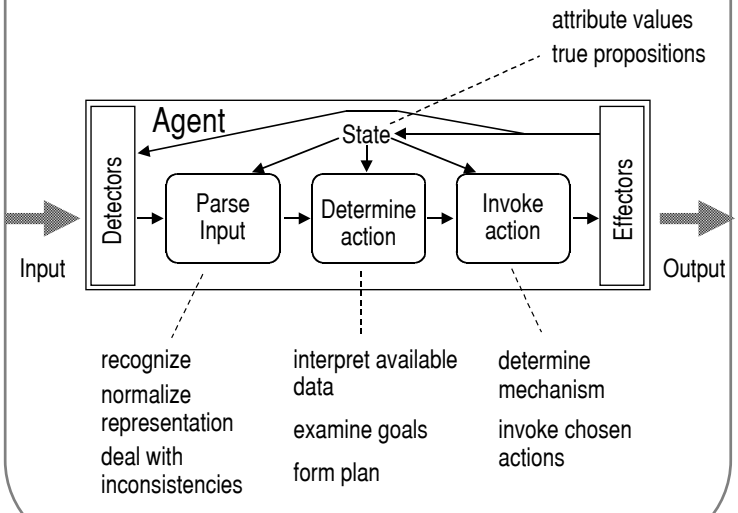


- **Input -** can be whatever the agent "perceives": assertions, queries, commands, state changes—that is, it is richer than OO messages.
- **Process -** interprets the perceived information, determines its action, and invokes it.
- **Output -** the initiated action, such as a change or communication to its world.

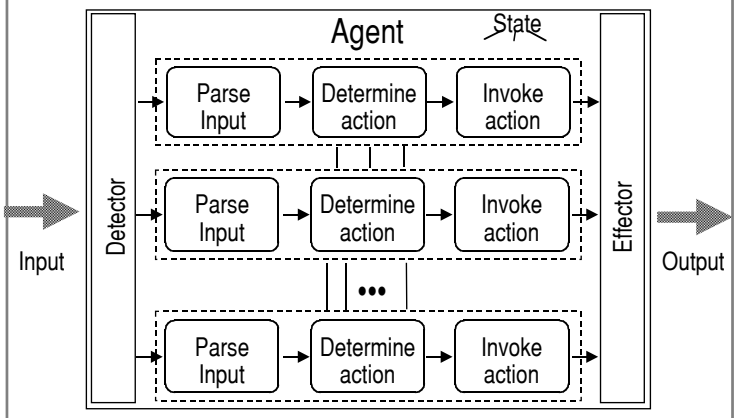
AGENT PROCESSING OVERVIEW

The primary purpose of an agent:

- is not just to interact with the environment,
- but rather to process and interpret the perceived information and to achieve some goal(s).



AGENTS CAN MULTIPROCESS



- ❑ **encourages concurrency**
- ❑ **minimizes single point of failure**
- ❑ **processing threads can communicate directly**
  - provides fast reaction capability
  - causes degradation when there is too much intercommunication
- ❑ **coordinators can be added to manage and control multiple threads**
  - reduces the need to embed all coordination functionality in every thread
  - increases processing overhead

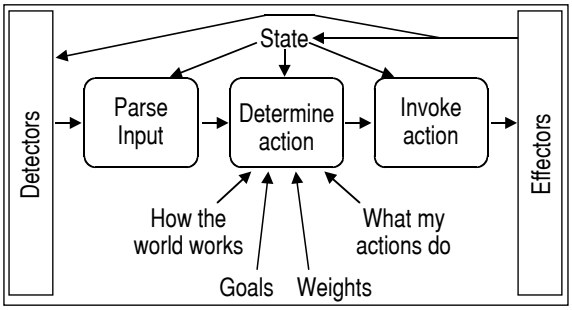
What are some examples of where agents can use multiprocessing?

REFLEX AND GOAL-BASED AGENTS

Goal-based agents

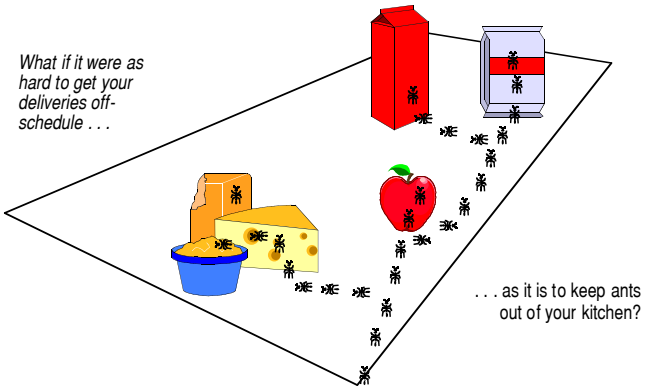
Have reflexes, as well as:

- ❑ Have goals to determine the selection of actions for a given circumstance.
- ❑ May optionally weigh the importance of certain goals (when there are several goals to choose from or conflicting goals).



IN THE MEANTIME,

if we built systems with ant-like simplicity,
we could create something very successful.

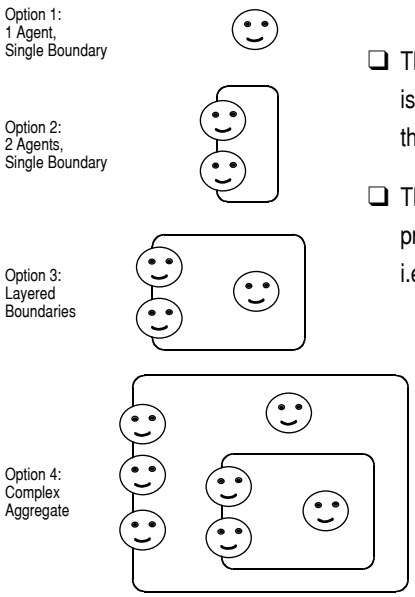


- ❑ Ants have been very successful for a very long time (7 million years).
- ❑ 15% of the Earth's biomass is composed of ants— much more than humans. (Another 17% is termites.)
- ❑ An ant has about 250,000 brain cells; humans have about 10 billion. (Simple rules; no intelligence)
- ❑ Ants can lift 20 times their own body weight. (They may be small, but they're more efficient.)
- ❑ If a person could run as fast for his size as an ant, he would match the pace of a race horse. (Ditto.)

Simple rules, high fault tolerance—and the colony thrives.
A great first-system model.

AGENTS AND AGGREGATE AGENTS

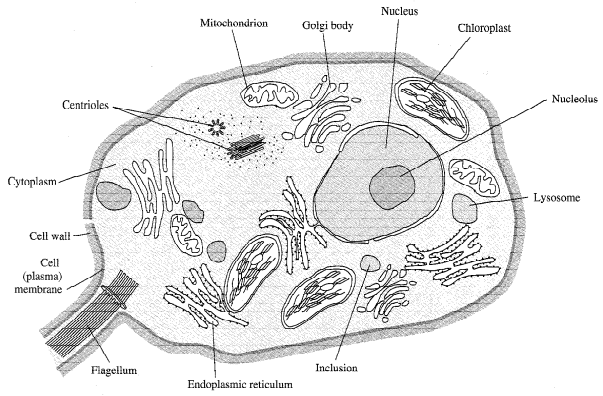
Communication can be with individuals as well as groups via agents defining the interface.



- ❑ The behavior of the aggregate is distinct from the behavior of the parts.
- ❑ The "membrane" agents provide the interface definition, i.e., the "cell" is encapsulated.

AGENTS AND AGGREGATE AGENTS

Using nature as an analogy

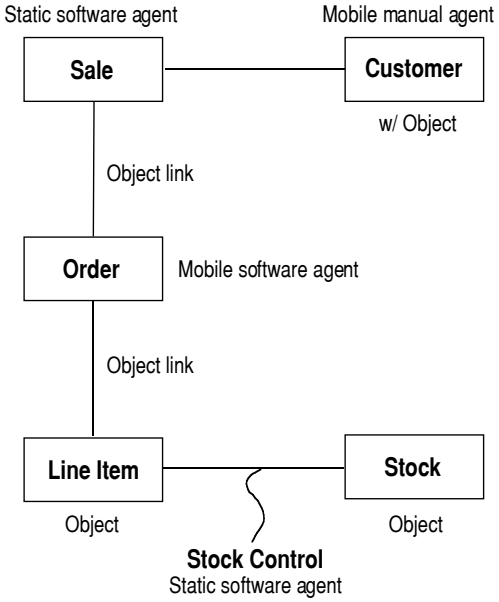


- ❑ parasitism
- ❑ symbiosis
- ❑ mimicry
- ❑ niche formation
- ❑ speciation
- ❑ sexual and asexual reproduction
- ❑ eukaryotic versus prokaryotic cells
- ❑ Darwinian evolution
- ❑ Lamarkian evolution
- ❑ neuron-based systems
- ❑ ...

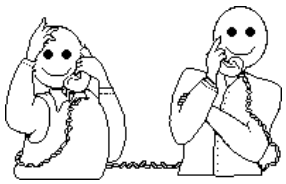
WHAT GETS AGENTED?

One technique: start with a conceptual-level class diagram and identify which objects and links are candidates for agent-hood.

An example with one possible solution.

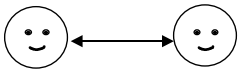


Agent Communication



CLIENT-SERVERS ARCHITECTURES AND SOFTWARE AGENTS

- ❑ Classical "me client, you server" architecture is too restrictive for agents.
- ❑ The peer-to-peer metaphor is closer.

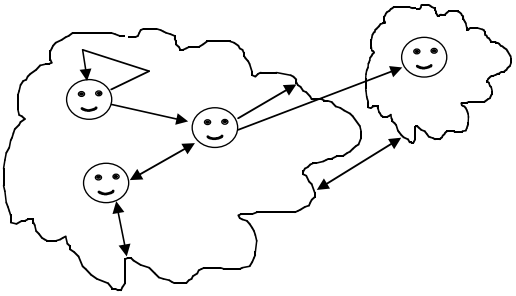


Agents can provide services at one moment and request services at another.

Furthermore, they can provide multiple services to multiple agents at the same time.

AGENT COMMUNICATION

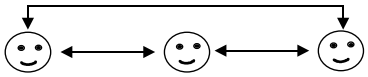
Agents can communicate with other agents—as well as with their environment.



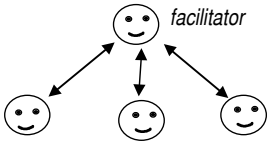
In fact, the environment itself can be treated as an agent, when appropriate.

INTERAGENT COMMUNICATION

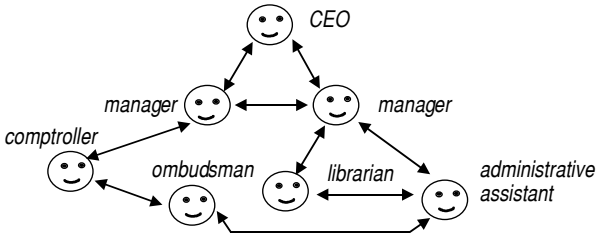
Uniform-agent architecture



Federated architecture



Highly specialized architecture



Corkill, Daniel D., and Susan E. Lander, "Organizing Software Agents: The Importance of Design to Effective System Performance," *Object Magazine*, 8:2, April 1998, pp. 41-47.

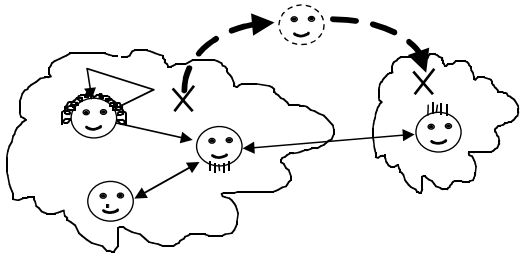
A COMMUNICATION ARCHITECTURE FOR SOFTWARE AGENTS INVOLVES:

- ❑ *Communication protocols*
  - **Unicast** - sending a packet when there is only one sender process and one specific recipient process.
  - **Broadcast** - sending only one packet and all the hosts in the network recognize and read it.
  - **Multicast** - sending only one packet and all the hosts that have registered interest recognize and read it.
- ❑ *Application protocols*
  - **Publish/subscribe** - decoupled, asynchronous, many-to-many, event-driven communication.
  - **Request/reply** - decoupled, synchronous, one-to-many, demand-driven communication.
  - **Solicit/response** - asynchronous request/reply.
- ❑ *Message routing*
  - **Subject-based**
  - **Content-based**
- ❑ *Message properties*
  - **Format repository service**
  - **Self-describing format**
  - **Transformation/translation service**
  - **Message priority**
  - **Message expiration**

Moreh, Jahan, "Publish & Subscribe: The Power behind Interactive Push Technology," *Distributed Computing*, 1:2, January/February, 1998, pp. 23-27.

STATIONARY VERSUS MOBILE AGENTS

Mobile agents are able to change platforms and environments; stationary agents are not.



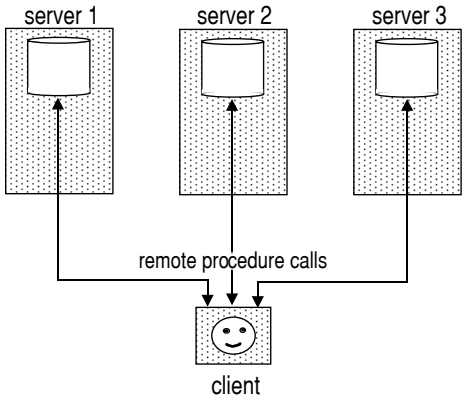
Stationary agents must use the network to exchange information. This:

- ✔ reduces complexity required by mobility.
- ✔ encourages specializations within platforms.
- ✔ employs well-established protocols.
- ✔ supports closed-environment philosophy.
- ✘ results in performance problems in situations requiring high volume or frequency.
- ✘ results in processing inefficiencies when the sum of the specialized agents makes more work than having a single mobile agent.
- ✘ reduces effectiveness when a connection is lost.

Many do not believe agent mobility is useful or necessary.

STATIONARY AGENTS

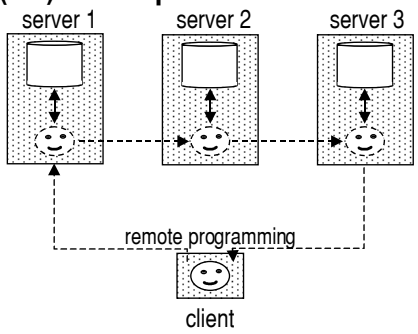
primarily use the Remote Procedure Call (RPC) technique for remote work.



- ❑ When an agent wants to use the services of another agent, a message (or request) conveys the intention to invoke a specific operation.
- ❑ The operation is then executed and the results (or reply) is returned to the requesting agent.
- ❑ Standard client-server protocols.

MOBILE AGENTS

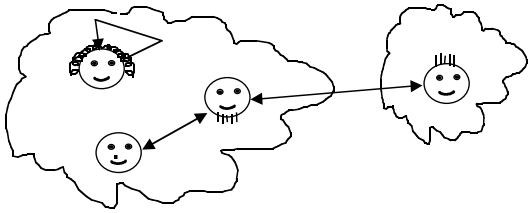
primarily use the Remote Programming (RP) technique for remote work.



- ❑ All structural and behavioral properties of the agent must be transferred during migration.
- ❑ Environmental differences must be changed or accommodated.
- ❑ The big issues:
  - How much time it takes to prepare for migration
  - How much data is actually transferred
  - The performance of the communication
  - Server-side burden
- ❑ Migrations can be handled by the agent which
  - ✔ reduces runtime environment complexity.
  - ✘ increases agent complexity.
- ❑ Migrations can be transparent to the agent which
  - ✔ reduces agent complexity.
  - ✘ increases runtime environment complexity

SINGLE VERSUS MULTIAGENT APPROACHES

Single agents have their use, but using many agents to solve problems can also be useful.

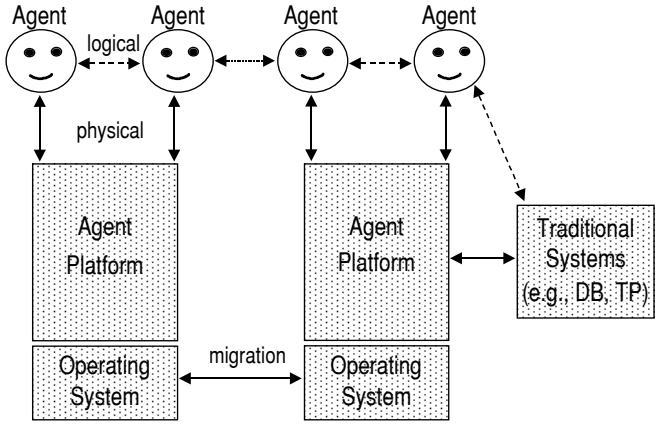


- ❑ **One agent *could* be constructed that does everything, but would represent a bottleneck for speed, reliability, etc.**
- ❑ **Dividing the functionality among many agents provides modularity, flexibility, modifiability, and extensibility.**
- ❑ **Support distributed processing and problem solving**
- ❑ **Specialized knowledge is often not available from a single agent.**
- ❑ **Knowledge is typically spread over various agents.**
- ❑ **Single-agent systems are much simpler because they they don't deal with cooperation, negotiation, etc.**

AGENT ARCHITECTURE

Agent comunication can be made in two ways:

- ❑ Directly with each other which
  - ✔ provides flexibility and freedom.
  - ✘ bypasses control and security.
- ❑ Through base software (preferred) which
  - ✔ resolves control and security problems.
  - ✘ requires "logical" communications which are physically resolved via the base software.



AGENT ARCHITECTURE

Three layer FIPA Agent Platform

Agent Management System	*Execution and monitoring of active agents* Basic functionality (API) - Identification - Query/Search - Directory Services - Negotiations - Registration - Mobility
Agent Platform Security Manager	*Secure transfer of messages and objects* Secure protocols Data encryption Digital signature Firewalls
Agent Platform Communication Channel	*Provision of base communication functions* Protocols, document formats RPC, remote programming Remote method invocation Object serialization

"FIPA Abstract Architecture Specification," FIPA Document PC00001, 2000.

Adaptation

ADAPTIVE AGENT

An agent that responds to its environment

Four primary ways of adapting:

- **reaction** - a direct, predetermined response to a particular event or an environmental signal.
- **reasoning** - ability to make inferences.
- **learning** - change that occurs during the lifetime of an agent.
- **evolution** - change that occurs over successive generations of agents.

ADAPTING BY REACTION

A direct, predetermined response to a particular event or environmental signal

Typically expressed in the form—

WHEN event, IF condition(s), THEN action:

- thermostats
- robotic sensors that can detect the presence of a nearby wall and activate a device for avoiding it
- washing machines and vacuum cleaners that use fuzzy logic

ADAPTING BY REASONING

A reactive response that uses inference rules.

A more advanced form of reactive adaptation using a set of rules to perform inferencing:

Typically *chains* of rules in the form—

WHEN event, IF condition(s), THEN action:

- ❑ patient diagnosis
- ❑ bulletin board or web foraging agents
- ❑ data mining

ADAPTING BY LEARNING

Change that occurs during the lifetime of an agent

Typical kinds of techniques:

- ❑ credit assignment
- ❑ Bayesian (or probablistic) rules
- ❑ neural networks
- ❑ classifier rules
- ❑ problem-specific structures

ADAPTING BY EVOLUTION

Change that occurs over successive generations of agents

Typical kinds of strategies:

- ❑ natural selection, i.e., survival of the fittest
- ❑ Darwinian versus Lamarckian evolution (e.g., genotype and phenotype)
- ❑ differentiation into ecosystem roles
- ❑ competition (e.g., increasing returns)
- ❑ cooperation (e.g., multiagent composition)
- ❑ coevolutionary arms races
- ❑ cultural transmission (e.g., Richard Dawkins' "memes")

ADAPTIVE AGENT

Four primary ways of adapting

Summary

- ❑ **Reaction (minimum requirement)**
- ❑ **Reasoning**
- ❑ **Learning**
- ❑ **Evolution**

} **Any or all in combination;**

e.g. Bayesian inference with memes and genes.

GENETIC PROGRAMMING

Solving quadratic equations

$$ax^2 + bx + c = 0$$

Familiar formula: $$x = \frac{-b \pm \sqrt{(b^2 - 4ac)}}{2a}$$

Given: `T = {A, B, C}` (3 coefficients)
`F = {+, -, *, %, SQRT}` (function set)

Generated code after 30 generations:

```
(% (- (% B (+ A A)) (% (SQRT (% (* (* (- (SQRT B)
   (SQRT B)) (* (+ (- C A) A) (% A B))) (* (SQRT
   (% B B)) (* (- (% (SQRT A) (SQRT (SQRT (SQRT
   A)))) (% (SQRT -1) (* C -3))) (- B B)))) (* A
   A))) (* C -3))) (- (% (% (- (SQRT (% (* (* (-
   (SQRT (% 3 -4)) (SQRT B)) (* (SQRT -1) (% A
   B))) (* (SQRT (% B B)) (* (- -2 C) (- B B))))
   (* A A))) (% (SQRT (SQRT (* (- B B) B))) (* C
   -3))) (- C A)) (SQRT -1)) (% (SQRT A) (% (SQRT
   (% (* (* (- (SQRT B) (SQRT B)) (* (SQRT -1) (%
   A B))) (* (SQRT (% B B)) (* (- -2 C) (- B
   B)))) (* A A))) (* C -3)))))
(% (SQRT (+ (+ B -5) (+ (- (* (- (+ B -5) 1) (%
   (+ 1 B) 4)) (+ (* C A) (- (+ (% (- B B) 4) -5)
   1))) (% (+ 1 (+ 1 B)) 4)))) A)).
```

Equivalent to: `(% (- B) (* 2 A))` *Plus/minus*
`(% (SQRT (- (* 0.25 B B) (* A C))) A))`

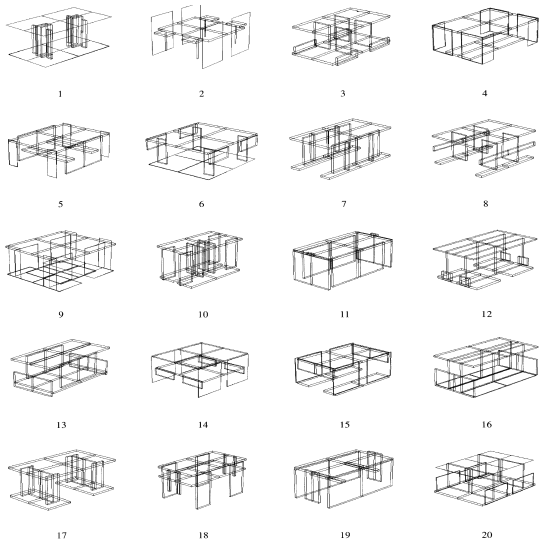
Koza, John R., *Genetic Programming: On the Programming of Computers by Means of Natural Selection.*, MIT Press, Cambridge, MA, 1992.

EVOLUTION OF DESK DESIGNS

Evaluation based on size, mass, flat upper surface, supportiveness, and unfragmented.

After 35 minutes, 20 "perfect" designs were evolved;

but which Is most pleasing to the eye?



The winner:

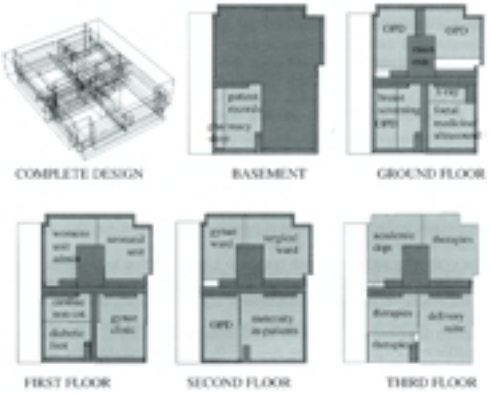
design number 4

Bentley, Peer J. ed., *Evolutionary Design by Computers*, Morgan Kaufman, San Francisco, 1999.

EVOLUTION OF HOSPITAL FLOOR PLAN

The GADES application evolved the size and location of 17 departments based on requirements, such as:

- ❑ Small, heavily constrained site in London (e.g. space, height, location of entrances and elevators)
- ❑ Usage frequency and volume, structural considerations (e.g. weight of equipment), lighting, regulations



- ❑ The problem was setup in a few hours and the design took 20 minutes.
- ❑ Generic enough to cope with many different design problems.
- ❑ An agent could periodically reassess and redesign as necessary.

Bentley, Peer J. ed., *Evolutionary Design by Computers*, Morgan Kaufman, San Francisco, 1999.

EVOLUTION OF CREATURES

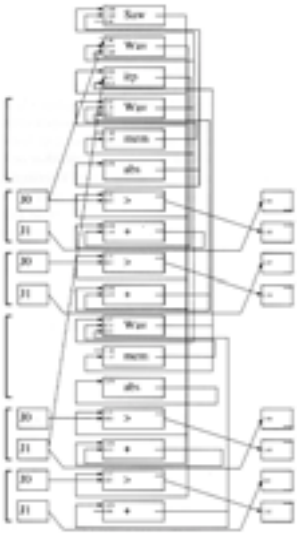
Karl Sims used genetic algorithms which drove both physical structure and its "nervous system. "

Physical structure includes:

- ❑ Nodes (e.g., size, shape, joint placement)
- ❑ Connectivity (e.g., position, orientation, recursion limit)
- ❑ Joint type (e.g., rigid, revolute, twist, bend)

Creature control includes:

- ❑ Sensors (e.g., joint angle sensors, contact sensors)
- ❑ Neurons (dataflow neural net style)
- ❑ Effectors (e.g., controlling joint degree of freedom, strength)

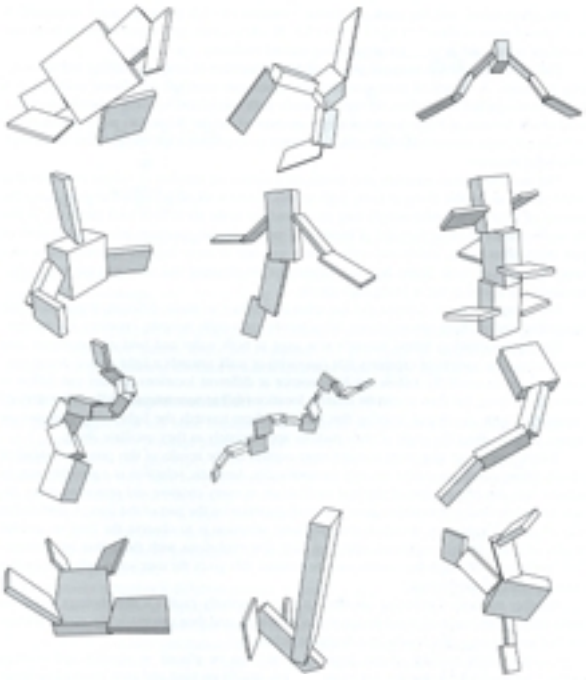


Creature gene

Bentley, Peer J. ed., *Evolutionary Design by Computers*, Morgan Kaufman, San Francisco, 1999.

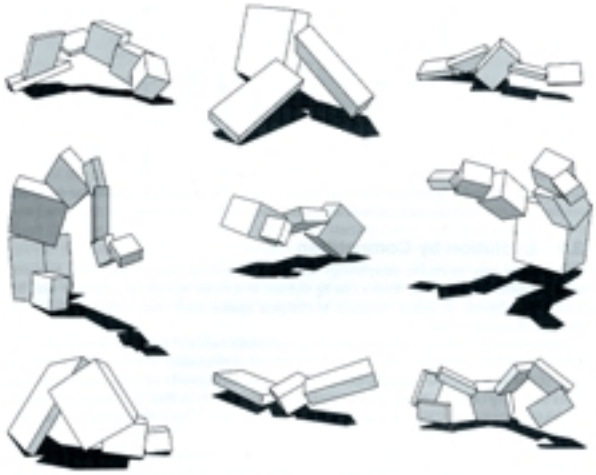
KARL SIMS EVOLUTION

Creatures evolved for swimming



KARL SIMS EVOLUTION

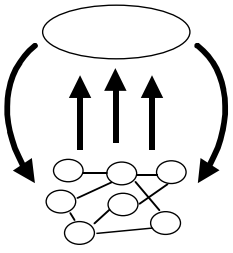
Creatures evolved for running



KARL SIMS EVOLUTION

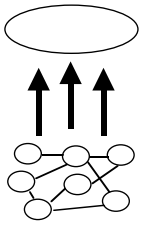
Video

Emergence



EMERGENCE

The appearance of a coherent pattern that arises out of interactions among simpler objects—but is* more *than just their summed behavior.

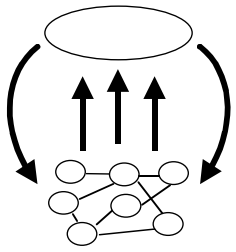


- ❏ Agents organize into a larger structure whose dynamics is greater than the sum of the components' dynamics.
- ❏ If a cluster is coherent and stable enough, it can usually serve as a building block for some larger cluster.
- ❏ At each level, new emergent structures can form and engage in behaviors that can lead to further levels of emergence.
- ❏ Building blocks at one level can combine to form building block at a higher level.
- ❏ Collections of agents can be homogeneous or heterogeneous
- ❏ Emergence is a property of systems, not of agents.

(e.g., 1000 people with $10 million and some stock don't make a market.)

Complexity is the science of emergence.

LOCAL INTERACTION, GLOBAL DYNAMICS



- ❏ Local interaction can give rise to global dynamics—creating a coherent structure.
- ❏ The global dynamics, in turn, can influence the local interaction.
- ❏ Here, the emergent structure is linked to the local interaction,
  - – influencing the boundary conditions of the local agents,
  - – as a result, local agents can adjust to the presence of the global dynamics,
  - – which might lead agents to change the conditions under which the agent behaves.

EMERGENCE— USING LIFE AS AN ANALOGY

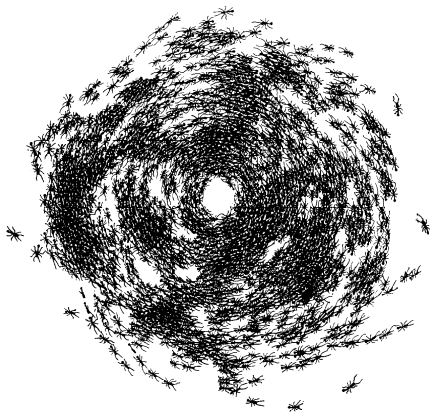
System (science)	Typical Mechanisms
Nucleus (physics)	Quarks, gluons
Atom (physics)	Protons, neutrons, electrons
Molecule (chemistry)	Bonds, active sites, mass action
Organelle (microbiology)	Enzymes, membranes, transport
Cell (biology)	Mitosis, meiosis, genetic operators
Multicellular organism (biology)	Morphogenesis, reproduction
Social group (biology)	Individuals, social relationships
Ecosystem (ecology)	Symbiosis, predation, mimicry

- ❏ Living systems are machines:
  - – Instead of being designed from top down the way human engineers do,
  - – living systems emerge from the bottom up from a population of much simpler systems.
- ❏ One possibility is that life isn't *just like* a computation. Life literally *is* a computation.

LIKE IT OR NOT, THE WORLD ISN'T STABLE.

It's full of evolution, upheaval, and surprise.

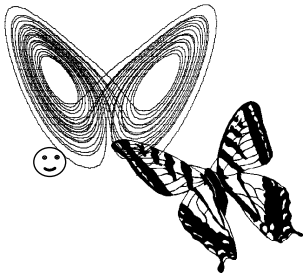
- ❏ Business is not a machine, but a kind of living system with all the spontaneity and complexity of life. New products, technologies, and markets are constantly arising and old ones are dying off.
- ❏ Tiny initial differences can produce enormously different effects. Simple dynamics can produce astonishingly complex behavior.



A circular mill of army ants that were cut off from the colony by rain. The workers were so attracted to each other that none left the group.

Holldobler, Bert and Edward O. Wilson, *The Ants*, Belknap Press, Cambridge, MA, 1990.

Between Chaos and Order

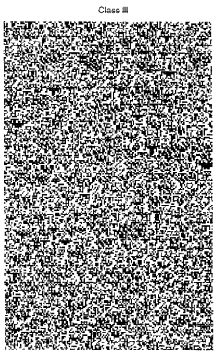


STEPHEN WOLFRAM'S CLASSIFICATION OF LONGTIME CA BEHAVIOR

disappears or becomes static or homogeneous

fixed finite size with indefinitely repeating structures

"chaotic"; little semblance of regularity

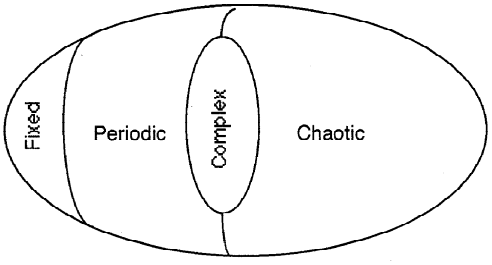


complex patterns grow and contract irregularly

(Courtesy of Andrew Wuensche, generated using his software "Discrete Dynamic Lab" from http://alife.santafe.edu/alife/software/ddlab.html)

BETWEEN ORDER AND CHAOS

Fitness is maximized between order and chaos.

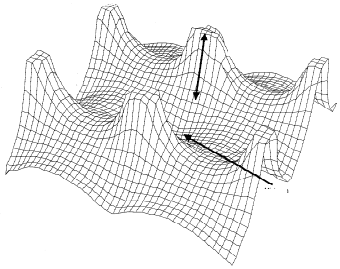


The highest average fitness occurs between ordered and chaotic behavior.

"The balance point—often called *the edge of chaos*—is where the components of a system never quite lock in place and yet never dissolve into turbulence, either."
"The edge of chaos is the constantly shifting battle zone between stagnation and anarchy."

— M. Mitchell Walthrop

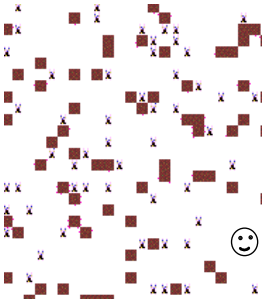
COEVOLUTION LANDSCAPE



The change in an individual or its species can alter the fitness landscape for other members of the ecosystem.

This can easily become chaotic.

Decentralization



DECENTRALIZATION IS HERE

On December 7, 1991, Boris Yeltsin met with the leaders of the Ukraine and Belarus. After two days of secret meetings, they issued a declaration: "The Union of Soviet Socialist Republics, as a subject of international law and geopolitical reality, is ceasing its existence."

The next day, IBM chairman John Akers publicly announced the decentralization of the computer giant into more than a dozen semiautonomous business units—each with its own financial authority and board of directors.

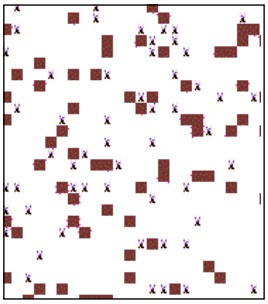
TOP-DOWN VERSUS BOTTOM-UP UNDERSTANDING

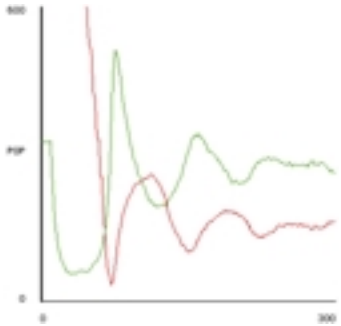
Example: Top-down, predator-prey models are based on sets of differential equations, know as the Lotka-Volterra equations:

$dn_1 / dt = n_1 (b - k_1 n_2)$, $dn_2 / dt = n_2 (k_2 n_1 - d)$, where:

n_1 = population density of prey, n_2 = population density of predator
b = birth rate of prey, d = death rate of predators, and
k_1 and k_2 are constants

Bottom-up would be creating a set of computer creatures that would interact and evolve.





Lotka-Volterra deals with aggregate quantities (population densities).

Simulation deals with the behaviors of individual creatures—from which the population dynamics emerge.

CENTRALIZATION VERSUS DECENTRALIZATION

Conclusion

- ❏ Resistance to decentralization exists. When people see a pattern, they often *assume* control is centralized.
- ❏ Centralized theories are necessarily wrong. *Some* phenomena are described quite well this way.
- ❏ Obviously, *one solution is not right for all situations.* Neither centralized nor decentralized is *the* solution.

The ones who thrive in decentralized environments are those who relish (not resist) unpredictability.

DECENTRALIZATION

Complexity from simplicity

When constructing agent systems:

- ❑ You control the action of the parts, not the whole.
- ❑ You act as a designer, but the resulting pattern is not designed.
- ❑ Self-organizing patterns are created without a central designer.
- ❑ You must have many agents acting in parallel to get "critical mass." A colony of 10 ants will not be sufficient.
- ❑ The parts must be interacting—parallelism is not enough. Without interactions, interesting colony-level behaviors will never arise.

Remember:

- ❑ A flock isn't a big bird.
- ❑ A traffic jam isn't just a collection of cars.

POSSIBLE GENERATIONS OF AGENT TECHNOLOGY

Generation	Description	Mainline Year
1	Agents are host based and standalone. They search the Web/Internet using fetch processing.	1994–2005
2	Agents are host based and capable of negotiating with computers and other agents, involving many business (and personal) functions.	1997–2005
3	Agents are mobile and highly personalized, but standalone.	1998–2010
4	Agents are mobile and capable of negotiating with computers and other agents.	1999–2010
5	Agents will also employ subagents.	2000–2020
6	Agents can activate and inhabit real-world robotics and pursue goals beyond software.	2001–2050
7	Agents are self-replicating and can design agents to specific needs. They are independent and self-motivating.	2005–2050

Murch, Richard, and Tony Johnson, *Intelligent Software Agents*, Prentice Hall, Upper Saddle River, NJ, 1999.