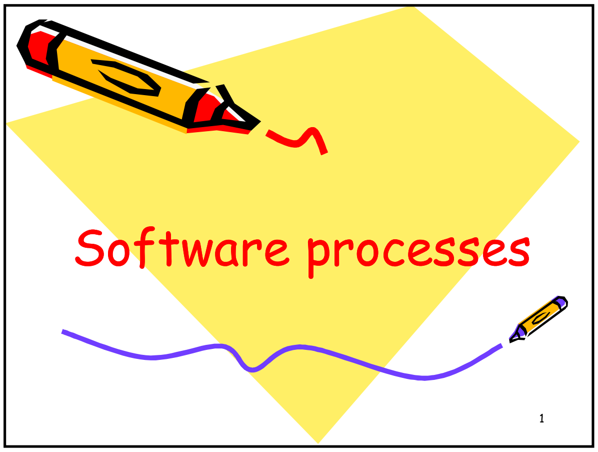
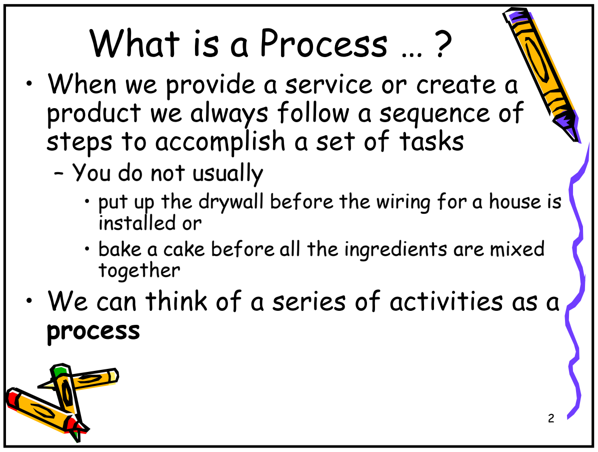
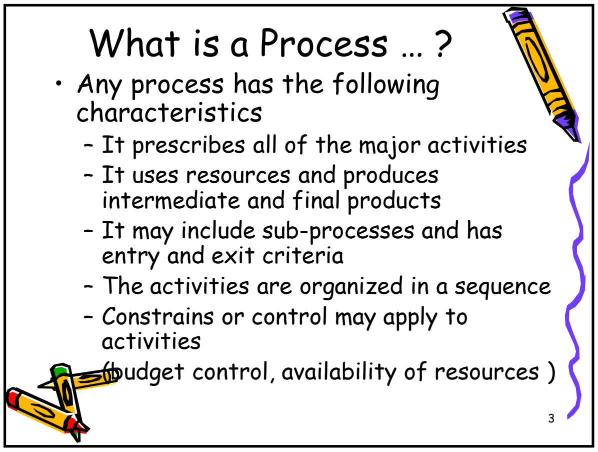


Software processes



What is a Process … ?

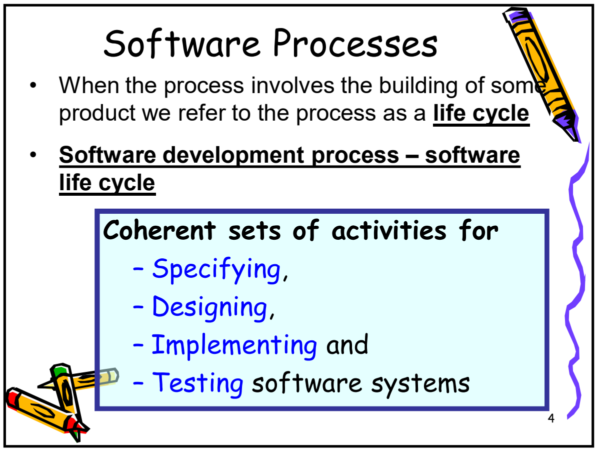
- When we provide a service or create a product we always follow a sequence of steps to accomplish a set of tasks
  - You do not usually
    - put up the drywall before the wiring for a house is installed or
    - bake a cake before all the ingredients are mixed together
- We can think of a series of activities as a **process**



What is a Process … ?

- Any process has the following characteristics
  - It prescribes all of the major activities
  - It uses resources and produces intermediate and final products
  - It may include sub-processes and has entry and exit criteria
  - The activities are organized in a sequence
  - Constrains or control may apply to activities

    (budget control, availability of resources)

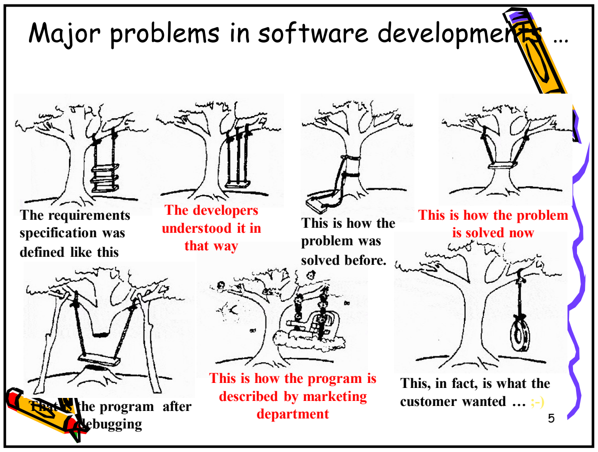


Software Processes

- When the process involves the building of some product we refer to the process as a **life cycle**
- **Software development process – software life cycle**

Coherent sets of activities for

- Specifying,
- Designing,
- Implementing and
- Testing software systems



Major problems in software developments …

The requirements specification was defined like this

The developers understood it in that way

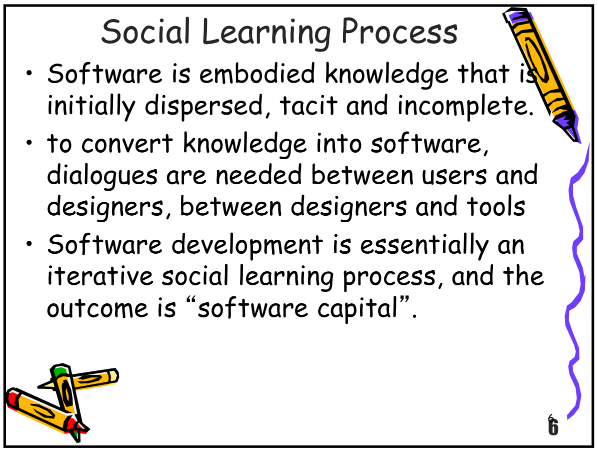
This is how the problem was solved before.

This is how the problem is solved now

That's the program after debugging

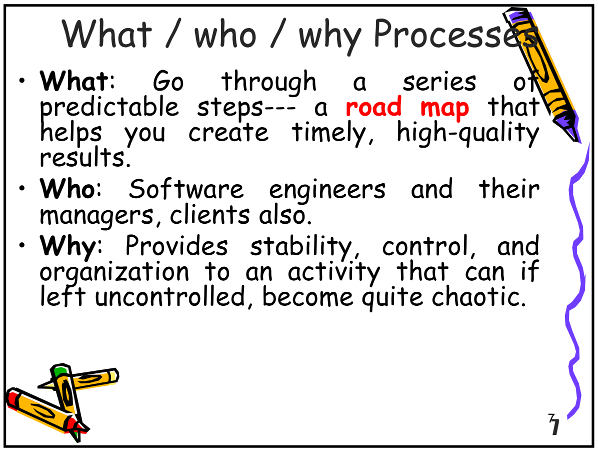
This is how the program is described by marketing department

This, in fact, is what the customer wanted … ;-)



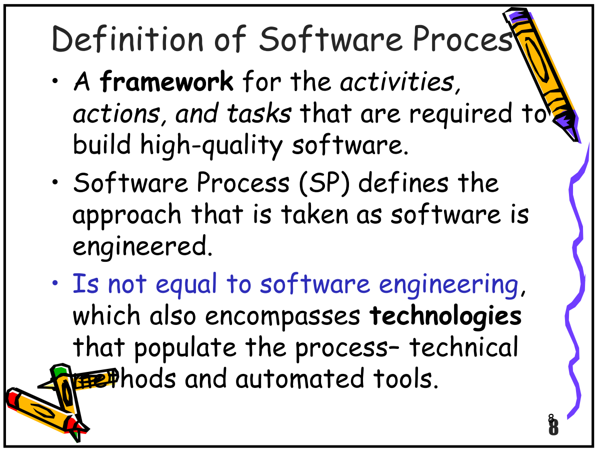
Social Learning Process

- Software is embodied knowledge that is initially dispersed, tacit and incomplete.
- to convert knowledge into software, dialogues are needed between users and designers, between designers and tools
- Software development is essentially an iterative social learning process, and the outcome is “software capital”.



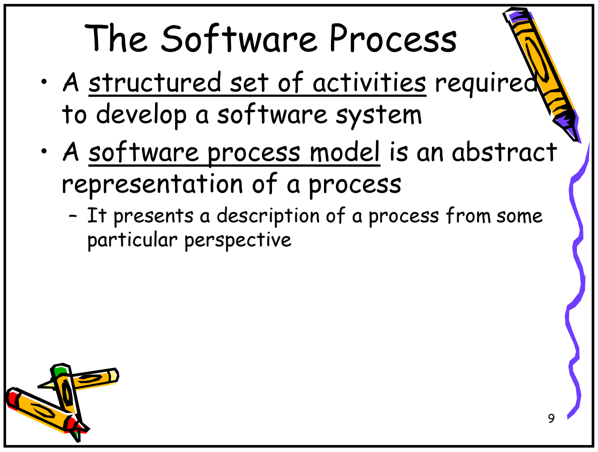
What / who / why Processes

- **What**: Go through a series of predictable steps--- a **road map** that helps you create timely, high-quality results.
- **Who**: Software engineers and their managers, clients also.
- **Why**: Provides stability, control, and organization to an activity that can if left uncontrolled, become quite chaotic.



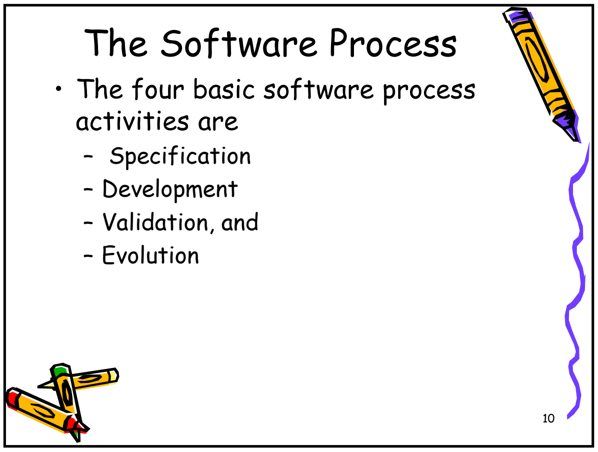
Definition of Software Process

- A **framework** for the *activities, actions, and tasks* that are required to build high-quality software.
- Software Process (SP) defines the approach that is taken as software is engineered.
- Is not equal to software engineering, which also encompasses **technologies** that populate the process– technical methods and automated tools.



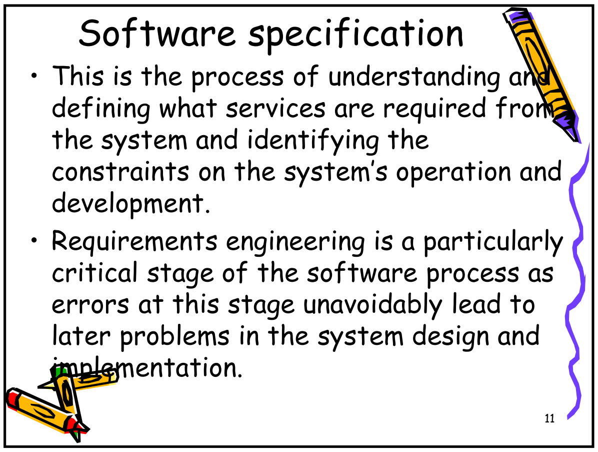
The Software Process

- A structured set of activities required to develop a software system
- A software process model is an abstract representation of a process
  - It presents a description of a process from some particular perspective



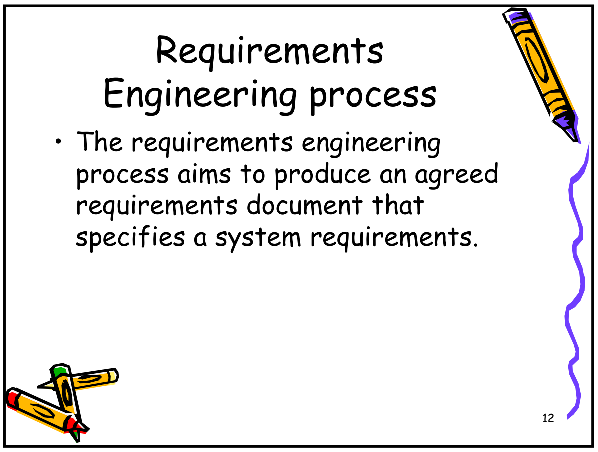
The Software Process

- The four basic software process activities are
  - Specification
  - Development
  - Validation, and
  - Evolution



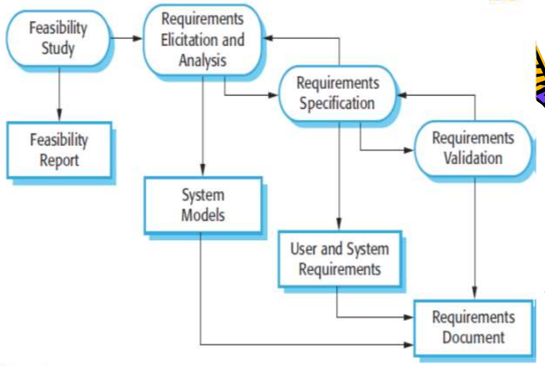
Software specification

- This is the process of understanding and defining what services are required from the system and identifying the constraints on the system's operation and development.
- Requirements engineering is a particularly critical stage of the software process as errors at this stage unavoidably lead to later problems in the system design and implementation.



Requirements Engineering process

- The requirements engineering process aims to produce an agreed requirements document that specifies a system requirements.



Requirements Engineering process

- There are four main activities in the requirements engineering process:
- Feasibility Study
- Requirements elicitation and Analysis
- Requirements specification
- Requirements Validation

Feasibility study

- An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies.
- The study considers whether the proposed system will be cost-effective from a business point of view and if it can be developed within existing budgetary constraints.

Feasibility study

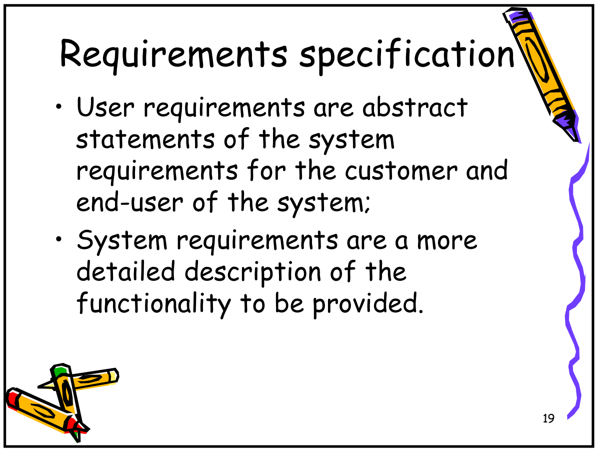
- A feasibility study should be relatively cheap and quick.
- The result should inform the decision of whether or not to go ahead with a more detailed analysis (feasibility report).

Requirements elicitation and analysis

- This is the process of deriving the system requirements through observation of existing systems, discussions with potential users and buyer, task analysis.
- This may involve the development of one or more system models and prototypes.

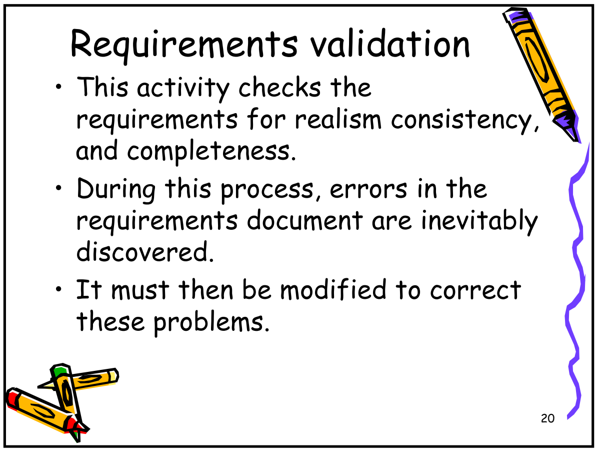
Requirements specification

- Requirements specification is the activity of translating the information gathered during the analysis activity into a document that defines a set of requirements.
- Two types of requirements may be included in this document.



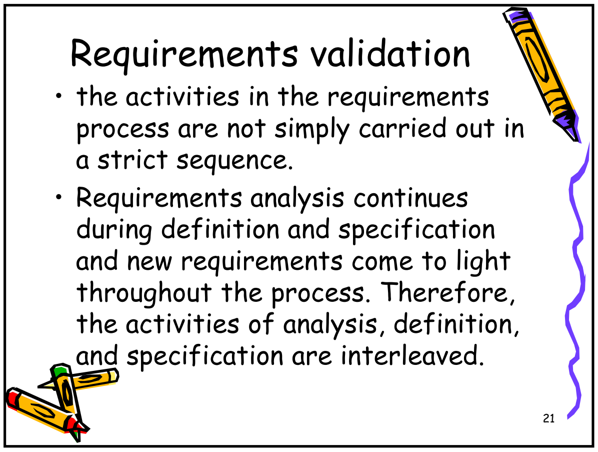
Requirements specification

- User requirements are abstract statements of the system requirements for the customer and end-user of the system;
- System requirements are a more detailed description of the functionality to be provided.



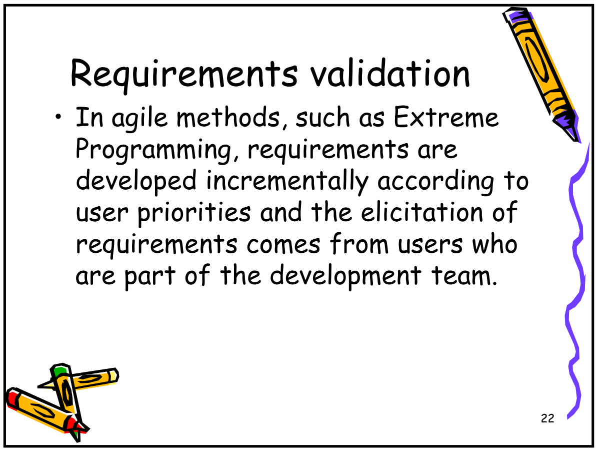
Requirements validation

- This activity checks the requirements for realism consistency, and completeness.
- During this process, errors in the requirements document are inevitably discovered.
- It must then be modified to correct these problems.



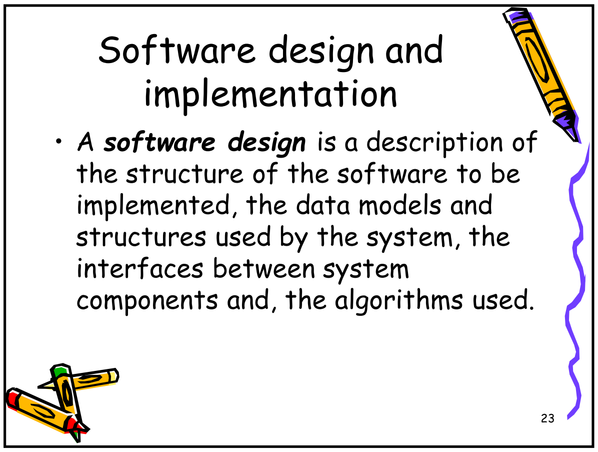
Requirements validation

- the activities in the requirements process are not simply carried out in a strict sequence.
- Requirements analysis continues during definition and specification and new requirements come to light throughout the process. Therefore, the activities of analysis, definition, and specification are interleaved.



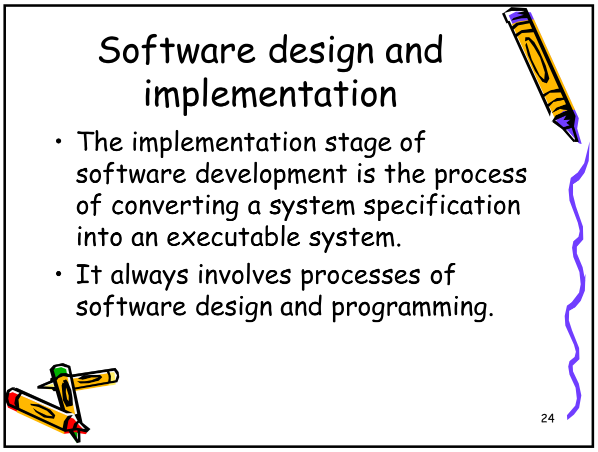
Requirements validation

- In agile methods, such as Extreme Programming, requirements are developed incrementally according to user priorities and the elicitation of requirements comes from users who are part of the development team.



Software design and implementation

- A ***software design*** is a description of the structure of the software to be implemented, the data models and structures used by the system, the interfaces between system components and, the algorithms used.

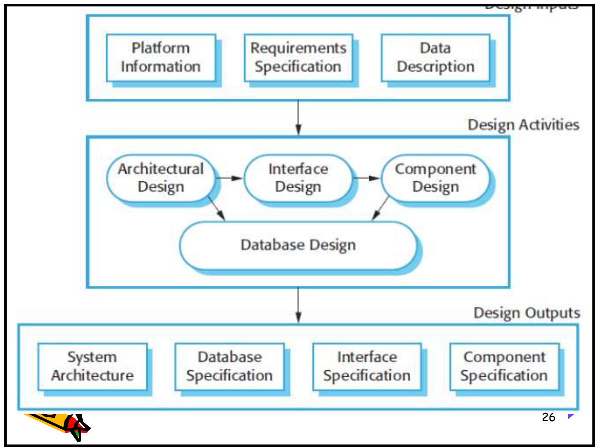


Software design and implementation

- The implementation stage of software development is the process of converting a system specification into an executable system.
- It always involves processes of software design and programming.

Software design and implementation

- The next Figure is an abstract model of this process showing the inputs to the design process, process activities, and the documents produced as outputs from this process



Design Inputs

Platform Information

Requirements Specification

Data Description

Design Activities

Architectural Design

Interface Design

Component Design

Database Design

Design Outputs

System Architecture

Database Specification

Interface Specification

Component Specification

Design Inputs

- Platform Information
  - 'software platform', is the environment in which the software will execute.
  - Information about this platform is an essential input to the design process, as designers must decide how best to integrate it with the software's environment.

Design Inputs

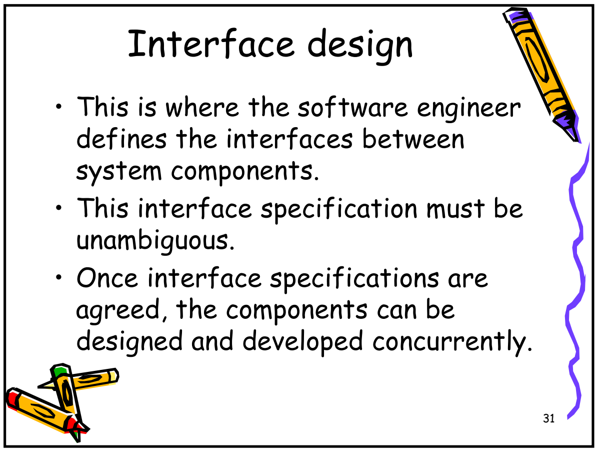
- The requirements specification
  - This is a description of the functionality the software must provide and its performance and dependability requirements.
- Data Description
  - If the system is to process existing data, then the description of that data may be included in the platform specification; otherwise, the data description must be an input to the design process.

The Design Process

- There are four activities that may be part of the design process for information systems as shown in the previous figure:
  - Architectural Design
  - Interface Design
  - Component Design
  - Database Design

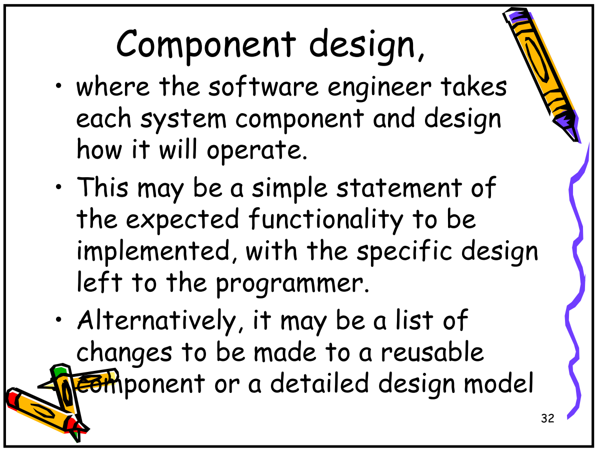
Architectural design

- This is where the software engineer identifies the overall structure of the system, the principal components (sometimes called sub-systems or modules), their relationships, and how they are distributed.



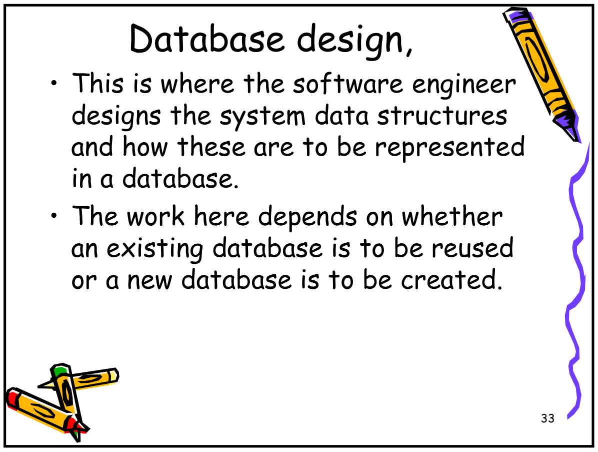
Interface design

- This is where the software engineer defines the interfaces between system components.
- This interface specification must be unambiguous.
- Once interface specifications are agreed, the components can be designed and developed concurrently.



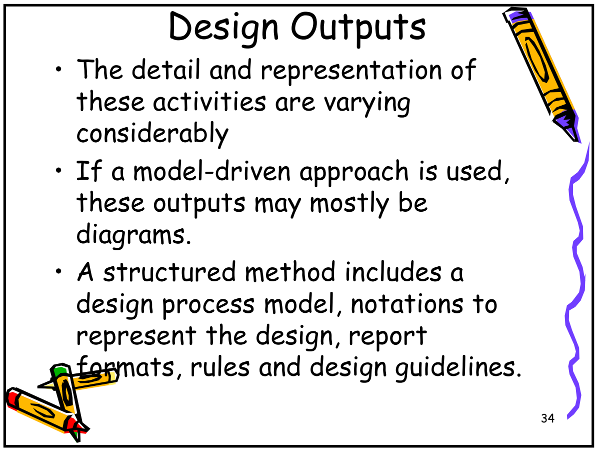
Component design,

- where the software engineer takes each system component and design how it will operate.
- This may be a simple statement of the expected functionality to be implemented, with the specific design left to the programmer.
- Alternatively, it may be a list of changes to be made to a reusable component or a detailed design model



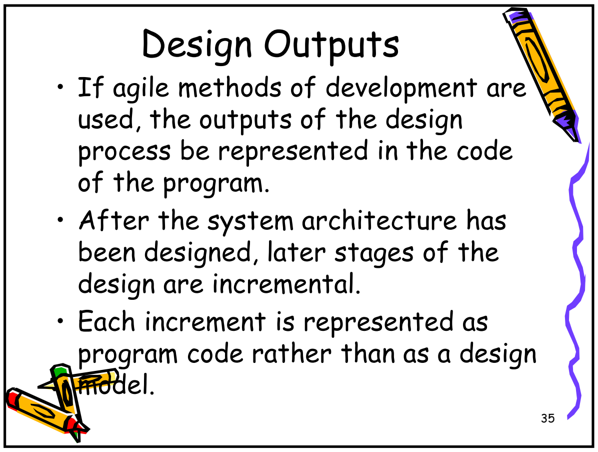
Database design,

- This is where the software engineer designs the system data structures and how these are to be represented in a database.
- The work here depends on whether an existing database is to be reused or a new database is to be created.



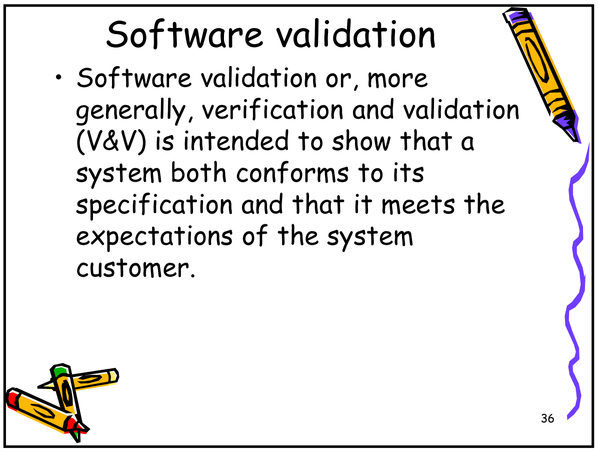
Design Outputs

- The detail and representation of these activities are varying considerably
- If a model-driven approach is used, these outputs may mostly be diagrams.
- A structured method includes a design process model, notations to represent the design, report formats, rules and design guidelines.



Design Outputs

- If agile methods of development are used, the outputs of the design process be represented in the code of the program.
- After the system architecture has been designed, later stages of the design are incremental.
- Each increment is represented as program code rather than as a design model.



Software validation

- Software validation or, more generally, verification and validation (V&V) is intended to show that a system both conforms to its specification and that it meets the expectations of the system customer.

Software validation

- Program testing, where the system is executed using simulated test data, is the principal validation technique.
- Validation may also involve checking processes, such as inspections and reviews, at each stage of the software process from user requirements definition to program development.

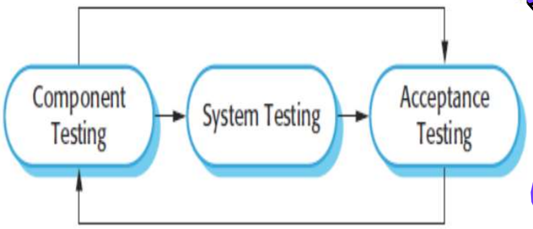
Software validation

- The next figure shows a three-stage testing process in which system components are tested then
- The integrated system is tested and, finally,
- the system is tested with the customer's data.
- Ideally, component defects are discovered early in the process, and interface problems are found when the system is integrated.

Software validation

- However, as defects are discovered, the program must be debugged and this may require other stages in the testing process to be repeated.
- Errors in program components are brought to light during system testing.
- The process is therefore an iterative one with information being fed back from later stages to earlier parts of the process.

Stages of testing



Component (or unite) testing:

- Individual components are tested to ensure that they operate correctly.
- Each component is tested independently, without other system components.
- Components may be simple entities such as functions or object classes, or may be coherent groupings of these entities.

System testing:

- System components are integrated to create a complete system.
- This process is concerned with finding errors that result from not expected interactions between components and component interface problems.

System testing:

- It is also concerned with showing that the system meets its functional and non-functional requirements, and testing the emergent system properties.
- For large systems, this may be a multi-stage process where components are integrated to form sub- systems that are individually tested before these sub-systems are themselves integrated to form the final

43

Acceptance testing:

- This is the final stage in the testing process before the system is accepted for operational use.
- The system is tested with data supplied by the system customer rather than with simulated test data.

44

Acceptance testing:

- Acceptance testing may reveal errors and omissions in the system requirements definition, because the real data exercise the system in different ways from the test data.
- Acceptance testing may also reveal requirements problems where the system's facilities do not really meet the user's needs or the system performance is unacceptable.

45

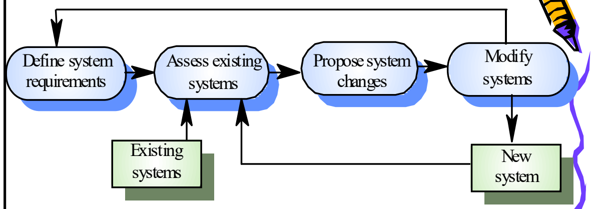
Software evolution

Software is inherently flexible and can change.

- As requirements change through changing business circumstances, the software that supports the business must also evolve and change
- Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new

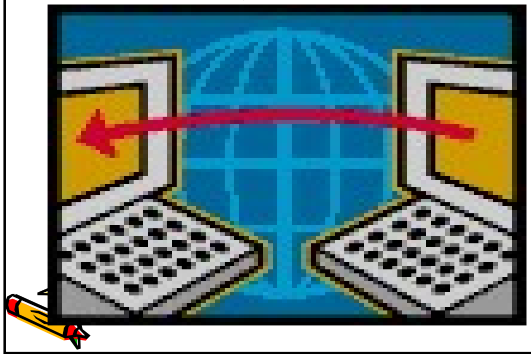
46

System evolution



47

Questions



48