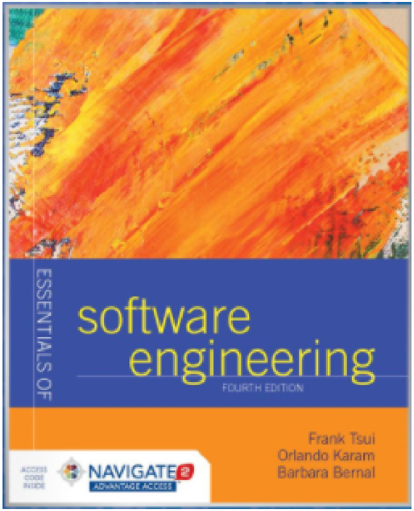
Design Characteristics and Metrics

CSC 355, Software Engineering II

Chapter 8: Design: Characteristics and Metrics



Characteristics of Good Design

- Besides the obvious “design should match the requirements,” there are two basic characteristics:
  1. Consistency across design
  2. Completeness of the design

Consistency

- Common user interface (UI)
  - Looks and logical flow
  - Location of buttons, etc.
- Common error processing
- Common reports
- Common system interfaces
- Common help

Completeness

- All the requirements are accounted for (Requirements Traceability Matrix)
- All parts of the design are carried to completion
- All design carried to the same depth level

Good Design Attributes

- Easy to:
    - Understand
    - Change
    - Reuse
    - Test
    - Integrate
    - Code
- We can get many of these if we consider:
    - Cohesion
    - Coupling

Cohesion

- Cohesion addresses the “degree of relatedness” within a unit, module, object, or component
- Degrees:

  1. Functional (best: performing one single function)
  2. Sequential
  3. Communicational
  4. Procedural
  5. Temporal
  6. Logical
  7. Coincidental (worst: performing more than one unrelated function)

Coupling

- Coupling addresses the “degree of independence” between software units, modules, or components
- Degrees:
    1. Content (worst: accessing the internal data or procedural information)
    2. Common
    3. Control
    4. Stamp
    5. Data
    6. None (best: passing only the necessary information)

Sneiderman's Golden Rules of Interface Design

1. Strive for consistency
   - Is the style of this element maintained across your site/app?
   - Does it follow the conventions for your chosen platform?
2. Enable frequent users to use shortcuts
   - Are there shortcuts available for your more experienced users?
   - Who is this product designed for?
3. Offer informative feedback
   - Does the user know where they are in the process?
   - How are you communicating feedback to your user?

Sneiderman's Golden Rules of Interface Design

4. Design dialogues to yield closure
   - Does the user have to do any guessing here?
   - Is it clear and obvious enough for your intended audience?
   - Are there any next steps?
5. Offer simple error handling
   - Have you done everything imaginable to prevent this error from happening on your end?
   - If the user does make an error, how easy is it to fix it?
6. Permit easy reversal of actions
   - How many steps does the use have to take to reverse their actions?
   - Will the user quickly realize they need to reverse the action?

Sneiderman's Golden Rules of Interface Design

7. Support internal locus of control
   - Will the user feel in control at this specific touch point in your app?
   - Will the user be surprised in an unpleasant manner?
   - Does the site feel easily navigable?
   - Does the user feel safe and in control?
8. Reduce the sort-term memory load
   - Are there enough visual clues here for the user to find the functionality or item?
   - Do they have to remember things to understand what is going on?

Mandel’s Golden Rules of Interface Design

1. Place the user in control
2. Reduce the user’s memory load (Miller’s 7 plus or minus 2)
3. Consistency (as described earlier)

UI Design Prototypes and Testing

- UI Prototypes
  - Low fidelity (with cardboards)
  - High fidelity (with storyboard tools)
- Usability “laboratories test” and statistical analysis
  - Number or subjects who can complete the tasks within some specified time
  - Length of time required to complete different tasks
  - Number of times help functions are needed
  - Number of times redo functions are used and where
  - Number of times short cuts were used

Law of Demeter

- A design guideline for Object-Oriented systems that originated from the Demeter system
- Addresses the design coupling issue through placing constraints on messaging among the objects
- Limits the sending of messages to objects that are directly known to it

Law of Demeter

- An object should send messages to **only** the following kinds of objects:
  - The object itself
  - The object's attributes
  - The parameters of the methods in the object
  - Any object created by a method in the object
  - Any object returned from a call to one of the methods of the object
  - Any object in any collection that is on of the above categories

Books About Design

- The design of everyday things, Don Norman
- Don't make me think, Steve Krug
- 100 Things Every Designer Needs to know about design, Susan M. Weinschenk