(SWE)

30/1/2023

Contextual Inquiry

plam
Initiate
Execute
close
Reflect - analyze data

Data Analysis

→ Affinity Diagram

* Generate Idea
* Display ideas
* Sort idea's into groups
* Create " group header "
* Draw finished diagram

functional Requirement

→ Abstraction

→ Decomposition and hierarchy

__/__/____

* In requirement specification we don't bother about the implementation details

Design:

Two issues:-
  * start?
  * representation

start
* Interface design
* Code design

* can also make use of some guidelines/thumb rules

Design Guidelines

* eight Golden rule
* Seven principle for interface design

Code design

* Main idea is make the code 'Manageable'

__/__/____

* Done based on SRS
* Two phases
  - preliminary (high-level) design
  - Detailed design (also called module-specification document)

High-level Design:

* Identification of modules
* Control relationships between modules
* Definition of interfaces between modules.

Detailed Design:

* we focus on individual modules which data structures and algorithms used

Design language

* Refers to a specifications of design

prototypes (language)

Code design language

__/__/____

* Natural language
* Semi-formal language – DFD (Data flow Diagram)
* formal language

→ Semi-formal – (UML) Unified Modeling Language

GUIDELINES

* provide starting point in the development life cycle

Shneiderman's Guidelines

* eight rules

Rule 1:-
strict consistency

* Internal consistency

* External Consistency

__/__/____

Rule 2

* Design for universal usability
Example :- menu, ctrl + s

Rule 3:

+ offer informative feedback
Eg: progress bar

Rule 4:

Design dialogues to yield closure

Organize activities into groups - beginning, middle end
some feedback at the end of each group
Eg:- online shoping (lots of subtasks, grouping helps)

Rule 5:

offer error prevention and simple error handling

* Design to keep error rates low (close and start options should not be kept close to each other)

+ Don't show complicated error messages

__/__/____

Rule-6

reversal of actions

* undo and Redo

Rule-7

Keep users in control

user should be able to perceive their interaction and change in system state

Rule 8

Reduce short term memory load

* should not force users to remember too many things

Norman principles:-

7 principles

Norman's Model of interaction

* Represents behaviour of a layman user of interactive systems in terms of a series of "actions"

* actions are cognitive and sensory motor

__/__/____

Two stages

Execution
- → Establish goal
- → Formulate intention
- → Specify action at the interface
- → Execute action (human motor action)

(Cognitive)

Evaluation
- → perceive system state
- → Interpret system state
- → Evaluate system state with respect to goal.

(Cognitive)

The gulf of execution :-
If the action we formulated and specify them with real tasks they may not be supported by the interfaces then we say there is a Gulf of execution

The gulf of evaluation :-

our perception, may not match with actual state of interface.

__/__/____

+ Slip
→ understand system and goal
→ Correct formulation of action
→ Incorrect action

+ Mistake
→ Unable establish a goal correctly

Slip - better interface design
Mistake - better understanding of system

+ This model also used to recommend the guidelines

principles:

(1) use both knowledge in the world and knowledge in the head
2. Simplify the structure of tasks
3 Make things visible: bridge the gulfs of execution and evalution
4 Get the mappings right
5 Exploit the power of constraints, both natural and artificial
6 Design - for error

__/__/____

7. When all else fails, standardise!

Prototyping

* purpose is served, a prototype can be discarded
* Can be incrementally refined

prototype categories:

Horizontal - the entries interface is depicted at the surface level without any functionality

* Interaction is not protyped
* No real work can be done
* Suitable to discuss, brain-storm or elicit feedback on the interface look and feel, primarily

Vertical: designed to represent interaction

* Few selected features implemented in-depth Starting from the first screen to the screen after the last action is performed.

__/__/____

* Suitable for analysis of interactions and features.

CREATION OF PROTOTYPES

* Low-fidelity
  * No technological intervention - made with paper cardboards, woods and so on
  * A popular example are the paper mock-ups for interface look, feel and even functionality
  * Quick and cheap
  * Good for horizontal

Interface sketches for interaction
+ story boarding - series of sketches

→ Medium-fidelity:-
  * When we use computer to create simple (low fidelity) prototypes.
  * If the computer is used to create static sketches only, it still remains horizontal prototypes since no real functionality is there

__/__/____

Ex: Adobe Flash tool
interaction can also be simulated with as simple a tool as a microsoft powerpoint TTM

→ Hi-fidelity prototypes

prototypes created with computer programs

* More sophisticated and requires much more effort than the other two

Ex: visual basic programming language, Java swing library and so on

usage of prototype

+ thrown away

+ Incremental: Each unit is separately prototyped and test
Afterwards, it is integrated into the system

* evolutionary: the whole system is prototyped and tested
* Eventually, it becomes final product.

__/__/____

Evaluation:

→ **Expert evaluation!**

quick and cheap evaluation

Two things

* a prototype (at least low-fidelity)
* An evalution team

* Each team member evaluates individually and produce Report
  → Report contains a list of usability issues

* All these reports combined to produce a final list

Cognitive Walk through

* An usability Inspection method
* Requirements
  → At least a low-fidelity with support for several tasks
  → 3-5 member team.

__/__/____

Additional Requirements

Task descriptions:-
specify scenario to perform tasks.

* Questions: We also need to frame a set of questions beforehand (usability issues)
* Evaluators report on these issues while they perform the tasks
* Evaluators need to give report about it

Heuristic Evaluation

* doesn't required end users.
* we can ask evaluators to tick on a checklist of features of the system as a whole

evaluate a system with a checklist
Nielsen's 10 Heuristics:
* Visibility of system status
* Match between system and the real world
User control and freedom

__/__/____

* consistency and standards
* Error prevention
* Recognition rather than recall
* Flexibility and efficiency of use
* Aesthetic and minimalist design
* Help users recognize, diagnose and recover from errors
* Help and documentation.

⇒ Reports are combined to determine heuristics that are violated.

Self reports and evaluation

Questionnaire:

post-task → After scenario Questionaire
Expectation measure

post-session questionnaires → system Usability scale
* Computer system Usability (CSUQ)
* Questionnaires for user interface Satisfaction (Quis)
* Usefulness, satisfaction, ease

__/__/____

SUS score as percentage

We can score in the example is 55 percent

* It is generally assumed that an SUS score around 70 percent or higher is desirable

* scores lower than that might indicate usability problems.

figma - medium fidelity prototype, vertical prototype,

Interface design to system

Two phases

- preliminary (high-level) design
- Detailed Design (also called Module-specification document)

High level design

Identification of modules

Control relationships between modules

Definition of interfaces between modules

__/__/____

DFD (Data flow Graph)

Graphical notations

Detailed Design

+ Identification of data structures and algorithms

Good Design

Coverage - Should implement All functionalities of SRS

Correctness - Should correctly implement all function -alities

Understandability - easy understanding

Efficiency - in terms resource required

Maintainability - should be amenable to change

Cohesion and coupling

+ Good software design requires
- clean decomposition of the problem into modules
- Neat arrangement of modules in a hierarchy

__/__/____

Modularization depends on cohesion and coupling

Cohesion (of a module)

Logical - if all functions perform similar operations

Temporal - if all functions should be performed in the same time span

procedural / functional - if all functions are part of the same procedure (algorithm)

Communication - if all functions refer to or update same data structure

Sequential - output from one element is input to the next element of the module

Coupling (between modules)

Data - if two modules communicate through a data item (e.g passing an integer between two modules)

Control - if data from one module is used to control the flow of instruction in the other module

Content - if two modules share code

__/__/____

→ High cohesion and low coupling → functionally independent

Basic Design Approaches

+ function oriented design (DFD)

+ object oriented design (UML)

DFD - Basics

+ flow of data through a process or a system

+ provides overview of
  + What data processed by a system
  + Transformations that are performed
  + Which data stored
  + what results are produced and where they go

+ good communication tool between
  → user and (system) designer
  → Designer and developer.

__/__/____

Components of DFD

* source / sinks
* processes — series of actions that transform data
* Data stores

Miracle - No input flow

Black hole - No output flow

Gray hole - No process ~~After~~ output flow

DFD levels

Level 0 - context diagram

Level 1 - overview diagram

Level 2 - detailed diagram

Entity Relationship Diagram

To express rich internal structure, organization and relationship in data stores

Components

* Entity
* Attribute
* Relationship