IF2261 Software Engineering

Interface Design

Program Studi Teknik Informatika
STEI ITB

Interface Design

- The interface design elements for software tell how information flows into and out of the system and how it is communicated among the components defined as part of the architecture

**** SEPA 6th ed, Roger S. Pressman***

Interfaces Design

- Inter-modular interface design
  - driven by data flow between modules
  - closely aligned with component level design
- External interface design
  - driven by interface between applications
  - driven by interface between software and non-human producers and/or consumers of information
- Human-computer interface design
  - driven by the communication between human and machine

SEPA 6th ed, Roger S. Pressman

User Interface Design

Three golden rules – Theo Mandel

- Place user in control
  - "What I really would like is a system that reads my mind. It knows what I want to do before I need to do it and makes it very easy for me to get it done. That's all, just that."
- Reduce the user's memory load
  - The more a user has to remember, the more error-prone interaction with the system will be
  - The system should 'remember'
- Make the interface consistent

Place User in Control

- Define interaction in such a way that the user is not forced into performing unnecessary or undesired actions
- Provide for flexible interaction (users have varying preferences)
- Allow user interaction to be interruptible and reversible
- Streamline interaction as skill level increases and allow customization of interaction
- Hide technical internals from the casual user
- Design for direct interaction with objects that appear on the screen

*** SEPA 6th ed, Roger S. Pressman**

Reduce User Memory Load

- Reduce demands on user's short-term memory (exp. providing visual cues)
- Establish meaningful defaults (“reset” option should be available)
- Define intuitive short-cuts (easy to remember)
- Visual layout of user interface should be based on a familiar real world metaphor
- Disclose information in a progressive fashion

*** SEPA 6th ed, Roger S. Pressman**

Make Interface Consistent

- Allow user to put the current task into a meaningful context
- Maintain consistency across a family of applications
- If past interaction models have created user expectations, do not make changes unless there is a good reason to do so

**** SEPA 6th ed, Roger S. Pressman***

User Interface Analysis and Design

- The models
  - The human engineer establishes a user model
  - The software engineer creates a design model
  - The end-user develops a mental image or user's mental model (system perception)
  - The implementers create an implementation model
- The process
  - User, task, and environment analysis and modeling
  - Interface design
  - Interface construction
  - Interface validation

**** SEPA 6th ed, Roger S. Pressman***

The Models

- User model
  - end user profiles:
    - Novice
    - Knowledgeable, intermittent user,
    - Knowledgeable, frequent users
- Design model
  - incorporates data, architectural, interface, and procedural representations of the software
- User's model or system perception
  - user's mental image of system
- Implementation model
  - look and feel of the interface and supporting media

** SEPA 6th ed, Roger S. Pressman*

The Process - User Analysis

- Understand who the end-users are
- What is likely to motivate and please them
- How they can be grouped into different user classes or profiles
- What their mental models of the system are
- How the user interface must be characterized to meet their needs

The Process - Task Analysis and Modeling

- Software engineer studies tasks human users must complete to accomplish their goal in the real world without the computer and map these into a similar set of tasks that are to be implemented in the context of the user interface
- Software engineer studies existing specification for computer-based solution and derives a set of tasks that will accommodate the user model, design model, and system perception
- Software engineer may devise an object-oriented approach by observing the objects and actions the user makes use of in the real world and model the interface objects after their real world counterparts

→ Know the user, know the tasks

** SEPA 6th ed, Roger S. Pressman*

The Process - Analysis of Display Content

- Type of content
  - Character-based reports
  - Graphical displays
  - Specialized information
- Source of content
  - Generated by components
  - Acquired from data store
  - Transmitted from systems external

** SEPA 6th ed, Roger S. Pressman*

The Process - Interface Design Activities

- Establish the goals and intentions of each task
- Map each goal/intention to a sequence of specific actions (objects and methods for manipulating objects)
- Specify the action sequence of tasks and subtasks (user scenario)
- Indicate the state of the system at the time the user scenario is performed
- Define control mechanisms → object and action
- Show how control mechanisms affect the state of the system
- Indicate how the user interprets the state of the system from information provided through the interface

**** SEPA 6th ed, Roger S. Pressman***

Interface Design Issues

- System response time
  - time between the point at which user initiates some control action and the time when the system responds
- User help facilities
  - integrated, context sensitive help versus add-on help
- Error information handling
  - messages should be non-judgmental, describe problem precisely, and suggest valid solutions
- Command labeling
  - based on user vocabulary, simple grammar, and have consistent rules for abbreviation

**** SEPA 6th ed, Roger S. Pressman***

User Interface Evaluation Cycle

1. Preliminary design
2. Build first interface prototype
3. User evaluates interface
4. Evaluation studied by designer
5. Design modifications made
6. Build next prototype
7. If interface is not complete then go to step 3

SEPA 6th ed, Roger S. Pressman

User Interface Design Evaluation Criteria

- Length and complexity of written interface specification provide an indication of amount of learning required by system users
- Number of user tasks and the average number of actions per task provide an indication of interaction time and overall system efficiency
- Number of tasks, actions, and system states in the design model provide an indication of the memory load required of system users
- Interface style, help facilities, and error handling protocols provide a general indication of system complexity and the degree of acceptance by the users

SEPA 6th ed, Roger S. Pressman

Data Design

- Data Design Principles
  - Systematic analysis principles applied to function and behavior should also be applied to data.
  - All data structures and the operations to be performed on each should be identified.
  - Data dictionary should be established and used to define both data and program design.
  - Low level design processes should be deferred until late in the design process.
  - Representations of data structure should be known only to those modules that must make direct use of the data contained within in the data structure.
  - A library of useful data structures and operations should be developed.
  - A software design and its implementation language should support the specification and realization of abstract data types.

Component Level Design

- The purpose of component level design is to translate the design model into operational software.
- Component level design occurs after the data, architectural, and interface designs are established.
- Component-level design represents the software in a way that allows the designer to review it for correctness and consistency, before it is built.
- The work product produced is the procedural design for each software component, represented using graphical, tabular, or text-based notation

Procedural Design Notation

- Flowcharts
  - arrows for flow of control, diamonds for decisions, rectangles for processes
- Box diagrams
  - also known as Nassi-Scheidnerman charts
- Decision table
  - subsets of system conditions and actions are associated with each other to define the rules for processing inputs and events
- Program Design Language
  - PDL - structured English or pseudocode used to describe processing details

Program Design Language Characteristics

- <u>Fixed syntax</u> with keywords providing for representation of all <u>structured constructs, data declarations, and module definitions</u>
- <u>Free syntax</u> of natural language for describing <u>processing features</u>
- <u>Data declaration</u> facilities for simple and complex data structures
- <u>Subprogram</u> definition and invocation facilities

Design Notation Assessment Criteria

- Modularity
  - notation supports development of modular software
- Overall simplicity
  - easy to learn, easy to use, easy to write
- Ease of editing
  - easy to modify design representation when changes are necessary
- Machine readability
  - notation can be input directly into a computer-based development system
- Maintainability
  - maintenance of the configuration usually involves maintenance of the procedural design representation

Design Notation Assessment Criteria (2)

- Structure enforcement
  - enforces the use of structured programming constructs
- Automatic processing
  - allows the designer to verify the correctness and quality of the design
- Data representation
  - ability to represent local and global data directly
- Logic verification
  - automatic logic verification improves testing adequacy
- Easily converted to program source code
  - makes code generation quicker