A2.2 DESIGN: ISSUES AND GUIDELINES

The RDD specifies customer and user requirements. The SRS specifies the software requirements that would fulfil customer and user requirements in terms of functions, features, quality, performance and so on. In order to achieve this, we need a system design that delivers the requirement. A system design transforms RDD and SRS into a set of deliverables. System design is a creative process intended to build a system that is dependable, cost effective and flexible, assuring quality and maintainability and satisfying customer expectations.

System design is a two-stage process. The first stage is conceptual design and the second stage is technical design. In conceptual design, what the customer wants is formalized. Technical design then specifies how it is achieved. The conceptual and technical designs are the views of the system designers to be translated by system developers into a workable system. A good conceptual design document uses customer's language and much less technical jargon to describe the functions, features and facilities that can be mapped to RDD and SRS. A good technical design talks about technology, platform, hardware and software configuration, system hierarchy in modules and clusters, data structures and data flow. It uses diagramming language extensively to convey the technical design to system developers.

Design Issues

In the task of building a good design, several issues crop up, which the software designer has to pay attention to. The issues are

- **Modular design**
- **Collaborative design**
- **Design of the user interface**
- **Management of concurrency in design**

Modular design is built by studying and analysing the system top-down through the process of abstraction. A module at the highest level is abstract and as you go down through decomposition it becomes less abstract and more clear. The purpose of modularity is to bring clarity into Inputs – Process – Output at each level. Modularity also throws up different views of the system. Advantage of such a system is that it enables you to choose different designs for different components.

Collaborative design is a need when a system is complex and large. Such systems are developed through distributed development strategy with different teams. The developed output of each team is then integrated to complete the design. When the team approach is taken for development, the issues that arise are distribution of broken-down components, documentation and co-ordination of the work of different teams to make a software system with good design features. When teams are organized on the basis of function or product, they have to work in a collaborative manner to produce an ultimate good design. Collaborative working however adds additional problems in the design process due to differences in team capabilities, personal experience, understanding, personal preferences and differences in skill sets and so on. These differences may arise within a team and among different teams as well. The problems due to these differences are aggravated if communications and documentation systems of good standard are not in place.

As stated, software design is a collaborative and iterative process. If this approach has to work satisfactorily, it is necessary to create shared common and precise understanding of business, customers, users, stakeholders, the application of domain knowledge and business environment. If this is supported by artifacts like notes, models, diagrams, prototypes and graphics, the quality of understanding improves considerably. The clear and transparent communication channels help to overcome problems arising out of differences in teams and team members.

Techniques for Better Design

The best technique of good design is to 'build it right first' instead of using a build and then improve policy. Some suggestions towards this goal are given below:

- Set the design goal correctly. This sets the baseline for measurement and user acceptance.
- Use diagramming and modeling tools to document and to convey the design. This improves the quality of design documentation and communication.
- Reduce the complexity of design by simplifying the design structure without loss in quality and design goals.
- Use decision trees and decision tables for understanding and coding the decision flow.
- Use Meyer's design by contract method, where requirements – Specifications – Processes – Deliverables etc. are so precise, that it can be contracted with commitment to meet obligations and benefits.
- Design prototypes for the part/component/function/feature first to understand its need and then to evolve precise specification. It helps to evolve correct RDD and then SRS. It is less costly if we make a mistake in prototype. Design prototype is built, demonstrated, discussed, and then finalized. It is always advisable to build prototypes for critical functions, features, processes and critical technology implementation.

  Prototype approach has the following benefits.
  - Learning in precise terms.
  - User accessibility after correct understanding.
  - Built of precise specifications.
  - Reduces efforts and cost on larger system.
  - Rise in skilled human resource productivity.
- Fault tree analysis

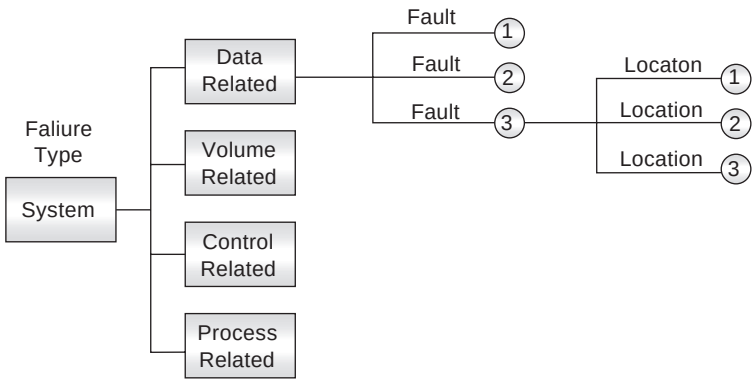
  This method is used in the design process to anticipate and identify possible faults and their locations. The method is originally developed for the US missile program. The way to proceed to draw a fault tree is to first identify areas and locations of failures, which could be due to design. Figure. A2.3 shows a model of a fault tree, where failure type, faults and location are shown as a model.

  Fault tree analysis helps to identify the faults, areas and location within it where the fault will hit and cause damage. The next step then is to review the design for redesigning by
  - removing the fault.
  - improving design by adding components of better quality.
  - imposing more checks - controls - conditions on input to detect fault and prevent execution.
  - adding components that will recover the system to its original position.

Fault tree analysis is not a fool-proof solution: however, it assists in handling the problem. Fault tree analysis is applied to critical aspects of design meeting

Fig. A2.3

Model of Fault Tree



- Data-related Faults: No Data, no Communication to Fetch or Send Data.
- Volume-related Faults: Volume is too High. Data Movement too Slow or too Fast.
- Control-related Faults: Communication Signals Fast, Slow, Incomplete, Misleading, and in Wrong Order.
- Process-related Faults: Not Sequential or Logical, Missing Process Steps and so on.

critical requirements of the user. Analyse the tree by assigning probabilistic value to each branch of the tree.

Once design is declared complete, it is subjected to an evaluation process through validation and verification of RDD and SRS *versus* the design specification.