CSE1720

Lecture 02

Winter 2013 ◆ Thursday, January 10, 2013

YORK UNIVERSITÉ UNIVERSITY

Objectives for this class meeting

- Review and discussion of course description
  - reference: Department of CSE “minicalendar”
- Presentation of course roadmap
  - Discussion and feedback
- Topic: Software Design vs User Centered Design
  - objective: understand the difference between the two methodologies, in a general way

YORK UNIVERSITÉ UNIVERSITY

CSE 1720 3.0 Building Interactive Systems

This course continues an introduction to computer programming within the context of image, sound and interaction, subsequent to CSE1710 3.0. The student’s foundation in basic programming will serve as a platform from which to explore the use of diverse media within interactive systems, including the WWW and simple game systems.

Topics include:

- User Interfaces (UIs)
- UI Elements
- Event driven programming
- Intro to threads
- User Interface Builders
- Guidelines for UI design
- Objects, classes and inheritance
- Interactive WWW-based systems - introduction to WWW and basic network concepts, HTML, Javascript, other WWW technologies (e.g. Flash), guidelines for WWW design
- How to design simple games and make them engaging

Prerequisites: CSE1710 3.0 *Course Credit Exclusions:* CSE1020 3.0, ITEC1620 3.0, ITEC1630 3.0

YORK UNIVERSITÉ UNIVERSITY

Course Roadmap

- **Design Task:** Create a single-player game
- **Design Iteration #1:** Storyboards, prototypes, graphics
  - UI elements, Guidelines for UI design
- **Design Iteration #2:** The data model
  - inheritance, collections framework
- **Design Iteration #3:** Connecting the graphics to the data model
  - event driven programming, threads
  - the observer pattern
- **Design Iteration #4:** Creating a controller to interpret user input actions and to implement game logic
  - model-view-controller

YORK UNIVERSITÉ UNIVERSITY

Course Evaluation

- **4 Written Tests 40%**
  - covering design iteration #1 (5%)
  - covering design iteration #2 (10%)
  - covering design iteration #3 (10%)
  - covering design iteration #4 (15%, final exam period)
- **4 Labtests 40%**
  - covering design iteration #1 (5%)
  - covering design iteration #2 (10%)
  - covering design iteration #3 (10%)
  - covering design iteration #4 (15%, final exam period)
- **Preparatory/In-Class Exercises 20%**
  - both written and code-based

Course Evaluation

- **During Term 70%**
  - exercises **20%**
  - written tests (x3) **35%**
  - labtests (x3) **35%**
- **Final Exam Period 30%**
  - written tests **15%**
  - labtests **15%**

- decision: 70-30 split vs 80-20 split

Course Content

- **Decision**
  - we work on the same game all together OR each works on their own game?
- **Comparison**
  - each student works on own game
    - game tailored to own interests
    - much more work and time investment
    - less direct guidance from instructor
  - same game
    - game tailored to class's common interest (to the extent possible)
    - greater degree of direct guidance from instructor

Design Methodologies

- Discuss similarities of and differences between:
  - The Waterfall Development Methodology (Ch 7, JBA)
  - The Iterative Software Development Methodology (Ch 7, JBA)
  - User-Centered Design (UCD) (many reference texts)

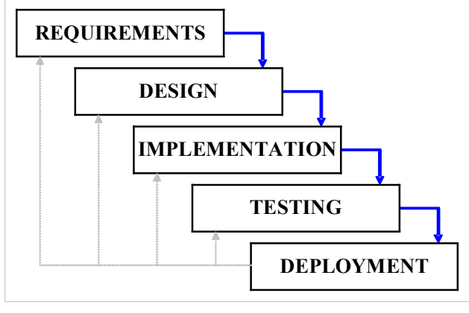
  - Engineering Design (professional accreditation)
  - "Design" (product design, digital media design, etc)

  - Development phase vs Production phase

Design Methodologies

- Contexts in which a methodology might be used:
  - software needs of private/public sector companies, provided on a fee-for-service basis or by a company's internal resources
    - capitalist domain
  - software needs for cultural objects and systems
    - things that are not sold, but exhibited, displayed, or otherwise shared; not capitalist
- Think of the roles that people occupy within these contexts and the expectations that would be attached
  - business client
  - end-users

The Waterfall Methodology



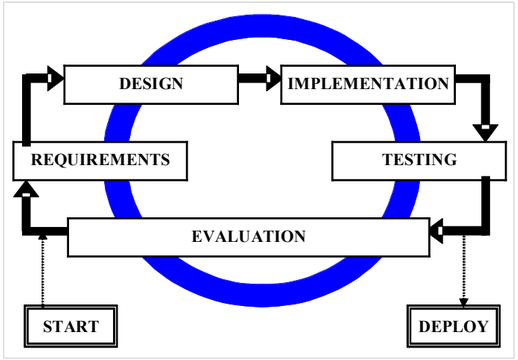
The Factor of Risk

- What kind of risk are we talking about?
  - Impact on user and/or use scenario **if the software operates differently than expected**
    - how crucial?
      - life or death? (to people, to the company)
      - or merely unpleasant/undesireable
- Risks arise from many sources:
  - software architecture itself
    - depend on other implementers? perhaps contain errors
    - complex functioning? perhaps problems exist
      - e.g., Therac-25 case study (google it)
  - the requirements may change during or after the design
  - the underlying assumptions may be wrong

Risk Mitigation

- Risk mitigation
  - also known as "risk reduction"
  - we cannot eliminate risk, but try to reduce the extent to which the client is exposed to risk
  - try to reduce the likelihood of occurrence
  - the use of a principled design methodology can serve to **systematically** mitigate risk
    - in other words, the design methodology **itself** is built to identify risks and to deal with them
- One technique for risk mitigation
  - try to expose risks earlier rather than later
  - "early exposure"

The Iterative Methodology



Specification vs Implementation

ref: Alan Cooper, About Face, 1995

Analogy (simplified):

- Houses are designed by architects and built by carpenters (somewhat true)
- If carpenters designed houses, they would certainly be easier or more interesting to build, but not necessarily better to live in.
  - example: position of light switches
- There is a tension between the needs/wants of the users and the needs/wants of the implementers
- The tension is kept in balance by the separation of design specification and implementation

Specification vs Implementation

In software development:

- Software is often designed by and built by the same person/team
- There is a tension between the needs/wants of the users and the needs/wants of the implementers
- The tension **may not be kept in balance**
  - desire to ease implementation may infiltrate the design specification
  - who is advocating for the end user?

User-Centered Design

- term coined by Donald Norman (1980's, The Psychology of Everyday Things, 1986)
- puts the user in the middle of the design process
- Three major components:
  - Iterative design
  - Early focus on users and tasks
  - Empirical measurement