CogSysIII Lecture 1: Introduction

*Human Computer Interaction*

*SS 2005*

Ute Schmid (lecture)
Emanuel Kitzelmann (practice)

Applied Computer Science, Bamberg University
Organization

Prerequisite: successful participation in CogSysI

Homepage of the course:

http://www.cogsys.wiai.uni-bamberg.de/teaching/

Practice:

- first part different aspects of practical computing, including text processing with LaTeX
- afterwards some assignments involve application of modeling systems and evaluation of HCI systems
Course Objective

- Techniques for design, implementation, and evaluation of interactive computer systems
- introduced in the context of different areas of application
- special regard to characteristics of human information processing
- core topics: psychological concepts for HCI design, cognitive architectures, and empirical methods for the evaluation of software systems
- further topics: user modeling, user adaptivity, enduser programming, anthropomorphic agents, tutor systems
Resources

- Text Books
  - and many more

- Journals and Conferences:
  - Conference on Human Factors in Computing Systems (CHI)
  - Human-Computer Interaction
  - and many more
  - see: http://www.hcibib.org/
Human-computer interaction (HCI) is:

concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them. (ACM SIGCHI, 1992, p.6)

Increasingly, more application areas, more technologies and more issues to consider when designing ’interfaces’!
Development of Interfaces

- 50s – Interface at the hardware level for engineers - switch panels
- 60-70s – interface at the programming level - COBOL, FORTRAN
- 70-90s – Interface at the terminal level - command languages
- 80s – Interface at the interaction dialogue level - GUIs, multimedia
- 90s – Interface at the work setting - networked systems, groupware
- 00s – Interface becomes pervasive RF tags, Bluetooth technology, mobile devices, consumer electronics, interactive screens, embedded technology
Consequences for HCI

- Current techniques make it easy to compose GUIs
- Danger 1: not enough regard for careful specification and design of underlying algorithms
  Good software design is prerequisite for HCI!
- Danger 2: overloading of the GUI with unnecessary features, just because the techniques are available
  Use only such features which are necessary to support the user!
Good and Poor Design

Jacques Carelman, “Coffeepot for Masochists”, Catalog of Unfindable Objects

- Support of functionality: bad
- Aesthetics: depends
Good and Poor Design cont.

- VCR: recording a different program than you wanted
- Voice mail in many organizations/hotels: input many numbers, lots of steps
- Taxi reservation: wait on phone until queue is smaller than 5
- Paper submission: first register as new user then get information about topics of interest, submission format etc.
- Internet shop: long video clip which cannot be interrupted as intro, slow and tedious procedure for filling out an online form – which often was not successful (company went bankrupt within a few month of going public)
Good and Poor Design cont.

“If you’d like to press 1, press 3.
If you’d like to press 3, press 8.
If you’d like to press 8, press 5...”

see http://www.baddesigns.com/ for more

→ Design Principles (Norman, 1988) and Usability Principles (Nielsen, 2001)
Design Activities

- Identifying needs and establishing requirements
- Developing alternative designs to meet those requirements
- Building interactive versions of the designs so that they can be communicated and assessed
- Evaluating what is being built throughout the process
Usability Goals

- **Effectivity** (how good is the system at doing what it is supposed to do)
- **Efficiency** (speed of information access, speed of doing what the user wants to do, easy to learn)
- **Safety** (protecting the user from dangerous conditions and undesirable situations)
- **Utility** (provides the right kind of functionality)
- **Learnability**
- **Memorability**
Putting 'Save' near 'Delete' and/or 'Quit' might produce unwanted errors
The Ten-Minute Rule

Assessing whether a system is easy to learn

Ten-minute-rule by Nelson (1980)

A computer system for architects is not expected to teach architecture. Quite the reverse: the ten-minute rule requires that what an architect already knows be helpful in learning to use the system. (Rubinstein & Hersh, 1984)

Ten-minute rule is inappropriate for complex systems
  - Flight support system for pilots
  - Word processor (diverse functionalities)
  - Video game (high level of skills)
User Experience Goals

- Satisfying, Enjoyable, Entertaining
- Helpful, Motivating, Supportive of Creativity
- Aesthetically pleasing
- ...

- How does the system feel like to the user
- Explicating the nature of user experience in subjective terms
- Trade-offs between usability and user experience goals:
  a process control system should be safe! (might be incompatible with being fun)
Norman’s Design Principles

- Visibility
- Feedback
- Constraints
- Mapping
- Consistency
- Affordance
Visibility and Feedback

Visibility:

- e.g.: controls for different operations in a car are visible (indicators, horn, hazard warning lights) and relationship between their positions make it easy for the driver to find the appropriate control for the task at hand
- The more visible functions are, the more likely users will be able to know what to do next

Feedback

- Sending back information about what action has been performed and what has been accomplished
- Audio, tactile, verbal, visual, and combinations
Constraints

- Physical: restriction of the movement of things (e.g., a disk can be placed into a drive only in one way)
- Logical: relying on people’s common sense reasoning about actions and consequences (e.g., disabling and shadowing menu options which are currently not executable)
- Cultural: exploiting learned conventions (e.g., using red for warning)
Mapping

- Relationship between controls and their effects in the world
- e.g., up and down arrows representing up/down movement of cursor
- Mapping of relative positions of controls is also of importance

![Diagram showing natural and unnatural mappings of control movements]
Consistency

- Similar operations for similar tasks (e.g., always left mouse button for selecting items)
- Similar elements for similar tasks
- Placement of related operations in a common menu (categorization)
- External Consistency: be consistent with what people do in the world
- Internal Consistency: be consistent within the system
- Problem with complex systems with thousands of different operations
- E.g., “sort” in Word is not in the “Tools” menu but in the “Table” menu
- In relation: principle of least astonishment
Principle of least astonishment

In user interface design, programming language design, and ergonomics: when two elements of an interface conflict or are ambiguous, the behaviour should be that which will least surprise the human user or programmer at the time the conflict arises, because the least surprising behavior will usually be the correct one.

Example: a user interface may have the behaviour that pressing Control-Q causes the program to quit. The same user interface may have a facility for recording macros, or sequence of keystrokes to be played back later, intended to be able to control all aspects of the program. The user may want to record a keystroke sequence with Control-Q as part (most likely the last part) of the macro. The principle of least surprise says that pressing Control-Q while recording a macro should not quit the program (which would surprise the user), but rather to record the keystroke.
Affordance

- Attribute of an object that allows people how to use it
- e.g., door handle invites pulling; cup handle invites grasping; mouse button invites pushing;
- Icons should be designed to afford clicking, scroll bars to afford moving
- “I put an affordance there”, a participant would say, “I wonder if the object affords clicking ...” affordance this, affordance that. And no data, just opinion. Yikes! What had I unleashed upon the world? (Norman, 1999)
Meeting Design/Usability Goals

- System design using software engineering methods to meet requirements of functionality, performance quality, safety (not topic of this lecture)

- Requisits for HCI design:
  - Knowledge about the system domain, general aspects of human information processing, special aspects of the kind of users which will use the system
  - Formulation of targets for usability goals
  - Methods to support system analysis and synthesis
System analysis

- How is the system likely to affect the user's activities?
- Can things be done faster with fewer errors?
- Build a model of the performed activities
- Derive performance measures
- Make usability assessment
HCI Methods

- Problem definition
- User study
- Usability analysis
- Controlled experiments
- Empirical evaluation
- Use of guidelines
- ...
Contributing Disciplins

**Academic Disciplines**
- Ergonomics
- Psychology
- Cognitive Science
- Computer Science
- Software Engineering
- Social Sciences (Sociology, Anthropology)

**Design Practices**
- Graphic Design
- Product Design
- Artist Design
- Industrial Design
- Film Industry
- Information Systems
- Human Factors
- HCI
- Cognitive Engineering
- Cognitive Ergonomics
- Computer–Supported Cooperative Work

**Interdisciplinary Fields**