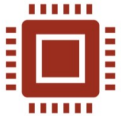
SOFTWARE DESIGN
COSC 4353/6353

Dr. Raj Singh

OUTLINE – WEEK 1



What is Software design?

Design concepts

Design considerations

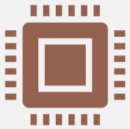
Software Modeling

Software development challenges

WHAT IS SOFTWARE DESIGN?

A process of implementing software solutions to one or more set of problems.

A process by which an agent creates a specification of a software artifact intended to accomplish goals using a set of primitive components and subject to constraints.



Software design can be considered as creating a solution to a problem in hand with available capabilities.

WHAT IS SOFTWARE DESIGN?

A process of converting the software requirements analysis (SRA) to list of specifications used for software development to solve problem/s.

Specifications could be as simple as

- flow chart
- UML diagram

The main difference between software analysis and design is that the output of a software analysis consist of smaller problems to solve.

SOFTWARE DESIGN VS. ANALYSIS

- Similar action but different output
- Design is focused on solution to the problem as whole that may consist sub-problems.
- Analysis consist of smaller (sub) problems to solve.
- Analysis must be same to the multiple designs to the same problem.
- Design may be platform-independent or platform-specific, depending on the availability of the technology used for the design.

DESIGN CONCEPTS

The design concepts are a foundation from which more sophisticated methods can be applied.

A set of concepts has evolved.

Abstraction

A process or result of generalization by reducing the information content.

Retain only information which is relevant for a particular purpose.

Refinement

More specific to a certain process.

Abstraction and Refinement are complementary concepts.

DESIGN CONCEPTS

Modularity

Software architecture is divided into components called modules.

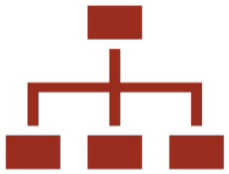
Software Architecture

Overall structure of the software.

A good architecture will yield to good quality product.

Effectiveness is measured in terms of performance, quality, schedule and cost.

DESIGN CONCEPTS



Hierarchy

A program structure that represents the organization of a program component.

Structural Partitioning

The program structure can be divided both horizontally and vertically.

Horizontal partitions define separate branches of modular hierarchy.

Vertical partitioning suggests that control and work should be distributed top down.

DESIGN CONCEPTS

Data Structure

It is a representation of the logical relationship among individual elements of data.

Software Procedure

It focuses on the processing of each modules individually

Information Hiding

Hide the implementation details.

DESIGN CONSIDERATIONS

There are many aspects to consider in the design of a piece of software based on the goals the software. Some of these aspects are:

Compatibility — The software is able to operate with other products.

Extensibility — New capabilities can be added to the software without major changes to the underlying architecture.

Fault-tolerance — The software is resistant to and able to recover from component failure.

DESIGN CONSIDERATIONS

Maintainability

A measure of how easily bug fixes or functional modifications can be accomplished. High maintainability can be the product of modularity and extensibility.

Modularity

Smaller modules for each individual task.

Easy to test, debug, reuse, and maintain.

Reliability

The software is able to perform a required function under stated conditions for a specified period of time.

DESIGN CONSIDERATIONS

Reusability	It can be reused in other application and can be extended easily.
Robustness	The software is able to operate under stress or tolerate unpredictable or invalid input.
Security	The software is able to withstand hostile acts and influences.

DESIGN CONSIDERATIONS

Usability	User friendly and self explanatory. Default values for the parameters.
Performance	The software performs its tasks within a user-acceptable time.
Scalability	The software adapts well to increasing data or number of users.

SOFTWARE MODELING LANGUAGE

Software models are used to represent software design.

Textual or graphical languages are used to express the software design.

For object-oriented software, an object modeling language such as UML is used to develop and express the software design.

Defined by a consistent set of rules.

SOFTWARE MODELING LANGUAGE

Most commonly used software modeling language is UML.

Unified Modeling Language (UML) is a general modeling language to describe software both structurally and behaviorally.

A standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

SOFTWARE DEVELOPMENT CHALLENGES

Software is

- Used for a long time
- Updated and maintained by people who did not write it

Software requirements specification

- Initially may be incomplete
- Clarified through extensive interaction between user(s) and system analyst(s)
- Needed at the beginning of any software project
- Designers and users should both approve it

SOFTWARE DEVELOPMENT CHALLENGES

Requirements change

- Users needs and expectations change over the time
- Software use reveals limitations and flaws
- Desire for increased convenience, functionality
- Desire for increased performance

Environment change

- Hardware, OS, software packages ("software rot")
- Need to interact with clients, parent org., etc.
- Law and regulations change
- Ways of doing business
- Style, "cool" factor, new trends

OTHER CHALLENGING FACTORS

Resources

Time, budget, expertise

Organizational changes

People, goals, mergers and acquisitions

Technology constraints

Operating system, software language, framework, database, security, ever changing technology, innovations

RISKS IN DEVELOPMENT

You write code based on what you know

When was the last time you had to change the design?

What happened after you changed it?

Does your code turn into a loose cannon towards the

deadline?

EFFORTS TO MINIMIZE RISK

Change is inevitable

No surprises

Feedback is critical

Frequent feedback is necessary

You want to know right away if you broke the code, isn't it?

Test to break it, break to test it.

MOTIVATION TO GOOD DESIGN

A good design leads to a good product.

In Engineering Construction is expensive, Design is relatively Cheap

In Software Development Construction is Cheap and design (which involves modeling and coding) is expensive

Can't we quickly test our design (since construction is cheap)?

Testing is the Engineering Rigor in Software Development

HOMEWORK

Review class notes.

Research software development methodologies and practices.

Start a discussion on Google Groups to clarify your doubts.