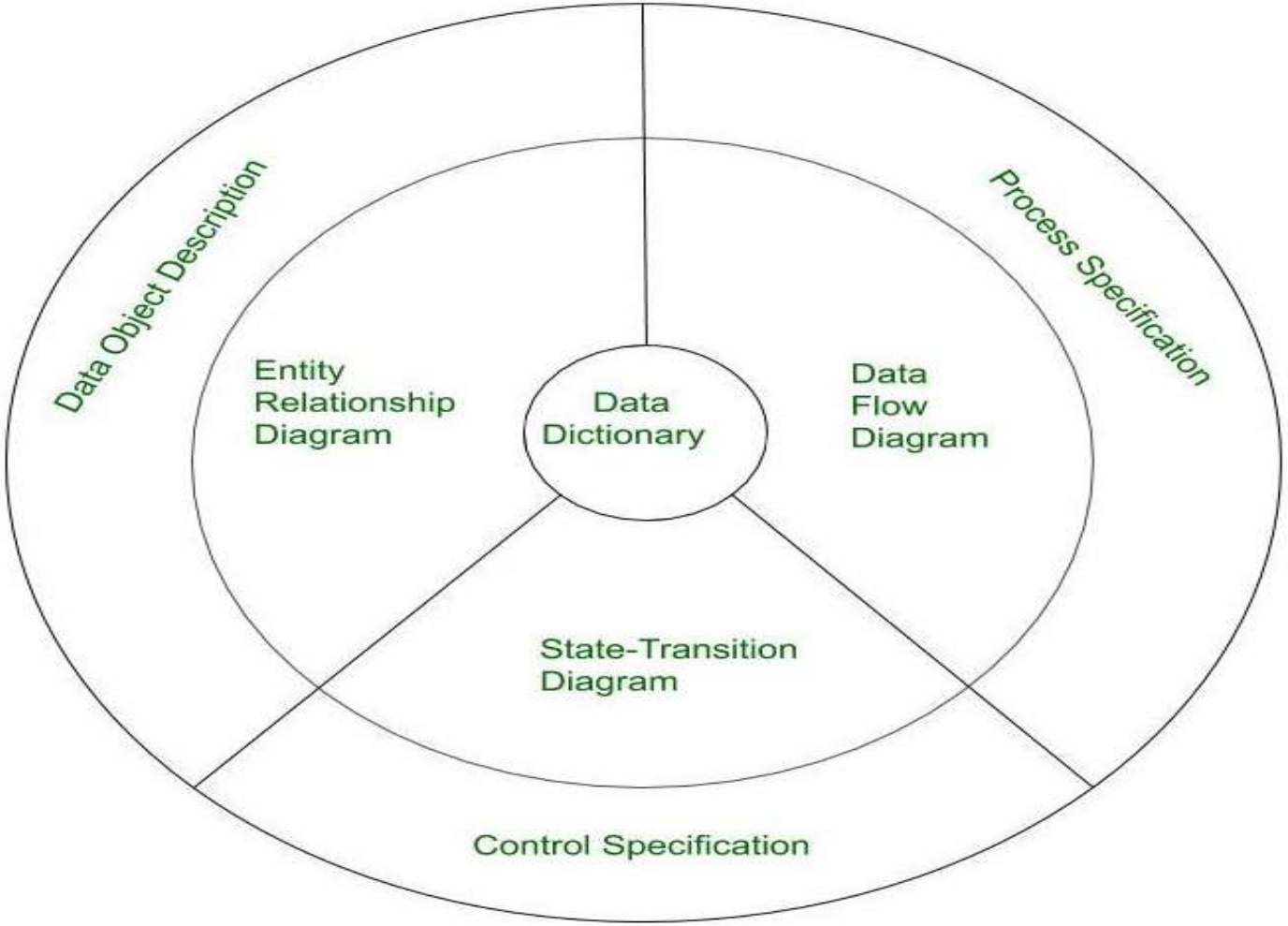
Design Engineering

Analysis Modelling in Software Engineering

- **Analysis Model** is a technical representation of the system.
- It acts as a link between the system description and the design model.
- In Analysis Modelling, information, behavior, and functions of the system are defined and translated into the architecture, component, and interface level design in the design modeling.

- **Objectives of Analysis Modelling:**
- It must establish a way of creating software design.
- It must describe the requirements of the customer.
- It must define a set of requirements that can be validated, once the software is built.

Elements of Analysis Model:



1. **Data Dictionary:**
   It is a repository that consists of a description of all data objects used or produced by the software. It stores the collection of data present in the software. It is a very crucial element of the analysis model. It acts as a centralized repository and also helps in modeling data objects defined during software requirements.

2. **Entity Relationship Diagram (ERD):**
   It depicts the relationship between data objects and is used in conducting data modeling activities. The attributes of each object in the Entity-Relationship Diagram can be described using Data object description. It provides the basis for activity related to data design.

3.Data Flow Diagram (DFD):

It depicts the functions that transform data flow, and it also shows how data is transformed when moving from input to output. It provides the additional information which is used during the analysis of the information domain and serves as a basis for the modeling of function.

4.State Transition Diagram:

It shows various modes of behavior (states) of the system and also shows the transitions from one state to another state in the system. It also provides the details of how the system behaves due to the consequences of external events. It represents the behavior of a system by presenting its states and the events that cause the system to change state. It also describes what actions are taken due to the occurrence of a particular event.

5.Process Specification:
It stores the description of each function present in the data flow diagram. It describes the input to a function, the algorithm that is applied for the transformation of input, and the output that is produced..

6.Control Specification:
It stores additional information about the control aspects of the software. It is used to indicate how the software behaves when an event occurs and which processes are invoked due to the occurrence of the event.

7.Data Object Description:
It stores and provides complete knowledge about a data object present and used in the software. It also gives us the details of attributes of the data object present in the Entity Relationship Diagram.

Elements of the Analysis Model

The specific elements of the analysis model are dictated by the analysis modeling method that is to be used. However, a set of generic elements is common to most analysis models.

Scenario-based elements: The system is described from the user's point of view using a scenario-based approach.

It always a good idea to get stakeholders involved. One of the best ways to do this is to have each stakeholder write use-cases that describe how the software engineering models will be used.

Functional—processing narratives for software functions

Use-case- descriptions of the interaction between an "actor" and the system.

Class-based elements: Each usage scenario implies a set of "objects" that are manipulated as an actor interacts with the system.

These objects are categorized into classes- a collection of things that have similar attributes and common behaviors.

Flow-oriented elements: Information is transformed as it flows through a computer-based system. The system accepts input in a variety of forms; applies functions to transform it; and produces output in a variety of forms.

Behavioral elements: The state diagram is one method for representing the behavior of a system by depicting its states and the events that cause the system to change state.

A *state* is any observable mode of behavior. Moreover, the state diagram indicates what actions are taken as a consequence of a particular event.

Design Engineering

- Software design encompasses the set of principles, concepts, and practices that lead to the development of a high-quality system or product.
- The goal of design is to produce a model or representation that exhibits firmness, commodity, and delight software.
- Software design changes continually as new methods, better analysis, and broader understanding evolve

- Software design sits at the technical kernel of software engineering and is applied regardless of the software process model that is used.
- Beginning once software requirements have been analyzed and modeled, software design is the last software engineering action within the modeling activity and sets the stage for construction (code generation and testing).

- Each of the elements of the requirements model provides information that is necessary to create the four design models required for a complete specification of design.
- The requirements model, manifested by **scenario-based, class-based, floworiented, and behavioral elements**, feed the design task.

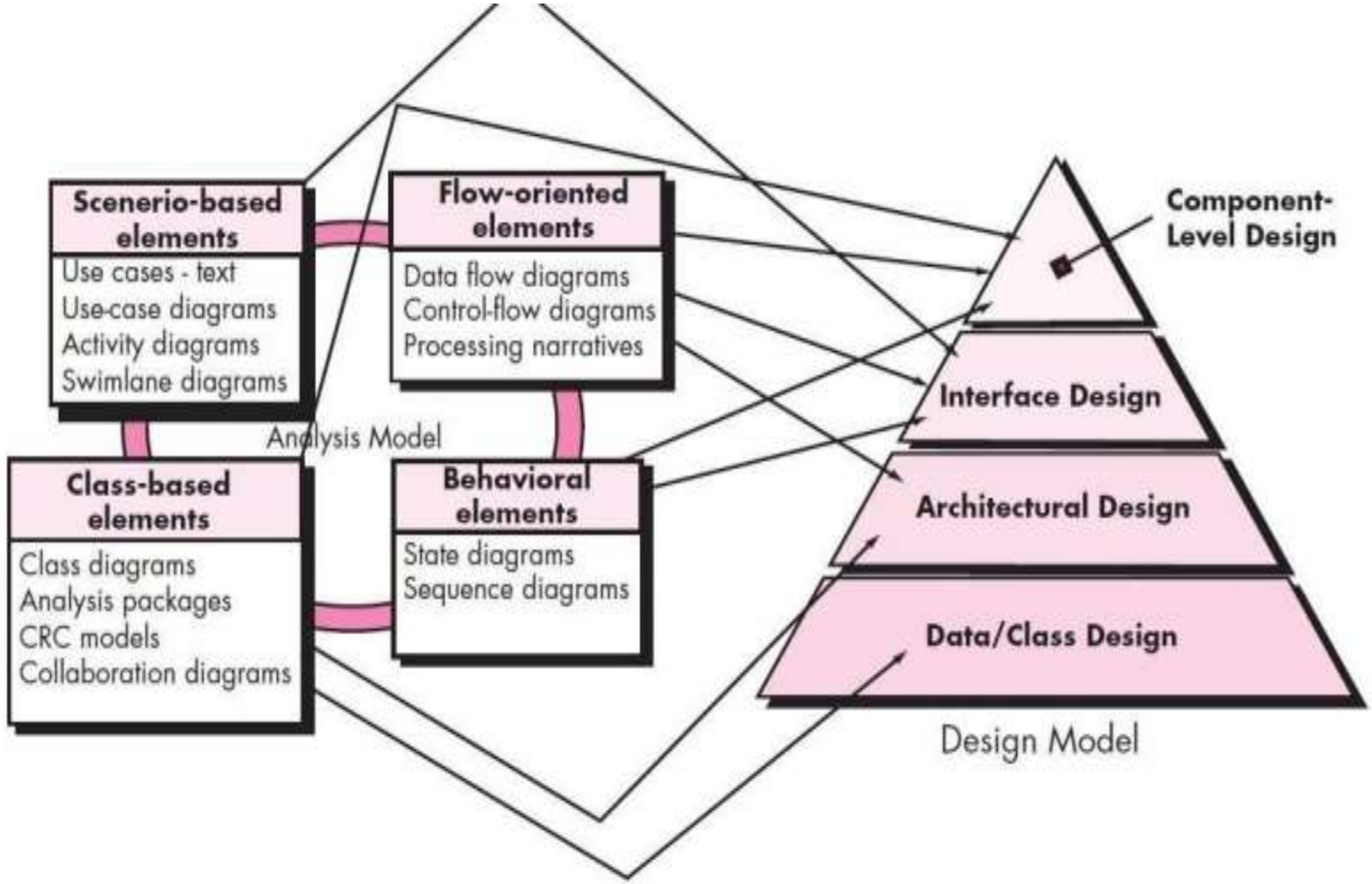


Fig : Translating the requirements model into the design model

- The **data/class** design transforms class models into design class realizations and the requisite data structures required to implement the software.
- The **architectural design** defines the relationship between major structural elements of the software, the architectural styles and design patterns that can be used to achieve the requirements defined for the system, and the constraints that affect the way in which architecture can be implemented. The architectural design representation—the framework of a computer- based system—is derived from the requirements model.

- The **interface design** describes how the software communicates with systems that interoperate with it, and with humans who use it. An interface implies a flow of information (e.g., data and/or control) and a specific type of behavior. Therefore, usage scenarios and behavioral models provide much of the information required for interface design.
- The **component-level design** transforms structural elements of the software architecture into a procedural description of software components. Information obtained from the class-based models, flow models, and behavioral models serve as the basis for component design.

- The importance of software design can be stated with a single word—**quality**.
- Design is the place where quality is fostered in software engineering. Design provides you with representations of software that can be assessed for quality. Design is the only way that you can accurately translate stakeholder's requirements into a finished software product or system.
- Software design serves as the foundation for all the software engineering and software support activities.

THE DESIGN PROCESS

- Software design is an iterative process through which requirements are translated into a “blueprint” for constructing the software.
- Software Quality Guidelines and Attributes:
- Three characteristics that serve as a guide for the evaluation of a good design:
- • The design must implement all of the explicit requirements contained in the requirements model, and it must accommodate all of the implicit requirements desired by stakeholders.
- • 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

- In order to evaluate the quality of a design representation, consider the following guidelines:
- 1. 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,thereby facilitating implementation and testing
- 2. A design should be modular; that is, the software should be logically partitioned into elements or subsystems.

- 3. A design should contain distinct representations of data, architecture, interfaces, and components.

- 4. A design should lead to data structures that are appropriate for the classes to be implemented and are drawn from recognizable data patterns.
- 5. A design should lead to components that exhibit independent functional characteristics

- 6. A design should lead to interfaces that reduce the complexity of connections between components and with the external environment.

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

- 8. A design should be represented using a notation that effectively communicates its meaning.

Quality Attributes

- Hewlett-Packard developed a set of software quality attributes that has been given the acronym **FURPS—functionality, usability, reliability, performance, and supportability.**

- The FURPS quality attributes represent a target for all software design:

- • Functionality is assessed by evaluating the feature set and capabilities of the program, the generality of the functions that are delivered, and the security of the overall system..
- • Usability is assessed by considering human factors, overall aesthetics, consistency, and documentation.

- • Reliability is evaluated by measuring the frequency and severity of failure, the accuracy of output results, the mean-time-to-failure (MTTF), the ability to recover from failure, and the predictability of the program

- • Performance is measured by considering processing speed, response time, resource consumption, throughput, and efficiency.

- • Supportability combines the ability to extend the program (extensibility), adaptability, serviceability—these three attributes represent a more common term, maintainability— and in addition, testability, compatibility, configurability, the ease with which a system can be installed, and the ease with which problems can be localized.

- Software Design is the process of transforming user requirements into a suitable form, which helps the programmer in software coding and implementation.
- During the software design phase, the design document is produced, based on the customer requirements as documented in the SRS document.
- Hence, this phase aims to transform the SRS document into a design document.
- The following items are designed and documented during the design phase:

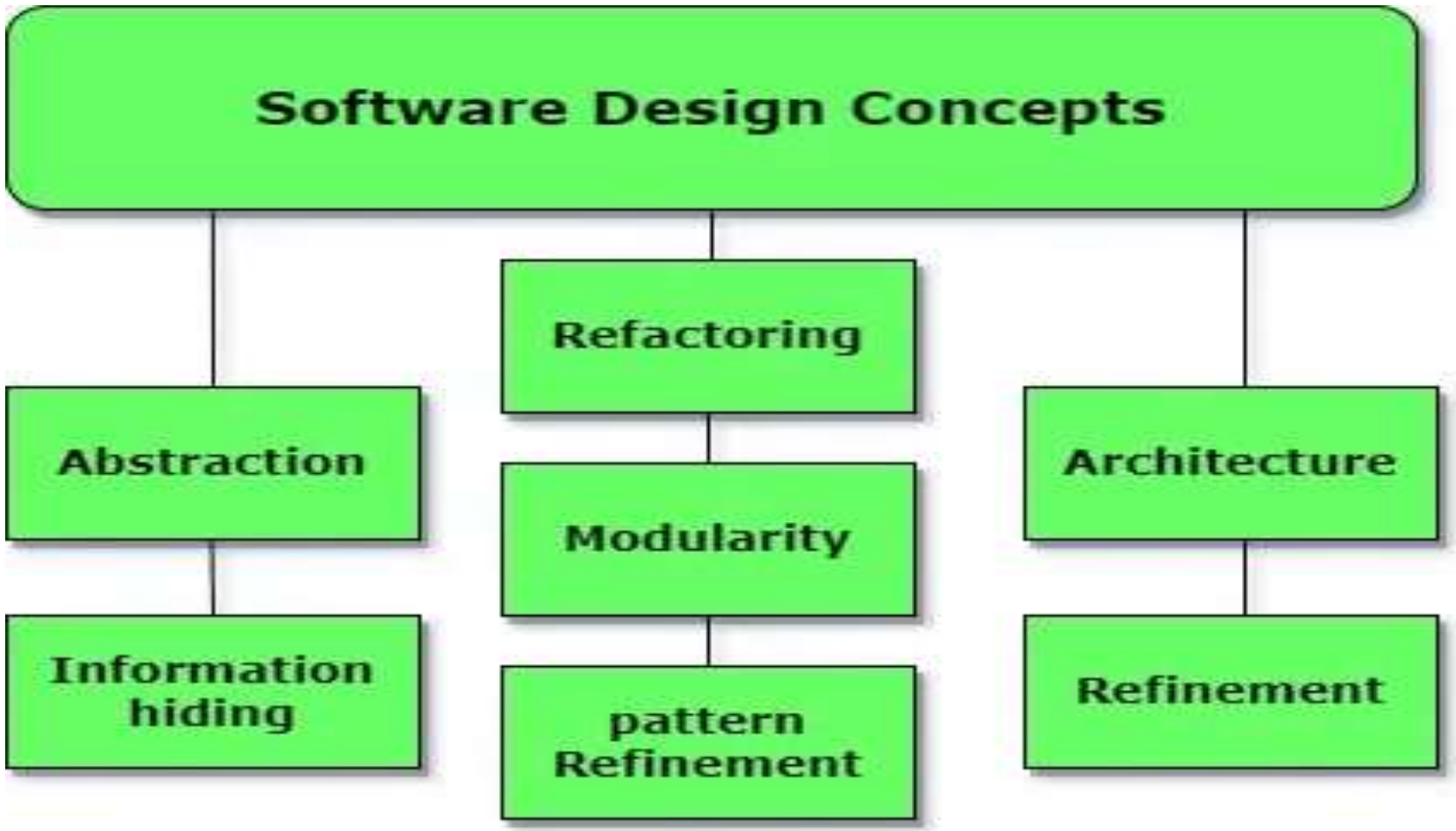
1. Different modules are required.
2. Control relationships among modules.
3. Interface among different modules.
4. Data structure among the different modules.
5. Algorithms are required to be implemented among the individual modules.

Objectives of Software Design

1. **Correctness:** A good design should be correct i.e., it should correctly implement all the functionalities of the system.
2. **Efficiency:** A good software design should address the resources, time, and cost optimization issues.
3. **Flexibility:** A good software design should have the ability to adapt and accommodate changes easily.
4. **Understandability:** A good design should be easily understandable, it should be modular, and all the modules are arranged in layers.
5. **Completeness:** The design should have all the components like data structures, modules, external interfaces, etc.
6. **Maintainability:** A good software design aims to create a system that is easy to understand, modify, and maintain over time.

Software Design Concepts

- . The **software design concept** simply means the idea or principle behind the design.
- It describes how you plan to solve the problem of designing software, and the logic, or thinking behind how you will design software.
- It allows the software engineer to create the model of the system software or product that is to be developed or built.


Software Design Concepts
Abstraction
Information hiding
Refactoring
Modularity
pattern Refinement
Architecture
Refinement

- **Abstraction** (**Hide Irrelevant data**)**:** Abstraction simply means to hide the details to reduce complexity and increase efficiency or quality. Different levels of Abstraction are necessary and must be applied at each stage of the design process so that any error that is present can be removed to increase the efficiency of the software solution and to refine the software solution. The solution should be described in broad ways that cover a wide range of different things at a higher level of abstraction and a more detailed description of a solution of software should be given at the lower level of abstraction.

- **Modularity (subdivide the system):** Modularity simply means dividing the system or project into smaller parts to reduce the complexity of the system or project. In the same way, modularity in design means subdividing a system into smaller parts so that these parts can be created independently and then use these parts in different systems to perform different functions. It is necessary to divide the software into components known as modules because nowadays, there are different software available like Monolithic software that is hard to grasp for software engineers. So, modularity in design has now become a trend and is also important. If the system contains fewer components then it would mean the system is complex which requires a lot of effort (cost) but if we can divide the system into components then the cost would be small.

- **Architecture (design a structure of something):** Architecture simply means a technique to design a structure of something. Architecture in designing software is a concept that focuses on various elements and the data of the structure. These components interact with each other and use the data of the structure in architecture.

- **Refinement (removes impurities):** Refinement simply means to refine something to remove any impurities if present and increase the quality. The refinement concept of software design is a process of developing or presenting the software or system in a detailed manner which means elaborating a system or software. Refinement is very necessary to find out any error if present and then to reduce it.

- **Pattern (a Repeated form):** A pattern simply means a repeated form or design in which the same shape is repeated several times to form a pattern. The pattern in the design process means the repetition of a solution to a common recurring problem within a certain context.

- **Information Hiding (Hide the Information):** Information hiding simply means to hide the information so that it cannot be accessed by an unwanted party. In software design, information hiding is achieved by designing the modules in a manner that the information gathered or contained in one module is hidden and can’t be accessed by any other modules.

- **Refactoring (Reconstruct something):** Refactoring simply means reconstructing something in such a way that it does not affect the behavior of any other features. Refactoring in software design means reconstructing the design to reduce complexity and simplify it without impacting the behavior or its functions. Fowler has defined refactoring as “the process of changing a software system in a way that it won’t impact the behavior of the design and improves the internal structure”.

Different levels of Software Design

- There are three different levels of software design. They are:

1. **Architectural Design:** The architecture of a system can be viewed as the overall structure of the system and the way in which structure provides conceptual integrity of the system. The architectural design identifies the software as a system with many components interacting with each other. At this level, the designers get the idea of the proposed solution domain.

2. **Preliminary or high-level design:** Here the problem is decomposed into a set of modules, the control relationship among various modules identified, and also the interfaces among various modules are identified. The outcome of this stage is called the program architecture. Design representation techniques used in this stage are structure chart and UML.

3. **Detailed design:** Once the high-level design is complete, a detailed design is undertaken. In detailed design, each module is examined carefully to design the data structure and algorithms. The stage outcome is documented in the form of a module specification document.

Elements of a System

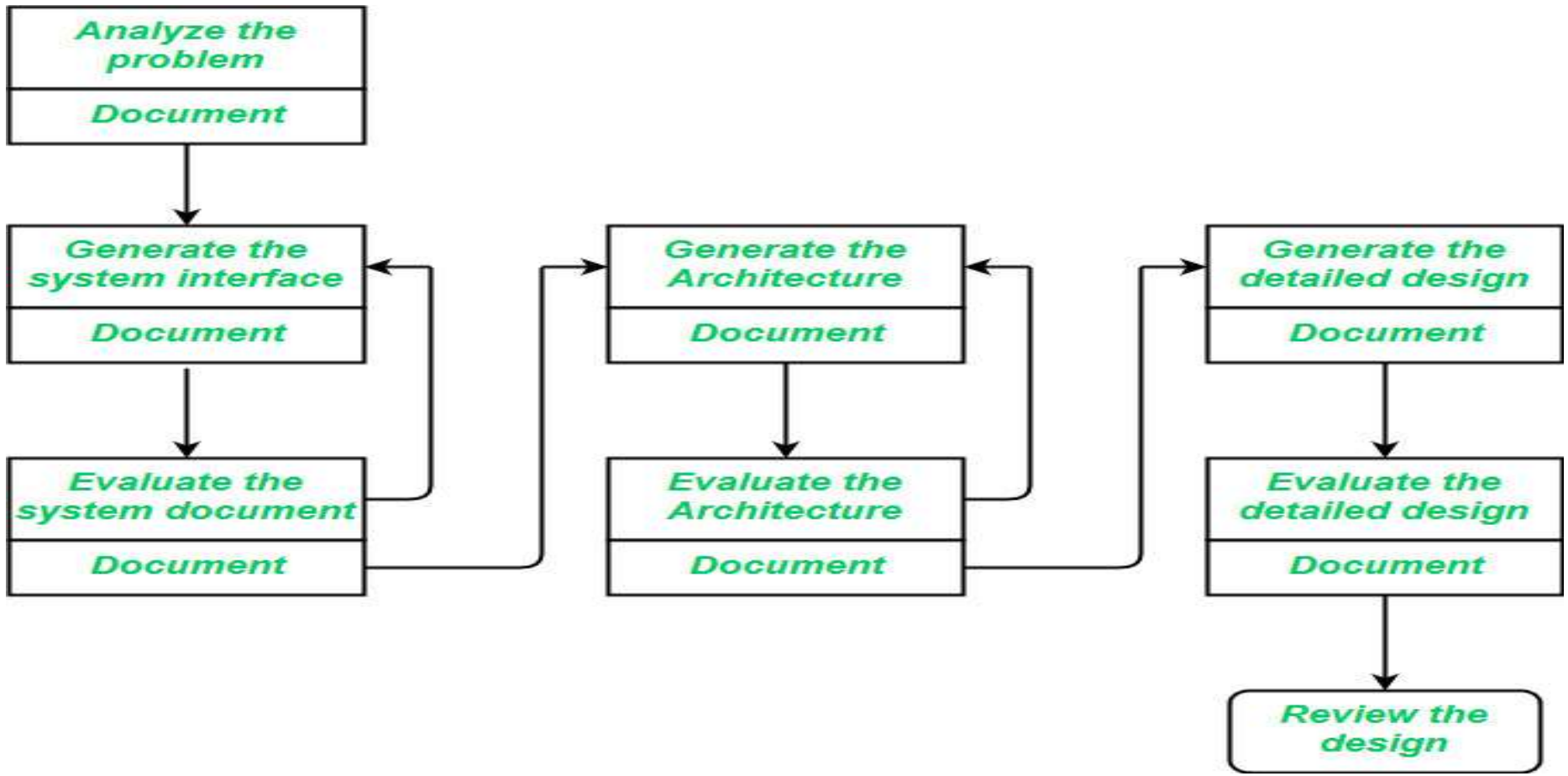
1. **Architecture:** This is the conceptual model that defines the structure, behavior, and views of a system. We can use flowcharts to represent and illustrate the architecture.

2. **Modules:** These are components that handle one specific task in a system. A combination of the modules makes up the system.

3. **Components:** This provides a particular function or group of related functions. They are made up of modules.

4. **Interfaces:** This is the shared boundary across which the components of a system exchange information and relate.

5. **Data:** This is the management of the information and data flow.


Analyze the problem
Document
Generate the system interface
Document
Evaluate the system document
Document
Generate the Architecture
Document
Evaluate the Architecture
Document
Generate the detailed design
Document
Evaluate the detailed design
Document
Review the design

Interface Design

- Interface design is the specification of the interaction between a system and its environment.

Interface design should include the following details:

1. Precise description of events in the environment, or messages from agents to which the system must respond.

2. Precise description of the events or messages that the system must produce.

3. Specification of the data, and the formats of the data coming into and going out of the system.

4. Specification of the ordering and timing relationships between incoming events or messages, and outgoing events or outputs.

Architectural Design

- Architectural design is the specification of the major components of a system, their responsibilities, properties, interfaces, and the relationships and interactions between them.
- In architectural design, the overall structure of the system is chosen, but the internal details of major components are ignored. Issues in architectural design includes:

1. Gross decomposition of the systems into major components.
2. Allocation of functional responsibilities to components.
3. Component Interfaces.
4. Component scaling and performance properties, resource consumption properties, reliability properties, and so forth.
5. Communication and interaction between components.

- The architectural design adds important details ignored during the interface design. Design of the internals of the major components is ignored until the last phase of the design

Detailed Design

- Detailed design is the specification of the internal elements of all major system components, their properties, relationships, processing, and often their algorithms and the data structures.

The detailed design may include:

1. Decomposition of major system components into program units.
2. Allocation of functional responsibilities to units.
3. User interfaces.
4. Unit states and state changes.
5. Data and control interaction between units.
6. Data packaging and implementation, including issues of scope and visibility of program elements.
7. Algorithms and data structures.