

**Brooks**

**Roll**

**Teams**

**Review**

**Analysis**

**Design**

**(Proposed) Exam Schedule:**

**Exam 1          February 22 or 24 ?**

**Exam 2          March 30**

**Exam 3          April 27**

**Final Exam      May 9**

## **Project: A Simple Spread Sheet**

**What does it do?**

**A 2 dimensional grid of cells containing text, numbers or formulas**

**May manipulate cells, rows or columns**

**How? Copy, Delete, etc.**

**What is a natural method to express (understand) what this  
Spread Sheet does?**

## Design - Methods of Design

What is design?

Software Design - is a process through which requirements are translated into a representation of software.

- Focus is on HOW the program will work.
- Try to avoid programming language and hardware specific details that affect HOW.
- Develop a program architecture and map requirements to portions of the architecture.

The Designer's goal -  
produce a model or representation of an entity  
that will later be built.

Why design?

Makes Implementation Mechanical  
Different from Requirement or Code

How is design done:

Many methods:

Structured

OO

Etc.

## NEED FOR DESIGN

Design is the only way we can accurately translate a customer's requirements into a finished software product

Without a design we risk building an unstable system:

- one that will fail when small changes are made (maintainability)
- one that may be difficult to test (testability)
- one whose quality can not be determined until late in the development process

Goal:

Define process (or system) in enough detail to  
Implement. (Create, realize)

What is designed:

Data

Architecture

Interface

Procedural

Data design:

Wasserman:

“Select logical representations of data objects”

Should apply same analysis principles as applied to function

Identify data structures and operations performed on them

Data dictionary

Defer low-level data decisions

Hide data representation in modules

Develop library of data structures and operations  
(reuse)

Use appropriate programming language

Architecture design:

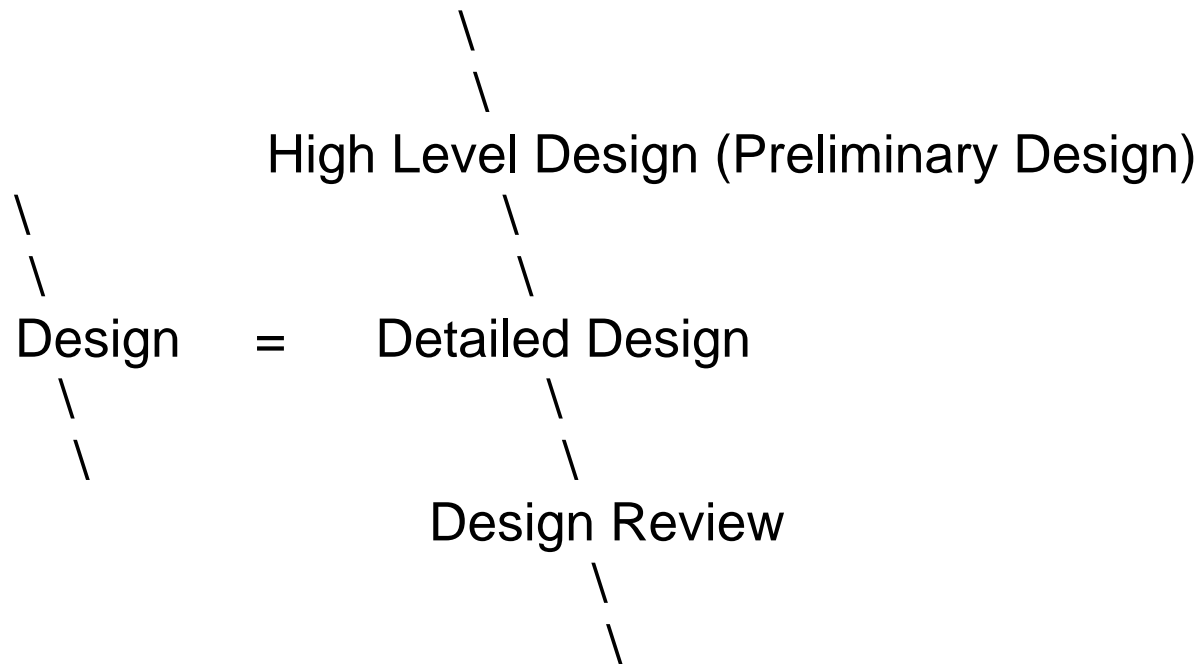
Develop modular program structure and show control relationships

This is very important: this shows the “outline” of the entire system.

## DESIGN PROCESS:

The design for a system is created in stages.

From a project management point of view  
(refinement of lifecycle)



(This started from “top-down” and “structured-programming” and “modular” ideas of late 1960’s and early 1970’s)

Data-flow oriented design:

Natural flow from analysis

Find type of information flow

Find flow boundaries

Map DFD to program flow

Factor control

Refine structure

Find type of information flow

Transform Flow  
Transaction Flow

Transform Flow

Incoming flow -> Transform center ->  
Outgoing flow

Overall flow is sequential, one of few paths  
Look at DFD for few (one) straight line paths

Transaction Flow

Data Item triggers other flows  
Select from many paths

Transaction->Transaction Center->  
->Many  
->Paths

In most systems BOTH flows are present.  
(Find flow boundaries)

Map DFD to program flow

Transform Mapping – DFD with Transform Flow mapped to template for program structure

Design Steps:

Review Level 0 DFD (system spec and SRS)

Refine Data Flow (Show Details)

Is DFD transform or transaction?

Find and isolate transform center (find in and out flow boundaries )

Do a first level factoring  
(top down distribution of control)

Do a second level factoring  
(individual DFD bubbles to modules)

Refine using heuristics for design

Transaction Mapping:

Similar to above, BUT

Find transaction center

Factoring is on path basis

## Afterwards:

- Need to describe processing for each module

- Describe Interfaces

- Local and Global Data structures

- Note Limits, Restrictions

- Review

- Optimize

## Design Optimization:

- Knuths Law: “Don’t”

- Use CASE tools

- Analyze hot spots (time hogs), use appropriate

  - Algorithms

- Use appropriate programming Language

- Use instrumentation to find heavy loaded

  - modules: SW or HW

- Redo those

## Interface Design:

Internal  
External  
Human

Some guidelines:

- Simplify information passing

- No globals

- Validate data

- Error handling and propagation: fix it,  
or pass it up.

User Interface

- KISS

- WIMP (?)

Who are users?

- User Model

  - Experiences, other background

  - Novices, intermittent, frequent users

## Design Issues:

### HELP:

How can User get “Help”?

Is help available for all functions?

Which?

How to show help?

How to get out of help?

### ERRORS:

How to show error messages

Provide some advice

Indicate any unexpected: (File lost, etc)

Don't blame User

### COMMANDS:

Short cuts

Menus

Customization

Offer feedback (when slow, etc.)

Hints:

Iterate

Prototype

Use tool kits

USE IT!

## User Interface Guidelines:

### General:

- Be consistent
- Undo
- Make destruction difficult

### Display:

- Show (only) important info
- Be consistent, predictable
- Compartmentalize
- Show analog displays (guage)

### Data Input:

- Minimize
- Consistent
- Let user control flow
- Customize

## Procedural Design:

### Structured:

Sequence, condition, repetition

### Graphical:

Flow Chart

Nassi-Shneiderman

### Tables:

Decision Tables: (Like PLA's)

### PDL (Program Design Language)

Pseudocode, structured English

### Combined:

CSD (Control Structure Diagram)

(Mostly with CASE tools)

Why is design Important?

Is Structured Design the way to go?

Pro:

- Popular
- Fits with SA
- Lots of Experience
- Tools
- The way you think
- Flexible

Cons:

- Like SA
- Is OO more “natural”
- Easy to omit, forget, etc. functions
- Maintenance
- Why Design at all?

## High Level Design

- Also called Preliminary Design
- Concerned with the transformation of requirements into data and software architecture
- PRIMARY INPUT is the SRS
- PRIMARY OUTPUT is an architectural design and data design

## Detailed Design

- Focuses on refinements to the architectural representation that lead to detailed data structure and algorithmic representations for software
- PRIMARY INPUTS are the outputs of the High Level Design step
- PRIMARY OUTPUT is the SDD (Software Design Description)
  - + contains architectural design (structure charts),
  - detailed design (module specifications)
  - data design (Design data dictionary)

and a Software Design Walkthrough document.  
(We don't use here anymore)

## STRIVING FOR QUALITY

Quality is an important objective of software design.

- Quality in the system really begins at the design stage
- Designs can be assessed for quality
  - One proven metric that can predict quality is  $(\text{Fan-Out})^2$   
i.e For a given module, as the number of other modules called by it increases, the complexity of the calling module is likely to increase, making that module more likely to contains mistakes.

## Guidelines for Design Quality (Pressman)

- 1) A design should exhibit a hierarchical organization that makes intelligent use of control among components of software.  
\*\* Abstraction
- 2) A design should be modular; that is, the software should be logically partitioned into components that perform specific functions and subfunctions.  
\*\* Cohesion
- 3) A design should contain distinct and separable representations of data and procedure.
- 4) A design should lead to modules (e.g functions or procedures) that exhibit independent functional characteristics.  
\*\* Coupling, Cohesion
- 5) A design should lead to interfaces that reduce the complexity of connections between modules and with

the external environment.

\*\* Coupling

- 6) A design should be derived using a repeatable method that is driven by information obtained during software requirements analysis.

\*\* methods, techniques

## Design Review

- Examines and evaluates design documents for completeness, correctness, quality, agreement with requirements.
- PRIMARY INPUTS are the outputs of the detailed design stage.
- PRIMARY OUTPUTS is an acceptance or rejection of the SDD, plus a list of issues or modifications to be handled.

Running through the High Level and Detailed Design Phases are:

Data Design - transforms the information domain model created during the requirements analysis into the data structures that will be required to implement the software. (We cover this more later when we talk about Databases.)

Architectural Design - Defines the relationships among major structural components of the system.

Procedural Design - transforms structural components into a procedural description of the software.

Interface Design (sometimes) - establishes layout and interaction mechanisms for human-machine interaction.

## Design - Methods of Design

What is design?

Software Design - is a process through which requirements are translated into a representation of software.

- Focus is on HOW the program will work.
- Try to avoid programming language and hardware specific details that affect HOW.
- Develop a program architecture and map requirements to portions of the architecture.

The Designer's goal -  
produce a model or representation of an entity  
that will later be built.

Object Oriented Methodolgy: Terms, ideas, techniques  
OO Concepts, OO Analysis, OO Design  
Design Concepts, Design Methods (Structured)  
Real Time Analysis and Design

What is designed:

Data

Architecture

Interface

Procedural

## Object Oriented Design:

Find objects, factor into good classes, define class hierarchies and interfaces, make it reusable, modifiable.

OO Design architecture stresses object relations rather than flow of control

What is different: (Fichman, Kemerer)  
(From SA)

Module Hierarchy representation  
Data Definition Spec  
Procedural Spec  
End-to-end processing sequences  
Class and Hierarchy definition  
Operations assigned to classes  
...

Bertrand Meyer suggests:

Decomposable – design method makes easier subproblems

Composable – reusable

Understandable – by module

Continuity – coupling (local)

Protection – coupling (isolation)

To achieve:

Few, small, explicit interfaces. Information hiding.

OOD:

Booch

Coad – Yourdon

Rumbaugh

Jackson – design a part of analysis!

Coad suggest (for any method): Define:

Problem Domain

User I/F

Task Management

Data Management

Design Patterns:

(This is another of the SE Buzz words!)

Recurring and reusing of classes and objects  
(Gamma)

Name of pattern

Problem to which applied

Characteristics

Consequences of applying

Should one use inheritance or composition?

(When both possible)

Use what is general, or make it specific?

Compare SA, SD versus OO:

Hospital system: Patient record system.

Spreadsheet

Is OO Important?

OO Prog languages: JAVA, et al

OO Interfaces:  
  Windowing systems

CORBA  
  Distributed objects