CogSys-HCI Lecture 1: Introduction

Human Computer Interaction

Ute Schmid (lecture and practice)

Applied Computer Science, Bamberg University

last change October 16, 2007
Organization

- Prerequisite: successful participation in CogSys-IA or equivalent
- Homepage of the course: [http://www.cogsys.wiai.uni-bamberg.de/teaching/](http://www.cogsys.wiai.uni-bamberg.de/teaching/)
- Practice: conducting a small empirical study
Course Objective

- Problems of and techniques for human-computer-interaction

- Core topics: characteristics of human information processing, psychological concepts for HCI design, cognitive architectures and cognitive modeling, empirical methods for the evaluation of software systems and usability studies

- Further topics: user adaptivity, enduser programming, anthropomorphic agents, tutor systems, social software, ...

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/
The HCI Challenge

*Human computer interaction is a difficult endeavor with glorious rewards.*

Designing interactive systems to be

- effective
- efficient
- easy to use
- enjoyable to use

so that people and society may realize the benefits of computation-based devices
Problem

Interleaved constraints with sometimes necessary trade-offs

- human
- machine
- algorithmic
- task
- social
- aesthetic
- economic
Reward

- Digital libraries where scholars can find and turn pages of medieval manuscripts located far away
- Medical instruments allowing complex neurosurgical operations
- Virtual worlds for entertainment and social interaction
- Responsive and efficient government services
- Smart phones
- ...
Definition of HCI

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
HCI as Subfield of Computer Science

- Computer science mainly concerned with the study of algorithms (sound, complete, efficient algorithms are a necessary condition for working systems)
- But: Human users and their contexts are major components of the design problem and cannot be wished away simply because they are complex to address
- Largest part of program code typically deals with interaction
The Science of HCI

- Domain of HCI is much more complex and fuzzy than classical computer science
- How to go beyond platitudes like “know the user”?
- Providing generative design principles
- Focus on both analytic and implementation methods:
  - Practitioners who can only evaluate but not design and build are under a handicap
  - Builders who cannot reason analytical about the systems they build and who do not understand human information processing or social contexts of their design are under a handicap
Examples for Bad Design

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

Design must support functionality!  \( \leftrightarrow \) Task analysis
Examples for Bad Design (2)

Menu-oriented word-processing package

- Save and delete both in menu “File” (makes sense)
- as adjacent items
- together with cursor controlled by a trackball this can be dangerous!
- Pop-up of a confirmation box, which looks very similar to the confirmation box for saving

→ violation of the usability goal “safety”
Examples for Bad Design (3)

- VCR: recording a different program than you wanted
- Voice mail in many organizations/hotels: input many numbers, lots of
- 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 succesful (company went bankrupt within a few month of going public)
- Car radios
- Mobile devices
- ...

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
Usability Goals

(Nielsen, 2001)

- 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
## Remember: Safe/Unsafe Menu

<table>
<thead>
<tr>
<th>File</th>
<th>Edit</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>New ...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open ...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Close</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Save</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Save As</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page Setup ...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print ...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>File</th>
<th>Edit</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>New ...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open ...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page Setup ...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print ...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Save</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Close</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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)
Fast Learnability vs. Efficient Use

- The ten-minute rule might be also not appropriate for expert computer users in general.

- Think about emacs + LaTeX vs. Word: where can you produce the desired outcome more efficiently?

- Think about a person writing one email per week vs. a professional writing 100 emails per day.

- Classic studies: keyboard shortcuts are more efficient than mouse input for expert users (Shneiderman, Designing the User Interface: Strategies for HCI, 1998).

- If keyboard shortcuts, then for everything (changing costs between mouse and keyboard).
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 inputs](image)
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
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
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**
A Closer Look at Interaction Design

Remember the 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
Users and Their Needs

Stakeholders: People or organizations who will be affected by the system and who have a direct or indirect influence on the system requirements (Kotonya and Sommerville, 1998)

Primary User: frequent and hands-on
Secondary User: occasional or through intermediary
Tertiary User: affected by introduction or influence purchase (Eason, 1987)

Frequently, the “formal client” who orders the system is very low on the list of those affected!

Identification of relevant stakeholders is crucial for user-centered development
User-Centered Approach

- Norman and Draper, 1986
- Focussing on the prospective users from the beginning and obtain real data about their needs
- → design based on empirical data and not on imagination
General and Specific Needs

- Interaction is a *human* activity: design must be consistent with physical and cognitive abilities and the social environment → take into account psychological research about perception, memory, etc.

- Particular expertise, responsibilities, training, working environments special demands because of impairments → type of user (e.g. passenger, telefon operator, ...) must be specified in problem/requirement definition

- Problem: for new inventions it is difficult to define users’ needs and representative tasks
Example Stakeholders

Who are the stakeholders?

- Suppliers
- Local shop owners

Check-out operators

Managers and owners

Customers
Requirement Analysis and Specification

Categories:
- target audience and target platform
- features of the application
- responsiveness
- usability benchmarks (e.g. "such and such a task must be completed within 2 minutes")
- the choice of general look and feel guidelines

established through: a combination of analysis and gathering information from target users, through interviews, observation, focus groups, surveys

Often only written down in contract development context

Validation: analytical and empirical (testing prototypes)
Alternative Designs

- Human tendency to stick with something which works “good enough”
- The “mysterie” of design creativity
- Pragmatic way: looking at other, similar designs
- Evolution: e.g., typewriter to word processor
- Typically, many, sometimes conflicting, constraints
- Example: Software running under Windows must conform to Windows look and feel
- Choosing between alternatives: empirical and analytical evaluation of prototypes
  - External: Usability characteristics, Performance times, ...
  - Internal: Correctness, consistency, ...
Creativity and Analogy

- Analogy as the basic mechanism of creativity
- Cross-fertilization between different domains
- Example: Steam engine / water kettle

Key components of analogy

- Retrieval of a suitable source (bottleneck, what is suitable?, pitfall of superficial similarity)
- Mapping of source and target structure
- Transfer: Inference of missing structural characteristics, Adaptation of a solution
Lifecycle Models

- Showing how activities are related: lifecycle models
- Software lifecycle models: waterfall model, spiral model, RAD model
Waterfall Model

Traditional ‘waterfall’ lifecycle

- Requirements analysis
- Design
- Code
- Test
- Maintenance

- focus on intended functionality, no user involvement
Spiral Model

- Barry Boehm (1988)
- Interactive framework with risk analysis and prototyping as important features
- No user involvement
- Iterations to take account of risks in development, i.e., any kind of uncertainty in requirements (might be resolved by prototyping)
- Idea of iterations adopted for HCI design models
Spiral Model

Spiral Lifecycle model

Determine objectives, alternatives, constraints

Evaluate alternatives; identify, resolve risks

Risk analysis

Risk analysis

Risk analysis

Final Prototype

Prototype 3

Prototype 2

Operation Concepts

Simulations

models

benchmarks

Product design

Detailed design

Code

Unit test

Integration test

Acceptance test

Design validation & verification

Acceptance test

Integration test

Develop, verify next-level product

Plan next phase

Development plan

Integration and test plan

Service

Life cycle plan

Requirements plan

REVIEW

From cctr.umkc.edu/~kennethjuwng/spiral.htm
Spiral Model in Usability Engineering

- 4 phases:
  - Analysis
  - Design
  - Development
  - Tests

- 3 Levels:
  - Paper Model
  - Prototype
  - Final Product
Rapid Applications Development

- User-centered view
- Minimize risks caused by requirements changes
- Joint Application Development (JAD): workshops with users and developers to gather requirements
RAD

A Lifecycle for RAD
(Rapid Applications Development)
Paper Mock-Ups

- Results of requirement analysis are modelled on paper.
- Tests with endusers possible: give concrete tasks and measure time to perform and difficulties with interaction (cognitive effort).
- Typically, about 80% of difficulties can be detected in this early phase!
Prototypes

- Prototyping: essential element of an iterative design approach
- create, evaluate and refine until desired performance or usability is achieved
- Low-fidelity prototypes: extremely simple sketches
- High-fidelity prototypes: systems that contain nearly all the functionality of the final system
Lifecycle Models in HCI

- Star model (Hartson and Hix, 1989)
- Emerged from empirical studies of interface designers
- Top-down: Analysis, system’s view
- Bottom-up: Synthesis, user’s view
- No specific order of activities
The Star Model (Hartson and Hix, 1989)
Integration of eXtreme Programming and Usability Engineering

Xtreme Programming (Kent Becker):
- A not undisputed approach ...
- Combining best practices of software development in an extreme way such that individual weaknesses are compensated
- Permanent code reviews by pairs of programmers who switch tasks every half day
- Permanent tests with at least one enduser
- YAGNI (you aren’t gonna need it): simple design
Monthly plan adaptation with stakeholders
Code releases approx. every three months
No specialization of developers, code is understood and refactored by all developers
Working along "user stories": one non-divisible feature which is of use for the end user
Development of test cases for each user story
Usability is one feature among others
Problem: calculating a budget in advance is nearly impossible with iterative approaches (no complete requirement/functional specification in advance)
Each XP iteration: increasing set of finished user stories and decreasing set of unfinished stories

Usability engineering in each iteration: use paper mock-ups for not implemented stories and user studies (e.g. analysis of think aloud and video protocols) for implemented stories

Getting rid of one crucial problem of usability engineering: taking into account too many details too early
Objectives of This Lecture

- Closer look to human information processing as the most important constraint for HCI
- Introduction of principle methods of empirical research as the most important way to assess the usability of a system