Peter Mileff PhD

SOFTWARE ENGINEERING

Software Specification
Software Design and Implementation
Software Validation

University of Miskolc
Department of Information Technology

Software Specification...

Software Specification

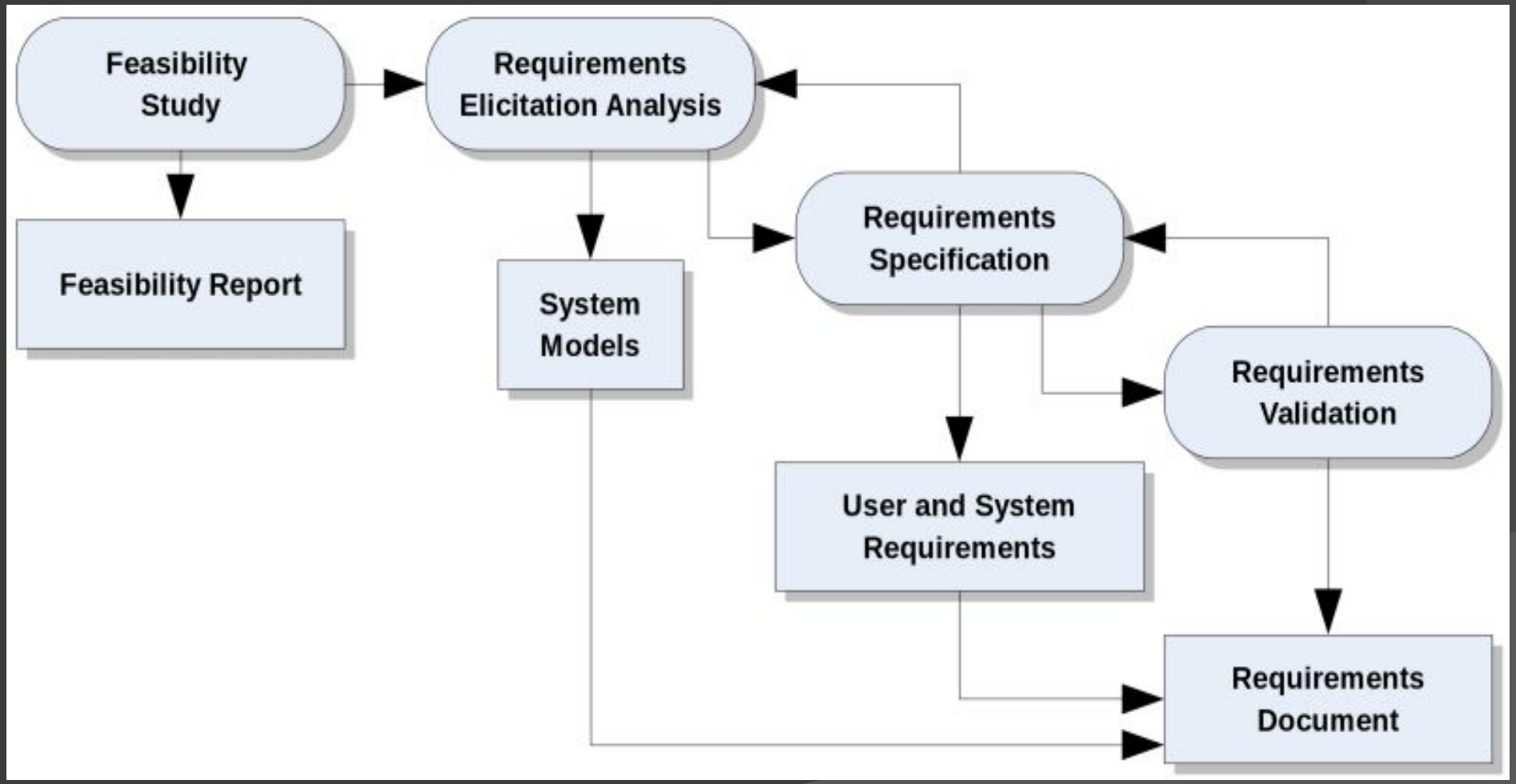
- **Software specification is a process, where**
  - we understand and define what services are required from the system
  - we identify the constraints on the system's operation and development.

- It is a particularly **critical stage** of the software process
  - errors at this stage inevitably lead to later problems in the system design and implementation.

Software Specification

- The process aims to produce **an agreed requirements document**
  - that specifies a system satisfying stakeholder requirements.
- Requirements are usually presented at two levels of detail:
  - **End-users and customers**
    - need a high-level statement of the requirements;
  - **System developers:**
    - need a more detailed system specification.

The Software Specification process



Four main activities of the process

- **Feasibility study:**
- This is an estimation process:
  - verify that software is feasible using current software and hardware technologies.
- **The result of the process is a study**
- The study considers the followings:
  - whether the proposed system will be cost-effective from a business point of view and
  - if it can be developed within existing budgetary constraints.
- 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.

Four main activities of the process

- **2. Requirements elicitation and analysis**
  - This process collects the system requirements
    - through observation of existing systems
  - Discussions with potential users and procurers and stakeholders
    - Talk to everybody who can provide information about the new software
  - The process may involve the development of one or more system models and prototypes.
    - These help you understand the system to be specified.

Four main activities of the process

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

Four main activities of the process

- **4. Requirements validation**
  - This activity checks the requirements for realism, consistency, and completeness.
    - Are they feasible?
    - Are they rational?
    - Is something missing?
  - The aim is to discover errors in the requirements document
    - It must then be modified to correct these problems.

Four main activities of the process

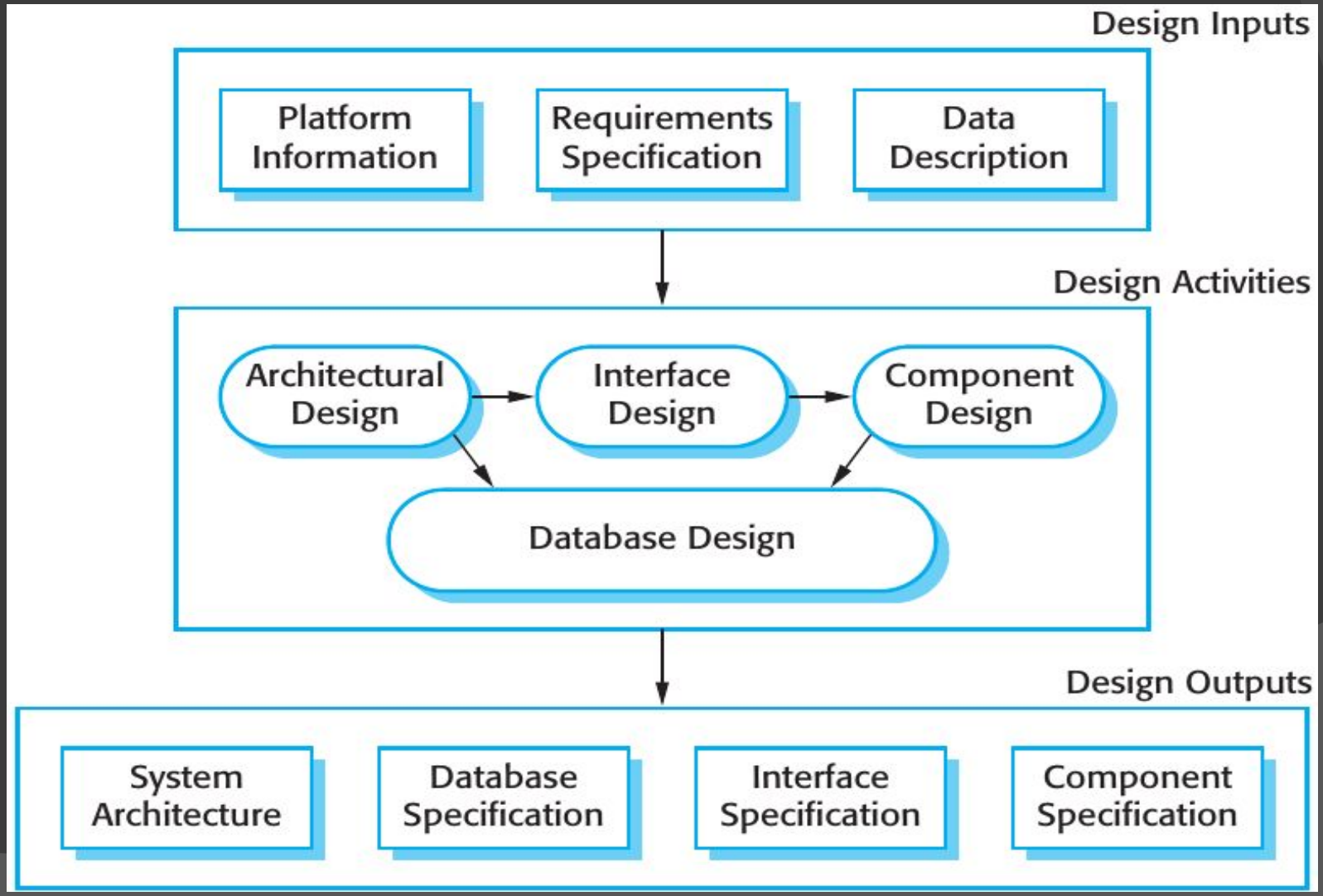
- The activities in the requirements process are not simply carried out in a strict sequence.
  - Requirements analysis continues during definition and specification
  - New requirements come to light throughout the process.
- Therefore, **the activities of analysis, definition, and specification are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

Software design and implementation...

The Software design process

- **Software design:** is a description of the structure of the software to be implemented
  - the data models and structures
  - the interfaces between system components
  - sometimes, the used algorithms
- Designers develop the design iteratively
  - A structure cannot be described without iteration
  - New information may become available
    - as we go forward
- They add formality and detail as they develop their design with constant backtracking to correct earlier designs.

The Software design process



The Software design process

- The diagram suggests that the stages of the design process are sequential.
  - However design process activities are interleaved.
  - Feedback from one stage to another inevitable in all design processes.
- The activities in the design process can vary
  - depending on the type of system being developed.
- **Example:**
  - real-time systems require timing design
    - but may not include a database
      - there is no database design involved.

Design activities

1. Architectural design:

- here we identify the overall structure of the system
  - What are the principal components
    - sometimes called sub-systems or modules
  - What are their relationships, and how they are distributed.

2. Interface design:

- defines the interfaces between system components.
- This interface specification must be unambiguous.
  - With a precise interface, a component can be used without other components having to know how it is implemented.
- Once interface specifications are agreed,
  - the components can be designed and developed concurrently.

Design activities

3. Component design:

- we take each system component and design how it will operate.
  - This may be a simple statement of the expected functionality to be implemented
    - 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.

- The design model may be used to automatically generate an implementation.

Design activities

4. Database design:

- Design the system data structures
  - if software functionality requires a database
  - how the data structures are to be represented in a database
    - database type, table names, connections, stored procedures, etc

- The work here depends on whether an existing database is to be reused or a new database is to be created.

Design activities

- These activities lead to a set of design outputs
  - Architectural Design
  - Database Specification
  - Interface Specification
  - Component Specification
- The detail and representation of these can be different
  - depends on the software we design
- For critical systems
  - detailed design documents
  - and accurate descriptions must be produced
- In other cases
  - Sometimes “lighter-style” documents with diagrams are enough
  - Or design is represented only in the code of the program

Implementation

- The implementation follows naturally the system design processes
  - It is more common for the later stages of design and program development to be interleaved.
- Software development tools may be used to generate a skeleton program from a design
- Programming is a personal activity and there is no general process that is usually followed.
  - Some programmers start with components that they understand
    - develop these, and then move on to less-understood components.
  - Others take the opposite approach

Implementation

- **Basic rules of development:**
  - **Well working code:** the code performs what is expected.
    - Reacts adequately even in case of an error.
      - no freezing
  - **Aesthetically adequate code:** the code shall be readable. We cannot presume that the code will be read only by us
    - we should apply certain rules that enhance readability.
      - For example the "**CamelCase**" naming convention.
      - Often the company specifies these rules by which it ensures subsequent further development of its software.

Implementation

- **Documented code:**
  - the quality of a code increases if it is documented on a certain level.
  - Especially good if we modify the code of somebody else
    - it is a great help if at the beginning of the fourfold for cycle you find a brief explanation on its aim.
- Comments should answer basic questions:
  - What does this code part do?
  - Why are these lines necessary?
  - Why should we be careful at the modification?
  - etc.

Implementation

- Normally, programmers carry out some testing of the code they have developed
  - This often reveals program defects that must be removed from the program.
  - This is called debugging.
- **Defect testing and debugging are different processes.**
  - Testing establishes the existence of defects.
  - Debugging is concerned with locating and correcting these defects.
- IDEs usually support debugging
  - the offered interactive debugging tools are usually very intelligent

Software validation...

Software Validation

- **More generally:** Verification and Validation (V&V)
- **Goal:** to show that a system conforms
  - to its specification (**Verification**)
  - and it meets the expectations of the system customer (**Validation**)
- The principal validation technique is program testing.
  - where the system is executed using simulated test data
- Validation may also involve other checking processes
  - at each stage of the software process from user requirements definition to program development.
    - such as inspections and reviews
    - verify the requirements, the specification document, the design, etc

Software evolution...

Software Evolution

- **Software development does not stop when a system is delivered**
  - but continues throughout the lifetime of the system.
- After a system has been deployed, it inevitably has to change if it is **to remain useful**.
- The usually reason of continuously evolution:
  - Business changes
    - new user expectations generate new requirements for the existing software
  - Parts of the software may have to be modified
    - to correct errors that are found in operation,
    - to adapt it for changes to its hardware and software platform,
    - or to improve its performance or other non-functional characteristics

Software Evolution

- **Software evolution is important!**
  - organizations have invested large amounts of money in their software
    - they are now completely dependent on these systems
  - Usually large companies spend more on maintaining existing systems than on new systems development
- **As the software is modified:**
  - its structure tends to degrade and changes become more and more expensive.
    - it is getting harder to maintain the codebase
  - This often happens after a few years of use
    - when other environmental changes, such as hardware and operating systems, are also often required.

Thank you for your attention!