UNIVERSITY OF CRETE
FACULTY OF SCIENCES AND ENGINEERING
COMPUTER SCIENCE DEPARTMENT

COURSE CS-364 (OPTIONAL)
HUMAN – COMPUTER INTERACTION

HCI Design
Overview

- What is Design, Design Model, Design Process?
  - Definitions of design, term disambiguation
  - Brief history of the development process and its evolution
    - Waterfall model inherited from engineering
    - The formality gap problem
    - Alternatives and Iterative Design
    - Who Designs what? (GUI Programmer vs. GUI / UX Designer)

- User Centred Design and the Design Thinking Process
  - HCI / Designer’s perspective
  - Designing Interactions
  - The 5-step Design Thinking Process

- User-Centred Design Challenges
  - Real World Problems
  - Benefits and Costs
What is Design

“Design” in Dictionaries:

- “A plan or drawing produced to show the look and function or workings of a building, garment, or other object before it is made” – Oxford
- “To create, fashion, execute, or construct according to plan” – Merriam – Webster
- “The creation of a plan or convention for the construction of an object, system or measurable human interaction” – Wikipedia
A **software process** (also known as **software methodology**) is a set of related activities that leads to the production of the software.

A **software process model** is a simplified representation of a software process. Each model represents a process from a specific perspective.

- Later on some general process models are presented. These generic models are abstractions of the process that can be used to explain different approaches to the software development. They can be adapted and extended to create more specific processes.

- Some methodologies are sometimes known as **software development life cycle (SDLC) methodologies**, though this term could also be used more generally to refer to any methodology.
There are two fundamental viewpoints of the design process

First, from the perspective of a project manager
- A part of the software development process, design is one link in the chain of product development
- Designers are expected to create a “good” design to be developed in due course and time

Second, from the perspective of the design team
- What do we design and how?
- But first, what is the design team actually designing in software?

From command-line interfaces to the GUI to today, through the years, a designer had different skills and duties
In the beginning (that is, at least until the 80s) software design meant system architecture design.

The user interface is a product of software engineers and little thought is given to the experience and knowledge of the user.

- The user is expected to learn how to use the product
- Command line interfaces dominate
- Large business applications are just batch-processing systems crunching data, not really interactive

The design process is handled by a systems analyst working with software engineers under the supervision of the project manager.

The development process is modelled after the Waterfall Model.
The basic software process – The Waterfall Model

- Requirements Specification
- Architectural Design
- Detailed Design
- Coding & Unit Testing
- Integration & Testing
- Operation & Maintenance

The life cycle activities reminisce a waterfall, in which each activity naturally leads into the next.
Inherited by the traditional engineering industry, the waterfall model was favored by project managers as it divides the development process into neat, distinct and manageable sections:
- Good for setting and monitoring deadlines
- Produces a rich amount of tangible deliverables (documents etc.)
- It won’t really go away for large projects as it is convenient for drawing up detailed contracts

But it is not ideal nor realistic for the software development process, especially for the production of user-centric software:
- Why? The formality gap
The Formality Gap

Real-world requirements and constraints

The formality gap

Real World

Structured Design

Requirements Specification
Architectural Design
Detailed Design
Coding & Unit Testing
Integration & Testing
Operation & Maintenance
Interactive Systems & the Software Lifecycle

- After the GUI (from the Mac and MS Windows onwards), modern systems developed today are highly interactive and it is vital to the success of any product that it is not only easy to operate but also efficient and fun to use.

- Users had to be involved in the process before product launch to make sure they are satisfied.

- New models were developed:
  - The Prototyping Model
  - The Incremental Development Model
  - The Spiral Model
  - Iterative Design Model
The Prototyping Model

- **Establish objectives**: The objectives of the prototype should be made explicit from the start of the process. Is it to validate system requirements, or demonstrate feasibility, etc.

- **Define prototype functionality**: Decide what are the inputs and the expected output from a prototype. To reduce the prototyping costs and accelerate the delivery schedule, you may ignore some functionality, such as response time and memory utilization unless they are relevant to the objective of the prototype.

- **Develop the prototype**: The initial prototype is developed that includes only user interfaces.

- **Evaluate the prototype**: Once the users are trained to use the prototype, they then discover requirements errors. Using the feedback both the specifications and the prototype can be improved. If changes are introduced, then a repeat of steps 3 and 4 may be needed.
Incremental development is based on the idea of developing an initial implementation, exposing this to user feedback, and evolving it through several versions until an acceptable system has been developed.

The activities of a process are not separated but interleaved with feedback involved across those activities.
The spiral model is a risk-driven model where the process is represented as spiral rather than a sequence of activities.

It was designed to include the best features from the waterfall and prototyping models, and introduces a new component; risk-assessment.

Each loop (from *review* till *service*) in the spiral represents a phase. Thus the first loop might be concerned with system feasibility, the next loop might be concerned with the requirements definition, the next loop with system design, and so on.
The core of the process lies in the central loop, which is characterized by extensive use of prototyping methods and testing with users or by performing heuristic evaluation.

At the beginning of the process, the loop would resume at the requirements engineering phase and gradually shift towards the design & implementation phases.
Requirements for an interactive system cannot be completely specified from the beginning of the life cycle.

The only way to be sure about some features is to build them and test them on real users.

This is the essence of iterative design:

- it tries to overcome the inherent problems of incomplete requirements specification by cycling through several designs, incrementally improving upon the final product with each pass.
The software process models presented are all significant improvements over the monolithic Waterfall Model.

Above all, they introduced user feedback into the loop from early on.

More models exist, such as Agile development or reuse-oriented software engineering.

However, there is still a significant problem:
- Who is the design team and what are they designing?
- Graphic designer? GUI programmer? GUI / Interaction / UX designer?
- A software engineer has a very different approach to the system than an interaction designer.
The design team and the skills required also evolved along with the interactivity level of software systems.

- GUI Designer → Interface Designer → Interaction or User Experience Designer

This change in job title and job description is also reflected in the design processes employed today.

- So, what are we designing and how?
## Comparing skills

<table>
<thead>
<tr>
<th>Role</th>
<th>Skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUI designer</td>
<td>• Task analysis, conceptual design</td>
</tr>
<tr>
<td></td>
<td>• Interaction design: context, high-level organization, task flow</td>
</tr>
<tr>
<td></td>
<td>• UI design: input and output</td>
</tr>
<tr>
<td></td>
<td>• Real-time responsiveness goals</td>
</tr>
<tr>
<td></td>
<td>• Usability evaluation, usability testing</td>
</tr>
<tr>
<td></td>
<td>• Assessing conformance to usability standards</td>
</tr>
<tr>
<td></td>
<td>• Layout</td>
</tr>
<tr>
<td>Graphic designer</td>
<td>• Creating recognizable images, intuitive symbols</td>
</tr>
<tr>
<td></td>
<td>• Production values, aesthetic appeal, brand awareness</td>
</tr>
<tr>
<td></td>
<td>• Making best use of the available display medium</td>
</tr>
<tr>
<td></td>
<td>• Conveying function graphically</td>
</tr>
<tr>
<td></td>
<td>• Layout, visual hierarchy</td>
</tr>
<tr>
<td></td>
<td>• Visual consistency</td>
</tr>
<tr>
<td>GUI programmer</td>
<td>• Dynamic prototypes</td>
</tr>
<tr>
<td></td>
<td>• Implementing specified design: internal architecture, programming</td>
</tr>
<tr>
<td></td>
<td>• Knowledge of GUI toolkit</td>
</tr>
<tr>
<td></td>
<td>• Maximizing performance, meeting real-time goals</td>
</tr>
<tr>
<td></td>
<td>• Assessing and explaining technical constraints, costs, and risks</td>
</tr>
</tbody>
</table>
Clarification of job skills

- GUI programmers are not GUI designers
  - Lack of GUI Design experience
  - Designing GUI to fit toolkit
  - Poor team-working skills

- Graphic designers are not GUI designers
  - Managers often think the user interface is just the appearance of an application, so they hire appearance experts—graphic designers—to design it
  - The result of such a decision can be a product that demos well but supports user tasks poorly
Interface and Interaction Design vs. Software Engineering

- User interfaces are built of software

- This is the reason why many software engineers believe that the well-established techniques for developing software in general will apply as well to user interface development.

- These techniques do apply to user interface software, but not to designing what the software would implement - namely the interaction with a user.

- The human element, brought into the equation by the user is not a consideration of conventional software engineering as a discipline.
Because of the ‘human factor’, user interaction development represents a domain with its own special problems, requiring its own skills and its own special development processes and new communication requirements that emerge as a result.

Software engineering techniques do not transfer well to engineering of user interaction.

The same kind of concepts apply - requirements, specifications, design, metrics, evaluation, maintenance, documentation, life cycle - but for user interaction, each has to be done in a different way.
“HCI..?”

Slide 22

CS-364: Human – Computer Interaction
What are we designing? (1/2)

- We design interactive products
  - This means naturally software systems, including websites
  - It also means many kinds of electronic devices: they are interactive products and involve the interaction of the user with a (“mini”) computer, most including a display
  - Example: a smartphone, a camera menu or an iPod

- We do not design JUST interfaces. We are also designing interactions, a series of things the user does to reach a goal
  - It is not just the thing, but what you do with it
  - The camera menu is only a part of the interaction. Other interaction involves looking through viewfinder or plugging it later to a computer

- However, it is not just the immediate interactions, like the key press or the menu selection or even the “delete” sequence
  - We design the entire flow of the interaction of the user and the device
  - The flow, in which the user is flowing through “photographs” and not flowing through a “camera menu”
The Stapler example:

- Everyone in the office has a personal manual stapler. The interaction flow is like this:
  - Write something, print it out, staple it together. → write, print, staple, write, print staple, ...

- A new electronic stapler is bought in the office, that can staple 50 and 100 sheets together in one go, located in a common area of the office. The interaction flow changes:
  - You write something, print it... then don’t bother going to the stapler until some work has piled up and make it worth the effort
  - So → Write, print, write, print, write, print... staple, staple, staple

- One change in the stapling technology leads to a huge change in the interaction

- And perhaps walking by a colleague’s offices if they have some piles as well, one can offer to staple them for them
  - The individual interaction has become a collaborative interaction

- With the interaction change come the new errors
  - A way is needed to make sure each printout belongs together in the pile in order to avoid stapling together the wrong sheets. An error that would not have happened in the old interaction flow where the user staples each job after it has been printed
We are also designing **interventions**

- Not just artifacts

Not just the system, but also

- Documentation, manuals, tutorials produced might change the way people behave with a system
- Design is about changing things
  - 60s printer story (see next slide)
- What we say and do is also an intervention
  - Ratners jewelry (see next slide)
Designing interventions (2/2)

- 60s printer story
  - A person was called to design a new printing solution. The office already had a printer, not computer controlled. Most of the staff were women, who cared about their appearance and they avoided their current setup as it was very dirty (old machine, inks and stuff). His budget was in the equivalent of tens thousands of pounds. He responded with “would you like to give me a budget of 200 pounds a year?”. They of course accepted. Instead of designing a new solution, the designer simply bought whitecoats. Suddenly, the machine that produced nothing leapt in productivity because the whole staff was using the current setup regularly.
  - The moral of the story is that of change. We are changing things. We intervene in an environment and change it (hopefully for the better!).

- Example from the business world: Ratners Jewelry
  - A very successful franchise of low-cost jewelry in the 80s revolutionized the jewelry market by displaying fluorescent orange posters advertising cut price bargains and by offering low price ranges amidst the high street. Their success was turned to disaster after Ratner, in a private City function in the early 90s, described the company’s products as “total crap”. One word and the change was phenomenal, the company near collapsed (its value plummeted by around 500 million pounds).
  - Notorious incident (known as the Ratner effect) as an example of the value of branding and image over quality
What is HCI design?

- A definition by Alan Dix (2012):
  - “achieving goals within constraints”

- Goals – purpose
  - Who is it for, why do they want it

- Constraints
  - Materials, platforms
  - Financial, time constraints, expertise

- Trade-offs
Understand your materials

1. Understand computers
   - Limitations, capacities, tools, platforms

2. Understand people
   - Psychological, social aspects
   - Human error

3. And their interaction

Understand your materials
Mugs: metal vs. ceramic

Understand your materials
Paintings: oil vs. acrylics vs. watercolors
To err is human

- Accident reports
  - Air crash, industrial accident, hospital mistake
  - Enquiry blames “human error”
  - Hawaii 2018 – false missile alert

- But
  - Concrete lintel breaks because of too much weight
    - Blame “lintel error?”
    - No! We say “design error”
    - We know how concrete behaves under stress

- Human “error” is normal
  - We know how users behave under stress
  - So design for it!

- Treat the user at least as well as the physical materials!
User – Centred Design
The HCI and Designer perspective
User - Centred Design

- User - Centred Design or Human – Centred Design is an approach to the software development process that puts emphasis on the user.

- Main characteristics:
  - Human-centric as opposed to technology centric
  - User involvement from early on
  - Empirical measurement
  - Iterations in design / prototyping

- Iterative development and the cycle of feedback is a process that is known in software engineering.

- The difference between UCD and the rest is one of perspective
  - Variations from the managerial perspective of the software process and the engineering perspective of the developer to the perspective of the UX designer and Design Thinking.
The Design Thinking Process
The Design Thinking Process

1. Empathise
2. Define
3. Ideate
4. Prototype
5. Test

- Learn about users through testing
- Tests create new ideas for the project
- Learn from prototypes to spark new ideas
- Tests reveal insights that redefine the problem

Empathise to help define the problem
1. Empathise

- The first stage of the Design Thinking process is to gain an empathic understanding of the problem you are trying to solve.
  - This involves consulting experts to find out more about the area of concern through observing, engaging and empathizing with people to understand their experiences and motivations, as well as immersing yourself in the physical environment to have a deeper personal understanding of the issues involved.
  - Empathy is crucial to a human-centred design process such as Design Thinking, and empathy allows design thinkers to set aside his or her own assumptions about the world in order to gain insight into users and their needs.

- Depending on time constraints, a substantial amount of information is gathered at this stage to use during the next stage and to develop the best possible understanding of the users, their needs, and the problems that underlie the development of that particular product.
During the Define stage, you put together the information you have created and gathered during the Empathise stage.

You will analyse your observations and synthesise them in order to define the core problems that you and your team have identified up to this point.

You should seek to define the problem as a problem statement in a human-centred manner.

To illustrate, instead of defining the problem as your own wish or a need of the company such as, “We need to increase our food-product market share among young teenage girls by 5%,” a much better way to define the problem would be, “Teenage girls need to eat nutritious food in order to thrive, be healthy and grow.”

The Define stage will help the designers in your team gather great ideas to establish features, functions, and any other elements that will allow them to solve the problems or, at the very least, allow users to resolve issues themselves with the minimum of difficulty.

In the Define stage you will start to progress to the third stage, Ideate, by asking questions which can help you look for ideas for solutions by asking: “How might we... encourage teenage girls to perform an action that benefits them and also involves your company’s food-product or service?”
3. Ideate

- During the third stage of the Design Thinking process, designers are ready to start generating ideas.

- You’ve grown to understand your users and their needs in the Empathise stage, and you’ve analysed and synthesised your observations in the Define stage, and ended up with a human-centered problem statement.

- With this solid background yourself and your team members can start to 'think outside the box' to identify new solutions to the problem statement you’ve created, and you can start to look for alternative ways of viewing the problem.

- There are hundreds of Ideation techniques such as Brainstorm, Brainwrite, Worst Possible Idea, and SCAMPER.
  - For a list of ideation techniques: [https://www.interaction-design.org/literature/article/introduction-to-the-essential-ideation-techniques-which-are-the-heart-of-design-thinking](https://www.interaction-design.org/literature/article/introduction-to-the-essential-ideation-techniques-which-are-the-heart-of-design-thinking)

- It is important to get as many ideas or problem solutions as possible at the beginning of the Ideation phase.

- You should pick some other Ideation techniques by the end of the Ideation phase to help you investigate and test your ideas to find the best way to either solve a problem, or provide the elements required to circumvent the problem.
### 4. Prototype

- The design team will now produce a number of inexpensive, scaled down versions of the product or specific features found within the product, so they can investigate the problem solutions generated in the previous stage.

- Prototypes may be shared and tested within the team itself, in other departments, or on a small group of people outside the design team.

- This is an experimental phase, and the aim is to identify the best possible solution for each of the problems identified during the first three stages.

- The solutions are implemented within the prototypes and, one-by-one, they are investigated and either accepted, improved and re-examined, or rejected on the basis of the users’ experiences.

- By the end of this stage, the design team will have a better idea of the constraints inherent within the product, the problems that are present, and have a better/more informed perspective of how real users would behave, think, and feel when interacting with the end product.
5. Test

- Designers or evaluators rigorously test the complete product using the best solutions identified during the prototyping phase.

- This is the final stage of the 5 stage-model, but in an iterative process, the results generated during the testing phase are often used to redefine one or more problems and inform the understanding of the users, the conditions of use, how people think, behave, and feel, and to empathise.

- Even during this phase, alterations and refinements are made in order to rule out problem solutions and derive as deep an understanding of the product and its users as possible.
The Non-Linear Nature of Design Thinking

- We may have outlined a direct and linear Design Thinking process in which one stage seemingly leads to the next with a logical conclusion at user testing.

- However, in practice, the process is carried out in a more flexible and non-linear fashion.

- For example, more than one stage may be conducted concurrently by different groups within the design team, or the designers may collect information and prototype during the entire project so as to enable them to bring their ideas to life and visualise the problem solutions.

- Also, results from the testing phase may reveal some insights about users, which in turn may lead to another brainstorming session (ideation) or the development of new prototypes.
In essence, the Design Thinking process is iterative, flexible and focused on collaboration between designers and users, with an emphasis on bringing ideas to life based on how real users think, feel and behave.

Design thinking tackles complex problems by:
- Empathising: Understanding the human needs involved
- Defining: Re-framing and defining the problem in human-centric ways
- Ideating: Creating many ideas in ideation sessions
- Prototyping: Adopting a hands-on approach in prototyping
- Testing: Developing a prototype/solution to the problem
User centred design challenges

Real world problems
“Good designers don’t need iteration”
- “Why are we wasting time considering design alternatives we won’t use? Don’t you know which design is better?”
- “Why do we need to test it? We hired you because you are supposed to be a good designer. Aren’t you?”
- “We don’t have time for testing and revising. We need you to make an effort to design it right the first time.”

UI designers gain experience only if they receive feedback for their design through usability testing
“We don’t have the luxury of usability design”

- Myth 1. Testing is expensive
  - There are many cheap usability tests
- Myth 2. Skipping testing will save money
  - It will save the manager money, not the company
  - Market research and sales staff feedback is unreliable, often useless data on how to improve the UI
Discounting the value of testing and iterative design (3/3)

- Allowing no time to fix usability problems – “We don’t have time to fix these things!”
  - UI is considered just as presentation and graphics, therefore such problems should be easy to fix
  - “User-friendly front-end” is just another feature, which may not “make the cut”, just like other features
  - Ego (“the design is awesome, it’s the test that is problematic”)
  - It was mandatory to do the test by the company development process – Fixing the problems wasn’t
Process: Anarchic development

- No Design
  - Agile design methods are claimed to be used
    - designs are supposed to be emergent and not defined up front
    - actually there is no design at all

- No standards or guidelines

- No oversight – “Hire nerds, tell ‘em what you want, lock’em in an office and throw pizza and t-shirts every few weeks” – actual quote from a software manager

- Wrong people doing the wrong job the wrong way
Process: No task expertise on the team

- Developers assume they are task-domain experts but this was true in an age when computers were the domain of scientists and engineers only.
  - Often convince (inexperienced) managers with their human-natured, overrated self-assessments on what users need.
  - Discounting users’ task knowledge:
    - “We develop technology. We know technology. Users don’t”
    - Users can range from Novice/Naive to Experts in knowledge of computer technology.
    - However, there is also task domain knowledge and knowledge of particular software product.
Benefits of User-Centred Design (1/2)

- Reduced production costs
  - The overall development times and costs can be reduced by avoiding over design and reducing the number of changes required late in design.

- Reduced support costs
  - Systems which are easier to use require less training, less user support and less subsequent maintenance.

- Improved product quality
  - User-centred design results in products which have a higher quality of use and are more competitive in a market which is demanding easier to use systems.
Benefits of User-Centred Design (2/2)

- Reduced costs in use
  - Systems better matched to user needs improve productivity and the quality of actions and decisions
  - Easier to use systems reduce stress and enable workers to handle a wider variety of tasks
  - Difficult to use systems reduce health and well-being, motivation and may increase staff turnover and absenteeism
  - Difficult to use systems are time consuming to use, and are not exploited to full advantage, as the user may be discouraged from using advanced features. In some circumstances they may never be used
  - An ineffective system may be a major financial liability for the user organisation

- The complete benefits of user-centred design come from calculating the total lifecycle costs of the product including conception, development, implementation, support, use and maintenance
The manager’s task initially appears more difficult

Project planning has to allow for iteration and for incorporating user feedback

More time will be required for effective communication between design team participants and for reconciling potential conflicts and trade-offs
But ...

- Project managers will benefit from the additional creativity and ideas from the extended development team and skill base

- Informal communication and discussion of usage issues early on in the project will result in a better design and significant savings at later stages when changes are more costly
User-Centred Design processes may be applied in big companies with ample budgets

In practice...
- Limited time $\rightarrow$ design trade-off
- Maximum benefits with a minimum effort
- Usability
  - deciding which problem to fix vs.
  - finding all problems and fixing them
The End!

Computing is about People, Not Machines