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

Computer-Supported Cooperative Work (CSCW)

Gerhard Fischer and Leysia Palen
Spring Semester 1999

April 7, 1999

Computer-Supported Cooperative Work (CSCW)

- explores the potential of computer technologies **to help people work together**

- CSCW is relevant to design because **collaboration** is playing a larger role in design projects

- large and complex design projects **cannot be accomplished by any single person**
  - they increasingly require expertise in a wide range of areas
  - software design projects involve: designers, programmers, HCI specialists, marketing people, user participants, ...

- **the design process takes place over many years**, with initial design followed by extended periods of maintenance, evolution and redesign

A Communication And Coordination Perspective

—

Support for Communication With

- **ourselves** (e.g., capturing our thoughts of the past, allowing us to create personalized information environments that extend the knowledge we can keep in our head —*reflexive CSCW*

- **tools** (e.g., knowing which tools exist, how the can be used, and how they can be tailored to our specific needs

- **colleagues** (e.g., supporting long-term, indirect collaboration)
  - community of practice
  - community of interests

- **agents and critics** (e.g., in the context of cooperative problem-solving systems)

Communication And Coordination Processes

- **between designers and clients**
  - clients do not know what they want,
  - designers and clients need shared knowledge and artifacts for mutual understanding
  - require “externalizations for mutual understanding / languages of doing” instead of formal representations

- **within design teams**
  - most real tasks are not done by individuals but by groups of people
  - members within such teams might have very different interests (for example, waterfall models in software design are the heaven for managers and the hell for creative programmers)

- **between designer(s) and design environments**
  - environments serve as group and design artifact memories that can be used to support indirect, long-term communication
  - discussions about the design must be embedded in the design

Dimensions of Collaboration

- **temporal (across time)**
  - asynchronous
  - indirect, long-term

- **spatial (across space)**
  - networks

- **social (among persons)**
  - “virtual communities”

- **technologically (persons and artifacts)**
  - human-computer interaction ---->*human problem-domain communication*

Formalization

- **different approaches and requirements:**
  - DAI (Distributed Artificial Intelligence)
  - CPS (Cooperative Problem Solving)
  - CSCW (Computer-Supported Collaborative Work)

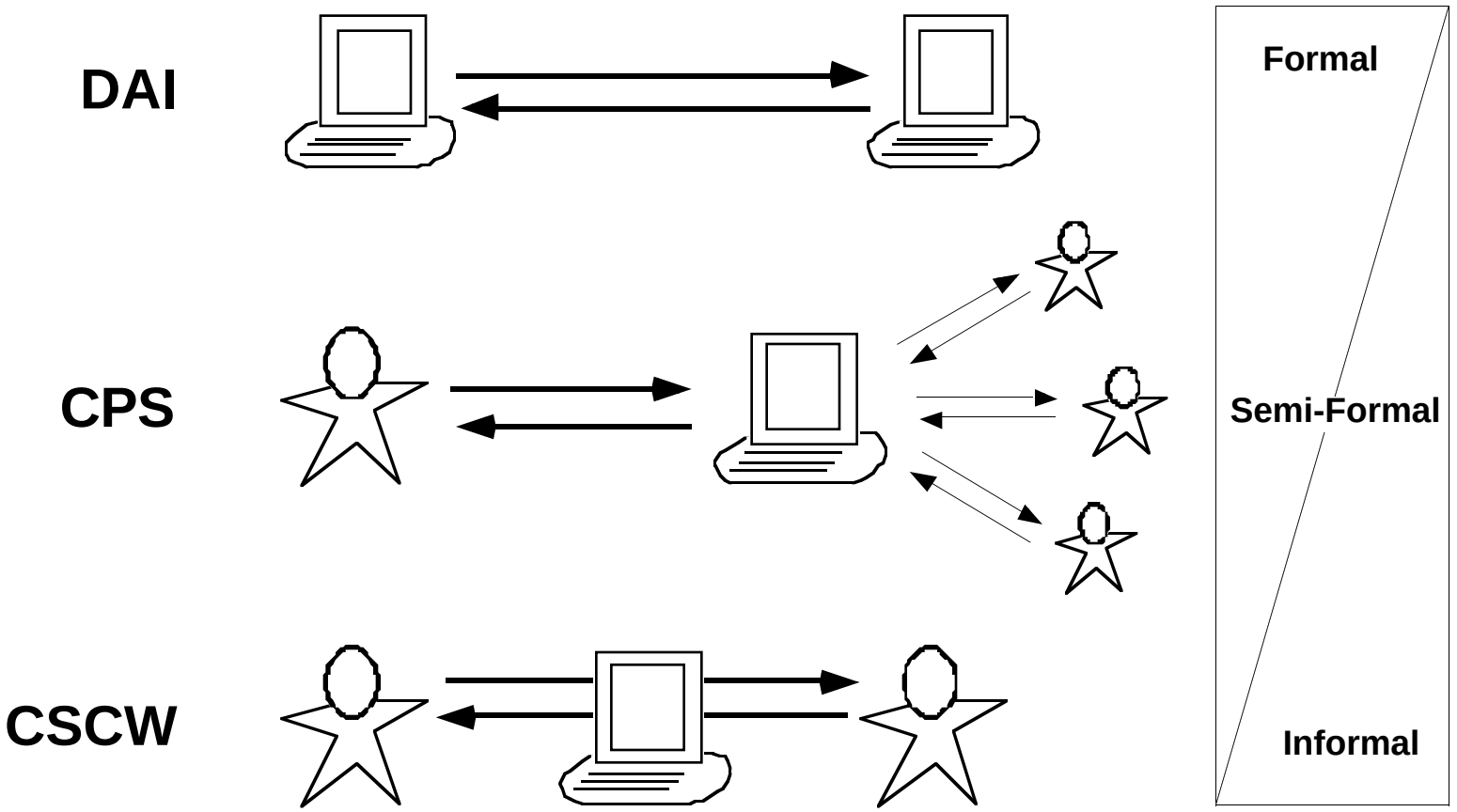
- **the rationale for incremental formalization:**
  - + of informal: less disruptive
  - + of formal: more computationally interpretable
  - see: Shipman, F. (1993) *Supporting Knowledge-Base Evolution with Incremental Formalization*, Ph.D. Dissertation, Department of Computer Science, University of Colorado at Boulder, Boulder, CO.

Incremental Formalization

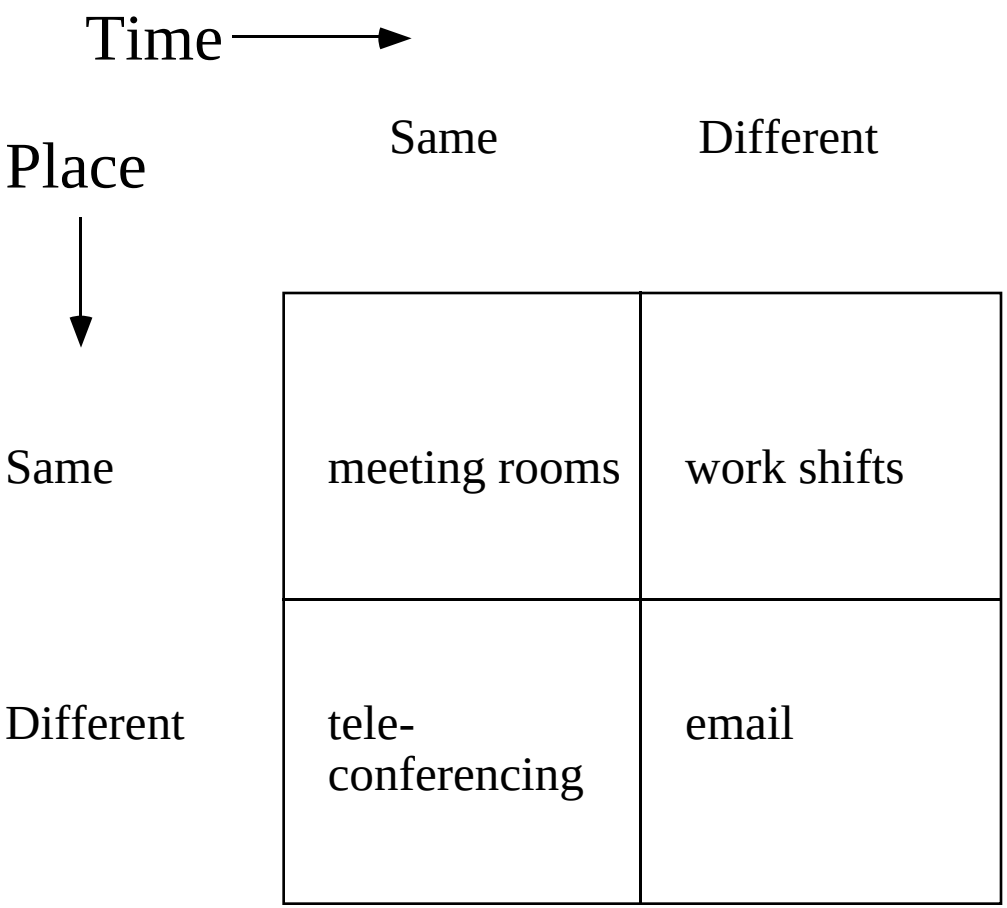
- **problems:**
  - the inhibiting effect of explicit structure
  - the need to exploit informal communication
  - the need for formalization

- **solutions:**
  - informal, unstructured input
  - support for incremental formalization

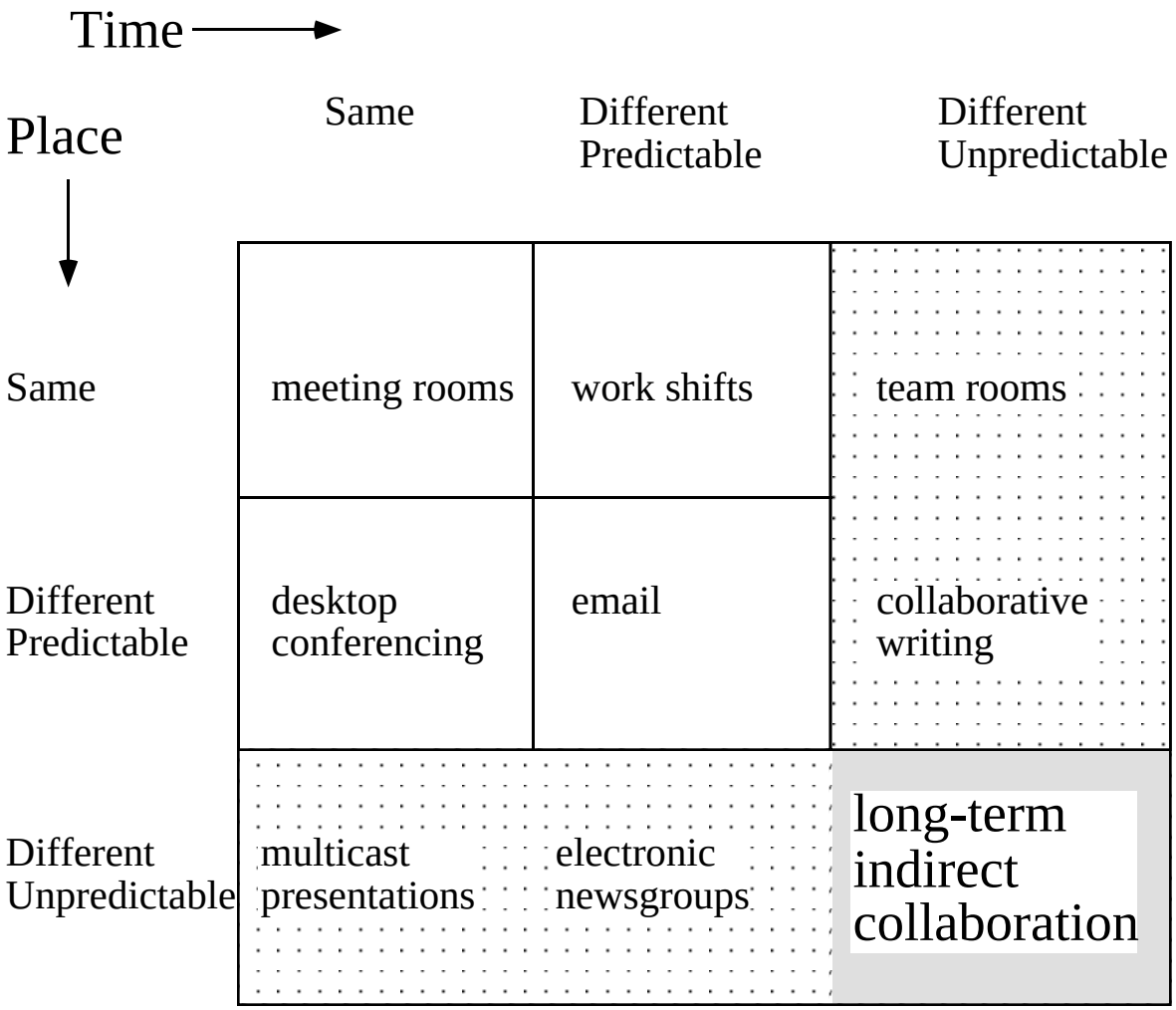
Alternate Communication Paradigms



The Standard Classification Scheme of CSCW Perspectives



An Extended Classification of CSCW Perspectives



Design, Use, Maintenance and Evolution
—
Who is Doing What?

- **traditional view:**
  - design and maintenance: by the software designer
  - use: by the user

- **our view — to support evolution:**
  - participatory design -- users are part of the design activity; they ***own*** the problem
  - “use” is not just use, but should lead to enhancements -- end-user modifiability

- **maintenance / evolution:**
  - 60% of overall software costs of a program occur after the program is delivered
  - 75% of maintenance are program enhancements ---> best done by the users

Examples of CSCW Systems

Email

- Email is the most successful groupware application.
- limitations as a tool for collaborative design:
  * typically managed by individuals, there is no shared information space
  * stored separately from artifacts, its contents are more difficult to access
  * isolated from the design artifact, users must supply more contextual information along with their message (deixis is impossible)

WorkFlow Systems

- seek to coordinate work within and across workgroups
- support scheduling, design verification (assurance that an artifact is implemented “to-specification”), and reporting

Groupware — examples:

- Coordinator system (structuring conversations according to a specific conversational model)
- Lotus Notes
- GIMME system stores email messages

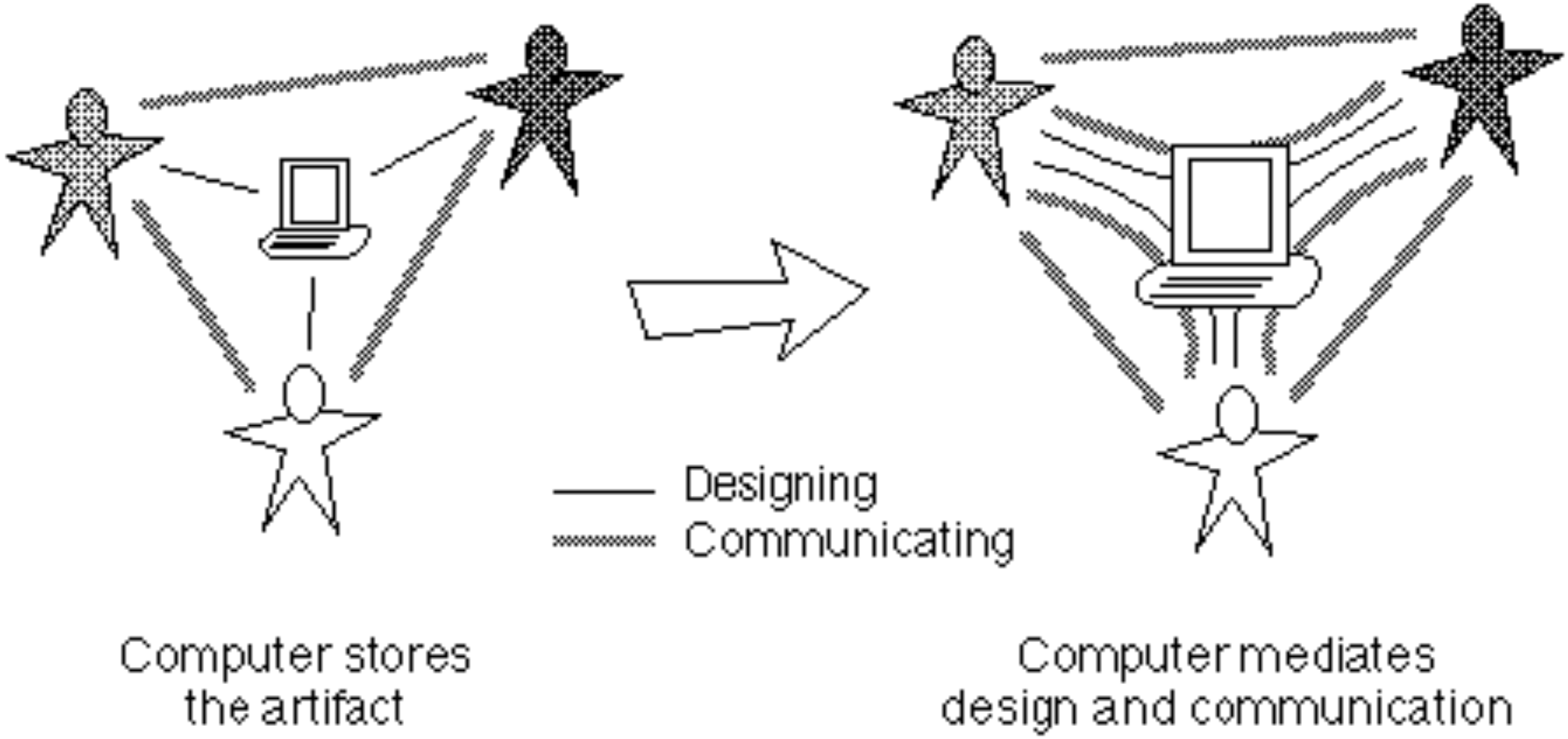
Domain-Oriented Design Environments as CSCW Systems

- **supporting indirect, long-term collaborative design with integrated, domain-oriented design environments in situations where:**
  - direct communication is impossible, impractical or undesirable
  - communication is shared around artifacts and information space evolution
  - designers need to be informed within the context of their work on real-world design problems

- support **reflection-in-action**, make argumentation serve design

- support **embedded communication**

Embedding Artifacts and Communication in Design Environments



Collaborative Work Practices

- **division of labor** is a cultural universal
- **claim:** domain-oriented design environments afford end-user more power, but not at the expense of ease of use
- **challenge:** to design both software systems and work practices that take advantage of the division of labor
- **work practices:**
  - local developers, gardeners, power users
  - representation for mutual understanding
  - face-to-face or media-supported direct communication
  - remote, indirect collaboration via reuse of artifacts
- **software systems:**
  - end-user modifiability
  - catalogs
  - design rationale
  - layered architectures
  - critiquing knowledge