



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**

Meta-Design: A Design Theory/Framework for End-User Development (EUD) and End-User Software Engineering (EUSE)

Gerhard Fischer

Center for LifeLong Learning & Design (L³D)

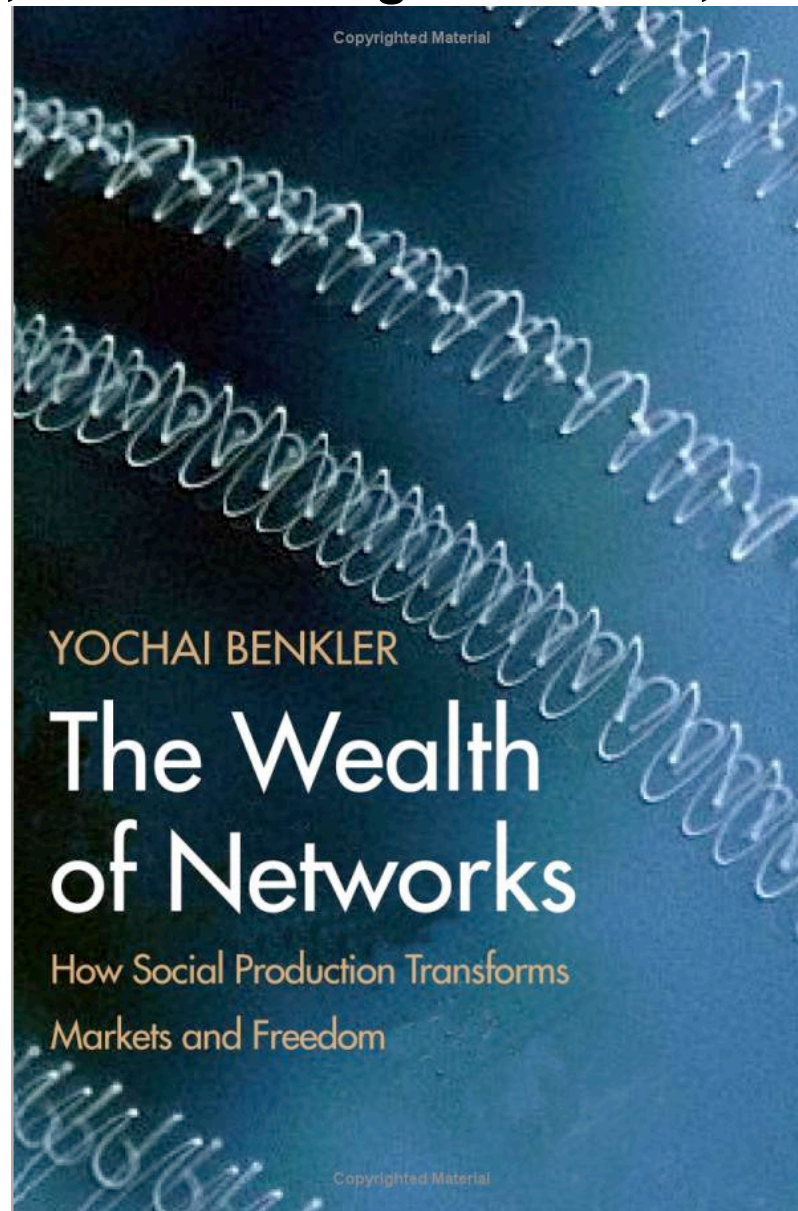
Department of Computer Science and Institute of Cognitive Science

University of Colorado, Boulder

<http://l3d.cs.colorado.edu/>

Presentation, L3D Meeting, February 7, 2007

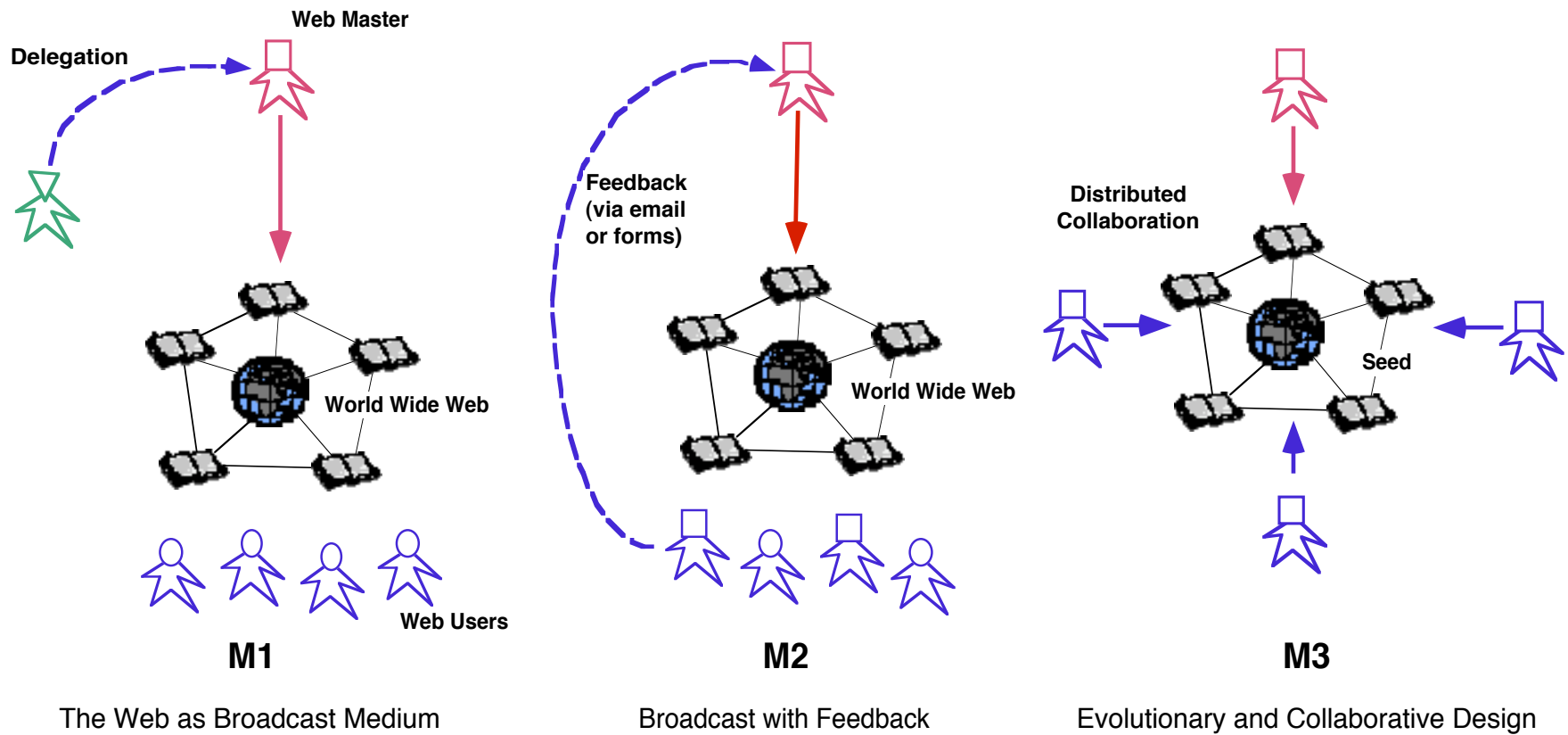
Social Production, Democratizing Innovation, and Social Creativity



The Networked Information Economy

- **claims:**
 - a series of changes in the technologies, economic organization, and social practices of production has created new opportunities for how we make and exchange information, knowledge, and culture
 - increasing role for non-market production in the information and cultural production sector, organized in a radically more decentralized pattern
- in the networked information economy, the physical capital required for production is broadly distributed throughout society
 - there are (a) no noncommercial automobile manufacturers and (b) no volunteer steel foundries
 - but: there exists widespread cooperative networks of volunteers that write the software and standards that run most of the Internet and enable what we do with it

WWW: From Broadcast to Collaboration Medium



Example: Web 2.0

- **source:** Tim O'Reilly *“What is Web 2.0 – Design Patterns and Business Models for the Next Generation of Software”*

Web 1.0

Britannica Online

→

personal website

→

publishing

→

content management systems

→

scheduled software releases

→

individual contributions

→

Web 2.0

Wikipedia

blogging

participation

wikis

continuous improvements

collective intelligence

claim: network effects from user contributions (= knowledge sharing) are the key to market dominance in the Web 2.0 era

Objective of End-User Software Engineering (EUSE)

<http://eusesconsortium.org/>

- The number of end users creating software is far larger than the number of professional programmers.
 - These end users are using various languages and programming systems to create software in forms such as spreadsheets, dynamic web applications, and scientific simulations.
 - This software needs to be sufficiently dependable, but substantial evidence suggests that it is not.

- Solving these problems involves not just software engineering issues, but also several challenges related to the users that the end user software engineering intends to benefit.
 - End users have very different training and background, and face different motivations and work constraints, than professional programmers.
 - They are not likely to know about such things as quality control mechanisms, formal development processes, system models, language design characteristics, or test adequacy criteria, and are not likely to invest time learning about them.

Science of Design and EUD / EUSE

- **software engineering**
 - a **human** activity → who are the humans engaged in software engineering?
 - a **design** activity
 - a **collaborative** design activity

- **our objective within a “science of design” → embed EUSE and EUD in a larger socio-technical context**
 - make all voices heard
 - increase quality / reliability of end-user contributions
 - beyond productivity → creativity and innovation
 - reflective practitioner → reflective community (including: software professionals, domain experts, power users)
 - invent new divisions of labor

Design

- Simon — the overriding **commonality of design**:
“change an existing state into a preferred one”
- Karl Marx: “Die Philosophen haben die Welt nur verschieden interpretiert; es kommt aber darauf an, sie zu verändern!”

in English: “Philosophers have been content to interpret the world in different ways — the challenge is to change it”
- a **rationale for meta-design** provided by Simon’s expectation and recommendation: *“strive for a design beyond that of leaving more responsibilities open to future generations than we ourselves inherited”*

Different Generations of Design Methods

(from the History of Architectural Design)

▪ **1st Generation (before 1970):**

- directionality and causality
- separation of analysis from synthesis
- major drawback: (a) perceived by the designers as being unnatural, and (b) does not correspond to actual design practice
- **example:** waterfall-type models

▪ **2nd Generation in the early 70'es:**

- participation — expertise in design is distributed among all participants
- argumentation — various positions on each issue
- major drawback: insisting on total participation neglects expertise possessed by a well-informed and skilled designer
- **example:** participatory design

▪ **3rd Generation (in the late 70'es):**

- inspired by Popper: the role of the designer is to make expert design conjectures
- these conjectures must be open to refutation and rejection by the people for whom they are made
- **example:** distributed intelligence, meta-design, post-structuralist approach

Putting Owners of Problems in Charge

a Necessity: not a Luxury

- **an interview, a geoscientist:**

“I spend in average an hour every day developing software for myself to analyze the data I collected because there is not any available software.

Even if there is a software developer sitting next to me, it would not be of much help because my needs vary as my research progresses and I cannot clearly explain what I want to do at any moment.

Even if the software developer can manage to write a program for me, I will not know if he or she has done it right without looking at the code.

So I spent three months to gain enough programming knowledge to get by. Software development has now become an essential task of my research, but I do not consider myself a software developer and I don't know many other things about software development.”

- **observations:**

- this geoscientist obviously is not just an end-user; his software has thousands of lines and he has considerable programming skills
- it is equally obvious that he is not a software professional and does not intend to become one

Unselfconscious Cultures of Design

(Christopher Alexander, Architect)

- breakdown and correction occur side by side (→ design-in-use)
- there is no formal set of rules describing how to repair breakdowns, since the breakdowns were not anticipated
- the knowledge to repair breakdowns comes from the knowledge of the user, who is best able to recognize a lack of fit, and how the artifact should be changed to improve its fit to the environment
- can cope with ill-defined problems
 - require the integration of problem framing and problem solving
 - ill-defined problems cannot be delegated

Most Important / Promising Design Activities for EUP /EUSE

- assumption: the goal of making systems modifiable / extensible by end-users does not imply transferring the responsibility of good system design to the user. Normal users will in general not build tools of the “quality / reliability” a professional designer would.
- coping with ill-defined problems
- exploring creative ideas
- everyday adaptive design (T. Moran) → the problems do not have to be anticipated and the users represented: they are at hand

Meta-Design — How We Think About It

- “if you give a fish to a human, you will feed him for a day — if you give someone a fishing rod, you will feed him for life” (Chinese Proverb)

- **meta-design** extends this to:

“if we can provide the knowledge, the know-how, and the tools for making fishing rods, we can feed the whole community”

Meta-Design

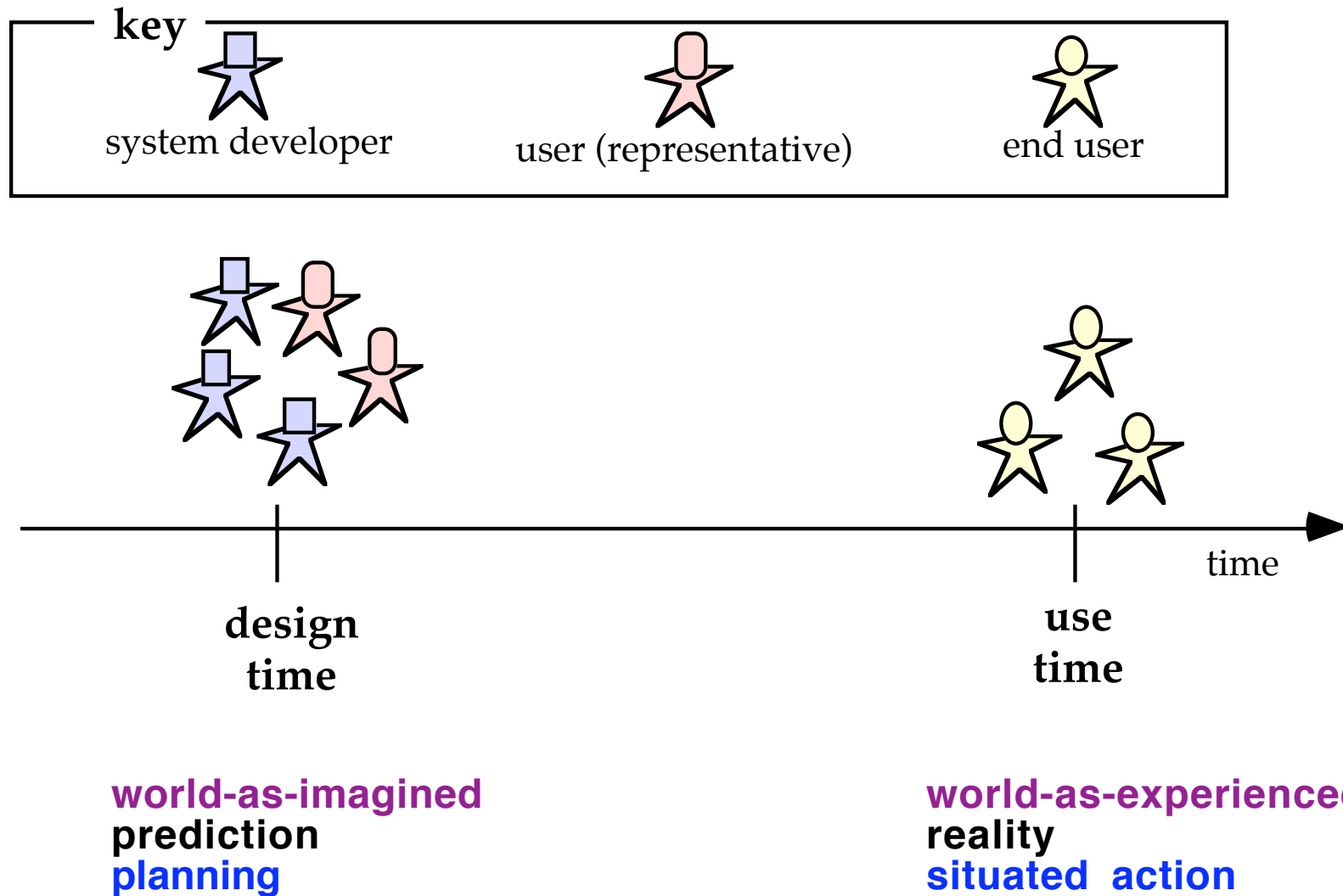
- **meta-design**

- new media that allow users to act as designers and be creative
- the creation of context rather than content
- puts the tools rather than the object of design in your hands
- does not define a product, but the conditions for a process of interaction
- a problem solving process that remains liquid and open (“final” is used only in a tentative sense)

- **why meta-design?**

- to avoid that most of the relevant knowledge gets forced to the earliest part of the design process, when everyone knows the least about what is really needed
- design for diversity (for “a universe of one” → CLever Project)
- design as a process is tightly coupled to use and continues during the use of the system
- addresses and overcome problems of closed systems
- prerequisite for social creativity and innovation
- transcends a “consumer mindset”

Design Time and Use Time



Computational Media

—

Extending Design Opportunities at Use Time

- **print media:** a fixed context for use time is decided at design time

- **computational media:**
 - presentations at use time can take advantage of contextual factors only known at use time (about tasks, users, social systems,.....)
 - examples: specification sheets and usage data, supporting dynamic forms, dynamic websites, user and task specific maps and traffic schedules....

- **evolving existing systems:** users (acting as designers) can transcend at use time the boundaries of the systems as developed at design time

Meta-Design: Extending Other Design Approaches

- **professionally-dominated design**
 - works best for people with the same interests and background knowledge
- **user-centered design:**
 - analyze the needs of the users
 - understand the conceptual worlds of the users
- **learner-centered design**
 - draws attention to the changing needs of users
 - combine HCI interaction principles with educational interaction support
- **participatory design**
 - involve users more deeply in the process as co-designers by empowering them to propose and generate design alternatives
 - focus on system development at design time by bringing developers and users together to envision the contexts of use
- **meta-design:**
 - create design opportunities at use time
 - improvisation, local knowledge, tacit knowledge getting activated in actual use situations is supported

What Do Meta-Designers Do?

- use their own creativity to create socio-technical environments in which other people can be creative
- create the **technical** and **social** conditions for broad participation in design activities which are as important as creating the artifact itself

End-User Development

- **end-users:**
 - are the owners of problems, have the domain knowledge
 - regard computers as useful machines capable of helping them work more productively, creatively, and with greater pleasure
 - like computers because they get their work done
 - are competent practitioners usually know more than they can say — tacit knowledge is triggered by situations, by breakdowns

- **computer scientist / programmers**
 - find computers intrinsically interesting
 - like computers because they get to program

- **ultimate goal / belief:** end-users will use, tailor, extend and create their own computational artifacts if they have a supportive socio-technical environment

- **communities of users** will develop: power users, local developers, gardeners

Design: Beyond Binary Choices

- **Turing Tar Pit:** *“Beware of the Turing Tar Pit, in which everything is possible, but nothing of interest is easy.”*
 - why are current interactive programming environments, such as Logo, Smalltalk, Squeak, Agentsheets, not sufficient for supporting meta-design?
 - claim: level of representation is still too far removed from the conceptual world of the domain workers
 - claim: they emphasize objective computability → the challenge: subjective computability
- **The Inverse of the Turing Tar Pit:** *“Beware of the over-specialized systems, where operations are easy, but little of interest is possible.”*
 - domain-specific tools (such as SimCity) provide extensive support for certain problem contexts
 - the ability to extend these environments is limited — even minor incremental changes are often impossible in these systems

Approaches to EUD

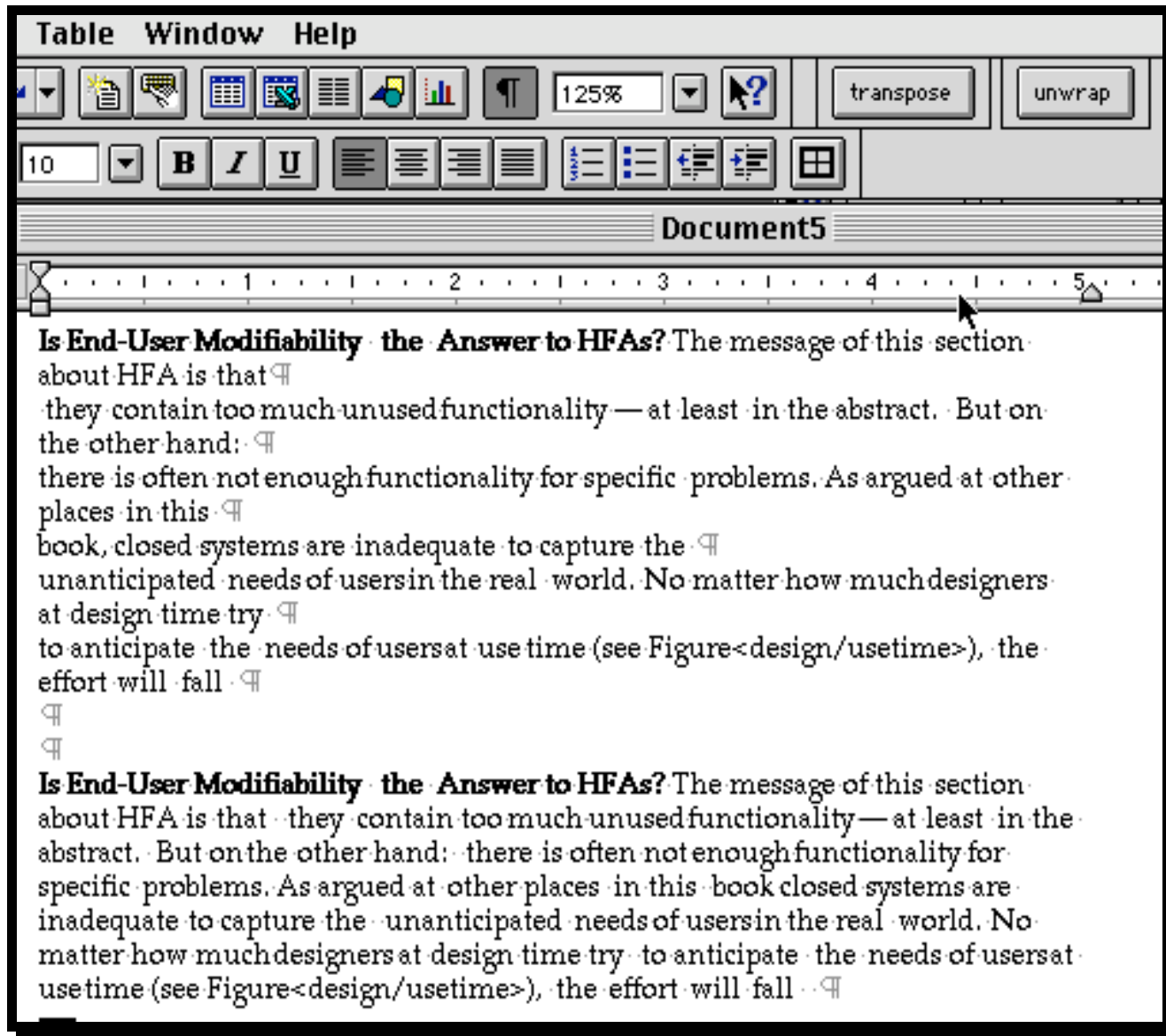
- scripting languages and macros
- DODEs and Layered Architectures
- communities of stakeholders with different expertise
- socio-technical environments giving all stakeholders a voice →
Envisionment and Discovery Collaboratory

Meta-Design Concepts (in Microsoft Word)

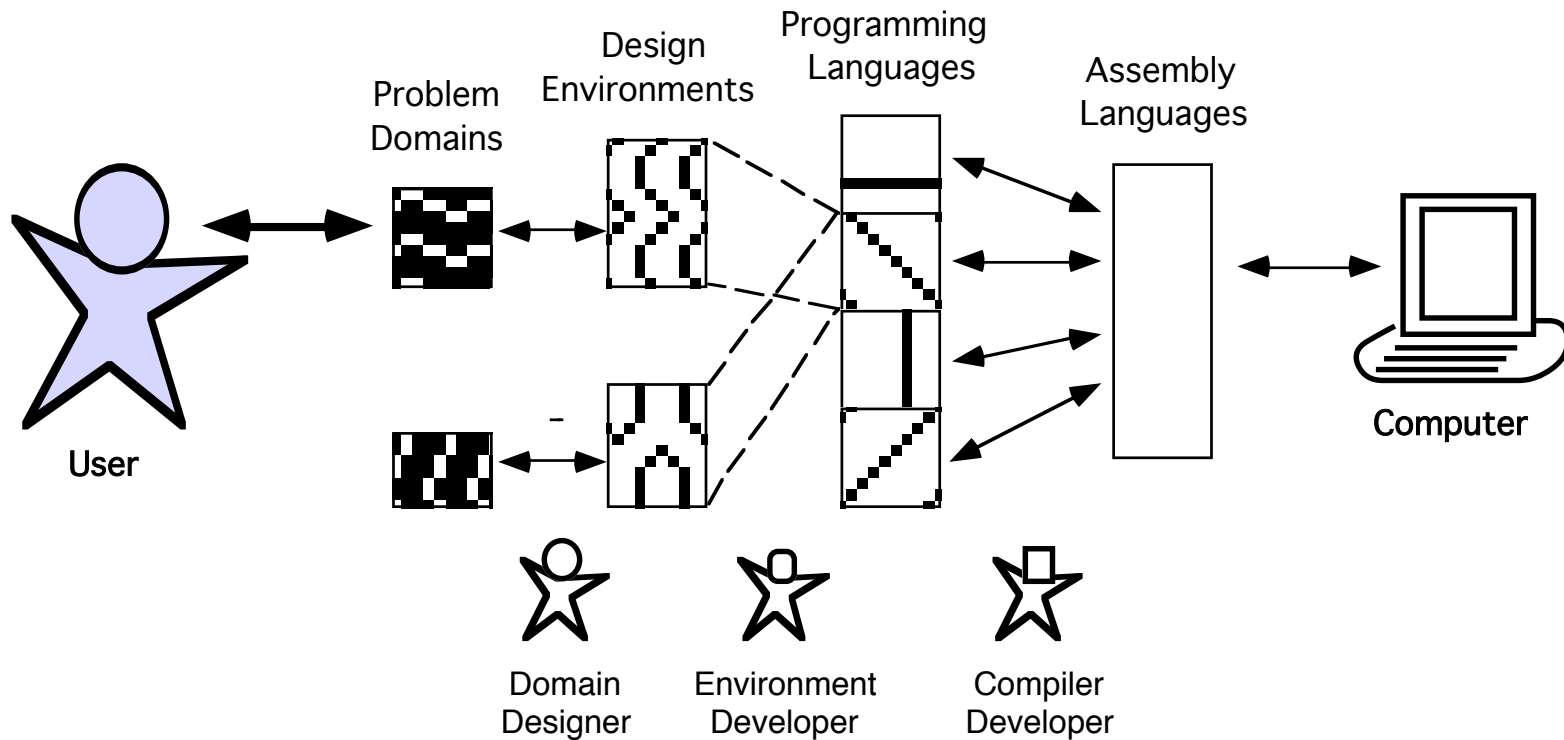
— Users as Co-Developers

- ***tailor*** and ***customize*** the system by setting different parameters as their personal preferences
- ***extend*** and ***evolve*** existing information structures (e.g., menus, spelling dictionaries, auto-correct tables, ...)
- write ***macros*** to create new operations (an example of “programming by example” or “programming by demonstration”)
- create ***programs in VisualBasic*** to extend the functionality of the system
- ***share*** the user-defined extensions



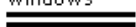
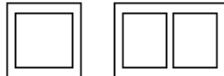

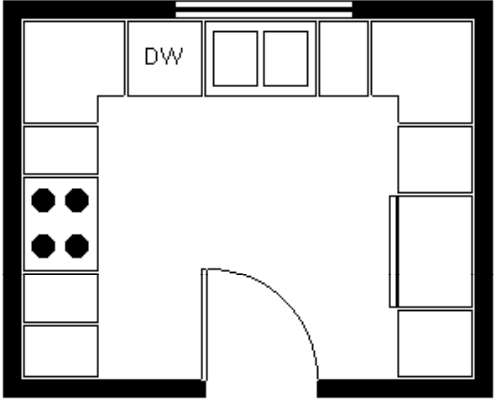
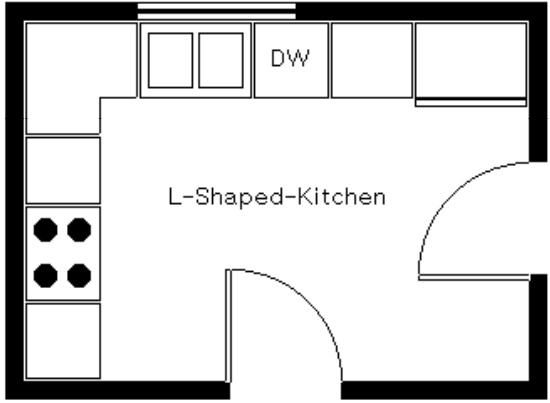

A Macro for Unwrapping Text



A Layered Architecture Creating New Divisions of Labor for Software Design



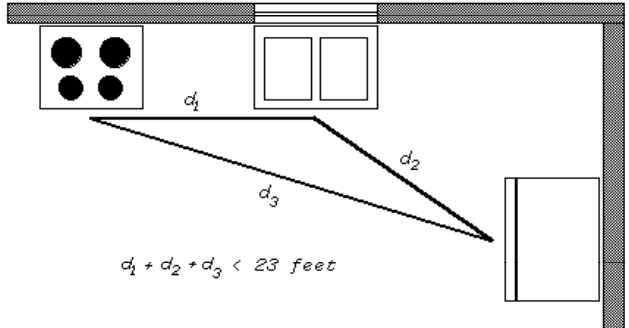
A DODE for Kitchen Design: Construction

<i>Janus-Construction</i>		Clear Work Area Load Catalog	Critique All Save In Catalog	Edit Global Descriptions Select Context
<p>Appliance Palette</p> <p>walls</p>  <p>doors</p>  <p>windows</p>  <p>sinks</p>  <p>stoves</p> 	<p>Work Area</p> 			
<p>Catalog</p> 	<p>Messages</p> <ul style="list-style-type: none"> ▪ The length of the work triangle (Double-Bowl-Sink-1, Four-Element-Stove-1, Single-Door-Refrigerator-1) is greater than 23 feet.  ▪ Single-Door-Refrigerator-1 is not near Four-Element-Stove-1. 			
<p>Commands</p> <ul style="list-style-type: none"> ▶ Critique All ▶ ■ 				

A DODE for Kitchen Design: Argumentation

Janus-Argumentation

Answer (Refrigerator, Sink, Stove)
 The distance between sink, stove and refrigerator, the *work triangle*, should be less than 23 feet.



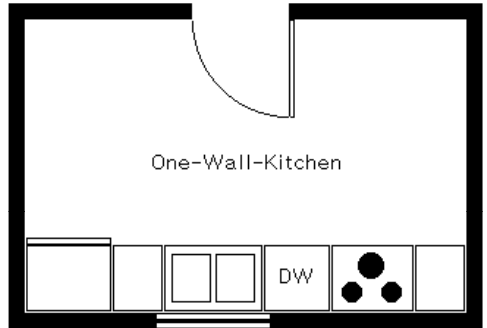
$d_1 + d_2 + d_3 < 23 \text{ feet}$

Figure 10: the work triangle

Argument (Walking Distance)
 The work triangle is an important concept in kitchen design. The work triangle denotes the center front distance between the three main appliances: *sink*, *stove* and *refrigerator*. This length should be less than 23 feet to avoid unnecessary walking and to ensure an efficient work flow in the kitchen!

Argument (Small Room)
 In small kitchens where the work triangle is less than 16 feet,

Catalog Example



One-Wall-Kitchen

The length of the work triangle (Stove, Refrigerator, Sink) is less than 23 feet.

Visited Nodes

- ➔ Answer (Refrigerator, Sink, Stove) Section

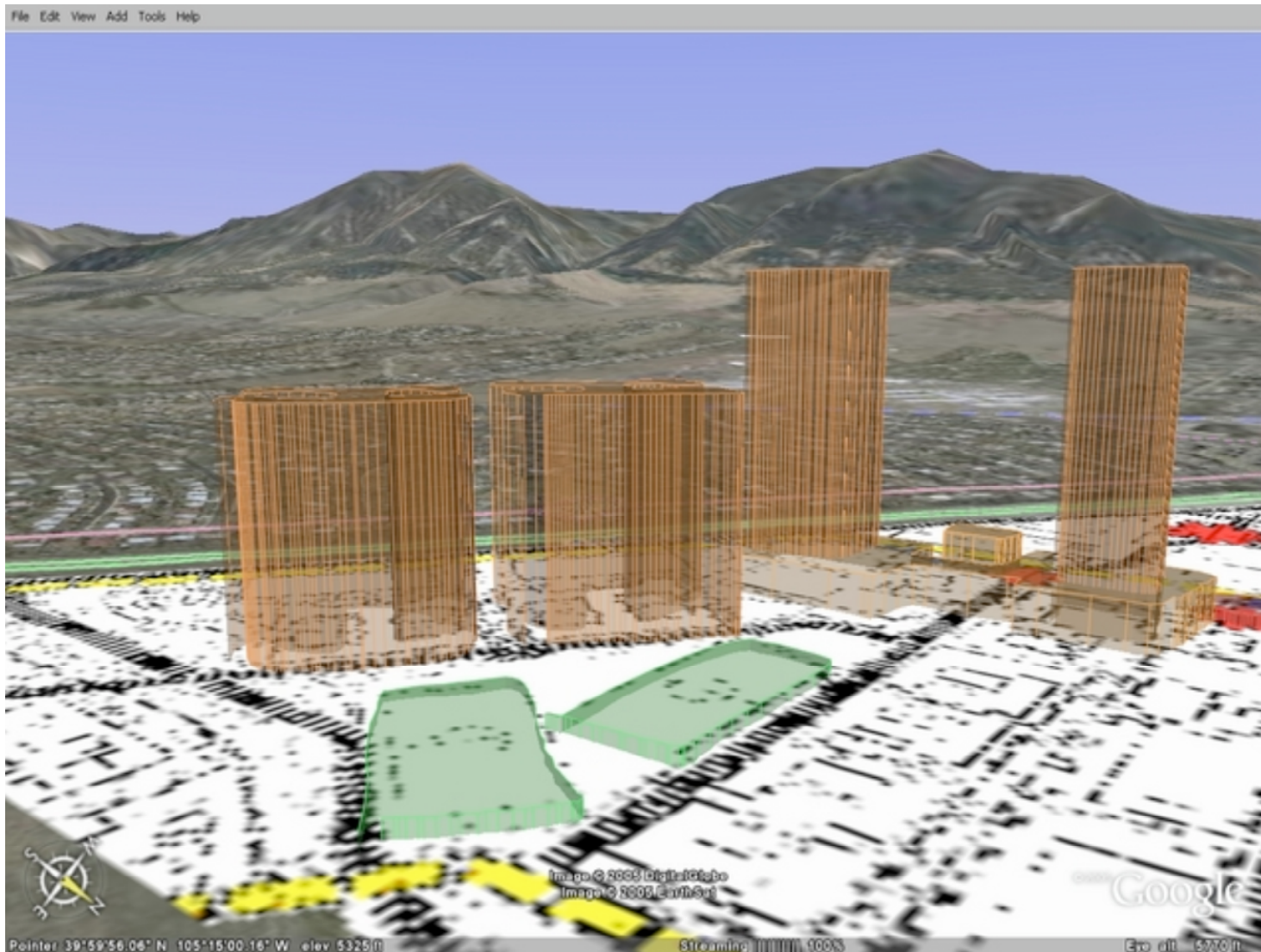
Viewer: Default Viewer

Commands

- ▶ Show Example: "Answer (Refrigerator, Sink, Stove)"
- ▶ Show Example Answer (Refrigerator, Sink, Stove)

- Show Outline
- Resume Construction
- Search For Topics
- Show Construction
- Show Argumentation
- Show Example
- Show Context
- Show Counter Example

Buildings Sketched into a Google-Earth Client



Meta-Design Aspects in the Envisionment and Discovery Collaboratory: Closed versus Open Systems

- **example for a closed system: SimCity** — too much crime
 - solution supported: build more police stations (**fight crime**)
 - solution not supported: increase social services, improve education (**prevent crime**)
- **important goal of EDC:** create end-user modifiable versions of SimCity, because:
 - background knowledge can never be completely articulated
 - the world changes
- **user control:**
 - end-user modifiability
 - conviviality (independence of high-tech scribes)
 - ownership (putting owners of problems in charge)

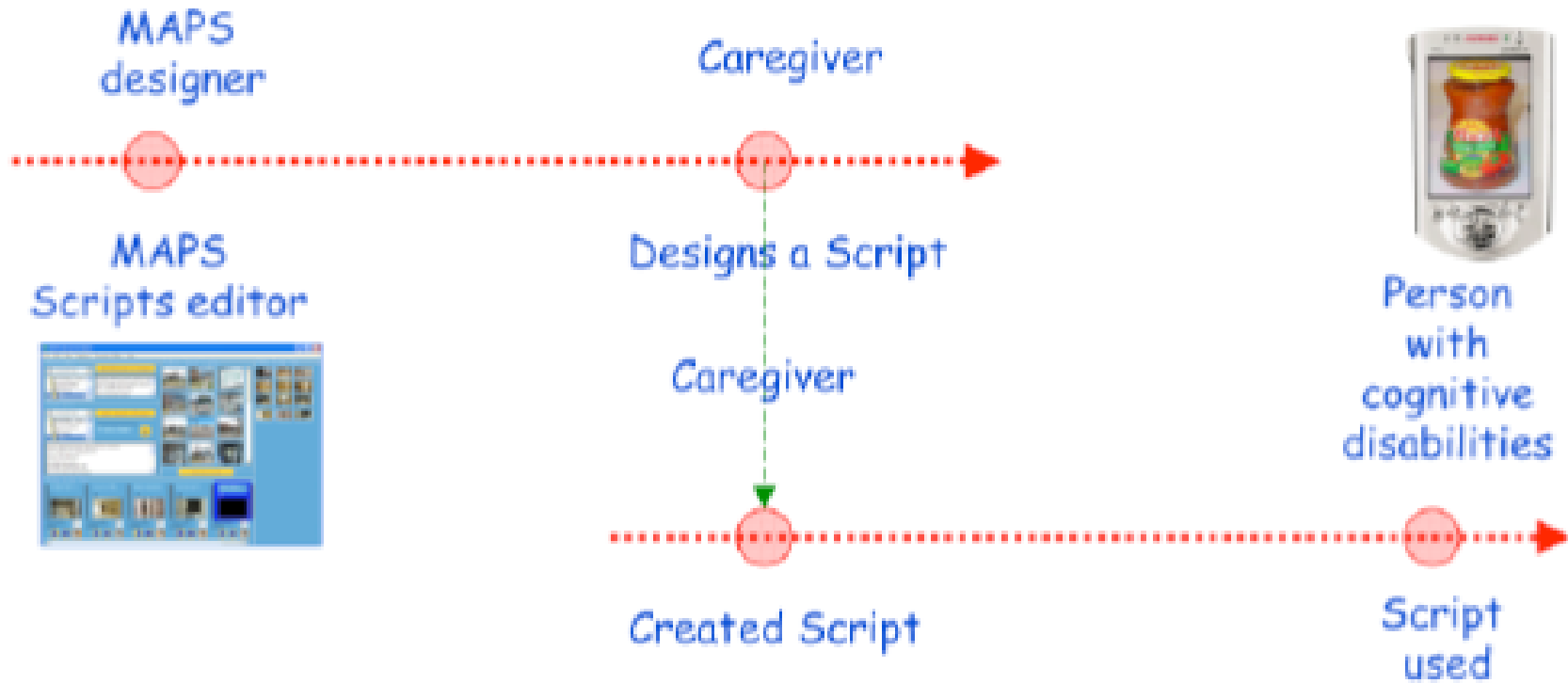
Meta-Design in the CLever Project

CLever: Cognitive Levers — Helping People Help Themselves

- **why is there a need:**
 - clients are a “universe of one” → move beyond “one size fits all” solutions based on “the average user myth”
 - clients limited abstraction capabilities
 - use ubiquitous, location-ware, mobile technologies to deliver personalized information tailored to individual needs and abilities

- **who are the end-users**
 - the caregivers

End-User Development in the MAPS Environment



MAPS Script Editor



MAPS Handheld Prompter

This is your
Bus,
Get ready to
get on



Consumer and Designers – Beyond Binary Choices

■ **claims:**

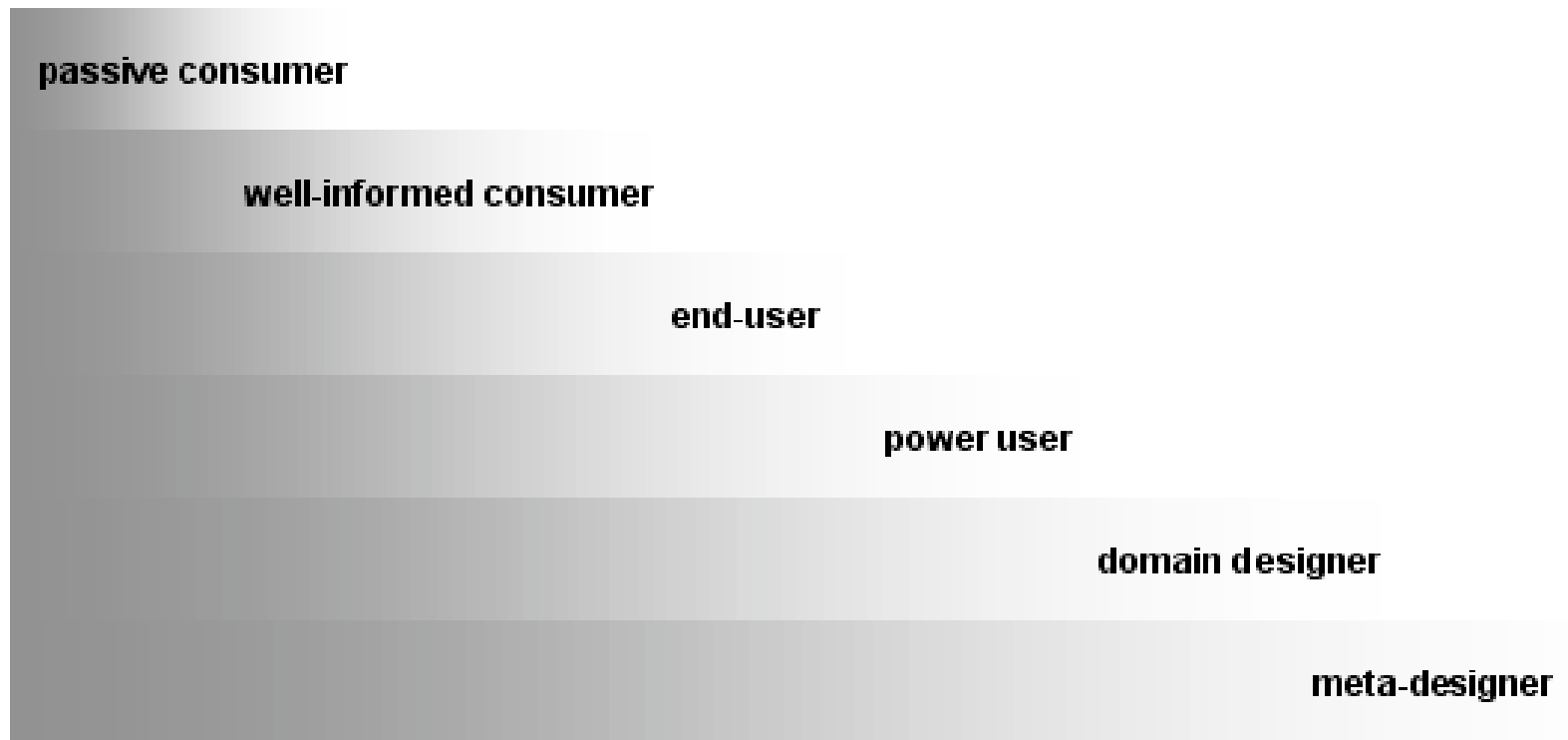
- there is nothing wrong about being a consumer (watching a tennis match, listening to a concert, ...)
- the same person wants to be a consumer in some situations and in others a designer
- consumer / designer is not an attribute of a person, but of a context
consumer / designer ≠ f{person} → f{context}

■ **problems:**

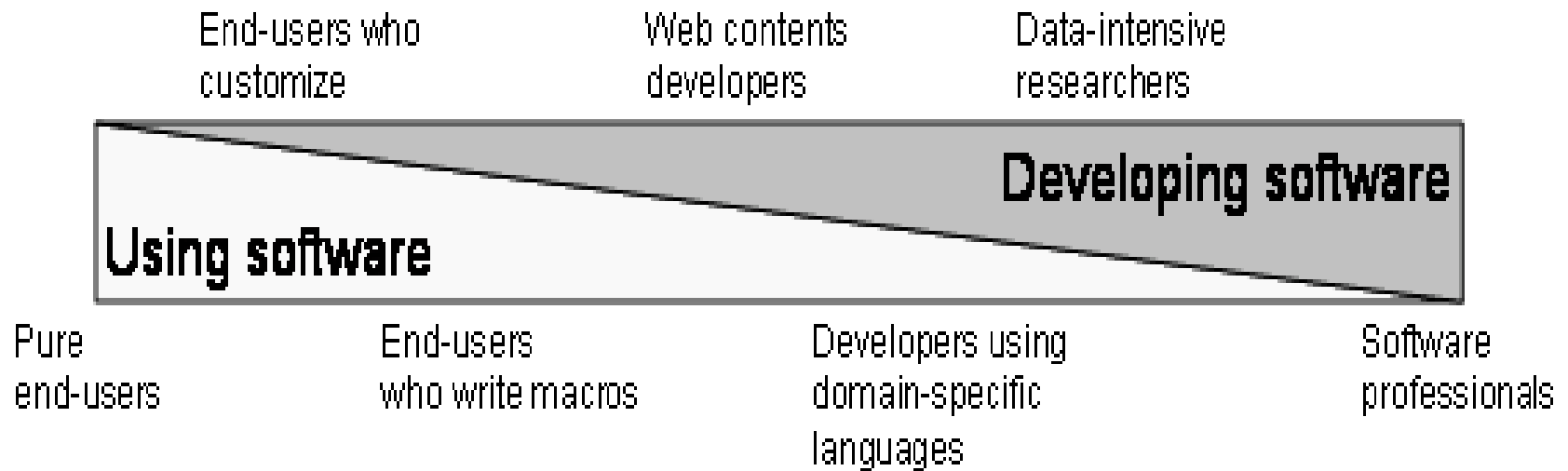
- someone wants to be a designer but is forced to be a consumer → ***personally meaningful activities***
- someone wants to be a consumer but is forced to be a designer → ***personally irrelevant activities***

Consumer and Designers – A Continuum

CONSUMER ←-----▶ DESIGNER



Consumer and Designers — A Continuum



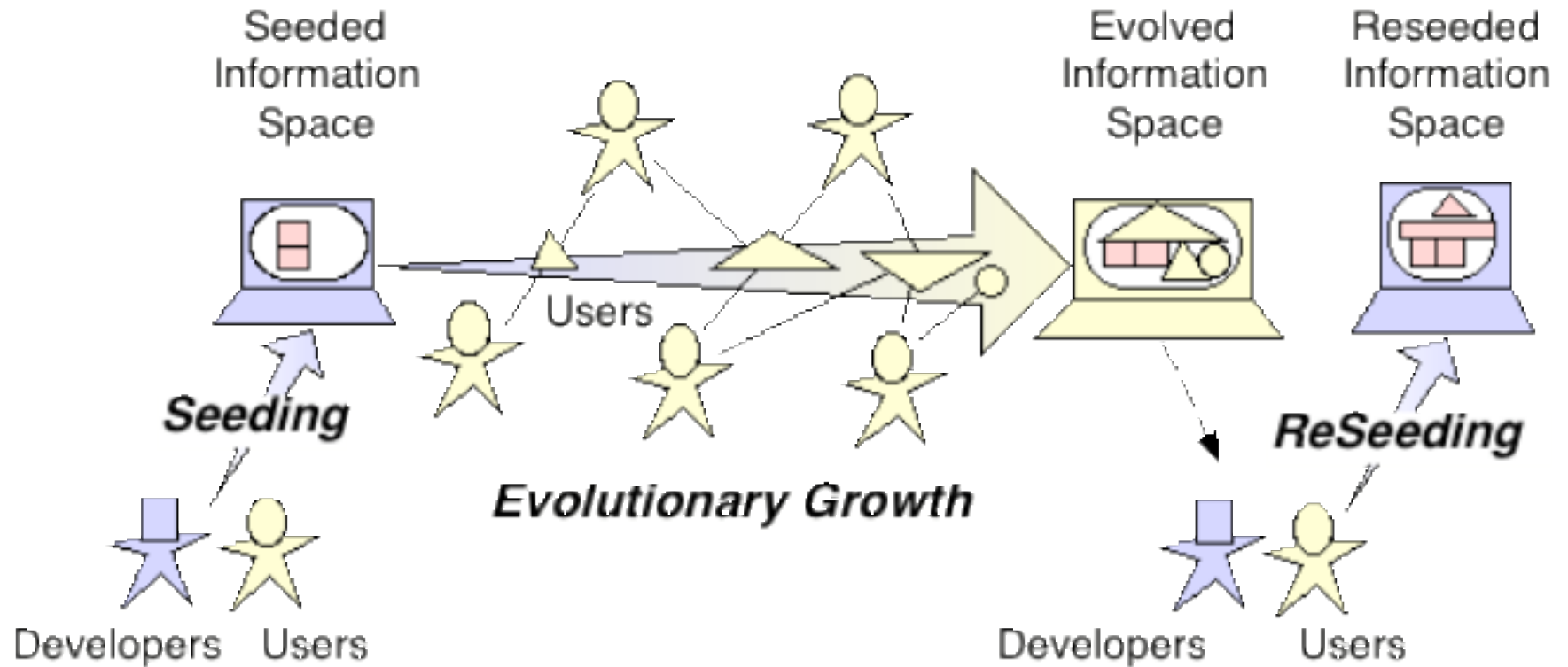
The **S**eeding, **E**volutionary Growth, **R**eseeding (**SER**) Model Supporting Meta-Design

- **at design time:**
 - development of an initial system that can change over time (seed)
 - underdesign: creating design options for users

- **at use time:**
 - support for “unself-conscious culture of design”: users will experience breakdowns by recognizing “bad fit” at use time
 - end-user modifications allow users to address limitations they experience
 - evolutionary growth through incremental modifications

- **reseeding:**
 - significant reconceptualization of the system
 - account for incremental modifications, mitigate conflicts between changes, and establish an enhanced system

The Seeding, Evolutionary Growth, Reseeding (SER) Model



Explore Technical Issues in Real-World Settings

—

Improvisations versus Standardization

- **example:** SAP Info, July 2003, p 33: “Reduce the Number of Customer Modifications”
- **rationale:**
“every customer modification implies costs because it has to be maintained by the customer. Each time a support package is imported there is a risk that the customer modification may have to be adjusted or re-implemented. To reduce the costs of such on-going maintenance of customer-specific changes, one of the key targets during an upgrade should be to return to the SAP standard wherever this is possible”
- **compare:**
 - “forking” in Open Source
 - “reseeding” in Seeding, Evolutionary Growth, Reseeding Model

Motivational Aspects and Meta-Design

- **what will make humans want to become designers/active contributors over time?**
 - serious working and learning does not have to be unpleasant but can be personally meaningful, empowering, engaging, and fun
 - comment by an artist: *“programming is not hard, but it is boring”*
- **what will make humans want to share?** → requires: mindset change, culture change, community knowledge bases, gift cultures, social capital
 - more details: Fischer, G., Scharff, E., & Ye, Y. (2004) "Fostering Social Creativity by Increasing Social Capital." In M. Huysman, & V. Wulf (Eds.), *Social Capital and Information Technology*, MIT Press, Cambridge, MA, pp. 355-399.
- **who is the beneficiary and who has to do the work?** → organizational rewards

Utility = Value / Effort

- **increase in value: motivation and rewards for a “design culture”**
 - feeling in control (i.e., independent from “high-tech scribes”)
 - being able to solve or contribute to the solution of a problem
 - mastering a tool in greater depth
 - making an ego-satisfying contribution to a group
 - enjoying the feeling of good citizenship to a community (“social capital”)
- **decrease in effort:**
 - meta-design is hard
 - extending meta-design to design for design communities

Meta-Design: Transforming Application Areas

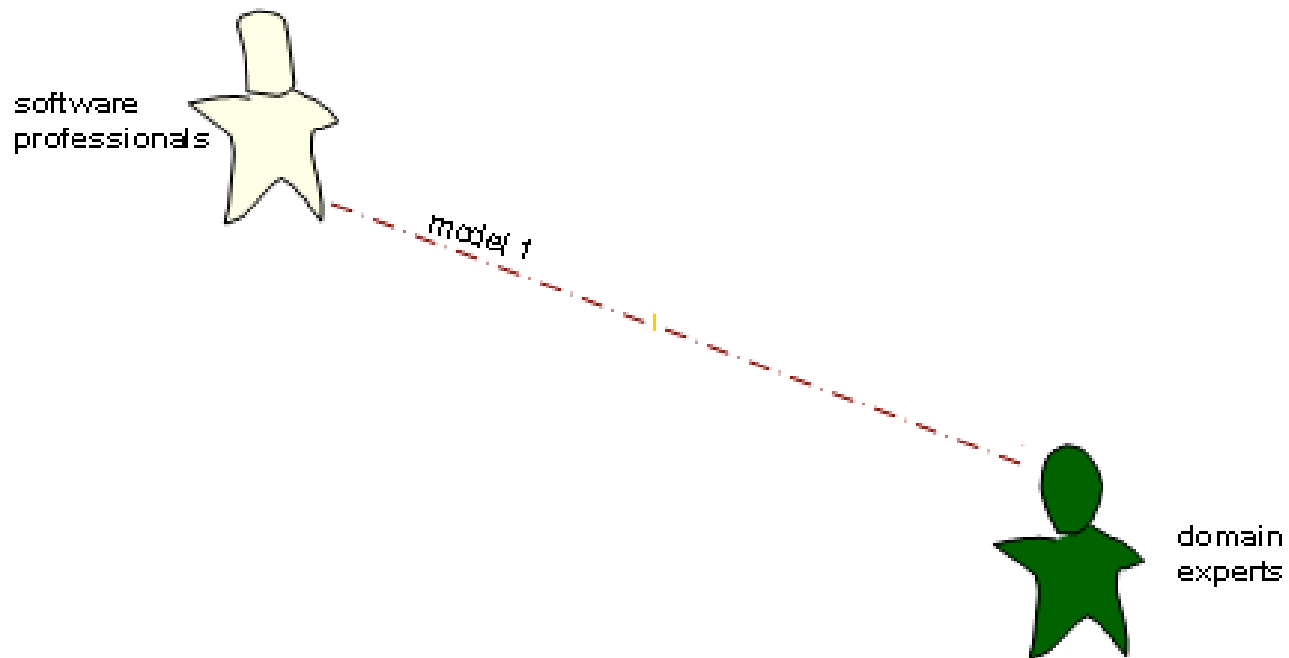
- **design:** customization, personalization, tailorability, end-user development, design for diversity — Lieberman, H., Paterno, F., & Wulf, V. (Eds.) (2006) *End User Development - Empowering people to flexibly employ advanced information and communication technology*, Kluwer Publishers, Dordrecht, The Netherlands.
- **architectural design:** underdesign, support for “unself-conscious culture of design” — Brand, S. (1995) *How Buildings Learn: What Happens After They're Built*, Penguin Books, New York.
- **teaching and learning:** teachers as facilitator, learning communities, courses-as-seeds — dePaula, R., Fischer, G., & Ostwald, J. (2001) "Courses as Seeds: Expectations and Realities," *Proceedings of the Second European Conference on Computer-Supported Collaborative Learning (Euro-CSCL' 2001)*, Maastricht, Netherlands, pp. 494-501.
- **informed participation:** beyond access, social creativity — Arias, E. G., Eden, H., Fischer, G., Gorman, A., & Scharff, E. (1999) "Beyond Access: Informed Participation and Empowerment," *Proceedings of the Computer Supported Collaborative Learning (CSCL '99) Conference*, Stanford, pp. 20-32.

Meta-Design: Transforming Application Areas – Continued

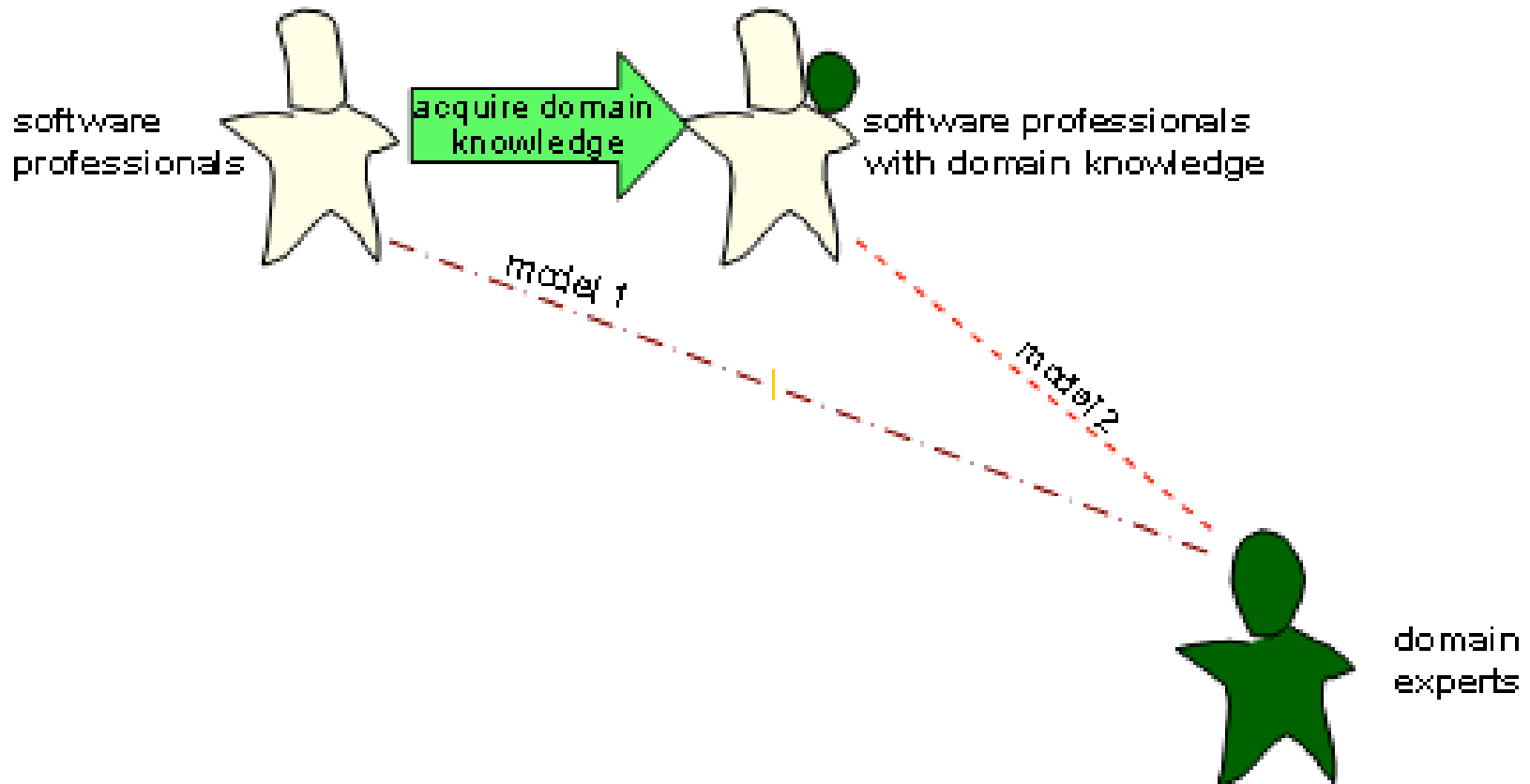
- **open source:** a success model of decentralized, collaborative, evolutionary development — Scharff, E. (2002) *Open Source Software, a Conceptual Framework for Collaborative Artifact and Knowledge Construction*, Ph.D. Dissertation, University of Colorado at Boulder.
- **living organizational memories:** living organizational memories such as Web2Gether — dePaula, R. (2004) *The Construction of Usefulness: How Users and Context Create Meaning with a Social Networking System*, Ph.D. Dissertation, University of Colorado at Boulder.
- **digital libraries:** community digital library — Wright, M., Marlino, M., & Sumner, T. (2002) *Meta-Design of a Community Digital Library*, D-Lib Magazine, Volume 8, Number 5, Available at <http://www.dlib.org/dlib/may02/wright/05wright.html>.
- **interactive art:** collaboration, co-creation, puts the tools rather than the object of design in the hands of users — Giaccardi, E. (2004) *Principles of Metadesign: Processes and Levels of Co-Creation in the New Design Space*, Ph.D. Dissertation, CAiiA-STAR, School of Computing, Plymouth, UK.

Reflective Communities

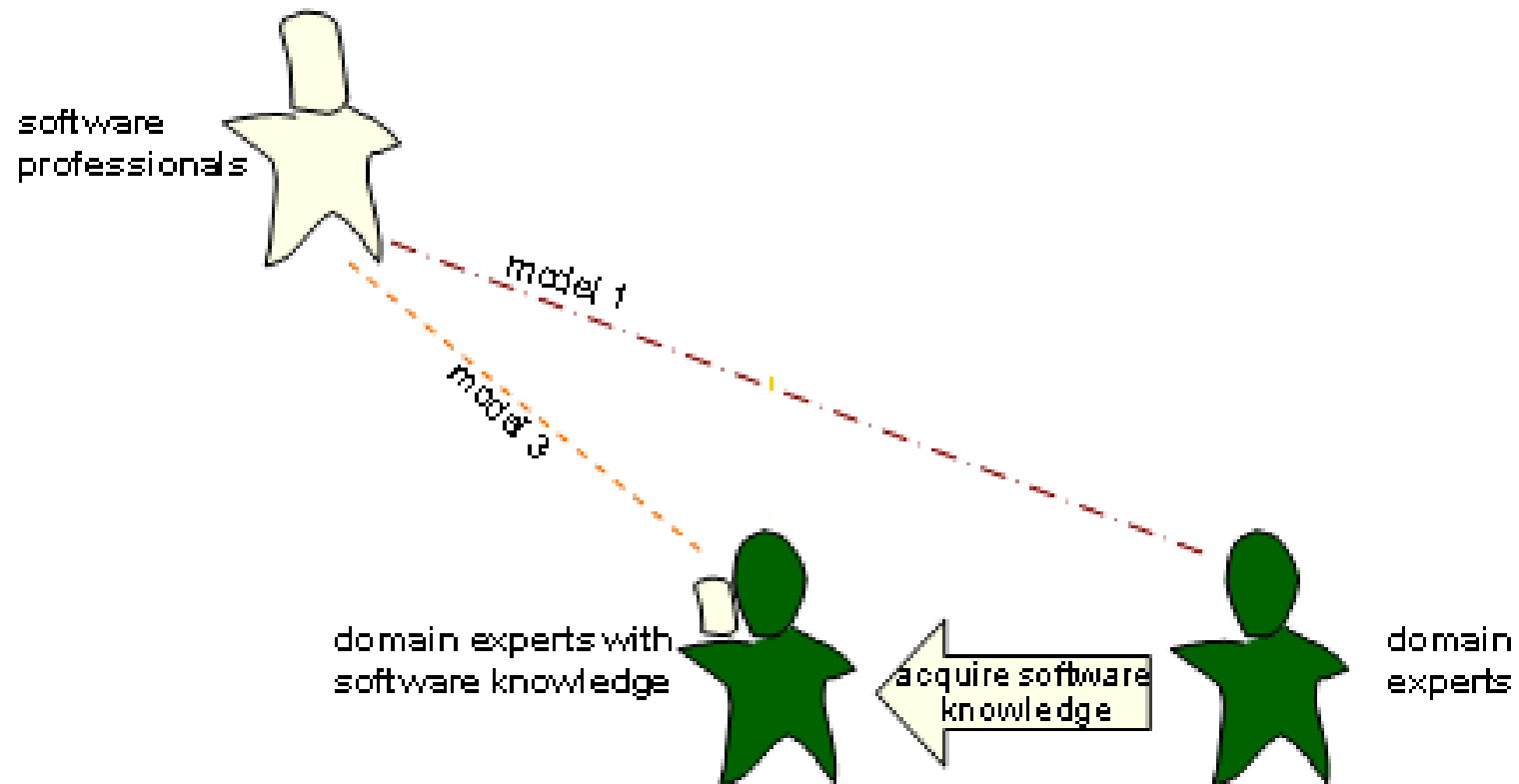
Large Conceptual Distance – Limited Common Ground



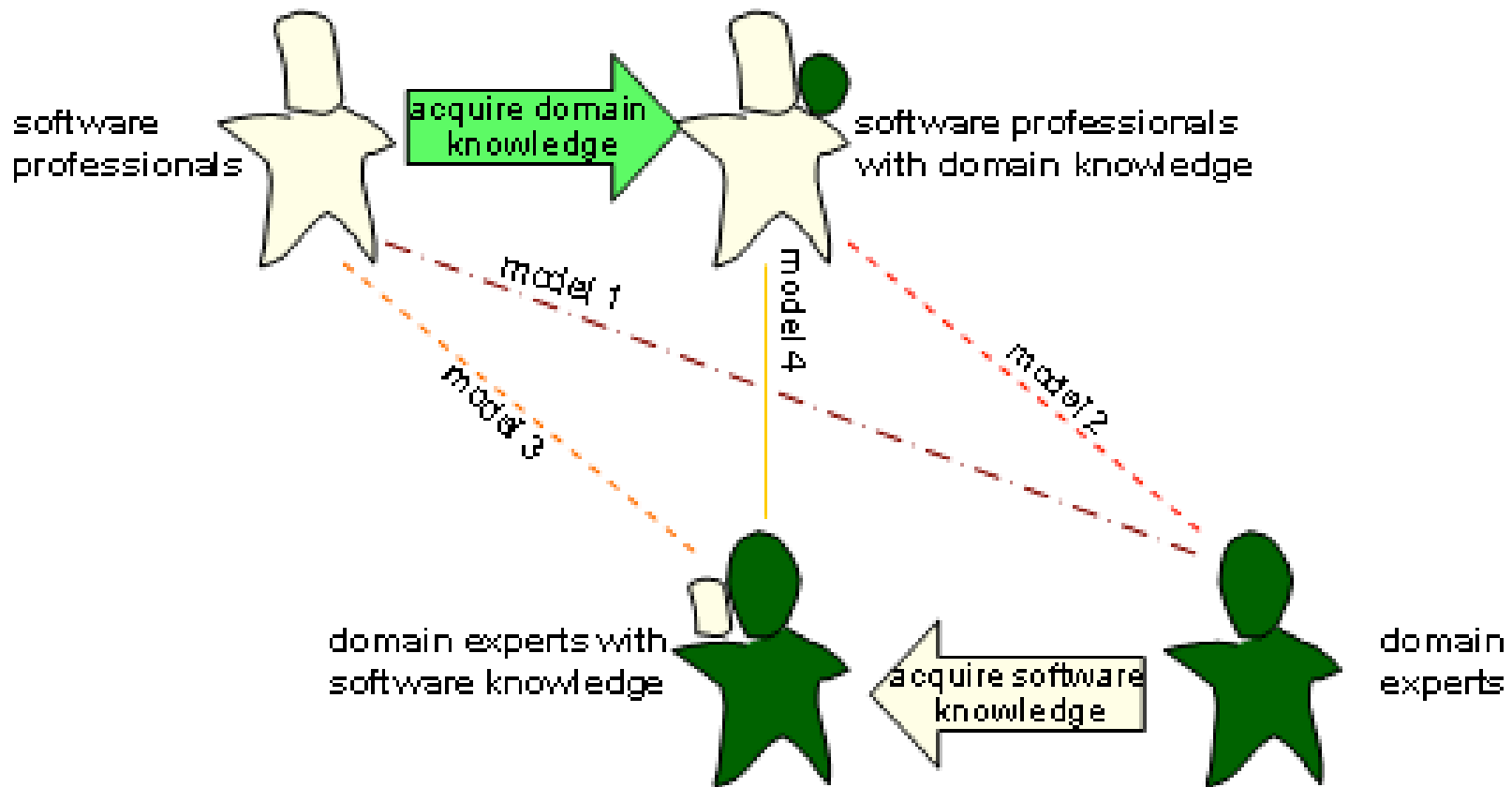
Software Professionals Acquiring Domain Knowledge



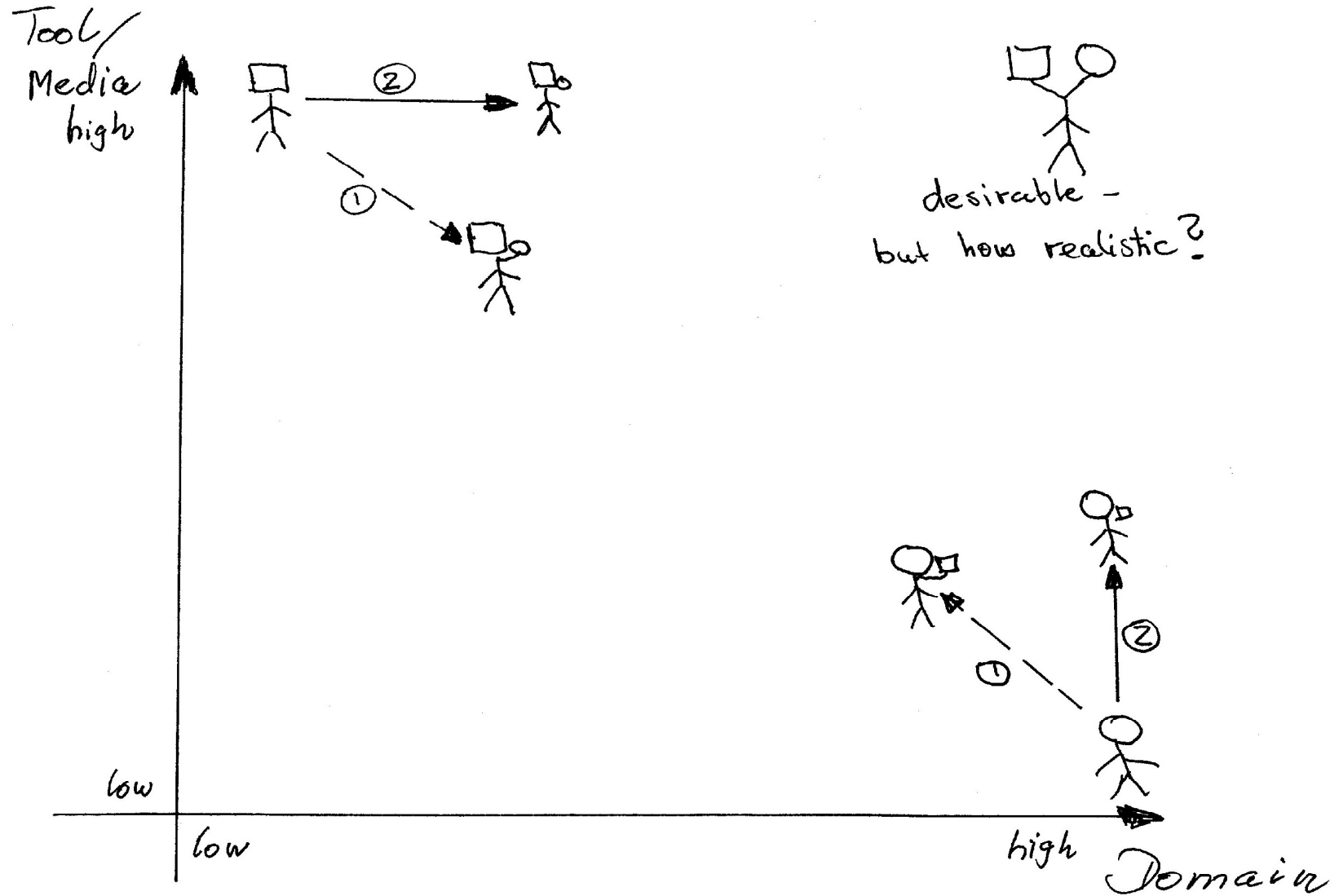
Domain Experts Acquiring Media Knowledge



From Reflective Practitioners to Reflective Communities



Renaissance Scholars or Reflective Communities



Software Design and Reflective Communities

- **claim:**

- to educate “Renaissance Scholars” such as Leonardo da Vinci, who was equally adept in the arts and the sciences is not a reasonable objective for the 21st century
 - to educate the superhuman who is equally adept in a content domain and a software professional is not a reasonable objective
 - *“the desire to show that with a computer **one person can do everything** may look not forward, but back to the stage in social evolution before anyone noticed the advantages of division of labor”* — Brown, J. S., & Duguid, P. (2000) *The Social Life of Information*, Harvard Business School Press, Boston, MA.
- *“system requirements are not so much analytically specified as they are collaboratively evolved through an iterative process of consultation between end-users and software developers”* — Computer Science Technology Board (1990) "Scaling Up: A Research Agenda for Software Engineering," *Communications of the ACM*, 33(3), pp. 281-293.
 - *“System development 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”* — Greenbaum, J., & Kyng, M. (Eds.) (1991) *Design at Work: Cooperative Design of Computer Systems*, Lawrence Erlbaum Associates, Hillsdale,

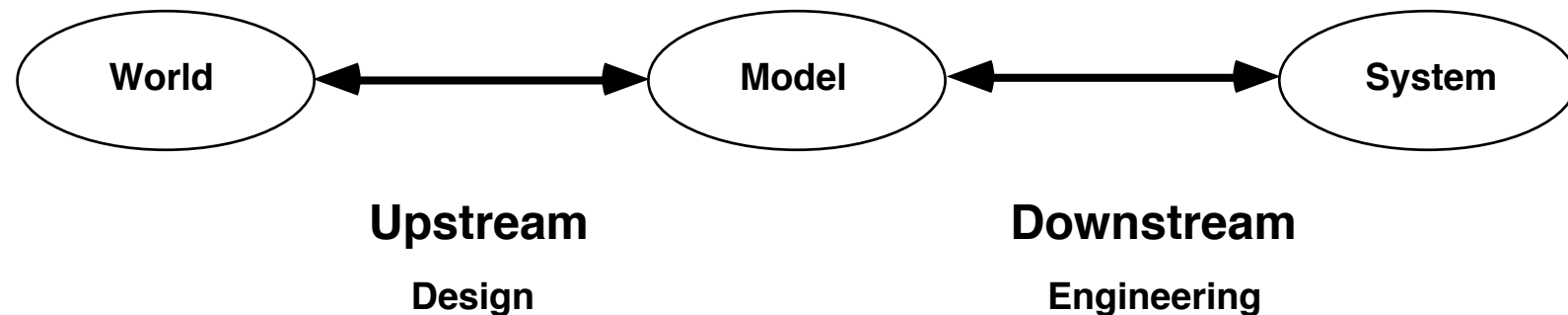
Outsourcing

- **an emerging question** for prospective computer science students: *“if the heart and soul of computing (programming) is being auctioned off to the lowest offshore bidder, what is the future for me?”*

- **question:** what will be the computing jobs, skills, and knowledge that are less likely to migrate offshore?

- **references:**
 - Friedman, T. L. (2005) *The world is Flat: A brief history of the twenty-first century*, Farrar, Straus and Giroux, New York.
 - Hagel III, J., & Brown, J. S. (2005) *The Only Sustainable Edge*, Harvard Business School Press, Boston.
 - William Aspray, Frank Mayadas, Moshe Y. Vardi (eds): “Globalization and Offshoring of Software”, A Report of the ACM Job Migration Task Force; at: <http://www1.acm.org/globalizationreport/>

Software Design: Upstream and Downstream Activities



- **upstream: world → model / specification**
 - ill-defined problem
 - integration of problem framing and problem solving
 - collaboration and communication between different stakeholders
 - failure leads to ***design disasters*** (wrong problem is solved)
- **downstream: model / specification → implementation / system**
 - well-defined problem
 - dealing with difficult technical problems
 - creating reliable code
 - failure leads to ***implementation disasters*** (wrong solution to the right problem)

Current Computer Science Education and Outsourcing

	upstream activities	downstream activities
themes	creative work, communication, collaboration, context, integration of problem framing and problem solving, fuzzy requirements, customer satisfaction	programming, programming languages, compilers, rule-based behavior (tax returns),....
emphasis in current CS programs	X	XXXXX
future jobs (not being outsourced)	XXXXX	X