Human-Computer Interaction is a discipline concerned with **Design, Evaluation, and Implementation** of **interactive computer systems** for human use, and with the study of the major phenomena surrounding them.

- **Computer scientists**
  - Modeling, specifying & analyzing interaction
- **Psychologists**
  - User as perceiver, thinker
- **Software engineer**
  - Interaction design is part of overall system development
  - Knowledge of users, tasks necessary to capture and understand requirements
- **Linguists and philosophers**
  - Interaction as communicative and conceptual activity
- **Artificial intelligence researchers**
  - Interactive systems exhibiting (or simulating) intelligent behavior
- **Ergonomists**
  - Interaction is physical as well as conceptual
- **Sociologists and anthropologists**
  - Interaction as social activity

- **Daily-Use Scenarios**
  - Main actions/goals that user needs to perform
  - Scenarios that need the most robust interaction support
- **Necessary-Use Scenarios**
  - Other actions that must be performed
  - May not be that frequently
- **Edge-Case Scenario**
  - Rarely used, but must be included (like configuration)

Four goals for Usability Requirements (UR):

1. Ascertain the **user's needs**

- Find out what **tasks** & **subtasks** must be carried out
- Include all tasks
  - [Daily Use] **Common tasks** are easy to identify
  - [Necessary Use] **Not very frequent** actions
  - [Edge Case] Occasional/**exceptional**/rarely-used tasks for emergency, repairing tasks for errors, configuration, etc.
- Functionality must **match user's need**; else users will reject or underutilize the product

2. Ensure **proper reliability**

- Actions must **function as specified**
- Data on screen must **reflect actual database value**
- Be sensitive to the user's **sense of mistrust**
  - Only a few errors and the user will not use your system.
  - Users trust is fragile! This is even more true in real world market.
- The system should **be available** as often as possible
  - No offline or long waits
- The system must **not introduce errors**
- Ensure the user's **privacy and data security** by protecting against unwarranted access, destruction of data, and malicious tampering

3. Promote appropriate **standardization, integration, consistency, and portability**

- **Standardization**: use pre-existing industry standards where they exist to aid learning and avoid errors
  - **Slight differences** not only increase learning times but also lead to annoying and dangerous errors
  - **Gross differences** require substantial retraining and burden users, e.g. incompatible storage formats, software versions, etc. cause frustration, inefficiency and delay.
- **Integration**: the product should be able to run across different software tools and packages (e.g. Unix)
- **Consistency**:
  - Compatibility across different product versions
  - Compatibility with related paper and other non-computer based systems
  - Use common action sequences, terms, units, colors, etc. within the program
- **Portability**:
  - Allow for the user to convert data across multiple software and hardware environments
  - Some UI building tools can generate user interfac code for Mac, Windows, Unix, etc.

4. Complete projects **on schedule** (and within budget)

Usability Measures

1. Time to learn

How long does it take for typical members of the community to learn relevant task?

2. Speed of performance

How long does it take to perform relevant benchmarks?

3. Rate of errors by users

How many and what kinds of errors are made during benchmark tasks?

4. Retention over time

How well do users maintain their knowledge after an hour, a day or a week?
Frequency of use and ease of learning help make for better user retention

5. Subjective satisfaction

How much did users like using various aspects of the interface? Allow for user feedback via interviews, free-form comments and satisfaction scales

Application Types and Motivation

1. Life-Critical Systems

Applications:
Air traffic control, nuclear reactors, power utilities, police & fire dispatch systems
Requirements:
reliability and effectiveness, error free performance
Not as important:
Cost, training time
user satisfaction: users are well motivated professionals

2. Industrial and commercial uses

Applications:
Banking, insurance, order entry, inventory management, reservation, billing system
Requirements:
Ease of learning is important to reduce training costs
Speed and error rates are relative to cost

Subjective satisfaction is fairly important to limit operator burnout
Speed of performance is important because of the number of transactions

3. Home and Entertainment

Applications:
Word processing, electronic mail, computer conferencing, and video game systems, educational packages, search engines, mobile device, etc.
Requirements:
Ease of learning, low error rates, and subjective satisfaction are paramount due to use is often discretionary and competition fierce
Interfaces must be **intuitive** with easy-to-use online help
Market competition often forces the need for **low cost**
Choosing functionality is difficult because the population has a wide range of both novice and expert users

4. Exploratory, creative, collaborative applications

Applications:
Web browsing, search engines, artist toolkits, architectural design, software development, music composition, and scientific modeling systems
Requirements:
Users may be knowledgeable in the task domain but **novices in the underlying computer concepts** (however, high in their expectations on the usability)
Due to exploratory nature, these systems are hard to design and evaluate
Designers should pursue the goal: **computer/UI should vanish** so that the user can be absorbed in their task domains

5. Social/Technological applications

Applications (usually used by govt.):
Voting, health support, identity verification, crime reporting
Requirements:
Trust, privacy, responsibility, and security are issues
Verifiable sources and status feedback are important
Diverse levels of users: Ease of **learning** for novices and **feedback** to build trust
Administrators need tools to detect unusual patterns of usage
Difficulties:
Huge systems that evolve over time

Universal Usability

1. Physical ability and workplace
   Dynamic measures (reach distance, strength, speed), screen-brightness (a knob to control), vision, touch, hand control, hearing. These physical abilities influence the elements of the interactive-system design, playing a prominent role in the design of workplace.
2. Cognitive and Perceptual abilities
3. Personality differences
4. Cultural and International Diversity
5. Users with Disabilities
6. Considerations for Elderly
7. Considerations for Children

Theories: **High-level theories** that describes objects and actions with consistent terminology to

support communication & teaching. Can be used to predict performance, errors, understanding, satisfaction of user.
Principles: Middle-level practices that can be applied to different guidelines, analyzing and comparing design alternative. User classification, "8 golden rules of UI design", etc.
Guidelines: Design-level practices and rules that make for good and consistent design (some based on theory). Examples: Apples guidelines on for UIs.

Why have guidelines, principles, and theories?

Help keep our UI designs focused and consistent.
Help avoid and remedy mistakes (e.g., cluttered display, tedious procedures, inadequate functionalities, etc.)
Provide theories and high-level description of interaction and design

Guidelines: best practices, from experience, good for starting point for all projects involving a UI, a shared language.
Guidelines consist of:

Rules

- Provides cures for design problems
- Provides cautions for potential danger
- Reminders based on experience

Examples

- Give details on how a design must be performed
- Style, color usage, window appearance, etc
- Interaction usage (when to use check-boxes, when to user buttons, etc)
  - These guidelines could be based on experience
  - Ex: developers found that users preferred a list+slider over a pull-down menu for a long list of choices
- Require all developers to follow the guidelines

Document

- Any serious large-scale UI design should have a "Guideline Document"
- Provides a "Shared language" that developers and customers can use
- Allows consistency within a design team
  - Especially a large-team working on a large project
- Guidelines document is not trivial
  - Think of the effort needed to specify "everything" pertaining to the UI, but it is necessary
- Can be used to specify many aspects of an interface:
  - Input and output formats
  - Action sequences
  - Terminology
  - Hardware devices/platforms
  - Provide Examples & counterexamples
- Pros:
  - Builds upon (good previous) experience
  - Continued improvements
- Cons:
  - Too specific

- Hard to innovate
- Not applicable/realistic to the situation
- Hard to apply
- What do you do when having an exception?

Guidelines for Disabled

- WWW Consortium adopted these guidelines for designing web pages for disabled
  - **Text** equivalent for every non-text element (images, image map, animations, applets, ascii art, frames, scripts, bullets, sounds, audio, video, etc.)
  - Any **time-based multimedia**, provide equivalent synchronized alternatives (captions, descriptions)
  - All **color** info can be captured by users without color – from context or markup
  - Title each frame, facilitating frame identification and navigation
- Enables screen readers or other technologies to have multiple methods to obtain the webpage info

Display Organization guidelines

- Consistency of data display
  - Terminology, abbrev., formats, colors, grammar, capitalization should be consistent!
- Efficient information assimilation by the user
  - Familiar format
  - Related to tasks at hand
  - e.g. spacing, formatting, labels, units/measurements, numbers of decimal points
- Minimal memory load on the user
  - Minimal carry information over from on screen to another
  - Require fewer actions
  - TAB key to move to next entry field vs. having to use the mouse
  - Labels and common formats should be provided for novice (Ex. SSN/phone #)
- Compatibility of data display with data entry
  - Entering data should look similar to the eventual viewing of the data
- Flexibility for user control of data display
  - User control for information display (e.g. sorting, ordering of columns and rows)

Data Entry guidelines

- Data entry can occupy a substantial portion of user's time and be the source of frustrating and potentially dangerous errors
- **Consistency** of data-entry transactions – similar sequence of actions, delimiters, abbrev.
- **Minimal input actions** by user
  - fewer actions = greater productivity and less error
  - Ex. single key-stroke vs. mouse selection, vs. typing is typically better
  - Ex. Command line vs. GUI
  - Too much hand movement is not good. Ex. Experts prefer to type 6-8 characters instead of moving a mouse, joystick, etc.
  - Avoid redundant data entry (waste of time, perceived effort, increased error). System should aid but allow overriding
- **Minimal memory load**
  - Don't use codes, complex syntactic strings.
    - Ex. Don't user codes for a country on a web form
  - Provide "selection" from a list

        - don't need to memorize choices
- **Compatibility** of data entry with data **display**
    - should matches display capability
- **Flexibility for user control** – Experienced vs. novice
    - Experienced may want "hot-keys", novice doesn't
    - All you Ctrl-F file people are happy!
    - Should be used cautiously, since it goes against consistency

Principles

- More fundamental, widely applicable, and enduring than guidelines
- Fundamental principles for all UI
    - **Determine user's skill levels** / Spiral design strategy
        - "Know thy user"
        - Start with population profile
            - Age, gender, physical and cognitive abilities, education...
        - Design goals based on skill level
            - Novice
                - Inexperience, anxiety
                - Restrict vocabulary, providing help, small number of actions, feedback, documents
            - Intermittent
                - Understand concepts and interface basics
                - May have difficulties in retaining the structure of menus, or location of features
                - Consistent sequences of actions, meaningful feedback, guides to frequent pattern of usage, protection from danger, help
            - Expert
                - Thoroughly familiar with task and interface
                - Goal is efficiency
                - Rapid response time, brief feedback, shortcuts, accelerator
            - Multi-layer strategy (level-structured, or spiral)
    - Identify the tasks of the application
        - Task analysis, decomposing high level tasks, relative task frequencies
        - User-need assessment

- **Five primary interaction styles**
    - Direct Manipulation
        - Visual representations, metaphor
        - Appealing to novices
        - Pro: fast feedback, easy to understand and retain, encourage exploration
        - Con: hard to program, interaction devices are harder to design or modify
    - Menu Selection
        - Select from a list of items
        - Novices and intermittent users
        - Frequent users if the display and selection mechanisms are rapid.
        - Pro: no memorization, few actions, clear structure, validity and consistency
        - Con: not easy to make actions understandable, careful task analysis
    - Form-fill in
        - Data entry into fields
        - Intermittent and frequent users

- Pro: rapid, for more advanced users, tools available for forms
- Con: must understand label and request format, respond to errors, training

- Command Language
  - For expert frequent users
  - Pro: feeling of in control, rapid, easy histories and macros, flexibility
  - Con: high error rates, training, poor retention, hard to create error messages
- Natural Language
  - Computer respond to arbitrary natural language
  - Pro: easy to learn
  - Con: unpredictable, requires clarification dialogues.

Eight golden rules of interface design

1. Strive for consistency
Consequence of actions, terminology(popup, menu, help), visual layout(colour, fonts)
Exception: confirmation for delete, password
2. Cater to universal usability
Needs for a diverse group, plasticity(transformation of content on any display), novice to expert, disabled
3. Offer informative feedback
Feedback for every action, common task modest feedback, uncommon/error substantial feedback, visual approaches
4. Design dialogues to yield closure
Action sequence should have beginning, middle and end; sense of accomplishment
5. Permit easy reversal of actions
Trash can relieves anxiety, history, undo, let the user know that they can reverse
6. Support internal locus of control
User in charge, rapid response, avoid acausality, make users initiators.
7. Reduce short term memory
User can remember 7+-2 chunks of information, keep display simple, training, online help
8. Prevent errors
Limit errors(gray out, no alphabets in numeric field), detect errors, simple constructive specific instructions
Better error messages, helps fix current errors, helps reduce future errors, increases satisfaction, specific constructive positive
Reduce chances of error, organizing info menu screen, distinctive choices of command and menu, state of interface, consistency of actions
Correct actions
Complete Sequences
Universal Usability(large buttons, readability)

Theory example: **Stages of Action**
1. Forming the goal
2. Forming the intention
3. Specifying the action
4. Executing the action
5. Perceiving the system state
6. Interpreting the system state
7. Evaluating the outcome

- Context of cycles of action and evaluation.
- **Gulf of execution**: Mismatch between the user's intentions and the allowable actions
- **Gulf of evaluation**: Mismatch between the system's representation and the users' expectations

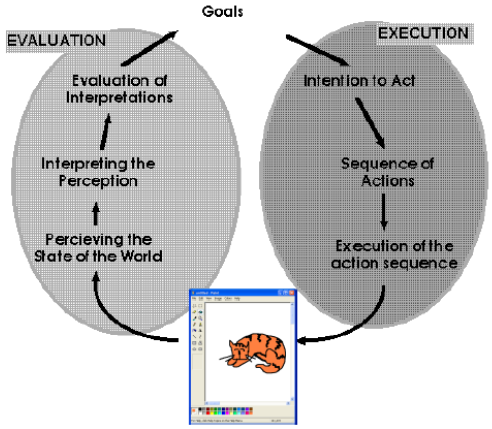
Issues in Execution and evaluation

- **Gulf of execution**
  - **Mismatch between the user's intentions and the allowable actions**
- **Gulf of evaluation**
  - **Mismatch between the system's representation and the users' expectations**

Gulf of evaluation error happens here.

Could be a real error in the system, or mismatch between what the user expected.

For example, imagine you want to paint the cat's head orange, but the program paints the entire image orange!



Gulf of execution error happens on this side.

User wants to paint the cat's head stripped.

There is no corresponding action to perform this.

The user is lost, confused, and makes errors.

59

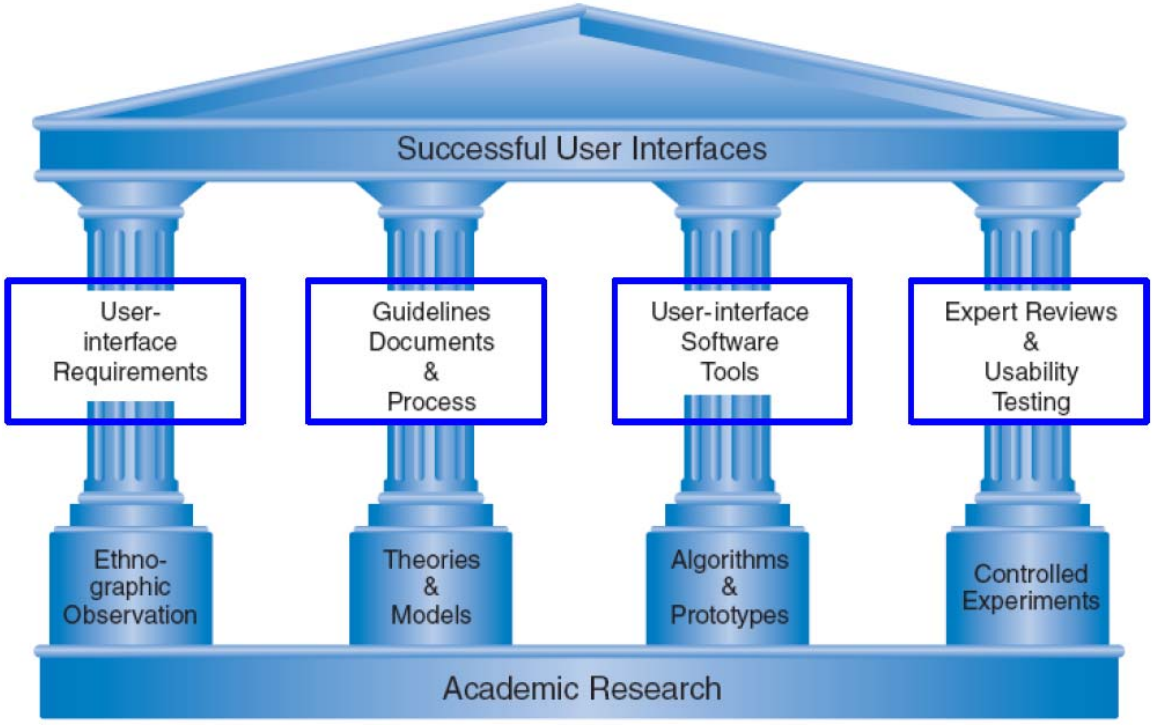
Four critical points where user failures can occur

- Users can form an inadequate goal
- Might not find the correct interface object because of an incomprehensible label or icon
- May not know how to specify or execute a desired action
- May receive inappropriate or misleading feedback

To avoid "gulf" errors, the following is proposed:

Four principles of good design

- Have a good conceptual model with a consistent system image
- State and the action alternatives should be visible
- Interface should include good mappings that reveal the relationships between stages
- User should receive continuous feedback



Ethnographic observation

- A **qualitative** method aimed to learn and understand cultural phenomena which reflect the knowledge and system of meanings guiding the life of a **cultural group**
- Observe "real world" users in their home or office environment
- Sends observers to people homes and offices to quietly observe user habits and methods
- Preparation
  - Understand organization policies and work culture.
  - Familiarize yourself with the system and its history.
  - Set initial goals and prepare questions.
  - Gain access and permission to observe/interview.
- Field Study
  - Establish rapport with managers and users.
  - Observe/interview users in their workplace and collect subjective/objective quantitative/qualitative data.
  - Follow any leads that emerge from the visits.
- Analysis
  - Compile the collected data in numerical, textual, and multimedia databases.
  - Quantify data and compile statistics.
  - Reduce and interpret the data.
  - Refine the goals and the process used.
- Reporting
  - Consider multiple audiences and goals.
  - Prepare a report and present the findings
- Pros:
  - more accurate information about tasks
  - more opportunity for users to influence design decisions
  - a sense of participation that builds users' ego investment in successful implementation

- potential for increased user acceptance of final system
- Cons:
  - be more costly
  - lengthen the implementation period
  - build antagonism with people not involved or whose suggestions rejected
  - force designers to compromise their design to satisfy incompetent participants

Prototyping

- UI Prototype ("mock up")
  - Does not have to be functional
  - Simply design the buttons, menus, etc
  - Can well be a hand drawn picture
- Develop prototypes early
  - Very difficult and costly to make changes near the end of a project
- Provides a "vision" for both develops and clients

Evaluating

Troubling aspect of **testing**: uncertainty remains even after exhaustive testing by multiple methods.

- Perfection is not possible in complex human endeavors, so must continue assessing and repairing problems during lifecycle of interface
- Decision must be made about completing prototype testing and delivering the product, even though problems may continue to be found
- Most testing methods will account fro normal usage, but performance in unpredictable situations with high levels of input such as nuclear reactor control, is extremely hard to test.

Expert reviews entail one-half day to one week effort, although a lengthy training period may sometimes be required to explain the task domain or operational procedures
There are a variety of expert review methods to chose from:

- Heuristic evaluation
  - Experts reviewers personal critic
- Guidelines review
  - Make sure UI adheres to established guidelines
- Consistency inspection
  - Check for consistency through-out interface
- Cognitive walkthrough
  - Simulate performing certain tasks
- Formal usability inspection
  - UI designers defend their choices against a "hostile" expert

Usability Testing and Laboratories

- The emergence of usability testing and laboratories since the early 1980s.
- Usability testing not only sped up many projects but has also produced dramatic cost savings.
  - Traditional managers and developers resisted at first, saying the usability testing may take time and resource away from development
  - They changed their mind when experience grew and successful projects gave credit to the testing process.
- The movement towards usability testing stimulated the construction of usability laboratories.

Usability Lab

- A typical modest usability lab would have two 10 by 10 foot areas, one for the participants to do their work and another, separated by a half-silvered mirror, for the testers and observers
- Participants should be chosen to represent the intended user communities, with attention to
- background in computing, experience with the task, motivation, education, and ability with the natural language used in the interface.
- Videotaping
- Think Aloud
- Paper mockups
  - Early usability study; inexpensive, rapid.
  - Flipping the (mockup of) screen displays to get reactions to wording, layout, etc.
- Discount usability testing
  - Only 3 to 6 test participants
- Competitive usability testing
  - Comparing new interface to previous versions or to similar products
- Universal usability testing
  - Diverse users, hardware/software platform, networks, etc.
- Field test and portable labs
  - New interface to work in realistic environments for a fixed trial period.
  - Test of new software or consumer products
- Remote usability testing
  - Tests online; less control over user behavior and observation of reaction
- Can-you-break-this tests
  - Game design: challenge energetic teenagers to beat new games, finding fatal flaws

Survey Instruments

- Written user surveys are a familiar, **inexpensive** and generally acceptable companion for usability tests and expert reviews.
- Keys to successful surveys
  - Clear **goals** in advance
  - Development of **focused items** that help attain the goals.
- Users could be asked for their subjective impressions about specific aspects of the interface such as the representation of:
  - task domain objects and actions
  - syntax of inputs and design of displays.
- Other goals would be to ascertain
  - users **background** (age, gender, origins, education, income)
  - **experience** with computers (specific applications or software packages, length of time, depth of knowledge)
  - job responsibilities (decision-making influence, managerial roles, motivation)
  - **personality** style (introvert vs. extrovert, risk taking vs. risk aversive, early vs. late adopter, systematic vs. opportunistic)
  - reasons for not using an interface (inadequate services, too complex, too slow)
  - familiarity with **features** (printing, macros, shortcuts, tutorials)
  - their feeling state after using an interface (confused vs. clear, frustrated vs. in-control, bored vs. excited).

Acceptance Test

- For large implementation projects, the customer or manager usually sets **objective** and **measurable goals** for hardware and software performance.
- If the completed product fails to meet these acceptance criteria, the system must be reworked until success is demonstrated.
- Rather than the vague and misleading criterion of "user friendly," **measurable criteria** for the user interface can be established for the following:
  - Time to learn specific functions
  - Speed of task performance
  - Rate of errors by users
  - Human retention of commands over time
  - Subjective user satisfaction

Evaluation During Active Use

- Successful active use requires **constant attention** from dedicated managers, user-services personnel, and maintenance staff.
- Perfection is not attainable, but percentage **improvements** are possible.
- Interviews
- Data logging
- Online/telephone consultants
- Email trouble reporting
- Discussion group/news group