Software Engineering—Software Process Activities (Part 3)

Requirements Engineering

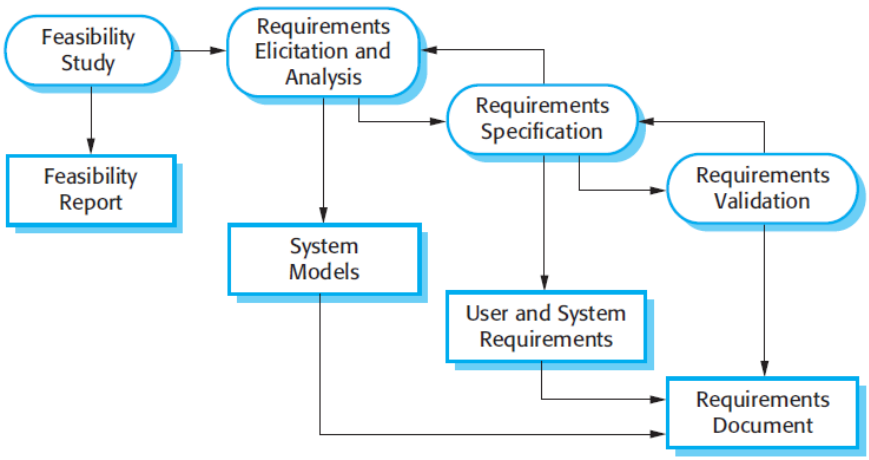
Software specification or requirements engineering is the process of understanding and defining what services are required and identifying the constraints on these services.

Requirements engineering processes ensures your software will meet the user expectations, and ending up with a high quality software.

It's a critical stage of the software process as errors at this stage will reflect later on the next stages, which definitely will cause you a higher costs.

At the end of this stage, a requirements document that specifies the requirements will be produced and validated with the stockholders.

There are four main activities (or sub-activities) of requirements engineering:



The requirements engineering process.

1. **Feasibility study:** An estimate is made of whether the identified can be achieved using the current software and hardware technologies, under the current budget, etc. The feasibility study should be cheap and quick; it should inform the decision of whether or not to go ahead with the project.

2. **Requirements elicitation and analysis:** This is the process of deriving the system requirements through observation of existing systems, discussions with stakeholders, etc. This may involve the development of one or more system models and prototypes that can help us understanding the system to be specified.

3. **Requirements specification:** It's the activity of writing down the information gathered during the elicitation and analysis activity into a document that defines a set of requirements. Two types of requirements may be included in this document; user and system requirements.

4. **Requirements validation:** It's the process of checking the requirements for realism, consistency and completeness. During this process, our goal is to discover errors in the requirements document. When errors are found, it must be modified to correct these problems.

Of course, the activities in the requirements process are not simply executed in a strict sequence, but, they are interleaved. For example, analysis activity continues during the specification as new requirements come to light.

> *In agile methods, requirements are developed incrementally according to user priorities and the elicitation of requirements comes from users who are part of the development team.*

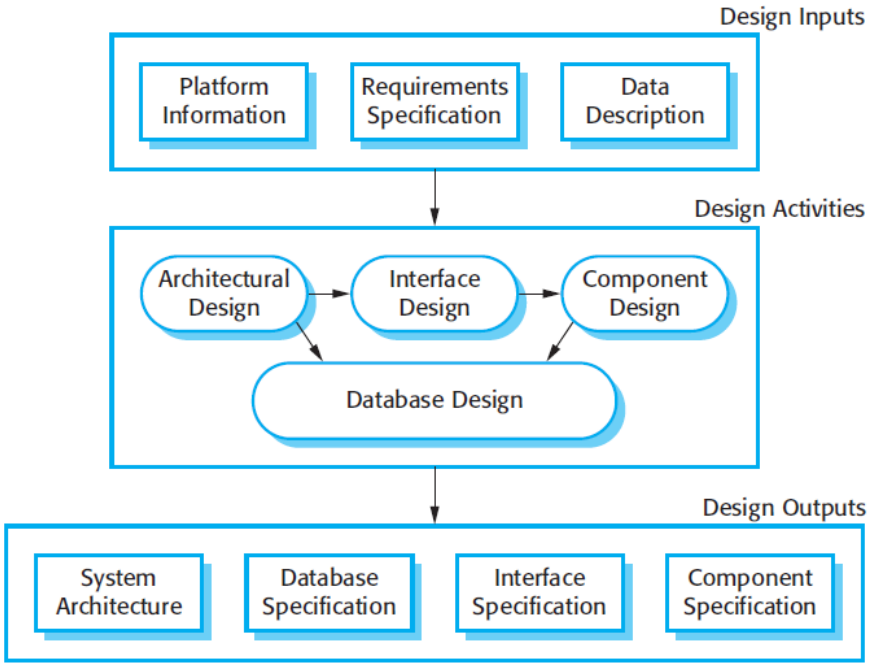
Software Design And Implementation

The implementation phase is the process of converting a system specification into an executable system. If an incremental approach is used, it may also involve refinement of the software specification.

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

The software designers develop the software design iteratively; they add formality and detail and correct the design as they develop their design.

Here's an abstract model of the design process showing the inputs, activities, and the documents to be produced as output.



The software design process

The diagram suggests that the stages of the design process are sequential. In fact, they are interleaved. A feedback from one stage to another and rework can't be avoided in any design process.

These activities can vary depending on the type of the system needs to be developed. We've showed four main activities that may be part of the design process for information systems, and they are:

1. **Architectural design:** It defines the overall structure of the system, the main components, their relationships.
2. **Interface design:** It defines the interfaces between these components. The interface specification must be clear. Therefore, a component can be used without having to know it's implemented. Once the interface specification are agreed, the components can be designed and developed concurrently.
3. **Component design:** Take each component and design how it will operate, with the specific design left to the programmer, or a list of changes to be made to a *reusable* component.
4. **Database design:** The system data structures are designed and their representation in a database is defined. This depends on whether an existing database is to be reused or a new database to be created.

These activities lead to a set of design outputs. The detail and representation vary based on the system being developed.

For example, in critical systems, detailed design documents giving a precise and accurate description of the system must be produced.

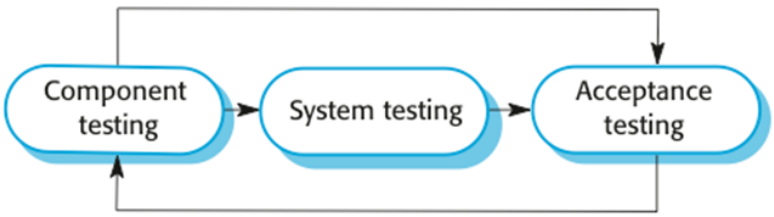
These outputs may be graphical models of the system, and in many cases, automatically generating code from these models.

Software Verification And 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 customer.

Validation may also involve checking processes, such as inspections or reviews at each stage of the software process, from defining the requirements till the software development.

Testing, where the system is executed using simulated test data, is an important validation technique.



The stages of testing

Testing has three main stages:

1. **Development (or component) testing:** The components making up the system are tested by the people developing the system. Each component is tested independently, without other system components.
2. **System testing:** System components are integrated to create a complete system. This process is concerned with finding errors that result from interactions between components. It is also concerned with showing that the system meets its functional and non-functional requirements.

3. **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 using simulated test data. It may reveal errors in the system requirements definition.

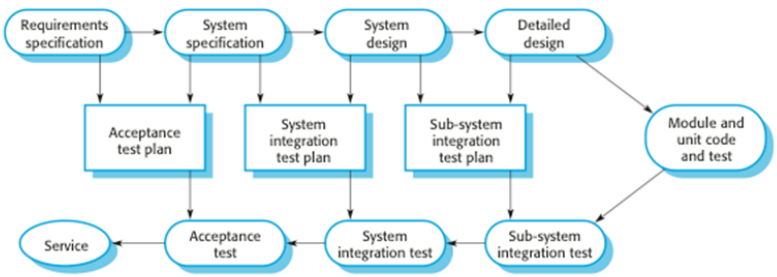
> *Components may be simple entities such as functions or object classes, or may be coherent groupings of these entities. Test automation tools, such as JUnit are commonly used to run component tests.*

Normally, component development and testing process are interleaved. Programmers tend o make up their own test data and incrementally test the code as it's developed.

In some other cases, tests are developed along with the requirements before the development starts. This helps the testers and developers to understand the requirements and reveals requirements problems.

When a plan-driven software process is used, testing is driven by a set of test plans, which created from the system specification and design.

How the test plans are the link between each phase of the development life cycle and its associated phase of testing can be demonstrated by a software process model called "V-model".



Testing phases in a plan-driven software process

Software Maintenance

Requirements are always changing, even after the system has been put into its operating environment. The flexibility of software systems is one of the main reasons why software is being used in large, complex systems.

Historically, there has always been a split between the process of software development and the process of software evolution (software maintenance).

However, this distinction is increasingly irrelevant, and it makes much more sense to see development and maintenance as a continuum.

Rather than two separate processes, it is more realistic to think of software engineering as an evolutionary process where software is continually changed over its lifetime in response to changing requirements and customer needs.