"Go to the Ant": Engineering Principles from Natural Multi-Agent Systems

Tristan Scheiner and Nyomi Morris

Introduction

- Influence of Code, State, Control:
  - Old software dependent on ‘caller’
  - New software ‘agents’ watch own internal responsibilities
- “100 agents, each with ten behaviors, require the programming of only 1000 individual behaviors, yet provide a behavior space on the order of 10^100”
  - Easier to maintain/adapt software
- Simpler agents work together more easily!

How do we teach the genius octopus in all its complexity to work with others?

Meanwhile the simple ants have it figured out!

Theoretical Context: Agenthood

Agents are defined by their interactions with their environment

$$Agents = \{Agent_1, \ldots, Agent_n\}$$

$$Agent_i = <State_i, Input_i, Output_i, Process_i>$$

- **State** - values defining the agent, variation between agents exists here
- **Input/Output** - coupled with environment, results from sensors & effectors
- **Process** - autonomously executing mapping changing the agent's State

Agents are related to other agents ***through* interactions with the environment

Theoretical Context: Environment

We can see the environment as a subset of an agent:

$$Environment = <State_e, Process_e>$$

- Environment unbounded by its own input/output
- Change in state coupled to agents' input/output
- Naturally aggregates low-level agents to high level ones → together they CHANGE the environment

Theoretical Context: Modeling

Table 2: Two Models of State and Process

Model	Discrete-Event Dynamical Systems	Time-Based Dynamical Systems
State	Discrete (symbols)	Continuous (real numbers)
Processing	Symbol manipulation, e.g., -- Rewrite rules -- Transition tables	Partial differential equations or difference equations
Progress	Discrete; event-based	Discrete (integer time) or Continuous (real time)
Time & State	Independent (clock speed)	Coupled (physical laws)
Communities	Computer Science, AI	Physics, Mathematical Ecology

It might be better for us to take a time-based approach to modelling swarm agents

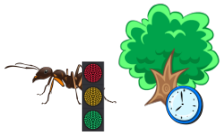
Theoretical Context: Coupling Agents + Environments

Homodynamic Systems

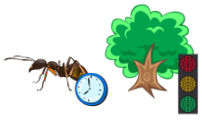
Directly relates agent action to change in environment→ can lead to frame problem

Models continuous energy flow → no single agent can predict the effects of its actions on the environment

Heterodynamic Systems



Agent's sensors must respond to environment flows and actuators provide energy flows



Reverse of the above→ unexplored territory research-wise, similar to HCI analysis

Natural Agent Systems: A Flocking Example

System Behavior—Flocks of birds stay together, coordinate turns, and avoid collisions with obstacles and each other.

Responsibilities (Simple Rules)—Each bird or fish follows three simple rules

1. Maintain specified minimum separation from the nearest object or other birds.
2. Match velocity (magnitude and direction) to nearby birds.
3. Stay close to the center of the flock.

Integration —Each entity's individual actions simultaneously respond to and change the overall structure of the flock. Each individual in the wolf-moose system both influences and is influenced by the entire system.

*We could probably use this structure in our use case style (See paper for others)

Other Engineering Principle Lists

Holland, *Hidden Order:* Understanding "Complex Adaptive Systems" (CAS's) by taking an analytical approach and offering four properties that these systems share and three mechanisms that they employ

Resnick, *Turtles, Termites, and Traffic Jams:* Wants to understand how people think about decentralized systems and gives five "guiding heuristics" that people can use to understand these systems

Kevin Kelly, *Out of Control:* Seeks out the laws "governing the incubation of something from nothing" and takes a perspective approach that is more philosophical

Agent Things, not Functions

Brief Summary: Agents should correspond to things in the problem domain rather than to abstract functions

- Classical software engineering leads towards “functional decomposition”
- Functional approach well suited for centralized systems
- In natural systems, functions emerge from the interaction of individual components

Keep Agents Small

Brief Summary: Agents should be small in mass (a small fraction of the total system), time (able to forget), and scope (avoiding global knowledge and action)

- Small in mass: Smaller agents are easier to design and develop and the impacts of failure are small compared to the entire system
- Small in time: The mechanism of forgetting plays a key role to prevent an agent from viewing the entire system and losing sight of individual goals
- Small in scope: Agents are only able to view their immediate surroundings and don't receive all information about the current state of the environment

Decentralized System Control

Brief Summary: The agent community should be decentralized, without a single point of control or failure

- A single agent control can lead to a single agent failure disrupting the rest of the system
- Single agent control can lead to bottlenecks in performance
- Single agents control can put a boundary on the system expansion

Support Agent Diversity

Brief Summary: Agents should be neither homogeneous nor incompatible, but diverse. Randomness and repulsion are important tools for establishing and maintaining this diversity

- Diversity and population size are not the same, a lot of homogeneous agents is different than fewer heterogeneous agents
- More diverse agents leads to more of the environment state space being explored
- Randomness can lead to simpler coordination among individuals and a higher amount of the state space being explored

Provide an Entropy Leak

Brief Summary: Agent communities should include a dissipative mechanism to whose flow they can orient themselves, thus leaking entropy away from the macro level at which they do useful work

- The addition of influence from outside the system is necessary for self-organization
- Agents must be able to orient themselves to environmental factors
- An agents actions must reinforce other agents similar actions

Enable Agents to Share Information

Brief Summary: Agents should have ways of caching and sharing what they learn about their environment, whether at the level of the individual, the generational chaoin, or the overall community organization

- Natural systems exchange information at three different levels: the species, the individual, and the society
- A community reduces the need for extensive searching by sharing information with one another
- Different levels learn by modifying different parts of the system

Plan and Execute Concurrently

Brief Summary: Agents should plan and execute concurrently rather than sequentially

- In more complicated industries, daily pre-planned schedules break apart really quickly leading to them being more general guides throughout the day
- Instead of making a plan and executing the plan, these systems should concurrently reason about a situation and act upon it
- Agents should seek to dynamically respond to their environment

Evaluation: Pros + Cons of Distributed Systems

Pros

- With good learning, agents can become as efficient as globally optimized systems
- Effective for conditions requiring adaptability
- Low cost maintenance

Cons

- Local autonomous agents could lose value captured by global approach
- Globally optimized competitors can kill swarm efforts before being fully realized

Summary: Rules for the Road

1. Agents correspond to things in **problem domain**, not abstract functions.
2. Agents are small in **mass** (fraction of total system), **time** (able to forget), and **scope** (local over global).
3. The agent community should be **decentralized**
4. Agents should be **neither homogeneous nor incompatible**, but diverse.
5. Agent communities should include a dissipative mechanism to whose flow they can orient themselves, thus leaking entropy away from the macro level at which they do useful work.
6. Agents have ways **sharing** what they learn about the environment
7. Agents should plan and execute **concurrently** rather than sequentially

Thank You! Questions?