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

The Software Technology of the 21st Century: From Software Reuse to Collaborative Software Design

Gerhard Fischer
Center for LifeLong Learning & Design (L3D)
http://www.cs.colorado.edu/~l3d/
Department of Computer Science and Institute of Cognitive Science
University of Colorado, Boulder, USA

ISFST2001 — International Symposium on Future Software Technology
November 5-7, 2001, ZhengZhou China

Overview

- **Fundamental Problems of Software Design**

- **Conceptual Frameworks**
    - Evolutionary Design
    - High-Functionality Applications
    - Software Reuse
    - Information Delivery
    - Collaboration (Social Creativity, Open Source)

- **Development of Systems**
    - **KID** —Domain-Oriented Design Environments
    - **CodeBroker** — Software Reuse and Information Delivery
    - **Envisionment and Discovery Collaboratory** — Collaborative Design

- **Conclusion**

The Basic Message

the fundamental challenge for software technologies of the future is to provide support for achieving a shared understanding among different groups of people that see the world in fundamentally different ways

Fundamental Problems of Software Design

- problems in semantically rich domains → thin spread of application knowledge

- modeling a changing world → changing and conflicting requirements

- complex problems → high-functionality applications (software reuse repositories)

- symmetry of ignorance → communication and coordination problems

Some Answers (and Systems) to the Fundamental Problems of Software Design

- problems in semantically rich domains → thin spread of application knowledge— **domain-oriented design environments (KID)**

- modeling a changing world → changing and conflicting requirements —**evolution (seeding, evolutionary growth, reseeding model)**

- complex problems → high-functionality applications (software reuse repositories) — **information delivery (CodeBroker)**

- symmetry of ignorance → communication and coordination problems — **representation for mutual understanding and mutual learning (Envisionment and Discovery Collaboratory)**

L3D's Path from Software Reuse to Collaborative Software Design

- Fischer, G. (1987) ***"Cognitive View of Reuse and Redesign,"*** *IEEE Software, Special Issue on Reusability,* 4(4), pp. 60-72.

- Fischer, G., Henninger, S. R., & Redmiles, D. F. (1991) ***"Cognitive Tools for Locating and Comprehending Software Objects for Reuse***." In *Thirteenth International Conference on Software Engineering (Austin, TX),* pp. 318-328.

- Fischer, G. (1994) "***Domain-Oriented Design Environments***," *Automated Software Engineering,* 1(2), pp. 177-203.

- Fischer, G., Redmiles, D., Williams, L., Puhr, G., Aoki, A., & Nakakoji, K. (1995) "***Beyond Object-Oriented Development: Where Current Object-Oriented Approaches Fall Short,"*** *Human-Computer Interaction, Special Issue on Object-Oriented Design,* 10(1), pp. 79-119.

- Fischer, G., Lindstaedt, S., Ostwald, J., Stolze, M., Sumner, T., & Zimmermann, B. (1995) "***From Domain Modeling to Collaborative Domain Construction."*** In G. M. Olson & S. Schuon (Eds.), *Proceedings of DIS'95 Symposium on Designing Interactive Systems,* ACM, New York, pp. 75 - 85.

- Fischer, G., Lemke, A. C., McCall, R., & Morch, A. (1996) "***Making Argumentation Serve Design.***" In T. Moran & J. Carrol (Eds.), Design Rationale: Concepts, Techniques, and Use, Lawrence Erlbaum and Associates, Mahwah, NJ, pp. 267-293.

L3D’s Path from Software Reuse to Collaborative Software Design — Continued

- Fischer, G. (2000) "***Social Creativity, Symmetry of Ignorance and Meta-Design***," Knowledge-Based Systems Journal (Special Issue on Creativity & Cognition), Elsevier Science B.V., Oxford, UK, 13(7-8), pp. 527-537.

- Fischer, G. & Scharff, E. (2000) "***Meta-Design—Design for Designers***," 3rd International Conference on Designing Interactive Systems (DIS 2000), pp. 396-405.

- Fischer, G., Grudin, J., McCall, R., Ostwald, J., Redmiles, D., Reeves, B., & Shipman, F. (2001) "***Seeding, Evolutionary Growth and Reseeding: The Incremental Development of Collaborative Design Environments***." In G. M. Olson, T. W. Malone, & J. B. Smith (Eds.), Coordination Theory and Collaboration Technology, Lawrence Erlbaum Associates, Mahwah, New Jersey, pp. 447-472.

- Fischer, G. & Ye, Y. (2001) "***Personalizing Delivered Information in a Software Reuse Environment.***" In M. Bauer, P. Gmytrasiewicz, & J. Vassileva (Eds.), Proceedings of User Modeling 2001 (8th International Conference, UM 2001), Sonthofen, Germany, Springer Verlag, Heidelberg, pp. 178-187.

References about L^{3}D Work in Japanese

- Fischer, G. (1998) **"Beyond 'Couch Potatoes': From Consumers to Designers."**
    - English Version: IEEE (Ed.) *1998 Asia-Pacific Computer and Human Interaction, APCHI'98,* IEEE Computer Society, pp. 2-9. At: http://www.cs.colorado.edu/~gerhard/papers/apchi.pdf
    - **Japanese Version (translated by K. Nakakoji): bit Magazine, Volume 31, No 4, April, Kyoritsu Shuppan, Tokyo, Japan, pp 11-21.**

- Fischer, G. (1999) **"A Group Has No Head — Conceptual Frameworks and Systems for Supporting Social Interaction**”
    - English Version: http://www.cs.colorado.edu/~gerhard/papers/IPSJ99.pdf
    - **Japanese Version (translated by M. Sugimoto): Information Processing Society of Japan (IPSJ) Magazine, 40(6), pp. 575-582.**

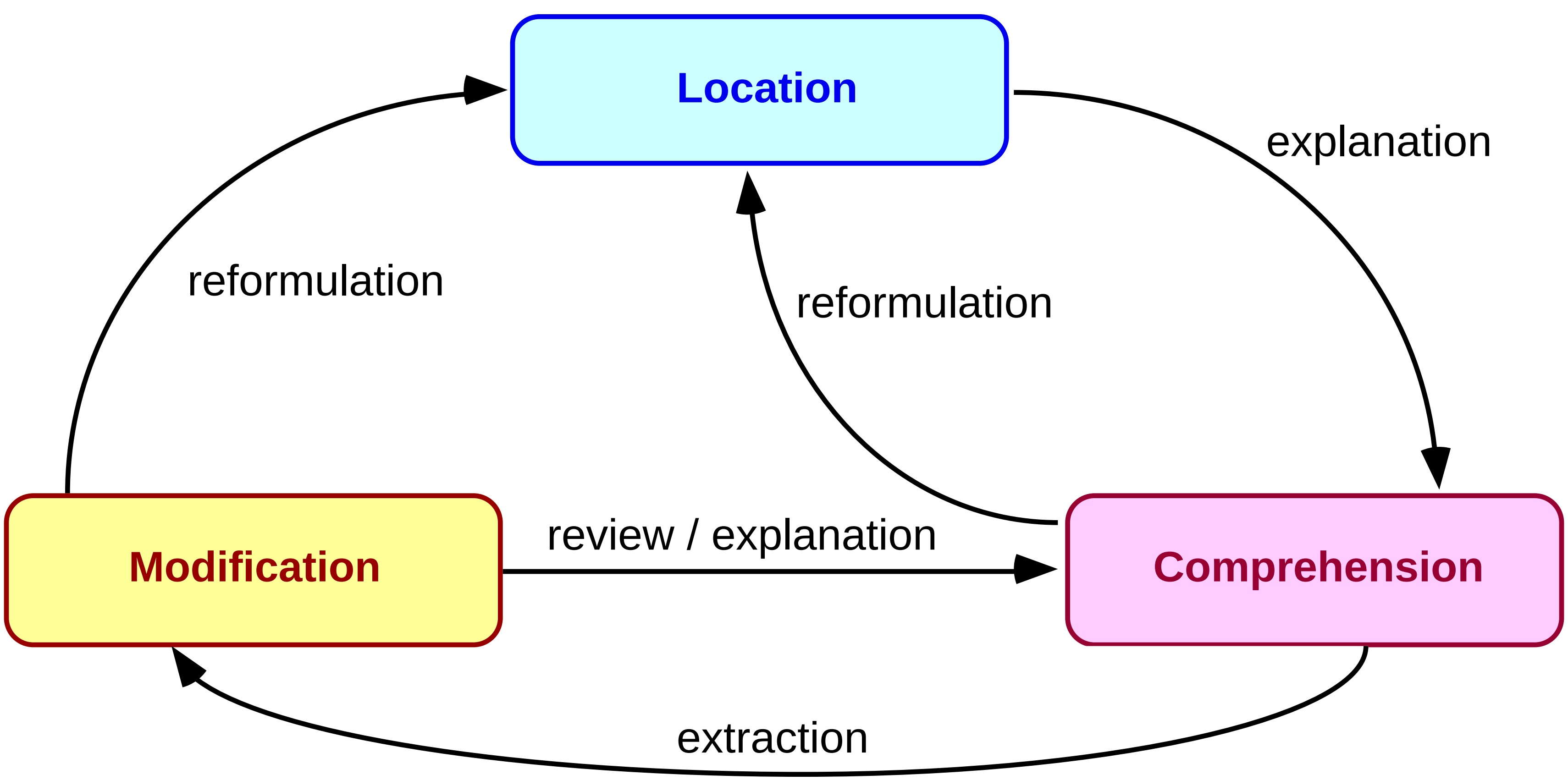
Past and Present Concerns for Software Technologies

dimension	past	present
limiting resource	information	human attention
models for collaboration	access	informed participation
design tools	focus: “downstream activities” — robust implementations of given specifications	focus: “upstream activities” — co-evolution between problem framing and problem solving
design products	finished systems	evolution
support for collaboration	file transfer	world-wide web (WWW)
model for creation	individual creativity	social creativity
documents	formal and informal objects of specific communities of practice	boundary objects: supporting collaboration between different communities
focus of software reuse	technical issues	cognitive, social issues
intellectual property	closed, company-owned	models for sharing (e.g., open source)

Overview of Conceptual Frameworks and Systems

Fundamental Challenge	Conceptual Frameworks	Systems	Author(s)
complex systems; high-functionality applications	software reuse; evolutionary design	Codefinder, Explainer, Modifier	S. Henninger, D. Redmiles, A.Girgensohn
problem-specific interaction	domain-oriented design environments	KID (Knowing-in-Design)	Kumiyo Nakakoji
organization of large bodies of knowledge; information overload	personalized information delivery	CodeBroker	Yunwen Ye
collaboration	collaborative design, social creativity	Envisionment and Discovery Collaboratory (EDC)	E. Arias, H. Eden, A. Gorman, E. Scharff

The Location-Comprehension-Modification Cycle



- **systems supporting these processes:**

-	Location	→	Codefinder	(Scott Henninger)
-	Comprehension	→	Explainer	(David Redmiles)
-	Modification	→	Modifier	(Andreas Girgensohn)

The Software Design Methodology of the Future
—
Evolutionary Design of Complex Systems

- complex (software) systems should be regarded as **“living” entities** which are open and evolve

- 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: allowing consumers to act as designers
  - decentralized system development
  - new conceptualization of the WWW
  - mindset changes in individuals and culture changes in organizations

Seeding, Evolutionary Growth, and Reseeding (SER)

A Process Model for the Evolutionary Design of Complex Software Systems

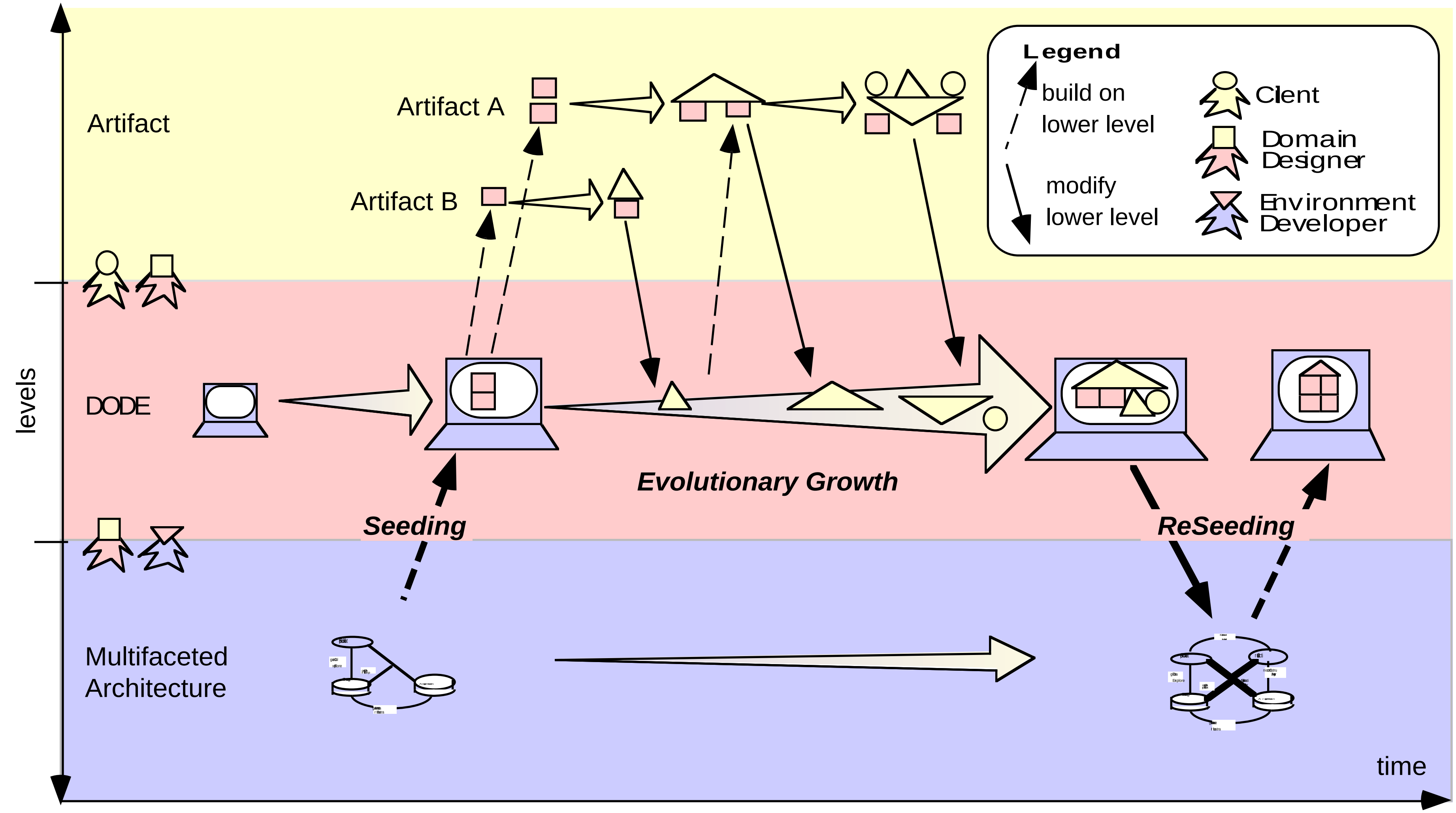
- **seeding**
  - seed a domain-specific domain-oriented design environments (DODE) using a domain-independent architecture
  - provide representations for mutual learning and understanding between the involved stakeholders (e.g., system developers and system users)
  - make the seed useful and usable enough for 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
  - domain-oriented design environments
  - open source projects

Seeding, Evolutionary Growth, and Reseeding (SER) Model



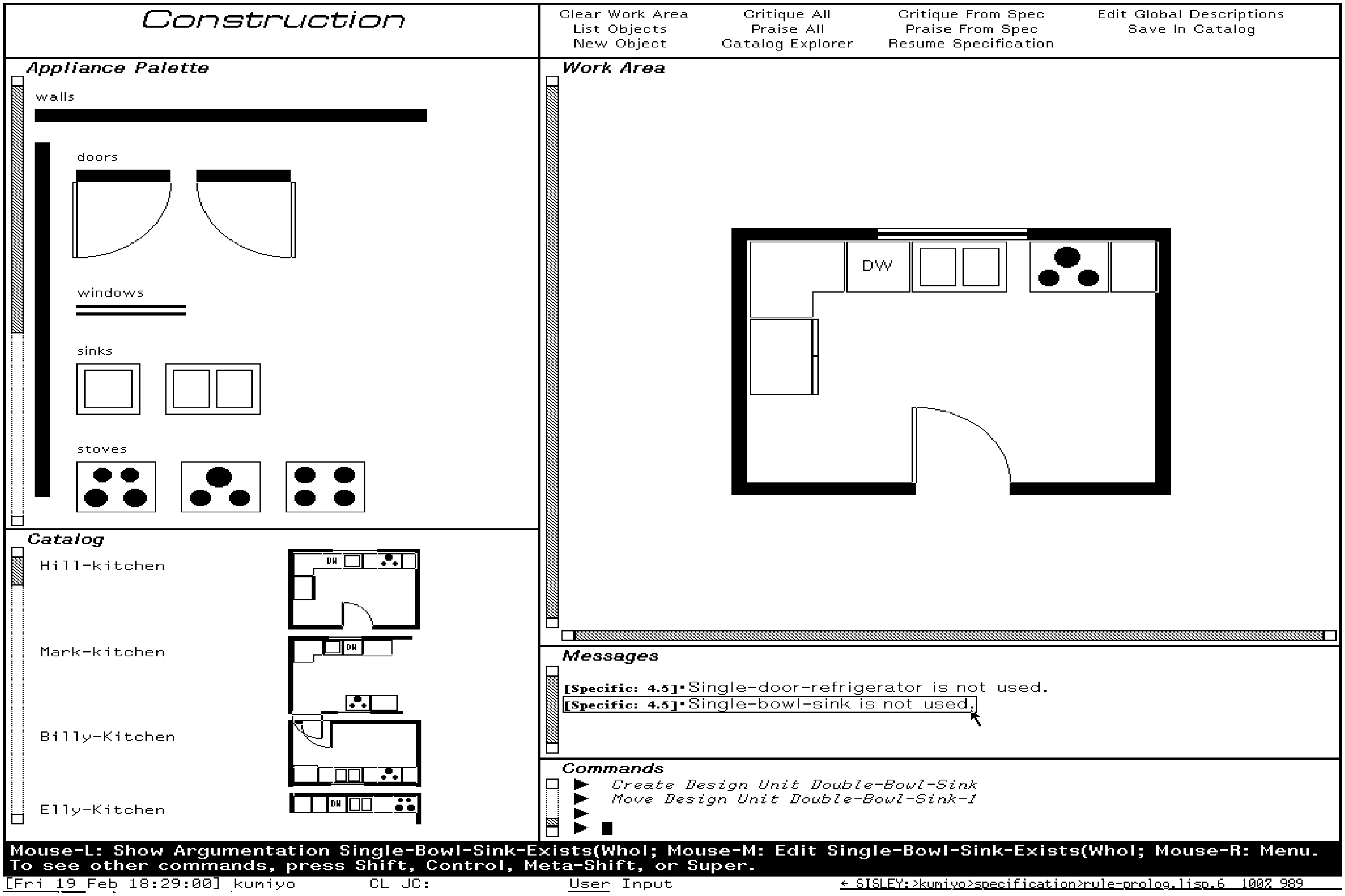
Domain-Oriented Design Environments (DODEs)

- problems in semantically rich domains → thin spread of application knowledge— **domain-oriented design environments**

- software design challenges:
  - develop domain models
  - support human problem-domain interaction (not only human computer interaction)
  - empower end-users as designers

- examples:
  - **KID (Knowing-in-Design): A DODE for Kitchen Design (Kumiyo Nakakoji)**
  - computer network design
  - voice dialog design
  - ……………………

KID (Knowing-in-Design) — A DODE for Kitchen Design



[Specific: 4.5]•Single-door-refrigerator is not used.
[Specific: 4.5]•Single-bowl-sink is not used.

```
▶  Create Design Unit Double-Bowl-Sink
▶  Move Design Unit Double-Bowl-Sink-1
▶
▶ ■
```

```
Mouse-L: Show Argumentation Single-Bowl-Sink-Exists(Whol; Mouse-M: Edit Single-Bowl-Sink-Exists(Whol; Mouse-R: Menu.
To see other commands, press Shift, Control, Meta-Shift, or Super.
[Fri 19 Feb 18:29:00] kumiyo       CL JC:           User Input           ← SISLEY:>kumiyo>specification>rule-prolog.lisp.6  100% 989
```

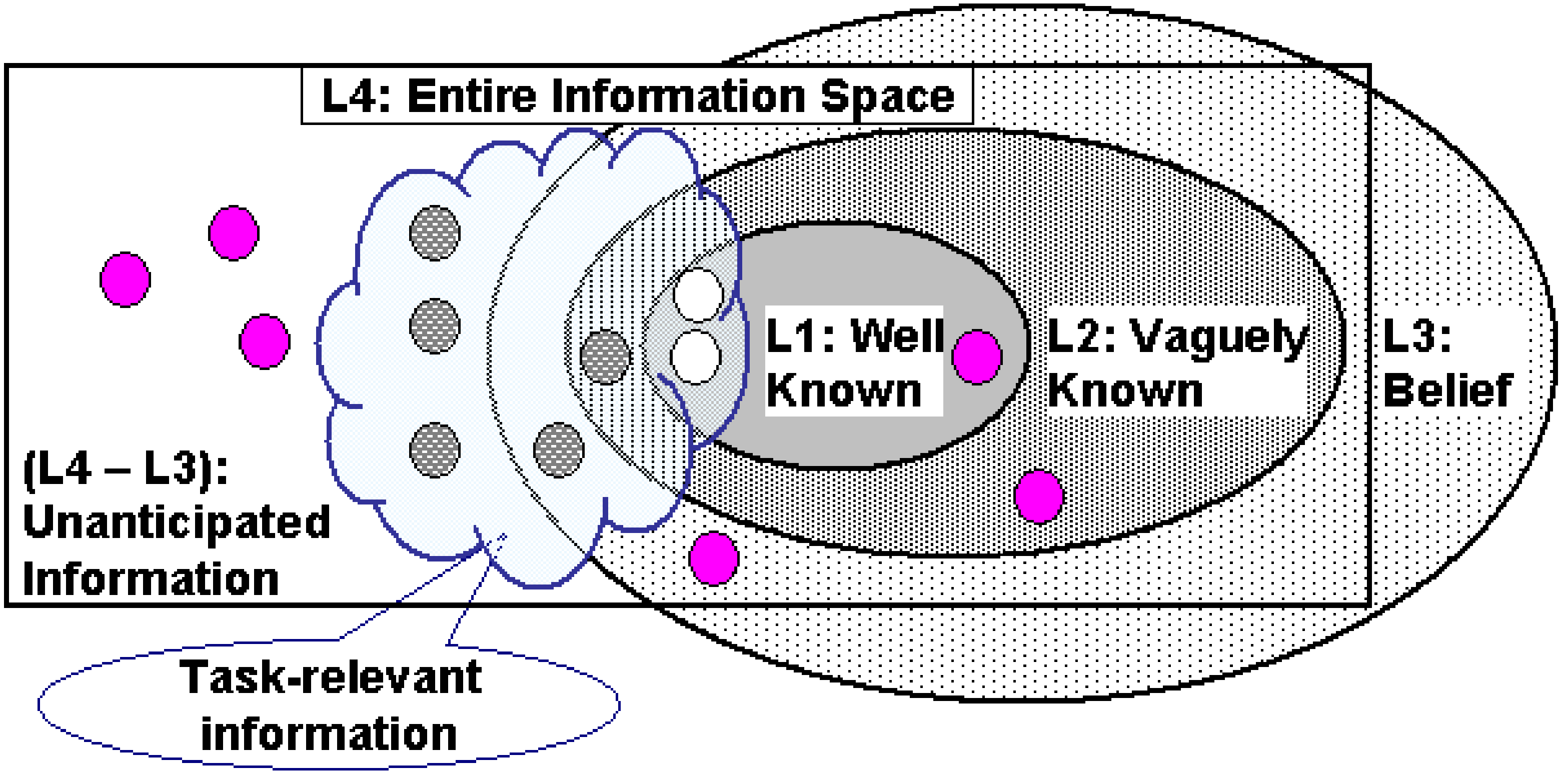
Making Software Reuse More Successful

- complex problems → high-functionality applications (software reuse repositories) — **information delivery**

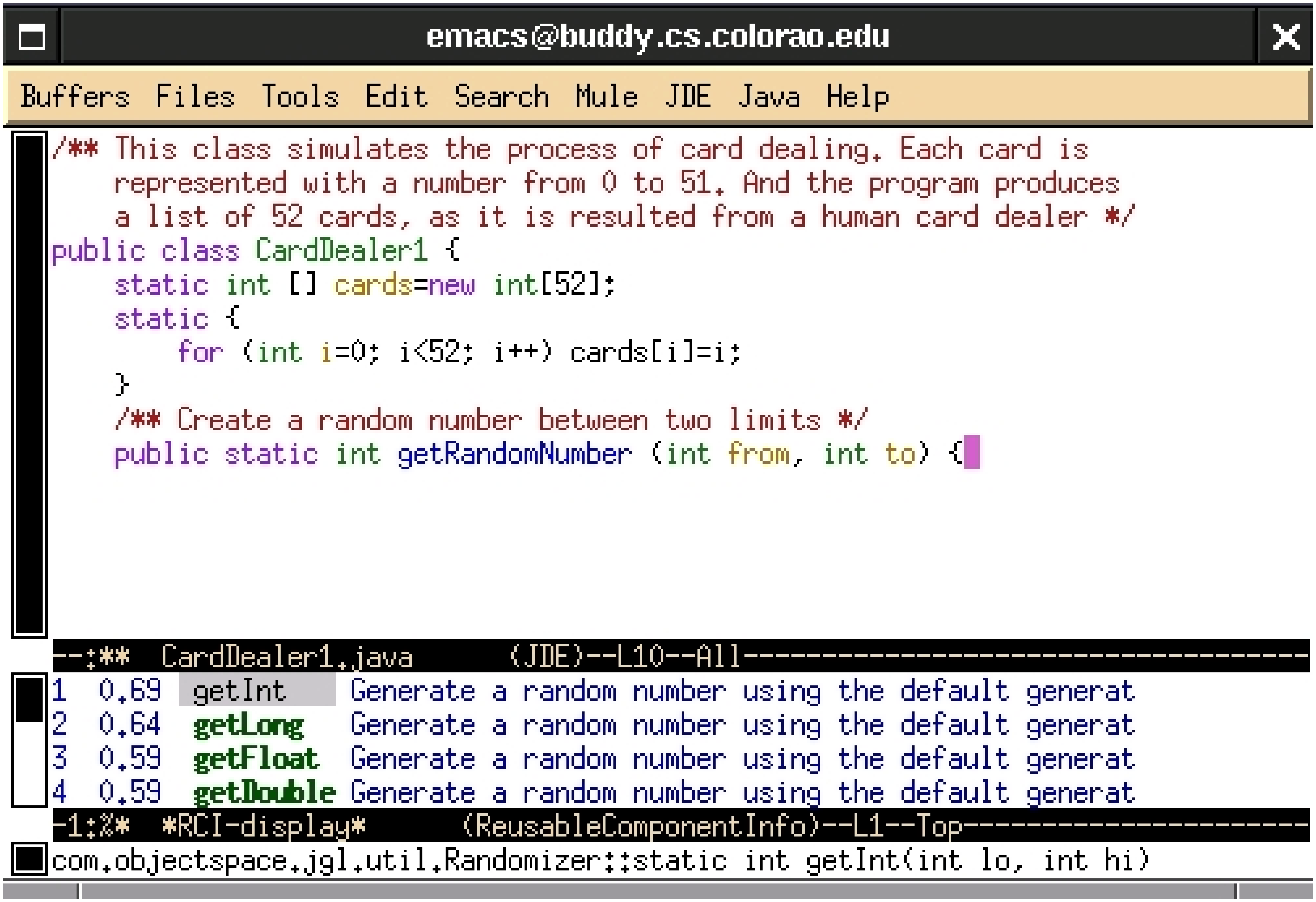
- beyond searching and browsing

- **system development: CodeBroker (Yunwen Ye)**
  - support Java programmers to take advantage of reuse libraries
  - analyze a partially written program to contextualize information to the task-at-hand
  - employ user models to personalize the delivered information

Different Levels of Knowledge about a High-Functionality Application (HFA)


L4: Entire Information Space
L1: Well Known
L2: Vaguely Known
L3: Belief
(L4 – L3): Unanticipated Information
Task-relevant information

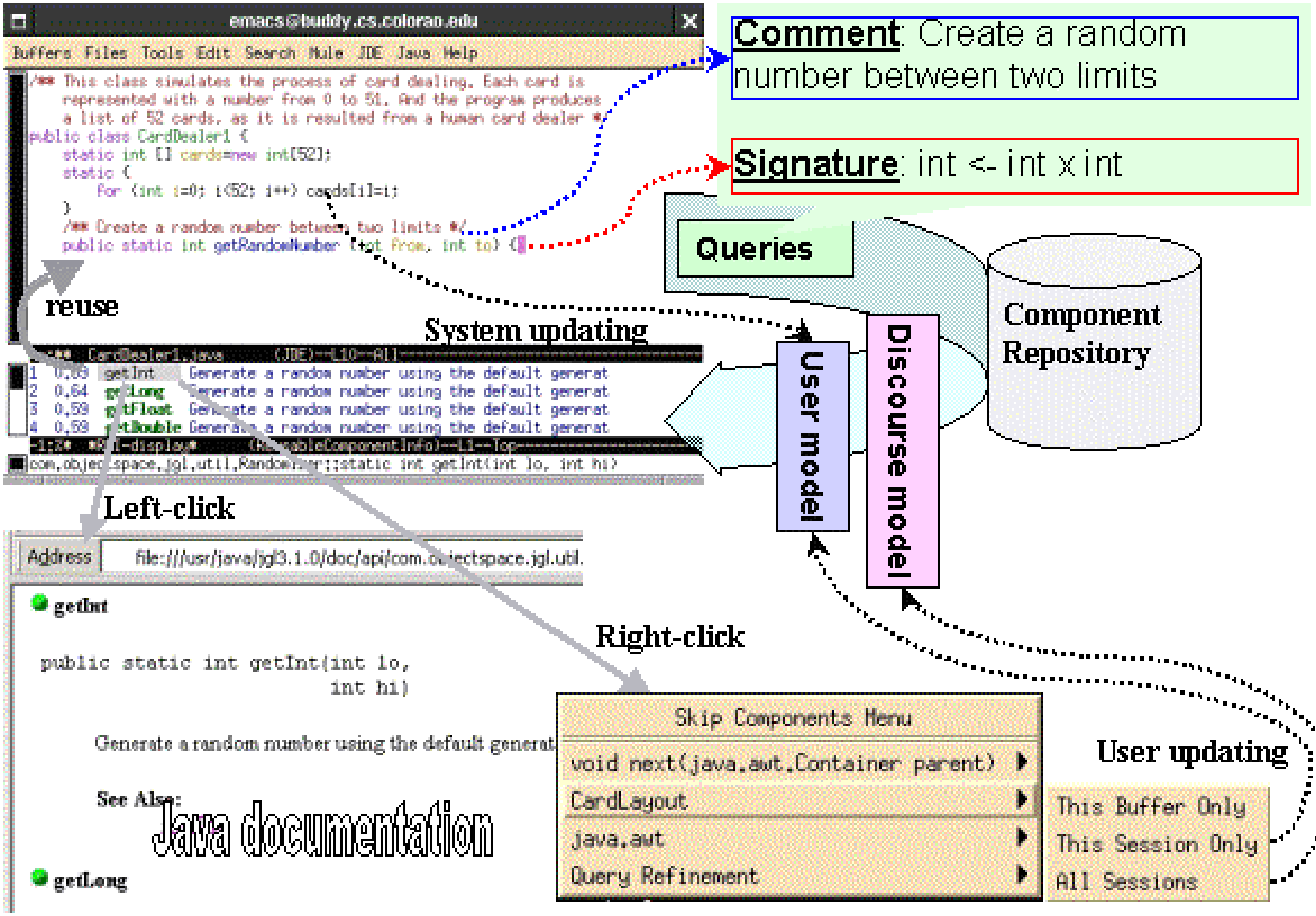
The User Interface of the CodeBroker System


emacs@buddy.cs.colorao.edu
Buffers Files Tools Edit Search Mule JDE Java Help

```
/** This class simulates the process of card dealing. Each card is
    represented with a number from 0 to 51. And the program produces
    a list of 52 cards, as it is resulted from a human card dealer */
public class CardDealer1 {
    static int [] cards=new int[52];
    static {
        for (int i=0; i<52; i++) cards[i]=i;
    }
    /** Create a random number between two limits */
    public static int getRandomNumber (int from, int to) {
```

```
--:**  CardDealer1.java      (JDE)--L10--All---------------------------------------
1   0.69  getInt    Generate a random number using the default generat
2   0.64  getLong   Generate a random number using the default generat
3   0.59  getFloat  Generate a random number using the default generat
4   0.59  getDouble Generate a random number using the default generat
-1:%*  *RCI-display*      (ReusableComponentInfo)--L1--Top---------------------
com.objectspace.jgl.util.Randomizer::static int getInt(int lo, int hi)
```

The Architecture of the CodeBroker System



```
/** This class simulates the process of card dealing. Each card is
    represented with a number from 0 to 51. And the program produces
    a list of 52 cards, as it is resulted from a human card dealer */
public class CardDealer1 {
    static int [] cards=new int[52];
    static {
        for (int i=0; i<52; i++) cards[i]=i;
    }
    /** Create a random number between two limits */
    public static int getRandomNumber (int from, int to) {
-**  CardDealer1.java      (JDE)--L10--All---------------------------
1  0.63  getInt     Generate a random number using the default generat
2  0.64  getLong    Generate a random number using the default generat
3  0.59  getFloat   Generate a random number using the default generat
4  0.59  getDouble  Generate a random number using the default generat
-1:%*  *RCI-display*    (ReusableComponentInfo)--L1--Top----------------
com.objectspace.jgl.util.Randomizer::static int getInt(int lo, int hi)
```

Address file:///usr/java/jgl3.1.0/doc/api/com.objectspace.jgl.util.

getInt

```
public static int getInt(int lo,
                         int hi)
```

Generate a random number using the default generat

See Also:

getLong

Collaborative Design: Transcending the Individual Human Mind
The “Wrong” Image? (“The Thinker” by Auguste Rodin)

A Conceptual Framework for Collaborative (Software) Design

- symmetry of ignorance → communication and coordination problems — **representation for mutual understanding and mutual learning**

- fundamental concepts for collaborative (software) design
  - symmetry of ignorance (or: asymmetry of knowledge)
  - social creativity
  - meta-design
  - boundary objects

- **system development: Envisionment and Discovery Collaboratory (Ernesto Arias, Hal Eden, Andy Gorman, Eric Scharff)**
  - **objective:** address the fundamental problem of software design that *"the process is difficult not because of the complexity of technical problems, but because of the social interaction when users and system developers learn to create, develop and express their ideas and visions"*

“Symmetry of Ignorance”

“If a lion could speak would we understand him?” — Wittgenstein

- **claim:** the heart of intelligent human performance is not the individual human mind but **groups of minds in interaction with each other and minds in interactions with tools and artifacts** (distributed cognition)

- **distinct domain of human knowledge exist**
  - C. P. Snow: “The Two Cultures”
  - of critical importance: mutual appreciation, efforts to understand each other, increase in socially shared cognition and practice, exploit the “symmetry of ignorance” for mutual learning

- create **boundary objects** (*shared* objects to “talk about” and to “think with”)

- **big challenge**: specialists have to put the case for their enterprise in language non-specialists can understand; combine creative scientific work with the communication of the new knowledge to a wider audience

Social Creativity

- **social creativity: requires designers not consumers**
  - from knowledge acquisition → knowledge construction
  - from access → informed participation

- **requires externalizations which**
  - talk back to all participants in a design process
  - can be analyzed, criticized, and incrementally improved
  - can serve as boundary objects

- **requires a willingness and social and technical support for collaboration** (in a competitive world) — e.g.: how do we effectively collect individual knowledge and make it accessible to entire organizations?

- **closed system → open and evolvable systems** (seeding, evolutionary growth, reseeding model)

Meta-Design

- **meta-design =** how to create new media which allows other humans to act as designers and be creative

- **concepts of meta-design:**
  - underdesigned systems
  - learning on demand
  - open, evolvable systems

- **impact of meta-design**

授人以鱼不如授人以渔

  - *"teach a person fishing rather than give him fish"(Chinese Proverb)*
  - can be extended to: *“if we can provide someone with the knowledge, the know-how, and the tools for making a fishing rod, we can feed the whole community”*

Boundary Objects

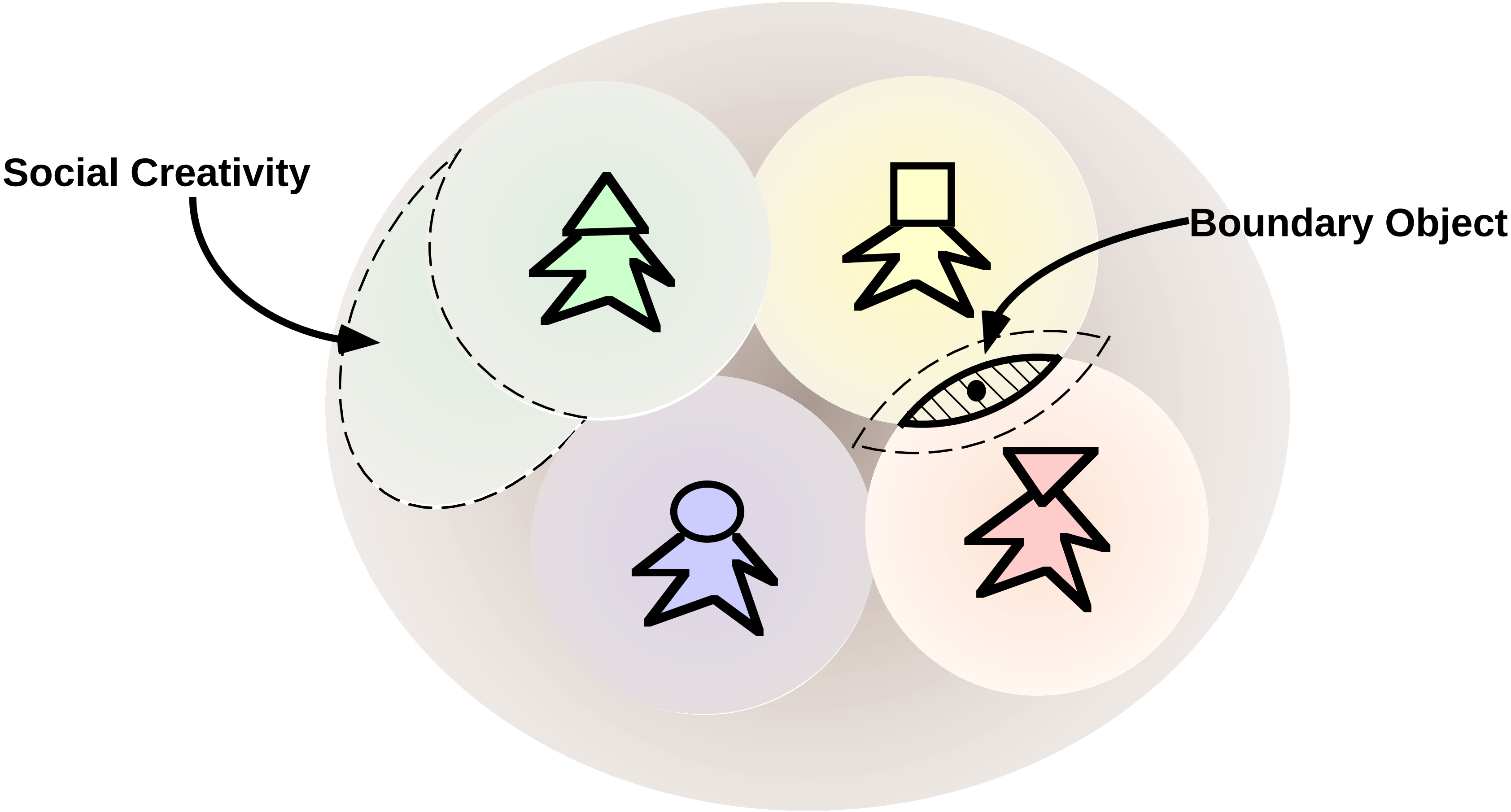
- **boundary objects serve**
  - to communicate and coordinate the perspectives of different communities (e.g. system developers and system users)
  - the interaction between users and (computational) environments

- perform a **brokering role** involving translation, coordination and alignment between the perspectives of different communities

- **examples:**
  - **externalizations**: can be critiqued, can be talked about, “talk-back”
  - **prototypes**: serve as boundary objects between developers and users in participatory system design
  - **simulations**: show the dynamic behavior

Social Creativity: Bringing Different Communities Together

“Innovations come from outside the city wall.”
Kouichi Kishida



The Envisionment and Discovery Collaboratory (EDC): Supporting Collaborative (Software) Design

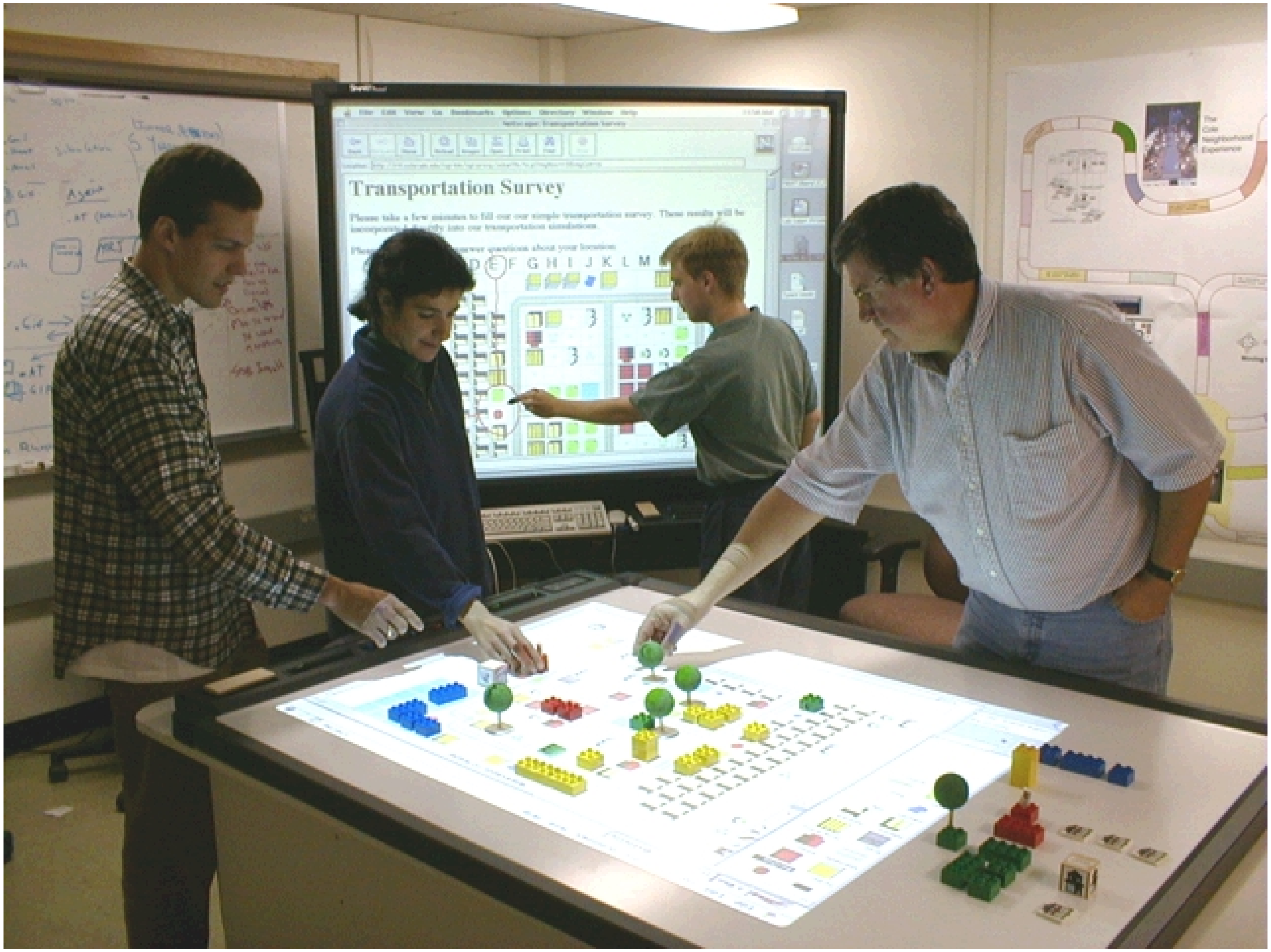
http://www.cs.colorado.edu/~l3d/systems/EDC

- **design objectives behind the EDC:**
  - designing complex software systems is an intrinsically collaborative process in which the major source of complexity arises from the need to synthesize different perspectives on the problems to be solved
  - these perspectives originate from the many stakeholders involved in system development

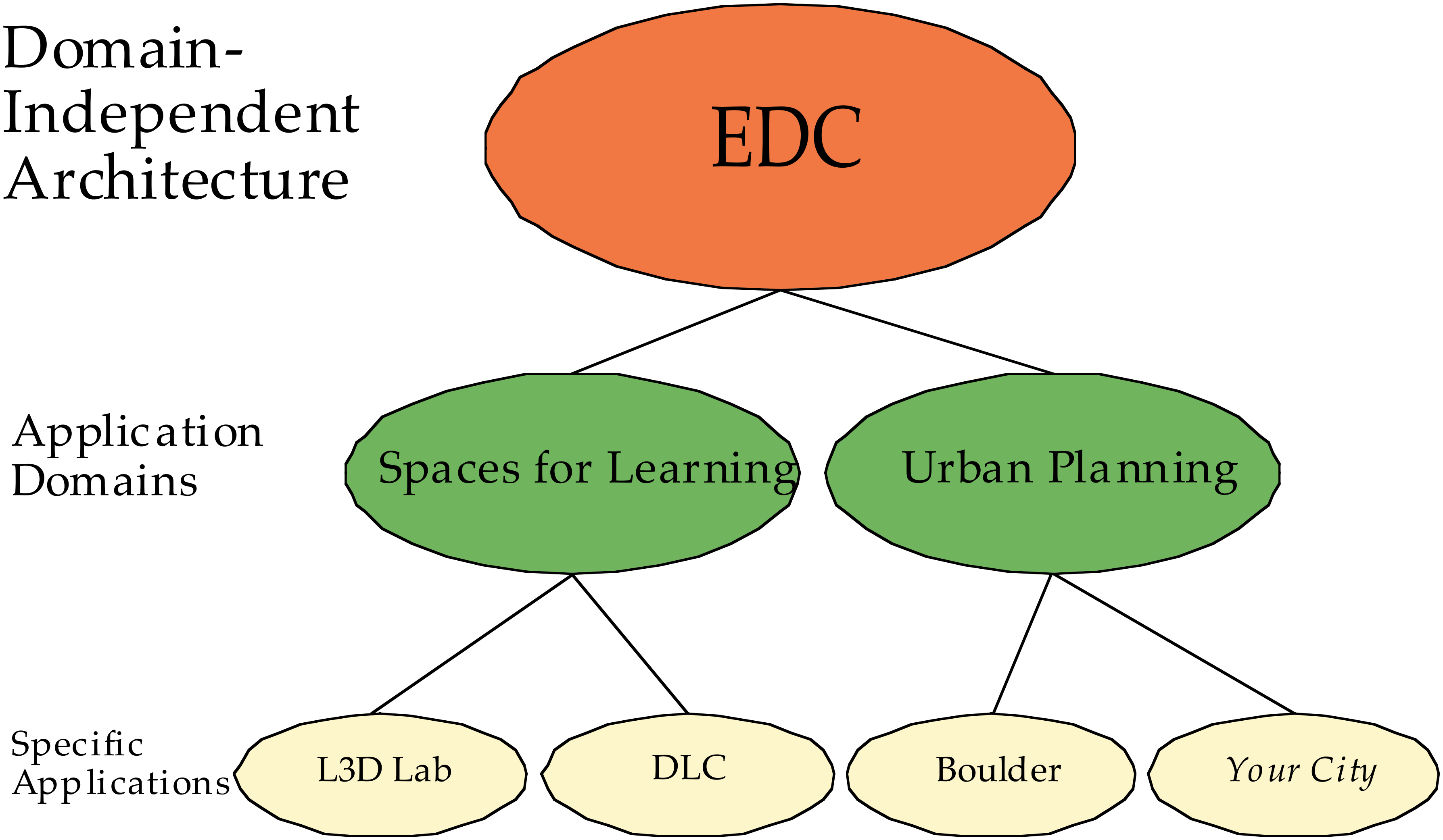
- **integration of physical and computational environments**

- **support for**
  - social creativity (exploit the symmetry of ignorance as a source of power)
  - the integration of problem framing and problem
  - open system (meta-design, SER model)

- **The Envisionment and Discovery Collaboratory (EDC)**



The Envisionment and Discovery Collaboratory



Open Source Software

- **open source software — what is it?**
  - a mindset?
  - a philosophy of life?
  - a business model?
  - see: Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida: *“Toward a Taxonomy of Open Source: A Case Study of Four Different Types of Open-Source Software Development Projects”* (submitted to ICSE’2002)

- **open source software as a new development process**
  - promotes rapid creation and deployment of incremental features and bug fixes in an existing code or knowledge base
  - leverage is gained by engaging the whole world as your talent pool

- the open source software process is unique in its participants' motivations and the resources that can be brought to problems

- learning and creating knowledge (the “Scientific Method”) is (or should be) an open source enterprise

“Open Source”: From Users to Co-Developers — Examples of Decentralized, Evolvable Information Repositories

- **Gamelan**
  - content: Java applets (an evolving community repositories of knowledge)
  - http://www.gamelan.com

- **Software Research Associates (SRA)**
  - **Kouichi Kishida:** Open Source Business Strategy
    http://www.srainc.com/osb/k2/OSS-Strategy-Public_e.html
  - **Jun** (platform for 3D graphic applications)
    http://www.sra.co.jp/people/aoki/Jun/Main_e.htm

- **Netscape Communicator**
  - distributed development and centralized integration
  - http://www.mozilla.org

- **Open Source (Cathedral and Bazaar) — http://www.tuxedo.org/~esr/**
  - Raymond, E. S. & Young, B. (2001): *“The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary”*, O'Reilly & Associates, Sebastopol, CA.

Conclusions

the fundamental challenge for software technologies of the future is to provide support for achieving a shared understanding among different groups of people that see the world in fundamentally different ways

—

the conceptual frameworks and systems presented in my talk and my paper make important contributions to this goal