UNIT IV

DESIGN & IMPLEMENTATION

Design Engineering -Architectural Design – Detailed Design - Design process - Design Quality- Design model-User interface Design – Implementation – issues in implementation.

DESIGN ENGINEERING

The result of the software requirements analysis (SRA) usually is a specification. The design helps us turning this specification into a working system.

Architectural Design: the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.

Detailed Design: the process of refining and expanding the preliminary design of a system or component to the extent that the design is sufficiently complete to begin implementation.

Functional Design: the process of defining the working relationships among the components of a system.

Preliminary Design: the process of analyzing design alternatives and defining the architecture, components, interfaces, and timing/sizing estimates for a system or components.

Hence software design includes architectural views, but also low-level component and algorithm implementation issues. Depending on the type, a software design may be platform-independent or platform-specific.

Design Considerations

There are many aspects to consider in the design of a piece of software. The importance of each should reflect the goals the software is trying to achieve. Some of these aspects are:

Compatibility - The software is able to operate with other products that are designed for interoperability with another product.

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.

Maintainability - The software can be restored to a specified condition within a specified period of time. For example, antivirus software may include the ability to periodically receive virus definition updates in order to maintain the software's effectiveness.

Modularity - the resulting software comprises well defined, independent components. That leads to better maintainability. The components could be then implemented and tested in isolation before being integrated to form a desired software system. This allows division of work in a software development project.

Packaging - Printed material such as the box and manuals should match the style designated for the target market and should enhance usability. All compatibility information should be visible on the outside of the package.

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

Reusability - the software is able to add further features and modification with slight or no modification.

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.

Usability - The software user interface must be usable for its target user/audience. Default values for the parameters must be chosen so that they are a good choice for the majority of the users.

ARCHITECTURAL DESIGN

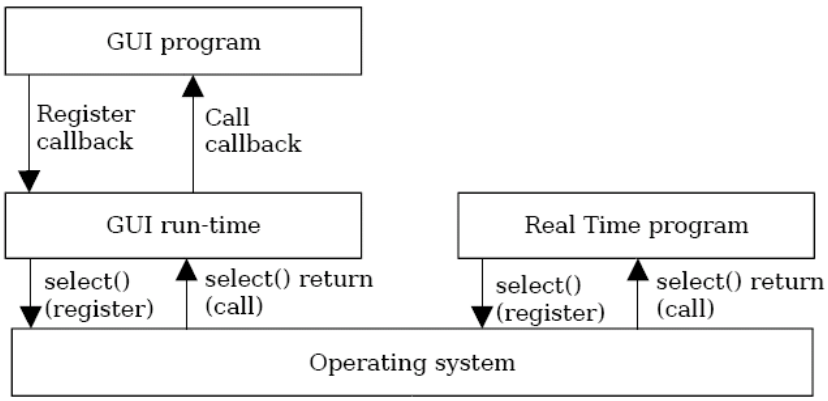
- The software architecture of a program or computing system is the structure or structures of the system which comprise
    - The software components
    - The externally visible properties of those components
    - The relationships among the components
- Software architectural design represents the structure of the data and program components that are required to build a computer-based system
- An architectural design model is transferable
    - It can be applied to the design of other systems
    - It represents a set of abstractions that enable software engineers to describe architecture in predictable ways

Architectural Design Process

- Basic Steps
    - Creation of the data design
    - Derivation of one or more representations of the architectural structure of the system
    - Analysis of alternative architectural styles to choose the one best suited to customer requirements and quality attributes
    - Elaboration of the architecture based on the selected architectural style
- A database designer creates the data architecture for a system to represent the data components
- A system architect selects an appropriate architectural style derived during system engineering and software requirements analysis

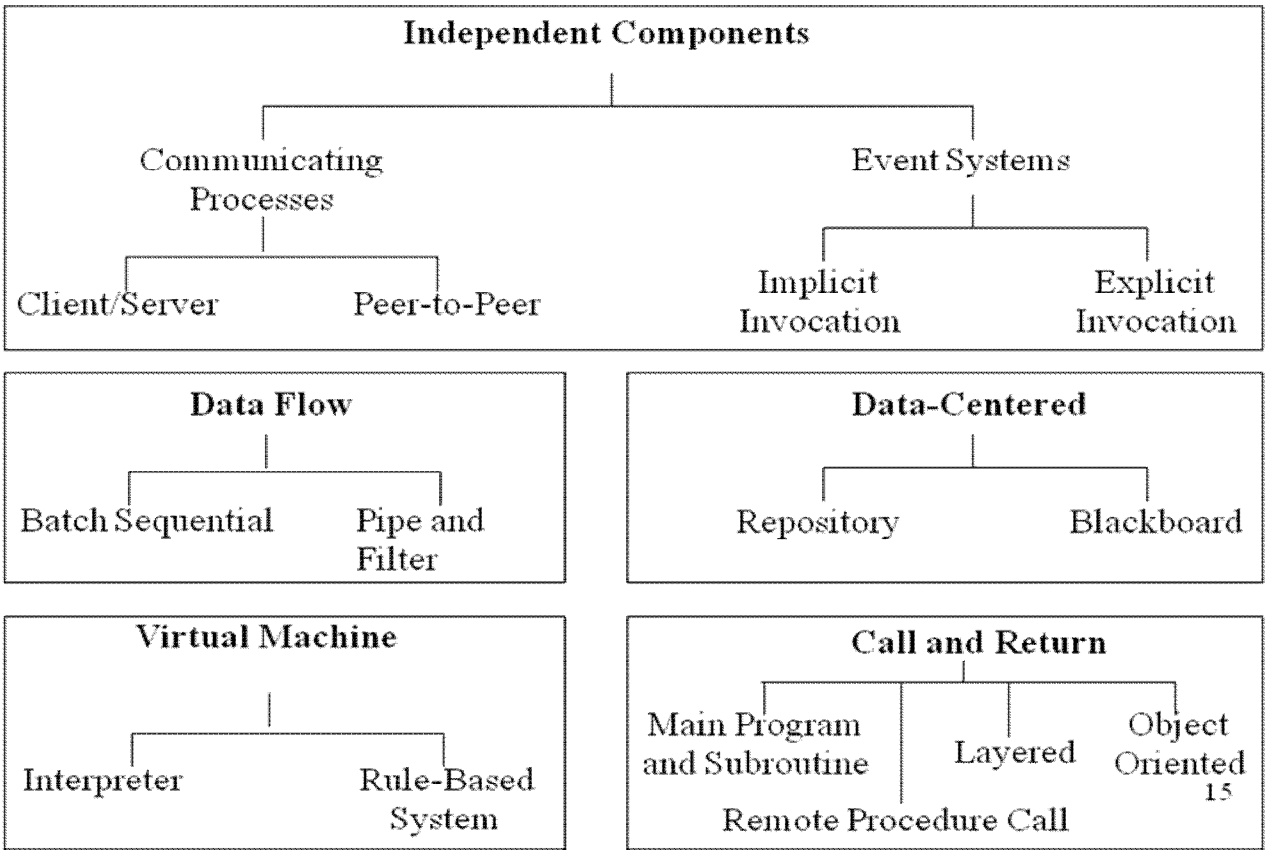
- A software architecture enables a software engineer to
    - Analyze the effectiveness of the design in meeting its stated requirements
    - Consider architectural alternatives at a stage when making design changes is still relatively easy
    - Reduce the risks associated with the construction of the software
- Focus is placed on the software component
    - A program module
    - An object-oriented class
    - A database
    - Middleware
- Representations of software architecture are an enabler for communication between all stakeholders interested in the development of a computer-based system
- The software architecture highlights early design decisions that will have a profound impact on all software engineering work that follows and, as important, on the ultimate success of the system as an operational entity

Example Software Architecture Diagrams



Software Architectural Styles

- The software that is built for computer-based systems exhibit one of many architectural styles
- Each style describes a system category that encompasses
  - A set of component types that perform a function required by the system
  - A set of connectors (subroutine call, remote procedure call, data stream, socket) that enable communication, coordination, and cooperation among components
  - Semantic constraints that define how components can be integrated to form the system
  - A topological layout of the components indicating their runtime interrelationships

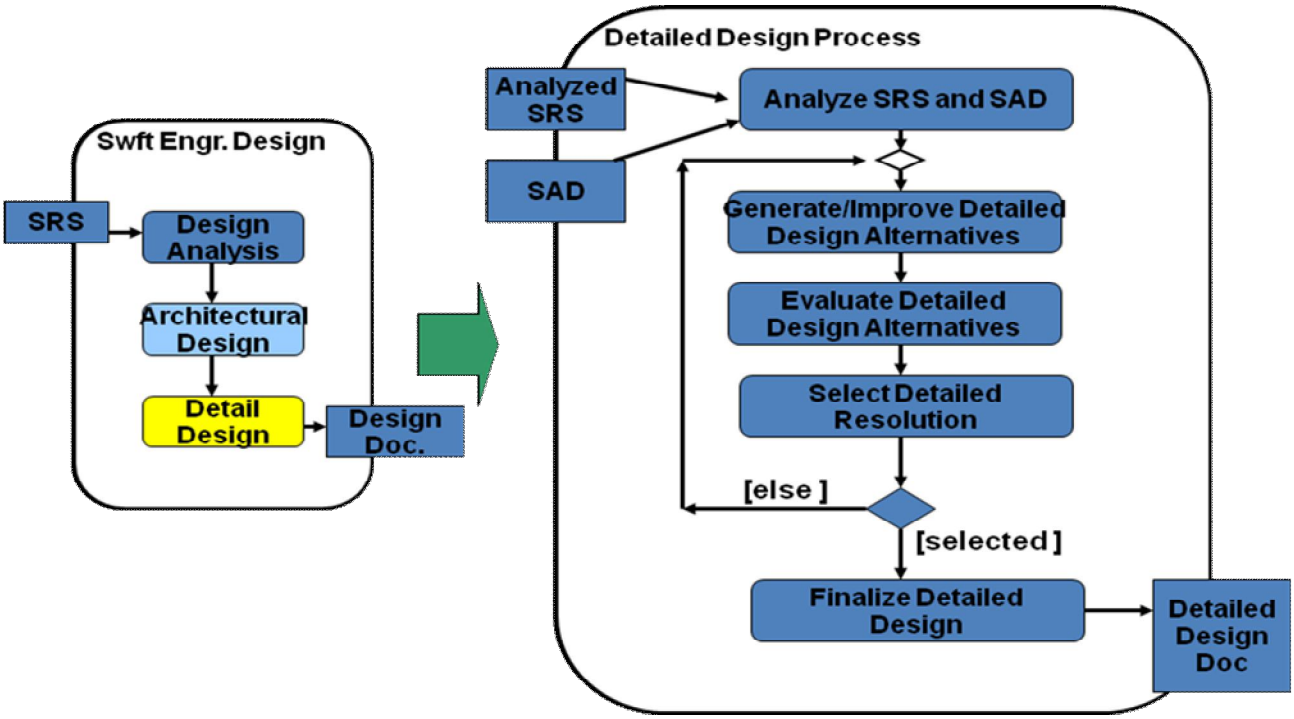


DETAILED DESIGN

- After high-level design, a designer's focus shifts to low-level design
- Each module's responsibilities should be specified as precisely as possible
- Constraints on the use of its interface should be specified
- pre and post conditions can be identified
- module-wide invariants can be specified
- internal data structures and algorithms can be suggested

Detailed Design starts with Software Architecture Design (SAD) and fill in the details of the architectural components - - - may cause re-work on SAD, too.

- Detailed Design has 2 levels:
    - *Mid-level*
    - Low-level



Mid levels of detailed design

- The mid-level design specifications include (DeSCRIPTR):

    1. Decomposed mid-level components
    2. States of these components
    3. Collaboration among these mid-level components
    4. Responsibilities of each of these mid-level components to perform some task or maintain data
    5. Interfaces that these mid-level components use to communicate with each other
    6. Properties (such as security, reliability, performance) of these components must be specified.
    7. Transition of states and state behavior of these components are described

Relationships (inheritance, aggregation, composition, etc.) among the components

Low levels of detailed design

- In contrast to mid-level, low level detailed design fills in the details for later programming purpose (PAID):

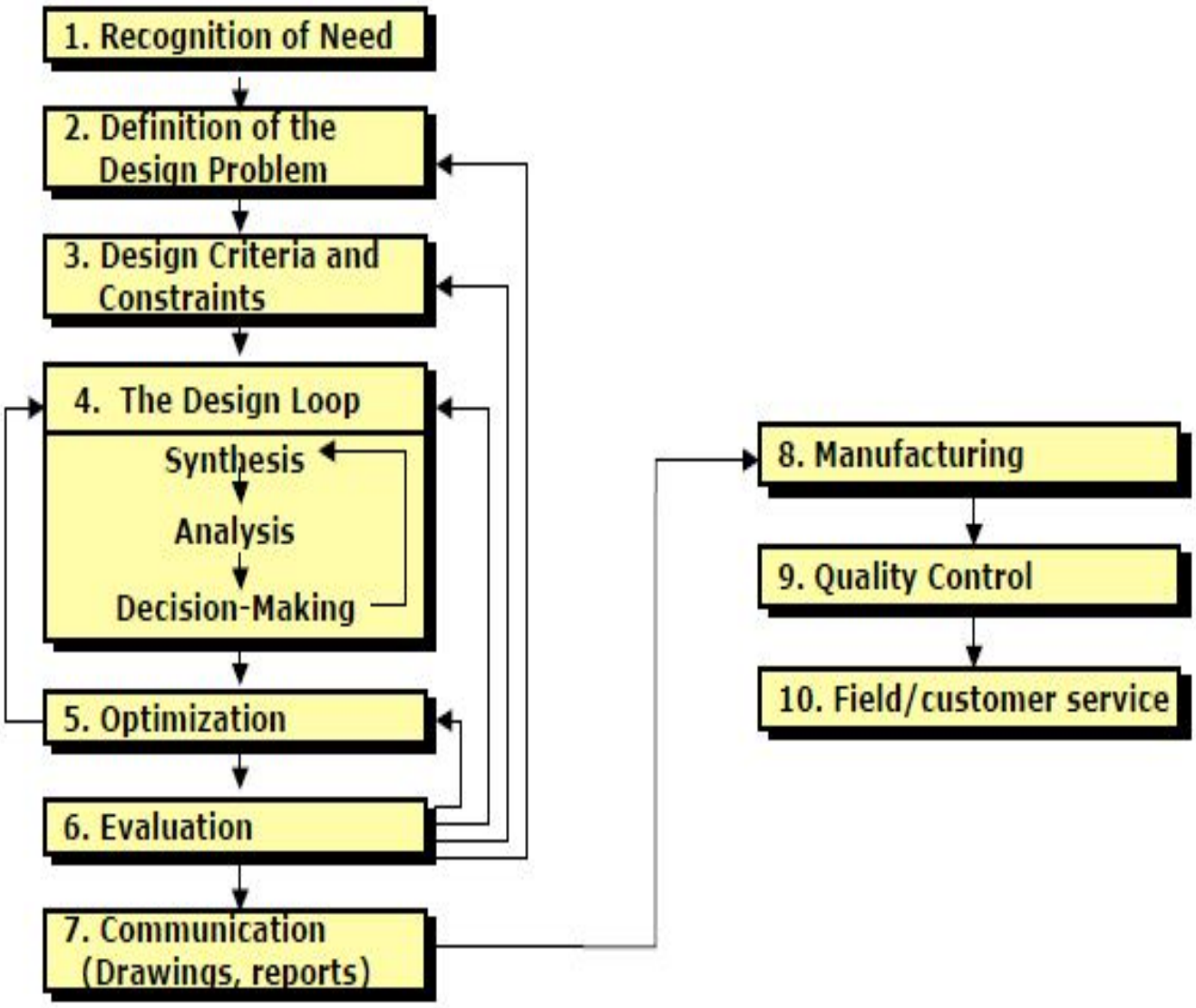
    1. Packaging of the code into compilation unit and libraries is decided
    2. Algorithms are described in detail steps
    3. Implementation issues such as visibility and accessibility of entities and low level relationships among entities are described
    4. Data structures and data types are specified

DESIGN PROCESS

An engineering design is a model of the product or structure to be engineered. The model is used to

- Evaluate suitability of proposed product/system
- Communicate proposed product to others

An engineering design process describes a set of steps for constructing an engineering design.



DESIGN QUALITY

- The design must implement all of the explicit requirements contained in the analysis model, and it must accommodate all of the implicit requirements desired by the customer.
- The design must be a readable, understandable guide for those who generate code and for those who test and subsequently support the software.
- The design should provide a complete picture of the software, addressing the data, functional, and behavioral domains from an implementation perspective.

Quality guidelines

- A design should exhibit an architecture that
  - (1) has been created using recognizable architectural styles or patterns,
  - (2) is composed of components that exhibit good design characteristics and
  - (3) can be implemented in an evolutionary fashion
    - For smaller systems, design can sometimes be developed linearly.
- A design should be modular; that is, the software should be logically partitioned into elements or subsystems
- A design should contain distinct representations of data, architecture, interfaces, and components.
- A design should lead to data structures that are appropriate for the classes to be implemented and are drawn from recognizable data patterns.
- A design should lead to components that exhibit independent functional characteristics.

- A design should lead to interfaces that reduce the complexity of connections between components and with the external environment.
- A design should be derived using a repeatable method that is driven by information obtained during software requirements analysis.
- A design should be represented using a notation that effectively communicates its meaning.

USER INTERFACE DESIGN

- System users often judge a system by its interface rather than its functionality
- A poorly designed interface can cause a user to make catastrophic errors
- Poor user interface design is the reason why so many software systems are never used
- Most users of business systems interact with these systems through graphical interfaces although, in some cases, legacy text-based interfaces are still used

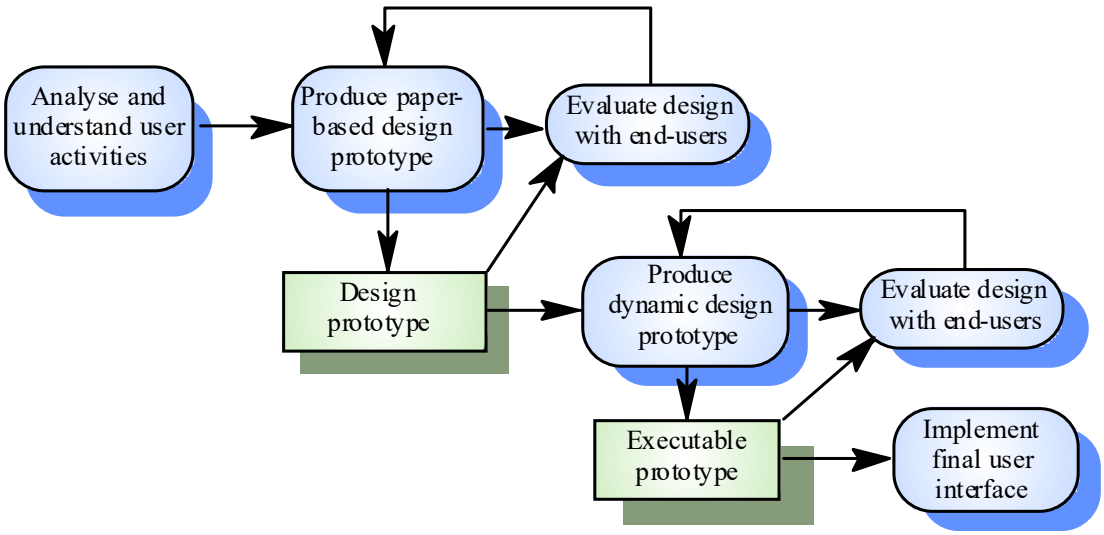
Characteristics of Graphical User Interface (GUI)

Characteristic	Description
Windows	Multiple windows allow different information to be displayed simultaneously on the user's screen.
Icons	Icons different types of information. On some systems, icons represent files; on others, icons represent processes.
Menus	Commands are selected from a menu rather than typed in a command language.
Pointing	A pointing device such as a mouse is used for selecting choices from a menu or indicating items of interest in a window.
Graphics	Graphical elements can be mixed with text on the same display.

GUI Advantages

- They are easy to learn and use.
    - Users without experience can learn to use the system quickly.
- The user may switch quickly from one task to another and can interact with several different applications.
    - Information remains visible in its own window when attention is switched.
- Fast, full-screen interaction is possible with immediate access to anywhere on the screen

User Interface Design Process



UI design principles

- UI design must take account of the needs, experience and capabilities of the system users
- Designers should be aware of people's physical and mental limitations (e.g. limited short-term memory) and should recognise that people make mistakes
- UI design principles underlie interface designs although not all principles are applicable to all designs

Principle	Description
User familiarity	The interface should use terms and concepts which are drawn from the experience of the people who will make most use of the system.
Consistency	The interface should be consistent in that, wherever possible, comparable operations should be activated in the same way.
Minimal surprise	Users should never be surprised by the behaviour of a system.
Recoverability	The interface should include mechanisms to allow users to recover from errors.
User guidance	The interface should provide meaningful feedback when errors occur and provide context-sensitive user help facilities.
User diversity	The interface should provide appropriate interaction facilities for different types of system user.

IMPLEMENTATION ISSUES

•Focus here is not on programming, although this is obviously important, but on other implementation issues that are often not covered in programming texts:

–Reuse Most modern software is constructed by reusing existing components or systems. When you are developing software, you should make as much use as possible of existing code.

–Configuration management During the development process, you have to keep track of the many different versions of each software component in a configuration management system.

–Host-target development Production software does not usually execute on the same computer as the software development environment. Rather, you develop it on one computer (the host system) and execute it on a separate computer (the target system).