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

Domain-Oriented Design Environments and Critiquing

Gerhard Fischer and Leysia Palen
Spring Semester 1999

February 24, 1999
Domain-Oriented Design Environments

- **goals:**
  - bring task to the forefront
  - analysis of work products
  - goal sharing (for agents, critics, task-based indexing)
  - information delivery
  - learning on demand
  - external simplicity through internal complexity

- **theory:**
  - collaborative problem solving
  - distributed cognition
  - integration of problem framing and problem solving
  - reflection-in-action
  - design-in-use
  - situational awareness
  - computational environments as “living” entities

- **users:**
  - skilled domain workers
  - stakeholders with different interest and different background knowledge
End-User Modifiable, Domain-Oriented Design Environments

- General Programming Environments, e.g., Lisp, ... \(\rightarrow\) limited reuse

- Object-Oriented Design, e.g., Smalltalk, Clos, C++, Java
  \(\rightarrow\) lack of domain-orientation

- Domain-Oriented Construction Kits, e.g., Pinball, Music Construction Kits
  \(\rightarrow\) no feedback about quality of artifact

- Constructive Design Environments, e.g., critics, explanations
  \(\rightarrow\) design is an argumentative process

- Integrated Design Environments, e.g., combining construction and argumentation
  \(\rightarrow\) lack of shared context

- Multifaceted Architecture
  \(\rightarrow\) limited evolution

- End-User Modifiable Design Environments
The Multi-Faceted Domain-Independent Architecture for DODEs
Examples of Domain-Oriented Design Environments

- user interface design — **Framer**
- floor plan design for kitchens — **Janus, KID**
- computer network design — **Network, Pronet, Webnet**
- Cobol programming and service provisioning — **GRACE** (with NYNEX)
- voice dialog design — **VDDE** (with USWest)
- lunar habitat design — **HERMES** (with NASA)
- graphic arts, information design, information visualization — **Schemechart, Chart ‘n’ Art**
- multi-media design environment — **eMMa** (with SRA)
Shared Context in Domain-Oriented Design Environments

• **increase on the system's side**
  - domain-orientation
  - construction
  - specification
  - embedded communication and history
  - incremental formalization

• **increase on the user's side**
  - “back-talk” of the situation (critics, simulation)
  - specification support through the argumentation component
  - making argumentation serve design (providing arguments behind critiquing messages)
  - access and delivery of cases (catalog examples) relevant to the task at hand
Why Critiquing?

- **support reflection-in-action**
  - the designer shapes the situation in accordance with his initial appreciation of it construction
  - the situation “talks back” with the help of the critics
  - in answers to the situations “back-talk”, the designer reflects-in-action on the construction of the problem argumentation

- **humans settle on plateaus of suboptimal behavior**

- **“virtual” stakeholders**
Rationale for Critiquing Systems

• **Claim**: as people take on more jobs that are more complex or more comprehensive, they need help accomplishing unfamiliar tasks that are part of an expanded job — e.g.: multi-media is a good example (charts, color, ...)

• **Kosslyn** (in “Elements of Graph Design”, p 2):
  - one reason for the abundance of bad graphs is the proliferation of low-cost microcomputers and “business graphics” packages, which often seduce the user into producing flashy, but muddled display
    - the ease of creating charts and graphs is a major selling point for personal computers, one rarely hears anything about the utility of the displays the machines produce

• **Travis** (in “Effective Color Displays”):
  - the standard IBM PC can now display 256 K colors and a Sun workstation can display 16.8 million — hardware is no longer a limiting factor to use color
    - *but*: when color is used inappropriately it can be very counter productive and few software designers have much experience with the use of color
Critiquing

- **critiquing** = presenting a reasoned opinion about a user’s product or action

- critics make the constructed artifact “**talk back**” to the users (beyond the “back-talk” provided by the materials)

- critics should be **embedded** into domain-oriented design environments

- **critiquing process:**
  - goal acquisition
  - product analysis
  - critiquing strategies (when, how, and where)

- **classes of critics:**
  - educational and/versus performance: primary objective is learning and/versus better product
  - negative and/versus positive
What is Critiquing?

• **exploiting the true power of computational media**
  - paper: passive — e.g.: style guides, design rationale systems
    (see Web Style manual: http://info.med.yale.edu/caim/StyleManual_Top.HTML)
  - computational media: active — critiquing, constraints, simulation,
    making argumentation serve design, contextualizing information to the
    task at hand, embedded critiquing

• **role distributions**
  - in our approach most of the time: human designs and computer
    critiques
  - proactivity (e.g., the Pronet system: the users designs the high-level
    architecture and the system fills in the details)
  - examples of computer designs and human critiques: Unix directory
    trees (the computer “knows” or can compute the information structure)

• **increase the back-talk of a situation**
  - how is failure or inadequacy of the form perceived in a design?
  - Rittel: “Buildings to not speak for themselves”
  - critics volunteer information
Examples

- spelling, grammar, color

- Lisp-Critic
  - all Lisp program could be critiqued
  - no knowledge about the problem to be solved (the macro example; compare to technical editor)

- Voice Dialog Design:
  - critiquing from multiple perspective
  - end-user control over intrusiveness

- critiquing at
  - the tool level (Lisp-Critic, spelling checker)
  - critiquing at the domain level (kitchen, VDDE, lunar habitat design)

- embedded critiquing
  - specific critics (left-handed, very short person)
  - interpretive critics (resale versus personal)
Embedded Critics

construction

critics

specification
Is the cook right- or left-handed?

catalog

design rationale

issue:
answer:
argument:

perspectives
resale
electrical
American

personal
plumbing
Japanese
Assessment Questions for Critiquing Systems

• differences in performance if the system is used with and without critics, catalog, and simulation component?

• integrate constraints (e.g., for building codes)

• trade-offs between running the system in a mode
  - to prevent problems to occur (constraints)
  - to let designers get in trouble

• intervention strategies (displaying enough information versus disrupting the work process)?

• does “making information relevant to the task at hand” prevent “serendipity”? 

• when are designers willing to suspend the construction process to access relevant information?

• when will designers/users challenge or extend the knowledge represented in the system? ---> end-user modifiability
Lessons Learned From Our System-Building Efforts

• DODEs support “human problem domain communication”

• DODEs are instrumental versions of systems that are simultaneously user-directed and computationally supportive

• critiquing
  - breakdown as opportunities
  - supports contextualized learning on demand
  - makes argumentation serve design

• seeds need to be functional enough that they are used by skilled domain designers in their work

• sociological structure of communities of practice with power users and local developers
Assessment of DODEs

• **current limitation of DODEs:**
  – limited success models — specifically lack of experience with evolutionary growth in naturalistic settings
  – tool mastery burden

• **research issue for DODEs**
  – design rationale
  – case-based reasoning
  – integrated artifact memories
  – multi-user DODEs
  – evolutionary growth through use
  – new contracts between stakeholders

• **challenges**
  – the question is how — not why?
  – how large or small, general or specific should a domain be?
  – cost-effectiveness: powerful substrates are needed
A Few References about DODEs and Critiquing


