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

High-Functionality Applications and User Modeling

Gerhard Fischer
Center for LifeLong Learning & Design (L^3D)
http://www.cs.colorado.edu/~l3d/
Department of Computer Science and Institute of Cognitive Science
University of Colorado, Boulder

Tutorial (December 5, 2000) at OZCHI 2000

High-Functionality Applications

- **high-functionality applications**
  - are used to model parts of existing worlds (and to create new worlds)
  - are complex systems because they serve the needs of large and diverse user populations → hypotheses/fact: *"if we ask 100 different people what features they would like to have in a particular application, we will end up with a very large number of features"*

- **examples: (VCRs), Unix, MS-Word, MS-Office, Photoshop, Eudora, Mathematica, …….**

- **the design of HFAs must address three problems:**
  - commonly used functionality should not be difficult to learn, use, and remember
  - unknown existing functionality must be accessible or delivered at times when it is needed
  - the unused functionality must not get in the way

Design Challenge: useable versus useful → usable and useful

"If ease of use was the only valid criterion, people would stick to tricycles and never try bicycles".
Doug Engelbart

main objective	usable	useful
users	novices	skilled users
size	limited functionality	broad functionality
most important design criteria	low threshold to get started	high ceiling for skilled users
expertise	"experts" exist	no "experts" (learning on demand is a necessity rather than a luxury)
models	understandable model of the complete system can be developed	no complete models
examples	original MacIntosh, ATMs, VCRs	Unix, MS-Word, MS-Office, Photoshop, Eudora, Mathematica

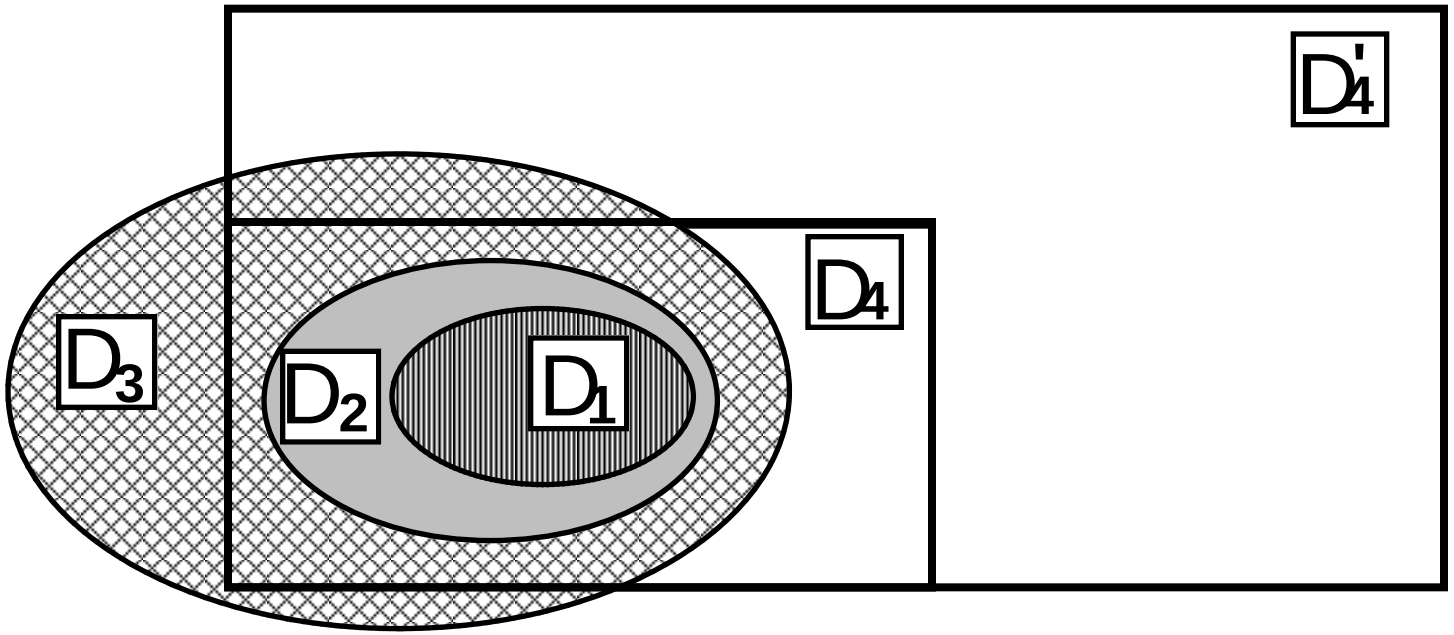
Success Story of a High-Functionality Application
—
A Large Hardware Store with Knowledgeable Sales Agents

- empirical study: McGuckin Hardware store in Boulder, Colorado — more than 350,000 different line items
- problem setting and problem solving are intertwined
- queries are articulated incrementally, situations talk back, examples are critical
- to determine the relevance of a found object requires domain knowledge (e.g., “simulation of use” — the plumber story)
- a shared understanding is incrementally achieved between customer and sales agent
- summary: “computer systems have the same functionality as McGuckin, but are operated like K-Mart”

High-Functionality Applications (HFA)

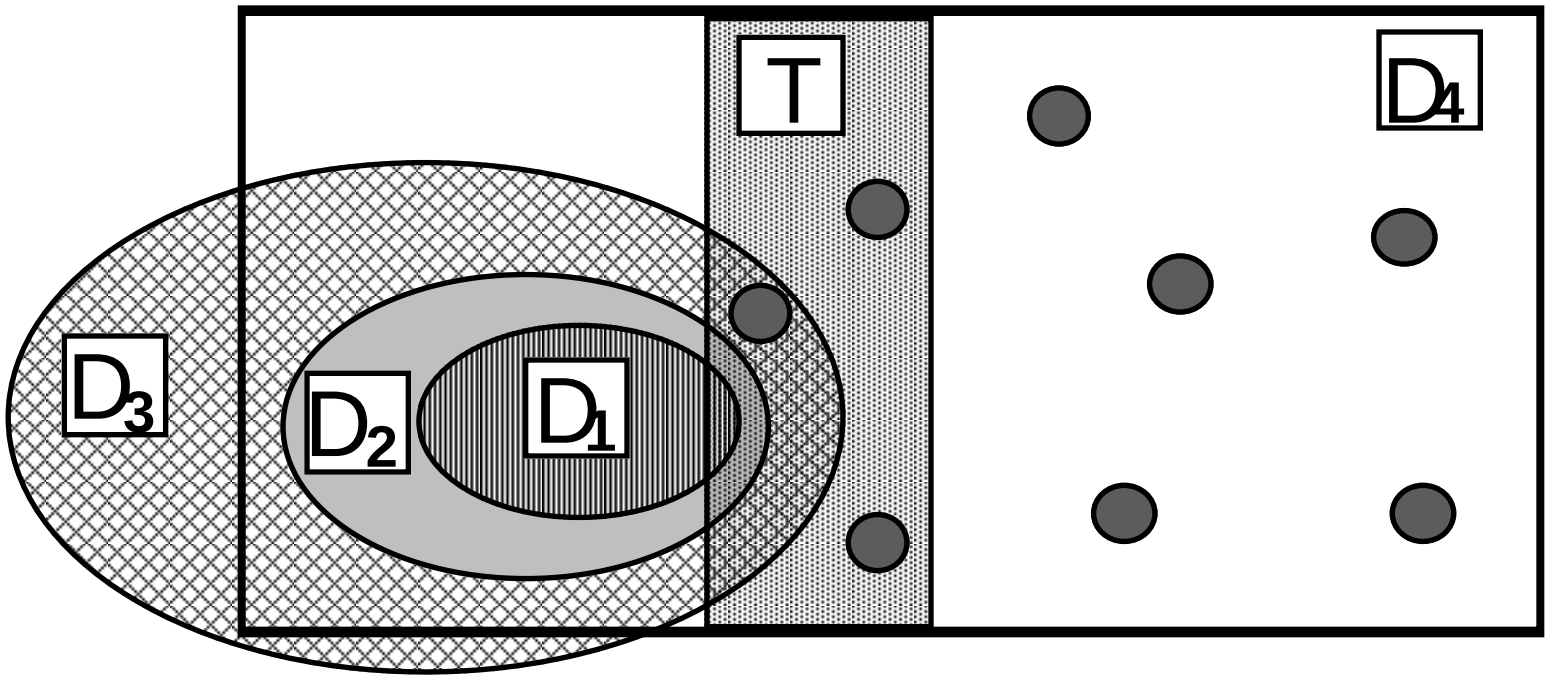
Levels of Users' Knowledge About a System's Information Spaces

(based on numerous empirical investigations)

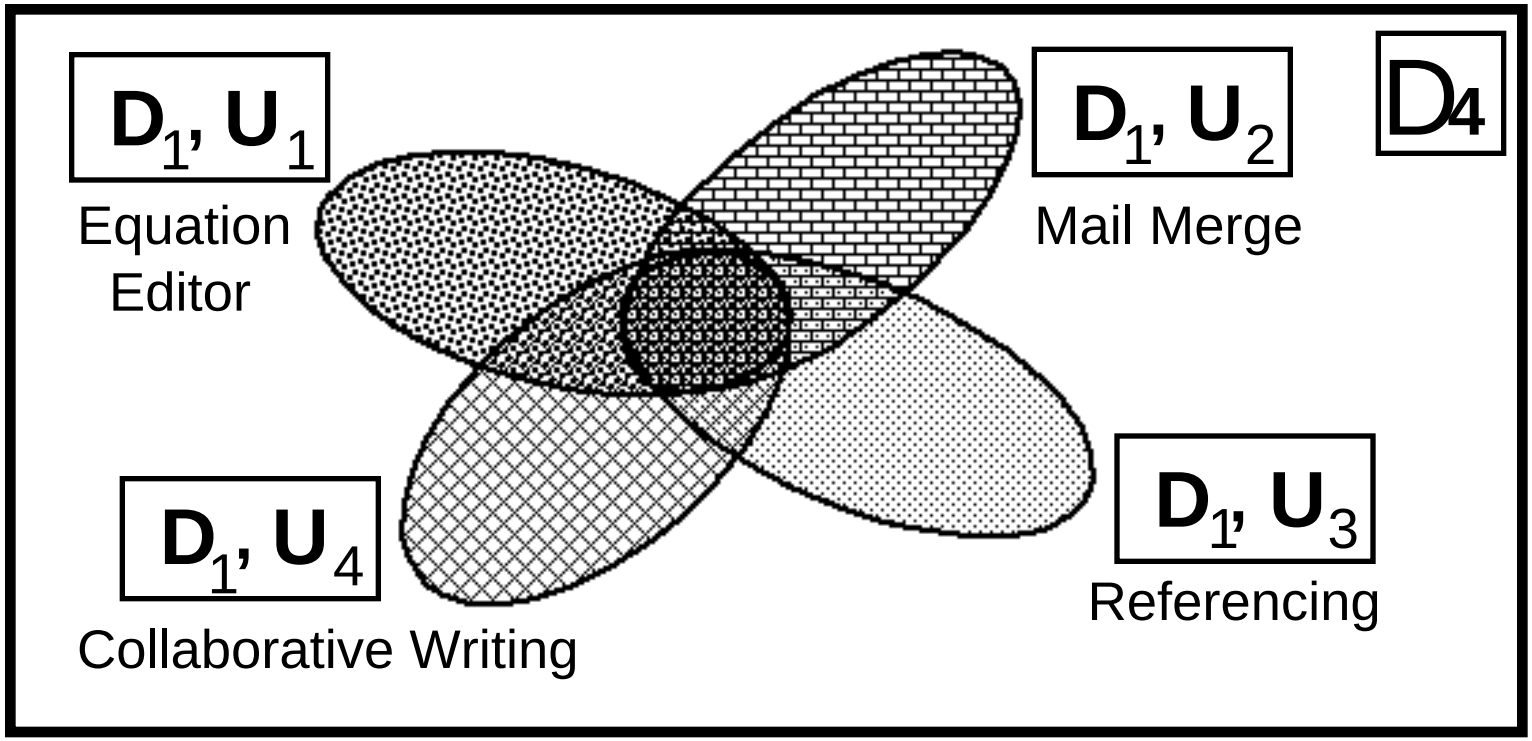


Functionality and its Relevancy to the Task at Hand in HFAs

Why “Did You Know (DYK)” and “MS Tip of the Day” is of limited success



Expertise in HFAs is an Attribute of a Context, not of a Person

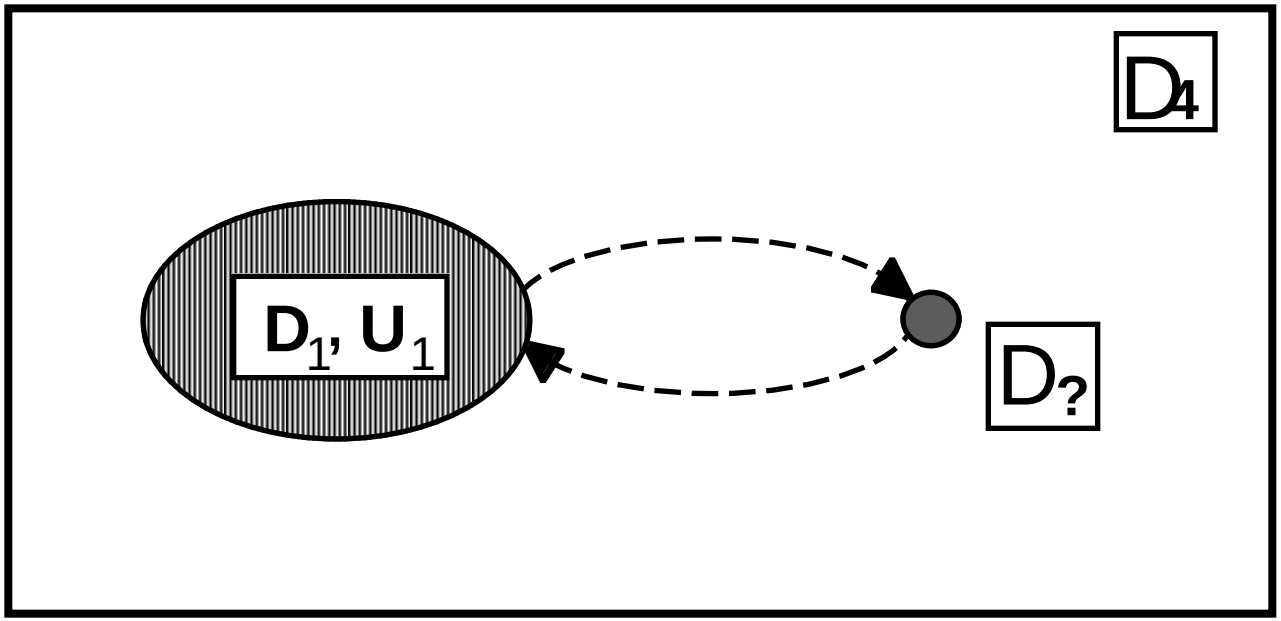


Problems with HFAs

- users do not know about the **existence of tools** ($D_4 \neg \wedge D_3$)
- users do not know how to **access tools**
- users do not know **when to use tools** — lack applicability conditions
- users cannot **combine, adapt, and modify tools** according to their specific needs
- additional complicating factor: in the real world problems are not given but emerge, implying that no precise goals and specifications can be articulated → **intertwining of problem framing and problem solving**

Entering Unknown Parts of D_4 — Opportunity or Problem

- **issues:** a user hits the wrong keys (but the keystrokes get interpreted in D_4); the system infers the "wrong" intentions from the users actions — ***"every wrong answer is the right answer to some other question"***
- **problem**: "smart" systems are often guessing wrong (e.g., in MS-Word: AutoCorrect, Tables, Bullets and Numbering, ……)
- **opportunity**: serendipity



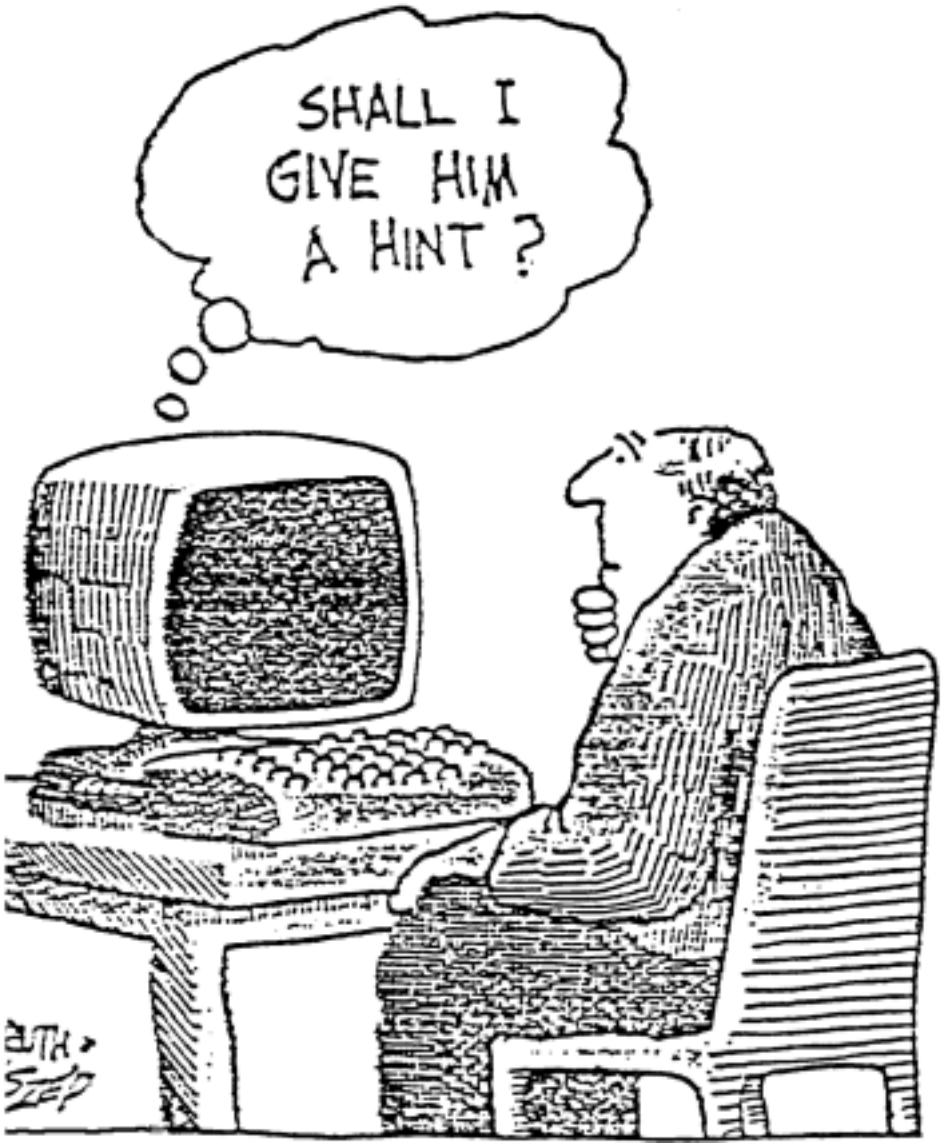
Problems with HFAs: Microsoft's View and Objectives

- some "routine" tasks could be and needed to be automated
- some tasks are used too infrequently by users to make it worthwhile for them to learn how to complete them and complex enough that users would need to relearn how to perform them each time they tried to accomplish the task **(→ main streets and side streets)**
- complex tasks may include options that could benefit the users - options that the user might never take advantage of
- users have different levels of expertise and backgrounds and therefore require different levels of support
- tasks supported by software are broad
- users don't want to become technical experts, they just want to get their tasks done
- users don't know about all software features that could help them
- help is insufficient, spread out over the user interface, hard to use, and requires prior knowledge of computer software lingo
- users want tailored help delivered in a friendly and easy to understand manner

How Our Research Addresses the Problems Created by HFAs

- ***active help systems*** — analyze the behavior of users and infer higher-level goals from low-level operations
- ***specification components*** — allow users to enrich the description of their tasks
- ***critiquing components*** — analyze and infer the task at hand; detect and identify the potential for a design information need; present contextualized knowledge for designers
- ***increase user and task relevance*** by integrating specification component and critiquing components; *generic critics* (defined at design time) → *specific critics* (information only known at use time)
- ***create malleable systems*** by integrating adaptive and adaptable components

Information Delivery, Contextualization and Intrusiveness



Some Challenging Research Problems for User Modeling

- **identify user goals from low-level interactions**
  - active help systems, data detectors
  - *"every wrong answer is the right answer to some other question"*
- **integrate different modeling techniques**
  - domain-orientation
  - explicit and implicit
  - give a user specific problems to solve
- **capture the larger (often unarticulated) context and what users are doing** (especially beyond the direct interaction with the computer system)
  - embedded communication
  - ubiquitous computing
- **reduce information overload by making information relevant**
  - to the task at hand
  - to the assumed background knowledge of the users
- **support differential descriptions** (relate new information to information and concepts assumed to be known by the user)

Early Example: Knowledge-Based Help Systems (CHI'85)

—

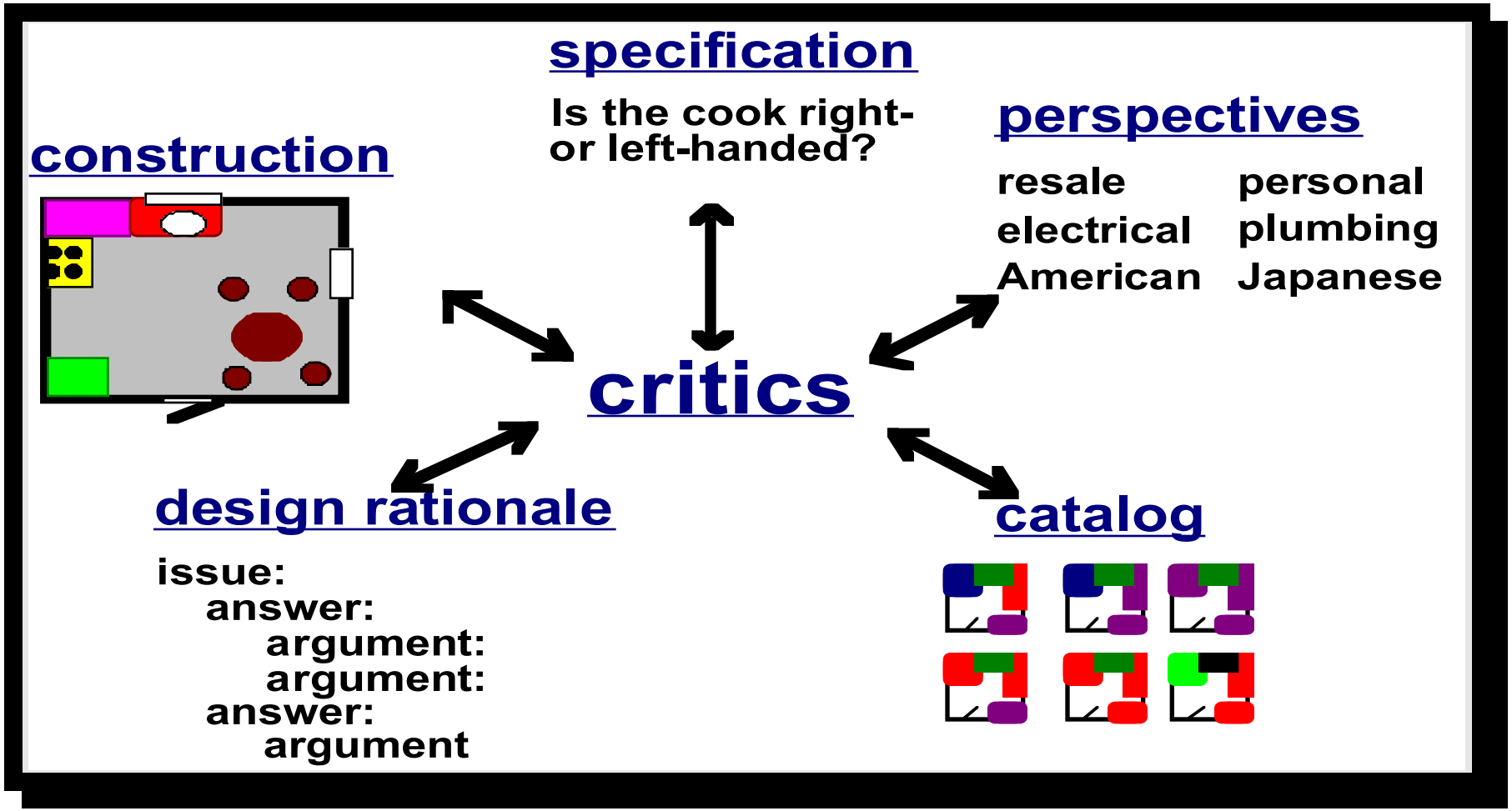
Activist (and Passivist)

- **Activist — an active help system** for an EMACS-like editor, deals with two different kinds of suboptimal behavior:
  - the user does not know a complex command and uses "suboptimal" commands to reach a goal ("suboptimal": main streets and side streets?)
  - the user knows the complex command but does not use the minimal key sequence to issue the command

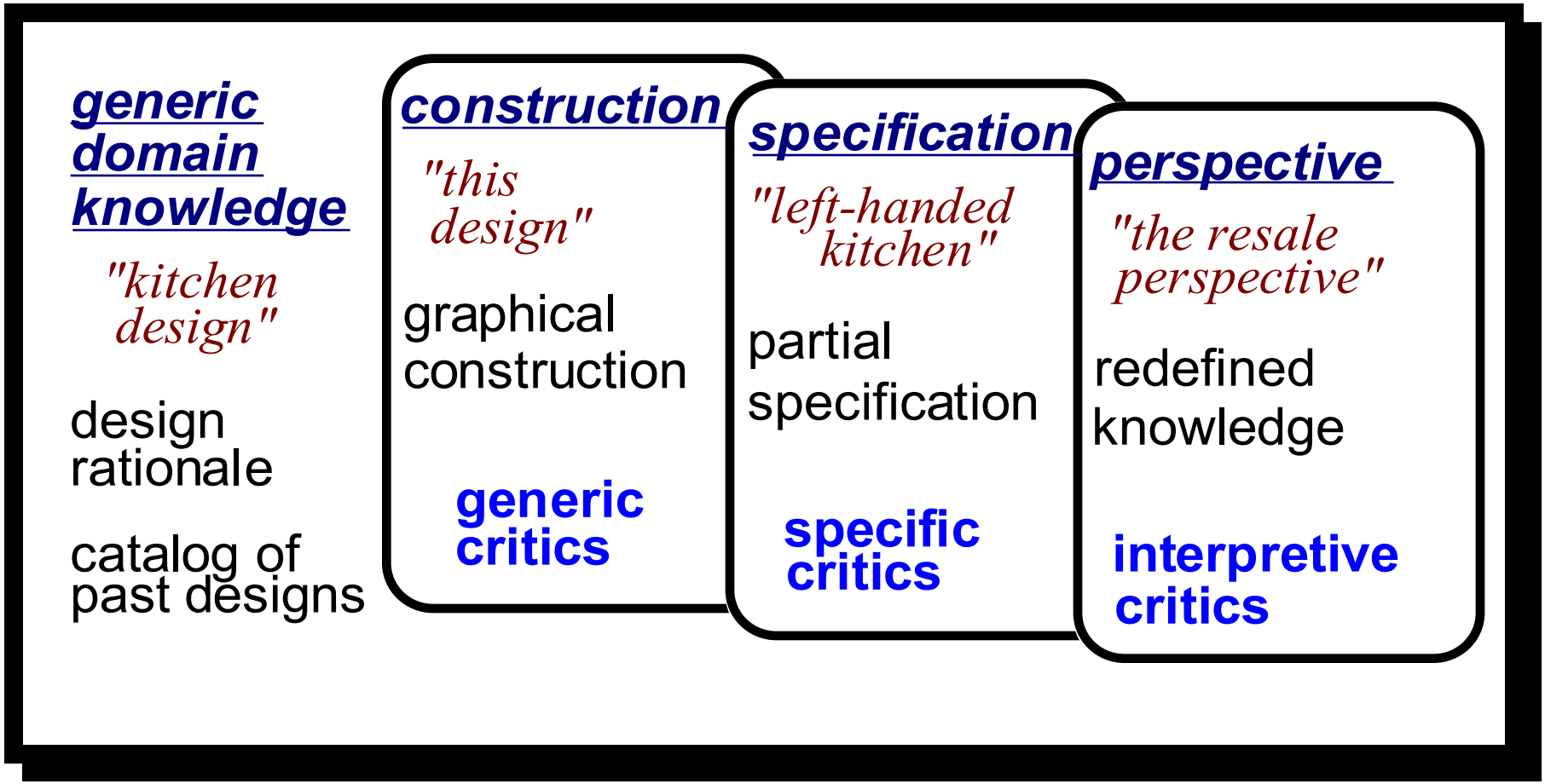
- similar to a human observer, **Activist handles the following tasks**:
  - recognizes what the user is doing or wants to do
  - evaluates how the user tries to achieve his/her goal
  - constructs a model of the user based on the results of the evaluation task
  - decides (dependent on the information in the model) *when* and *how* to interrupt (tutorial intervention)

- the recognition and evaluation task is delegated to **20 different plan specialists**

Domain-Oriented Design Environments



Embedding Critics in the Contexts of Design (Context defined by appropriate User Characteristics)



Embedding Critics

Saying the "right" thing

at the "right" time,

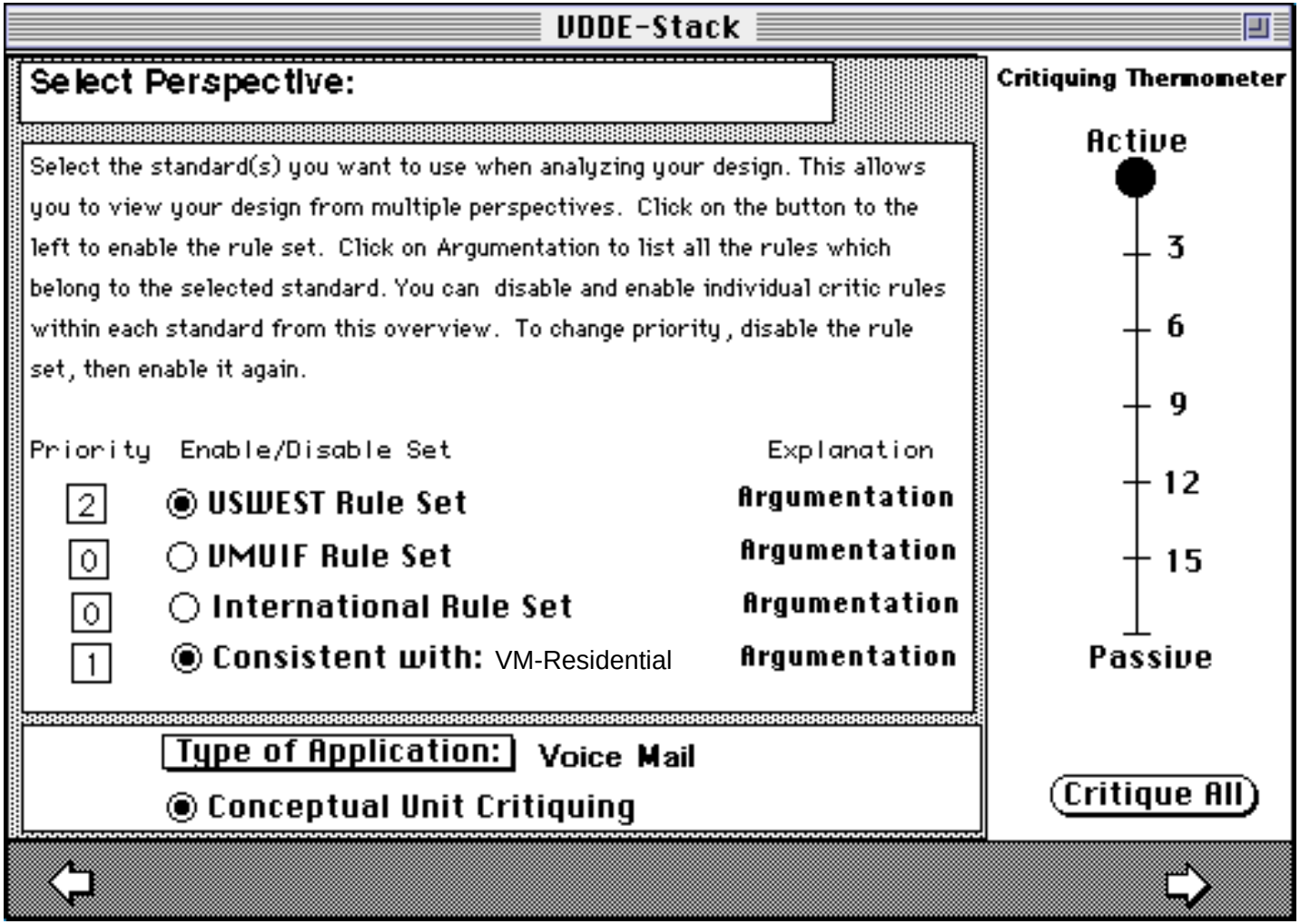
in the "right" way

- **benefits of embedding critics**
  - integrate design environment components
  - allow system to infer the task at hand and user characteristics
  - enable only the most relevant critic rules
  - modify critic rules to reflect task at hand and user characteristics
  - deliver more relevant information

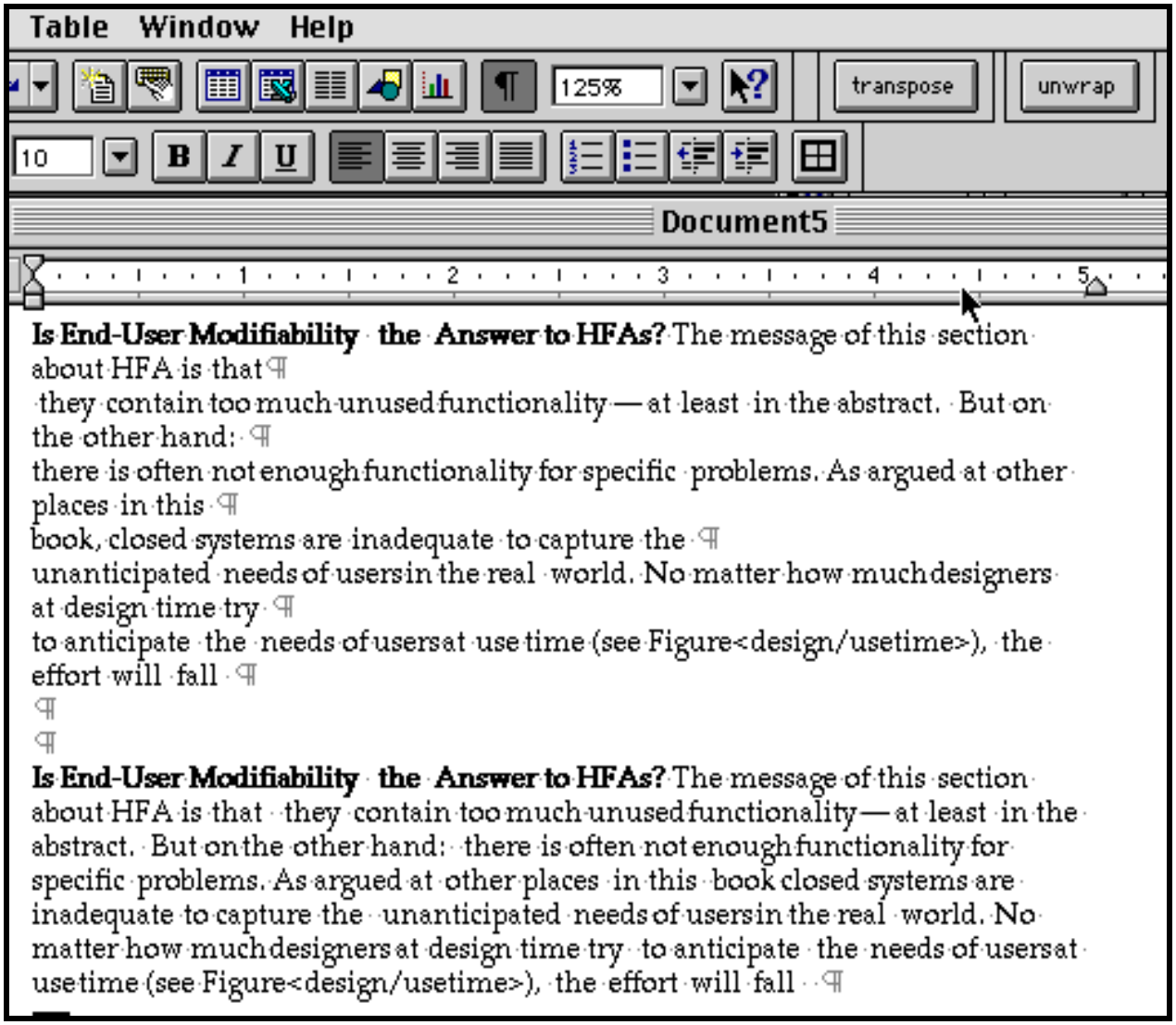
A Comparison between Adaptive and Adaptable Systems

	Adaptive	Adaptable
Definition	dynamic adaptation by the system itself to current task and current user	user changes (with substantial system support) the functionality of the system
Knowledge	contained in the system; projected in different ways	knowledge is extended
Strengths	little (or no) effort by the user; no special knowledge of the user is required	user is in control; system knowledge will fit better; success model exists
Weaknesses	user has difficulty developing a coherent model of the system; loss of control; few (if any) success models exist (except humans)	systems become incompatible; user must do substantial work; complexity is increased (user needs to learn the adaptation component)
Mechanisms Required	models of users, tasks and dialogs; knowledge base of goals and plans; powerful matching capabilities; incremental update of models	layered architecture; human problem-domain communication; "back-talk" from the system; design rationale
Application Domains	active help systems; critiquing systems; differential descriptions; user interface customization	end-user modifiability; tailorability; filtering; design-in-use

Adaptation Mechanism to Control Different Critiquing Rule Sets and Different Intervention Strategies



Adaptation Mechanisms in HFAs — Success or Failure?



Conclusion

- **the challenge is**

not only to create more functionality

but: to design usable, useful, learnable, memorizable, applications