5 Software Design

Software design is more creative process than analysis because it deals with the development of the actual mechanics for a new workable system. While analysing, it is possible to produce the correct model of an existing system. However, there is, no such thing as correct design. Good design is always system dependent and what is good design for one system may be bad for another.

The design of the new system must be done in great detail as it will be the basis for future computer programming and system implementation. Design is a problem-solving activity and as such, very much a matter of trial and error. The designer, together with users, has to search for a solution using his/her professional wisdom until there is an agreement that a satisfactory solution has been found.

For small projects (such as student's projects), one can sit with the specifications and simply write a program. For larger projects, it is necessary to bridge the gap between specifications and the coding with something more concrete. This bridge is the software design.

5.1 WHAT IS DESIGN?

Design is the highly significant phase in the software development where the designer plans "how" a software system should be produced in order to make it functional, reliable and reasonably easy to understand, modify and maintain. A software requirements specifications (SRS) document tells us "what" a system does, and becomes input to the design process, which tells us "how" a software system works. Designing software systems means determining how requirements are realized and result is a software design document (SDD). Thus, the purpose of design phase is to produce a solution to a problem given in SRS document.

A framework of the design is given in Fig. 5.1. It starts with initial requirements and ends up with the final design. Here, data is gathered on user requirements and analysed accordingly. A high level design is prepared after answering questions of requirements. Moreover, design is validated against requirements on regular basis. Design is refined in every cycle and finally it is documented to produce software design document.
Fig. 5.1 shows the design framework.

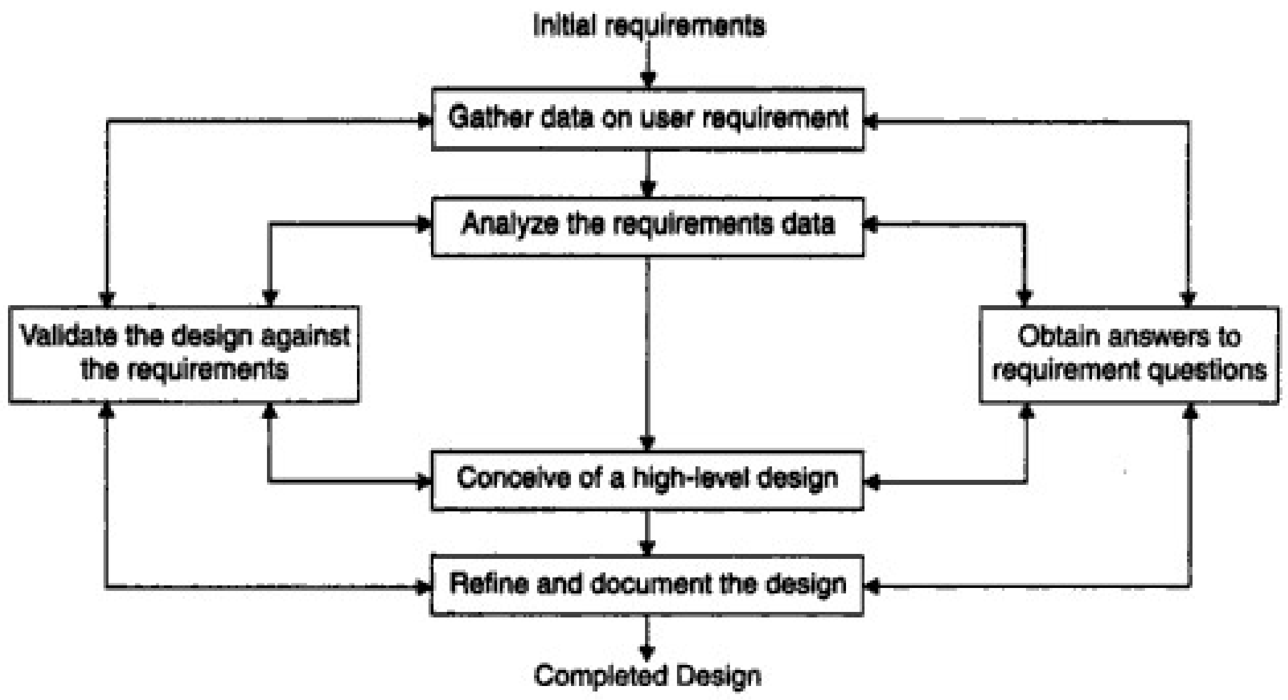
Initial requirements
Gather data on user requirement
Analyze the requirements data
Validate the design against the requirements
Obtain answers to requirement questions
Conceive of a high-level design
Refine and document the design
Completed Design

Fig. 5.1: Design framework.

5.1.1 Conceptual and Technical Designs

The process of software design involves the transformation of ideas into detailed implementation descriptions, with the goal of satisfying the software requirements. To transform requirements into a working system, designers must satisfy both customers and the system builders (coding persons). The customers understand what the system is to do. At the same time, the system builders must understand how the system is to work. For this reason, design is really a two part, iterative process. First, we produce conceptual design that tells the customer exactly what the system will do. Once the customer approves the conceptual design, we translate the conceptual design into a much more detailed document, the technical design, that allows system builders to understand the actual hardware and software needed to solve the customer's problem. This two part design process is shown in Fig. 5.2.

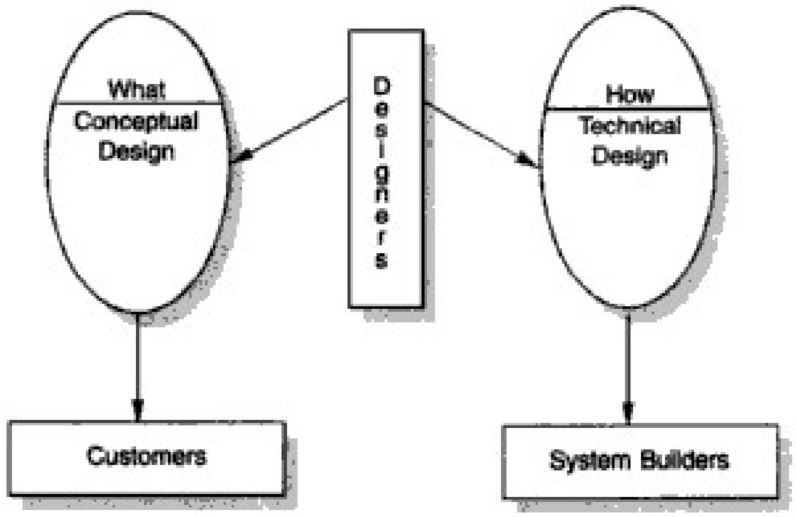

Fig. 5.2: A two-part design process.

The two design documents describe the same system, but in different ways because of the different audiences for the documents. The conceptual design answers the following questions [PFLE98].

- Where will the data come from?
- What will happen to the data in the system?
- How will the system look to users?
- What choices will be offered to users?
- What is the timing of events?
- How will the reports and screens look like?

The conceptual design describes the system in language understandable to the customer. It does not contain any technical jargons and is independent of implementation.

By contrast, the technical design describes the hardware configuration, the software needs, the communications interfaces, the input and output of the system, the network architecture, and anything else that translates the requirements into a solution to the customer's problem.

Sometimes customers are very sophisticated and they can understand the "what" and "how" together. This can happen when customers are themselves software developers and may not require conceptual design. In such a cases comprehensive design document may be produced.

5.1.2 Objectives of Design

The specification (*i.e.* the "outside" view) of a program should obviously be as free as possible of aspects imposed by "how" the program will work (*i.e.* the "inside" view). It is seldom a document from which coding can directly be done. So design fills the gap between specifications and coding; taking the specifications, deciding how the program will be organized, and the methods it will use, in sufficient detail as to be directly codeable.

If the specification calls for a large or complex program (or both), then the design is quite likely to work down through a number of levels. At each level, breaking the implementation problem into a combination of smaller and simpler problems. Filling a large gap will involve a number of stepping-stones! The wider the gap, the larger the number of stepping-stones. The design needs to be

- Correct and complete
- Understandable
- At the right level
- Maintainable, and to facilitate maintenance of the produced code

Software designers do not arrive at a finished design document immediately but develop the design iteratively through a number of different phases. The design process involves adding details as the design is developed with constant backtracking to correct earlier, less formal, designs. The starting point is an informal design which is refined by adding information to make it consistent and complete and this is shown in Fig. 5.3 [SOMM2K].

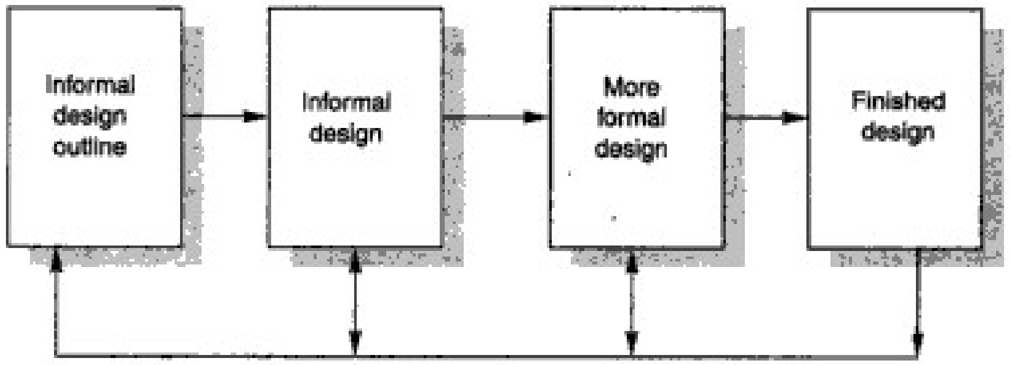

Fig. 5.3: The transformation of an informal design to a detailed design.

5.1.3 Why Design is Important?

A good design is the key to successful product. Almost 2000 years ago Roman Architect Vitruvious recorded the following attributes of a good design:

- Durability
- Utility and
- Charm

A well-designed system is easy to implement, understandable and reliable and allows for smooth evolution. Without design, we risk building an unstable system:

- One that will fail when small changes are made
- One that will be difficult to maintain
- One whose quality can not be assessed until late in the software process.

Therefore, software design should contain a sufficiently complete, accurate and precise solution to a problem in order to ensure its quality implementation.

There are three characteristics that serve as a guide for the evolution of a good design.

- 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 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 behavioural domain from an implementation perspective.

5.2 MODULARITY

There are many definitions of the term "module". They range from "a module is a FORTRAN subroutine" to "a module is an Ada package" to "procedures and functions of PASCAL and C", to "C++ / Java Classes", to "Java packages" to "a module is a work assignment for an individual programmer" [FAIR2K]. All of these definitions are correct. A modular system consist of well defined, manageable units with well defined interfaces among the units. Desirable properties of a modular system include: