



Center for
**LifeLong
Learning
& Design**

University of Colorado at Boulder

Wisdom is not the product of schooling
but the lifelong attempt to acquire it.
- Albert Einstein

Design, Learning and Collaboration

Fundamental Processes for Complex Systems

Gerhard Fischer and Leysia Palen
Spring Semester 1999

April 26, 1999

The Past and The Future

Theme	Past	Future
focus of interest	algorithm	complex system
relevant theories	physics, mathematics	biology
design methodology	building from scratch	reuse, redesign, adaptation, evolution

- **claims/challenges:**

- (many) software systems must evolve (they cannot be completely designed prior to use)
- (many) software systems must evolve at the hands of the users
- (many) software systems must be designed for evolution

Complex Systems: Why Do They Need to Evolve and How Can Evolution Be Supported

- **the basic message:** computational systems of the future
 - will be complex, embedded systems
 - need to be open and not closed
 - will evolve through their use by collaborating communities of practice
- **examples:**
 - domain-oriented design environments (DODEs)
 - * kitchen design: extensions for microwaves, critics checking appliances against the wall (unless island kitchens), designs for disabled people (blind, in wheelchairs)
 - * computer network design: new computers, new communication devices
 - Envisionment and Discovery Collaboratory (EDC) (versus SimCity)
 - operating systems (Linux) and high-functionality applications (MS-Word, Canvas,
 - courses as seeds
 - electronic journals (Journal of Interactive Media in Education (JIME))
 - buildings (see Stewart Brand: "How Buildings Learn - What Happens after they're built")

An Example of Closed versus Open Systems: SimCity

- **SimCity** allows users to build a city within a *given* framework (with specific object sets and constraints provided)
- **SimCity is a closed system** (apart from the SimCity Urban Renewal Kit (SCURK); an add-on module to allow users to change the appearance of objects)
- **example:** too much crime
 - solution supported: build more police stations (fight crime)
 - solution *not* supported: increase social services, improve education (prevent crime)
- **claim:** SimCity fails when applied to “real” city-planning problems (evidence: our collaboration with the City of Boulder and the Boulder County Healthy Community Initiative in transportation planning)
- **challenge:** build SimCity-like environments which are open and can be evolved (not by their designers, but by their users)

Theory and Practice of Design—A Quest for Evolution

Dawkins — “The Blind Watchmaker”: big-step reductionism cannot work as an explanation of mechanism; we can't explain a complex thing as originating in a single step

Simon — “The Sciences of the Artificial”: complex systems evolve faster if they can build on stable subsystems

Petroski — “To Engineer Is Human”: the role of failure in successful design

Brooks — “No Silver Bullet”: successful software gets changed, because it offers the possibility to evolve

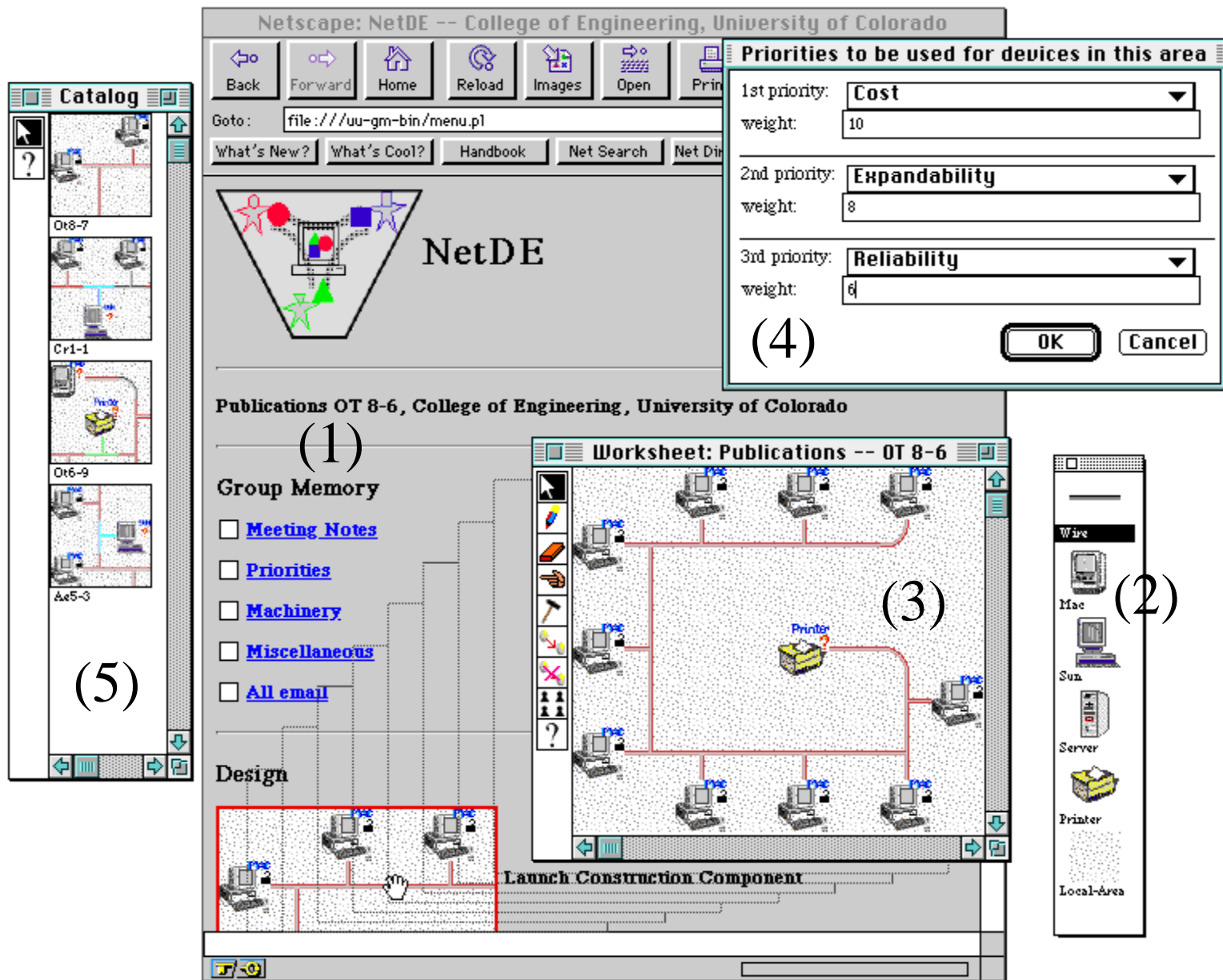
Polanyi — “The Tacit Dimension”: knowledge is tacit ----> we know more than we can say

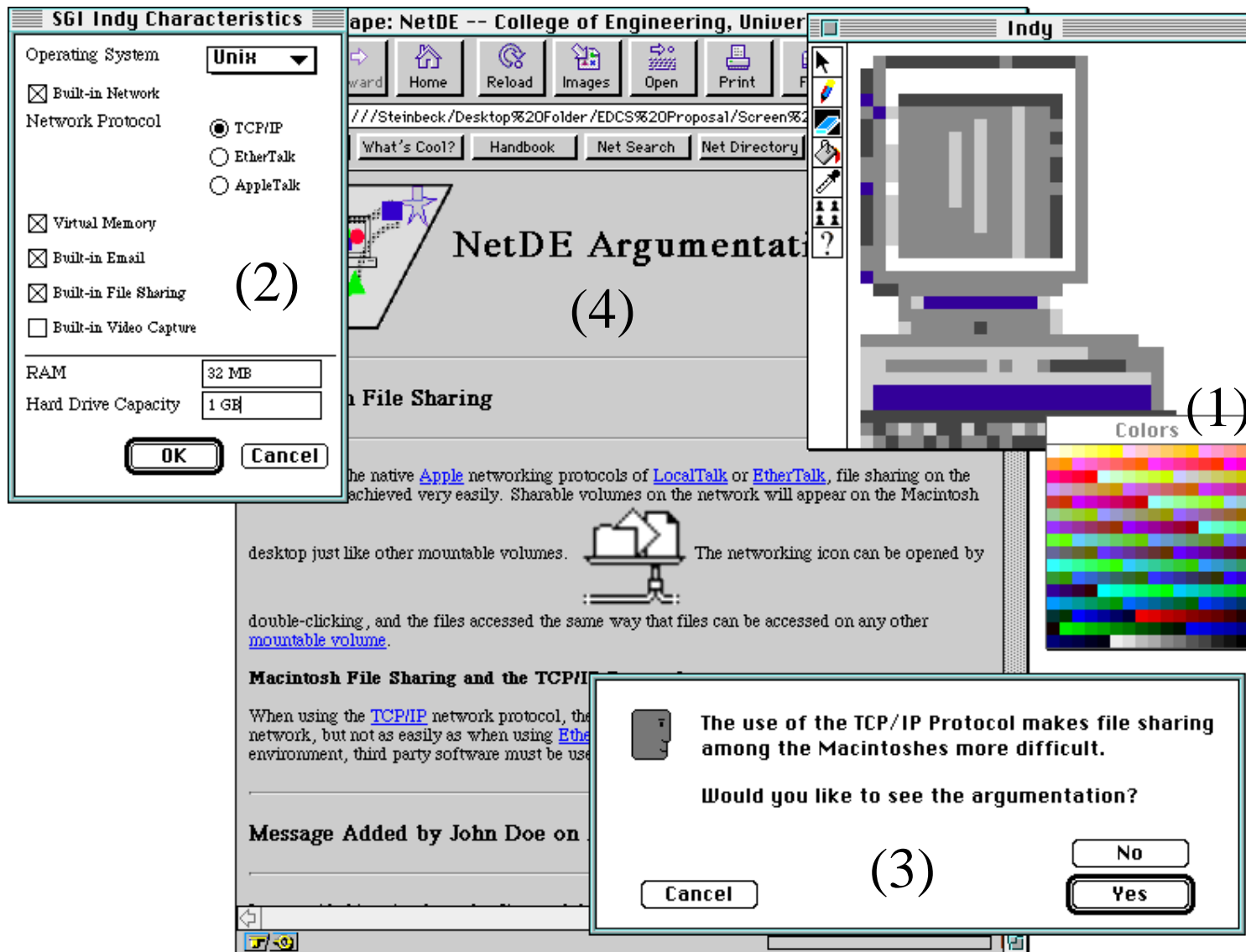
Karl Popper: Conjectures and Refutations

- John Archibald Wheeler: “Our whole problem is to make the mistakes as fast as possible.” (foreword to the book) — **breakdowns as opportunities**
- criticism of our conjectures is of decisive importance and all of our knowledge grows only through the correcting of our mistake— **critiquing systems**
- there are all kinds of sources of our knowledge but none has authority — **symmetry of ignorance and mutual competency**
- the advance of knowledge consists in the modification of earlier knowledge — **evolution**

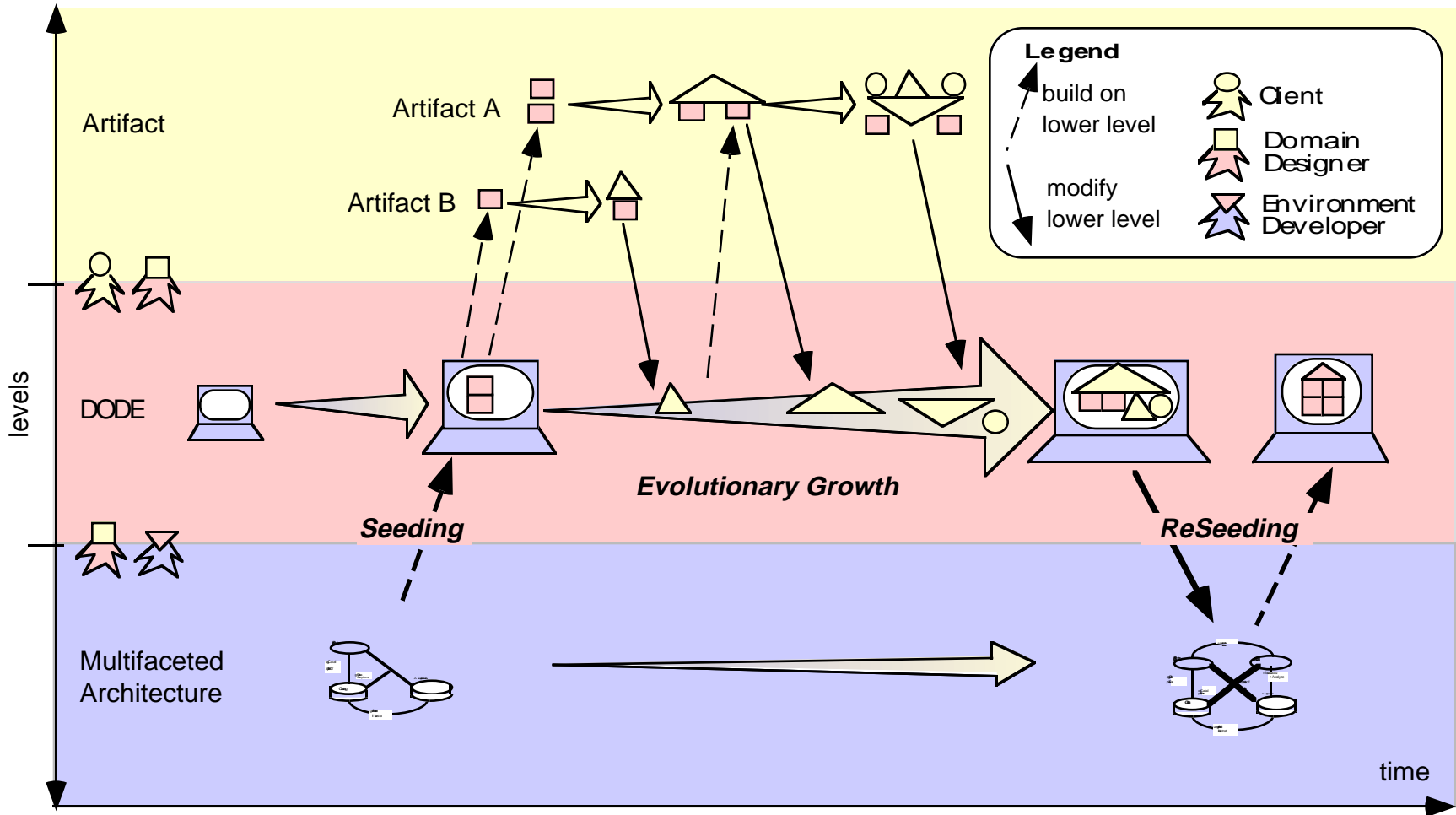
Domain-Oriented Design Environments and Evolution

- support the construction and evolution of **domains** (program families)
- empirical fact: **reuse** is most successful within domains
- **not just objects, but:**
 - case libraries (different granularity)
 - critiquing (accumulated “wisdom” of a community of practice, “virtual” stakeholders)
 - specification component — partial characterization of a situation model
 - simulation — to understand the behavior
 - argumentation — to explore the rationale behind the artifact
- the **Seeding, Evolutionary Growth, and Reseeding (SER)** model is a process model underlying the design, development and evolution of domain-oriented design environments



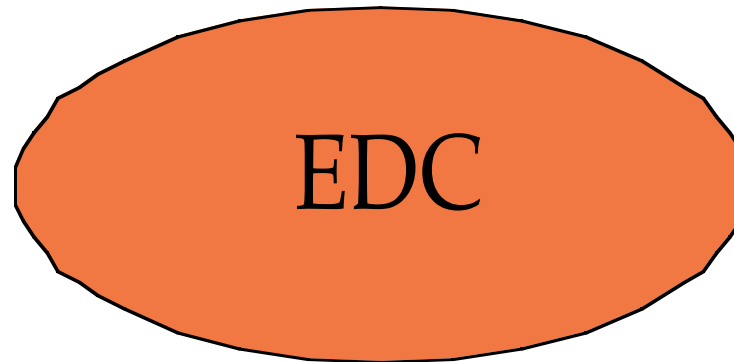


Seeding, Evolutionary Growth, and Reseeding (SER) Model



Overview of the EDC Environment

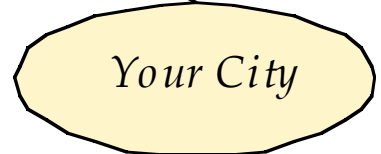
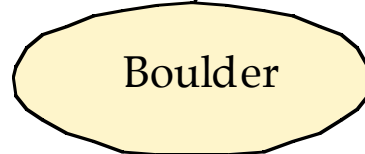
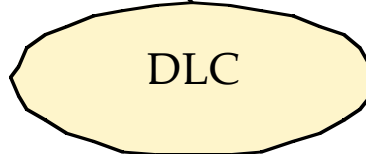
Domain-Independent Architecture



Application Domains



Specific Applications



Seeding, Evolutionary Growth, and Reseeding

- **seeding**
 - seed a domain-specific DODE using the domain-independent, multi-faceted architecture
 - provide representations for mutual learning and understanding between the involved stakeholders
 - make the seed useful and usable enough that it is used by domain workers
- **evolutionary growth**
 - co-evolution between individual artifacts and the DODE
 - learning on demand and end-user modifiability complement each other
- **reseeding**
 - formalize, generalize, structure
 - a social and technical challenge
- **success example of the SER model:**
 - development of operating systems
 - “communities of practice”

Evolution at All Three Levels

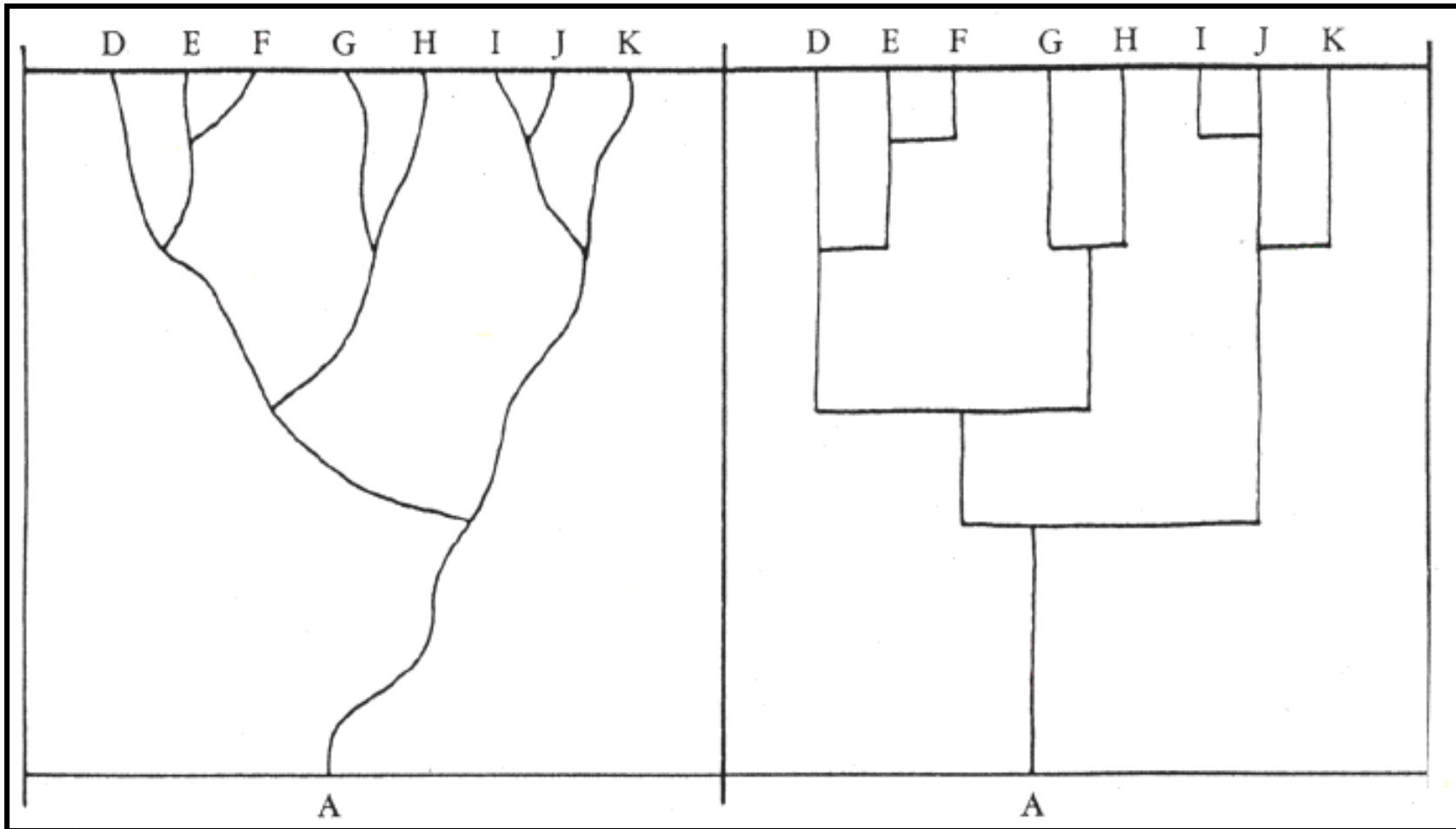
- evolution at the **conceptual framework** level
 - end-user modifiable DODEs
 - example: multifaceted, domain-independent architecture
- evolution of the **domain**
 - evolution is driven by new needs and expectations of users as well as new technology
 - example: the domain of computer network design
- evolution of **individual artifacts**
 - long-term, indirect collaboration
 - design rationale
 - example: the specific computer network at CU Boulder
- **co-evolution**
 - problem framing and problem solving (specification and implementation)
 - individual artifact and generic, domain-oriented design environment

Evolution in Biology versus Evolution in the Human-Made World — a Word of Caution

- **the evolutionary metaphor must be approached with caution because**
 - there are *vast differences* between the world of the made and the world of the born
 - one is the result of purposeful human activity, the other the outcome of a random natural process

- **does software develop according to the “punctuated equilibrium” theory?**
 - if yes, what causes the periods of increased change (subroutines, object-oriented programming, the World Wide Web (WWW))?

Punctuated Equilibrium



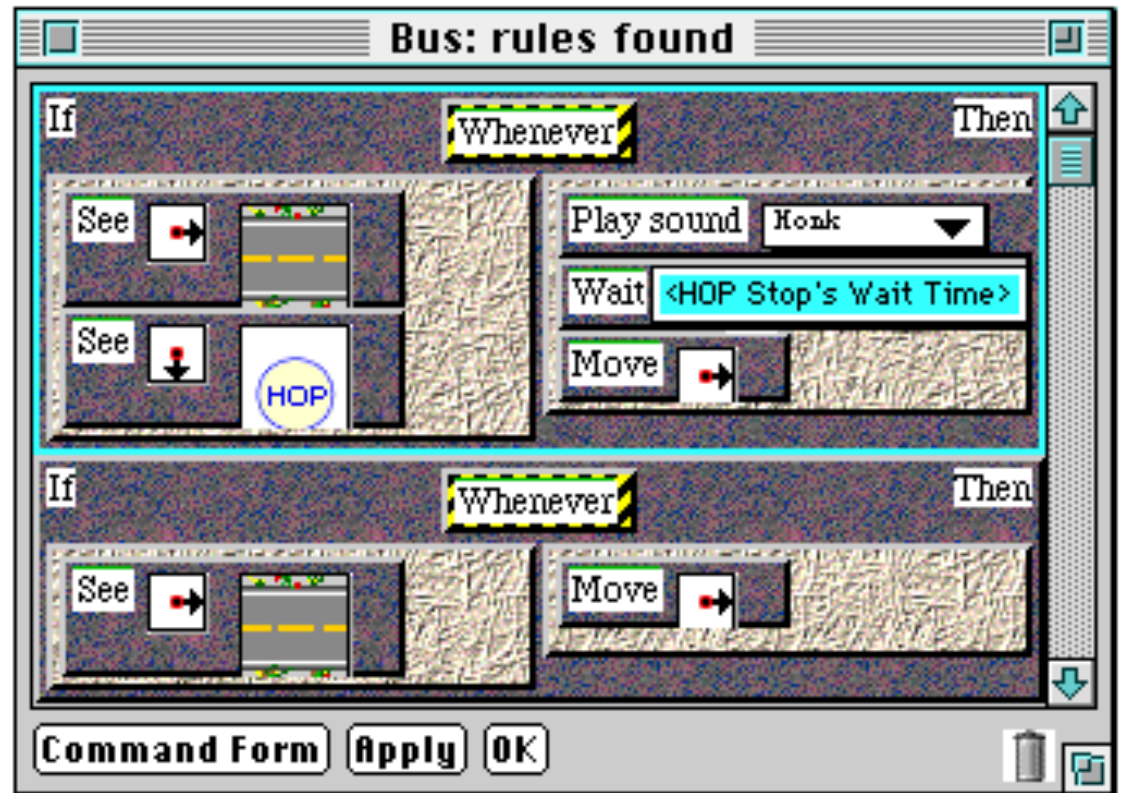
Prototypes of Systems Supporting Evolution

- **Modifier** (End-user modifiability component of Janus)
 - mechanisms to add new objects and new behavior by the domain designer
- **Gimme**
 - web-based group memory system
 - supports communication between all stakeholders
- **Expectation Agents** (with NYNEX, UC Irvine)
 - support communication between developers and end-users
 - observe actions of end-users and compare them to descriptions of the intended use
- **Chart 'n' Art** (self-disclosure): a gentle transition from direct manipulation interfaces to end-user programming
- **Visual AgenTalk (VAT)**
 - representations of conditions, actions and rules as graphical objects
 - interface support (drag and drop) for end-user programming
- **Behavior Exchange**: evolution by a community of practice over the WWW

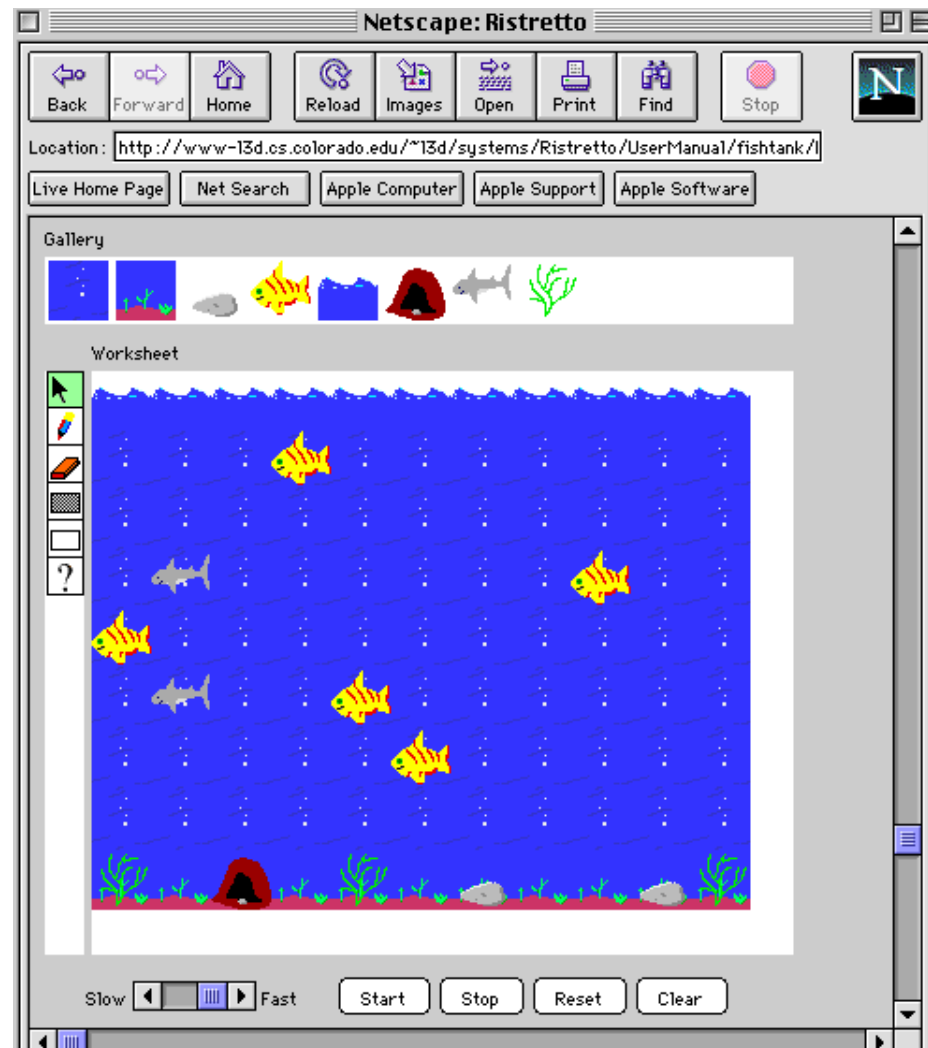
Visual AgentTalk



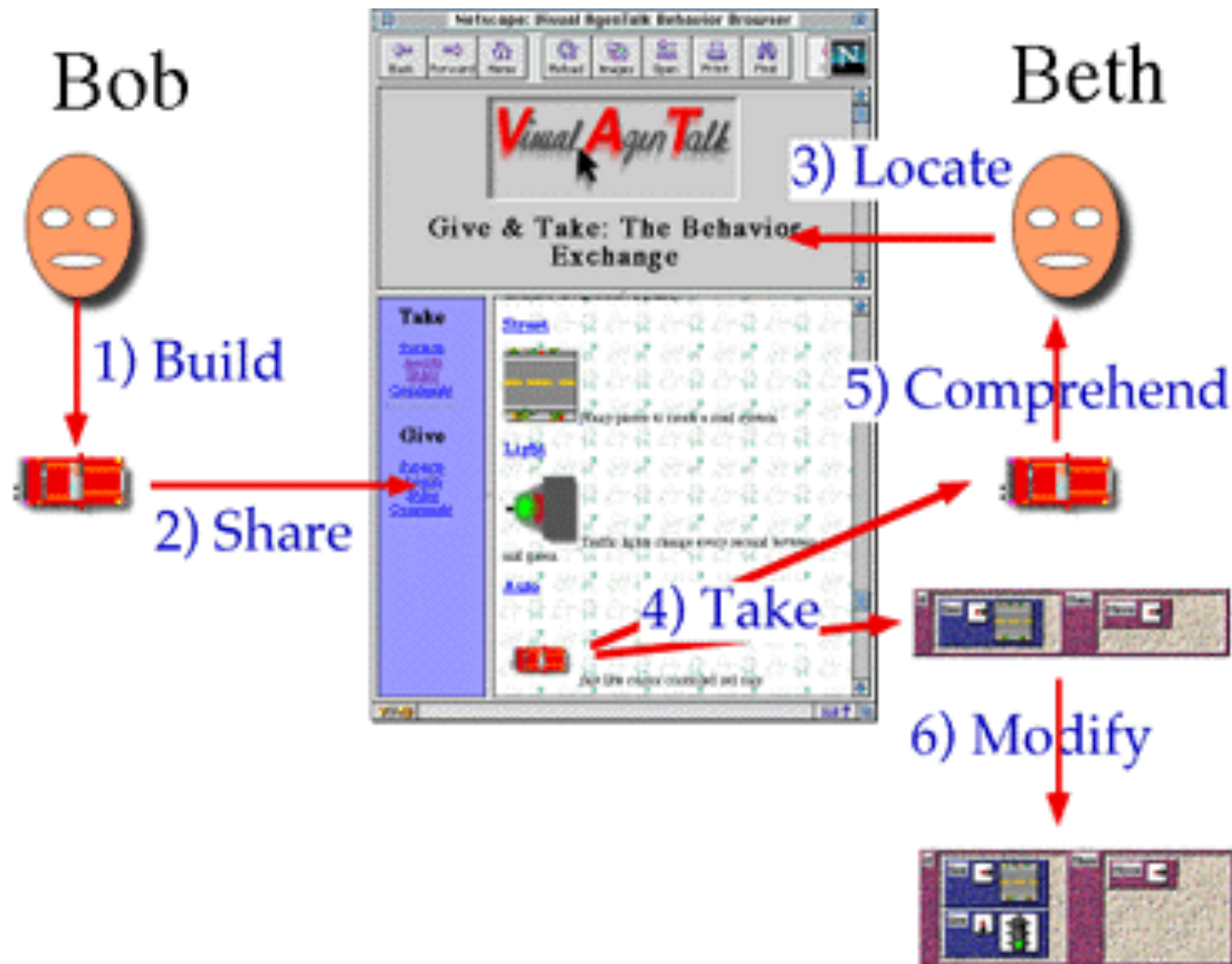
Visual Behavior Query

A screenshot of a graphical user interface for editing rules. The window title is "Bus: rules found". It contains two rule blocks, each starting with "If" and "Whenever" and ending with "Then". The first rule has two "See" conditions: one for a bus and one for an "HOP" sign. Its "Then" actions are "Play sound" (set to "Honk"), "Wait" (set to "<HOP Stop's Wait Time>"), and "Move". The second rule has one "See" condition for a bus and one "Move" action. At the bottom, there are buttons for "Command Form", "Apply", and "OK".

The FishTank — Created by a Community Using the Behavior Exchange



Processes Underlying the Behavior Exchange



Conclusions

- complex (software) systems should be regarded as **“living” entities** which are open and evolve
- the **seeding, evolutionary growth, reseeding (SER) model** is a feasible model for the evolutionary design of complex software systems
- complex (software) systems need to be **evolvable by their users**, not just by their developers
- these requirements create many interesting **research challenges** for
 - end-user modifiability
 - decentralized system development
 - new conceptualization of the WWW
 - culture changes in individuals (consumers --> designers) and organizations