Within computer science, there is already a large subdiscipline that addresses the management and technical issues of the development of software systems – called software engineering.

Section:6.2
6.2.1
6.2.1 Activities in the life cycle
We describe the activities of this waterfall model of the software life cycle next.
Requirements specification: In requirements specification, the designer and customer try to capture a description
of what the eventual system will be expected to provide. This is in contrast to determining how the system will provide the expected services, which is the concern of later activities. Requirements specification involves eliciting information from the customer about the work environment, or domain, in which the final product will function.
Architectural design
As we mentioned, the requirements specification concentrates on what the system is supposed to do. The next activities concentrate on how the system provides the services expected from it. The first activity is a high-level decomposition of the system into components that can either be brought in from existing software products or be developed from scratch independently.

Detailed design
The architectural design provides a decomposition of the system description that allows for isolated development of separate components which will later be integrated

Coding and unit testing
The detailed design for a component of the system should be in such a form that it is possible to implement it in some executable programming language. After coding, the component can be tested to verify that it performs correctly, according to some test criteria that were determined in earlier activities.
Integration and testing
Once enough components have been implemented and individually tested, they must be integrated as described in the architectural design.
Maintenance
After product release, all work on the system is considered under the category of maintenance, until such time as a new version of the product demands a total redesign or the product is phased out entirely.

6.2.2 Validation and verification

design must be checked to ensure that it both satisfies
the high-level requirements agreed with the customer and is also complete and internally consistent. These checks are referred to as validation and verification,

Respectively.

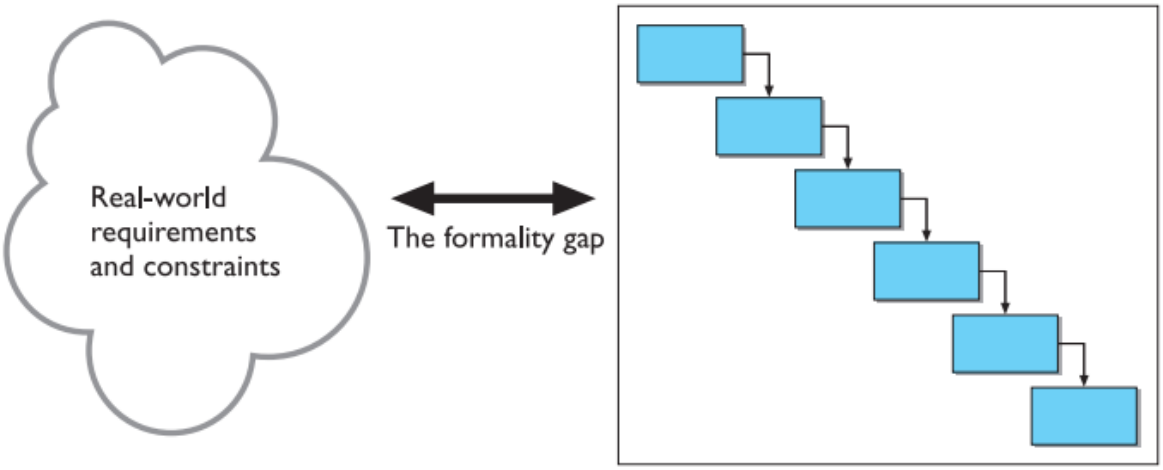

Figure 6.3 The formality gap between the real world and structured design

The formality gap means that validation will always rely to some extent on subjective means of proof.

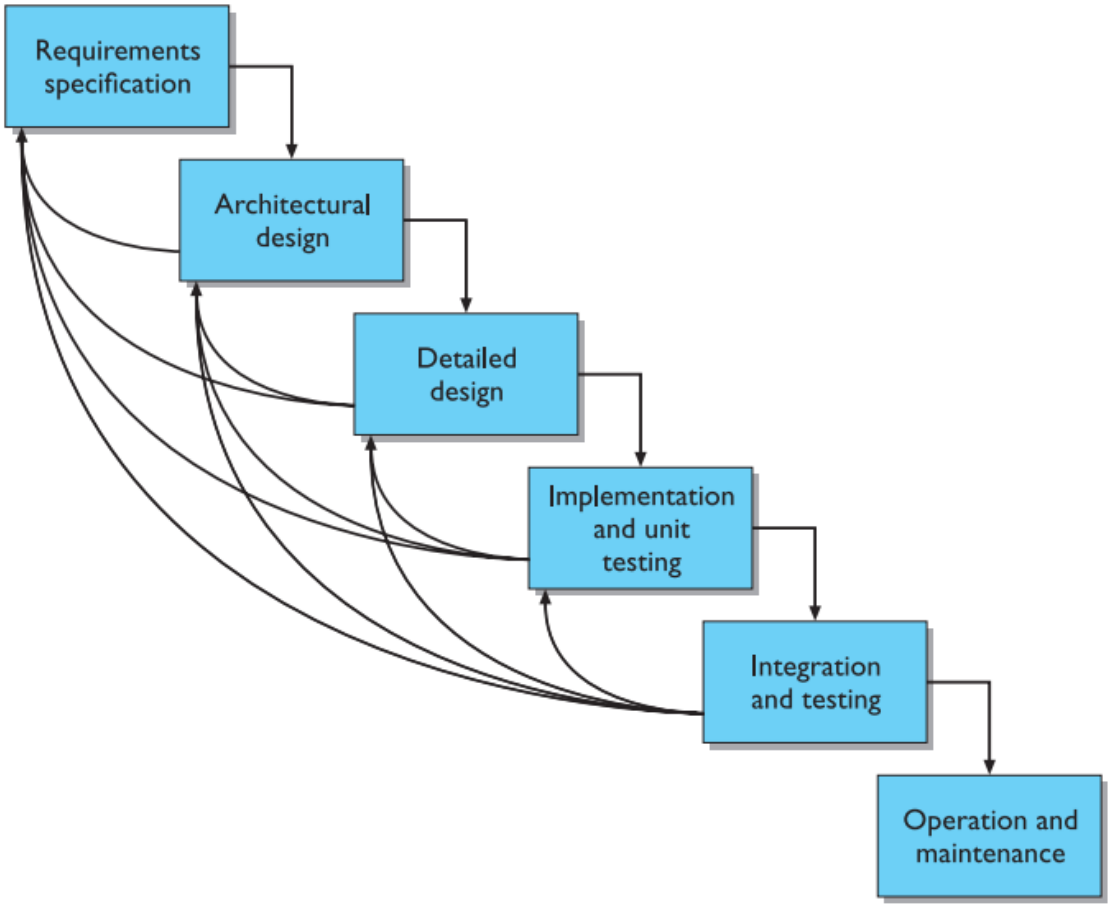

Figure 6.4 Representing iteration in the waterfall model

6.3 USABILITY ENGINEERING

One approach to user-centered design has been the introduction of explicit usability engineering goals into the design process, as suggested by Whiteside and colleagues at IBM and Digital Equipment Corporation

The ultimate test of a product's usability is based on measurements of users' experience with it. Therefore, since a user's direct experience with an interactive system is at the physical interface, focus on the actual user interface is understandable

Table 6.1 Sample usability specification for undo with a VCR

Attribute:	Backward recoverability
Measuring concept:	Undo an erroneous programming sequence
Measuring method:	Number of explicit user actions to undo current program
Now level:	No current product allows such an undo
Worst case:	As many actions as it takes to program in mistake
Planned level:	A maximum of two explicit user actions
Best case:	One explicit cancel action

In this example, we choose the principle of recoverability, described fully in Chapter 7, as the particular usability attribute of interest. Recoverability refers to the ability to reach a desired goal after recognition of some error in previous interaction. The recovery procedure can be in either a backward or forward sense. Current VCR design has resulted in interactive systems that are notoriously difficult to use; the redesign of a VCR provides a good case study for usability engineering. In designing a new VCR control panel, the designer wants to take into account how a user might recover from a mistake he discovers while trying to program the VCR to record some television program in his absence. One approach that the designer decides to follow is to allow the user the ability to undo the programming sequence, reverting the state of the VCR to what it was before the programming task began.

The backward recoverability attribute is defined in terms of a *measuring concept*, which makes the abstract attribute more concrete by describing it in terms of the actual product. So in this case, we realize backward recoverability as the ability

Table 6.4 Examples of usability metrics from ISO 9241

ITERATIVE DESIGN AND PROTOTYPING
6.4

The design can then be modified to correct any false assumptions that were revealed in the testing. This is the essence of iterative design, a purposeful design process which tries to overcome the inherent problems of incomplete requirements Specification.

three main approaches to prototyping:
Throw-away The prototype is built and tested. The design knowledge gained from this exercise is used to build the final product, but the actual prototype is discarded.

Incremental The final product is built as separate components, one at a time.
Evolutionary: Here the prototype is not discarded and serves as the basis for the next iteration of design.

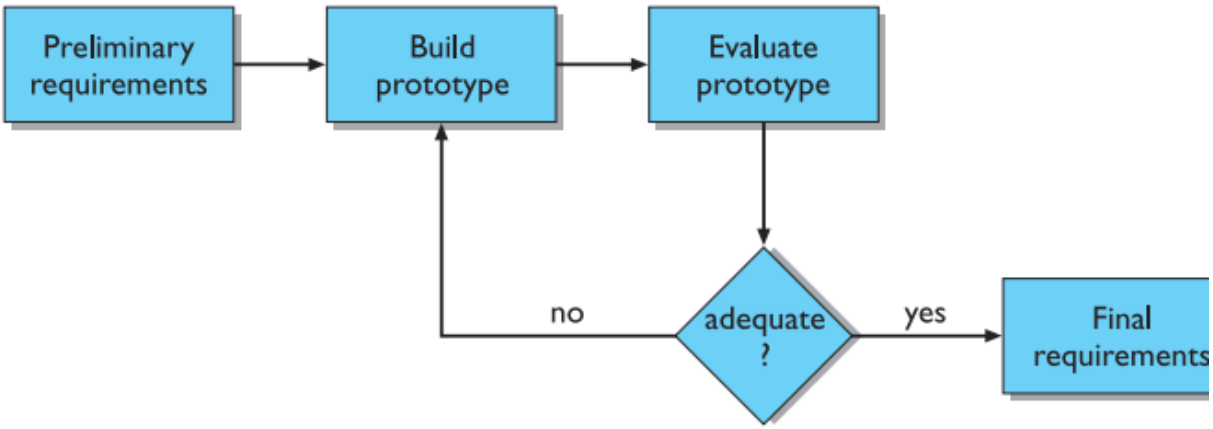

Figure 6.5 Throw-away prototyping within requirements specification

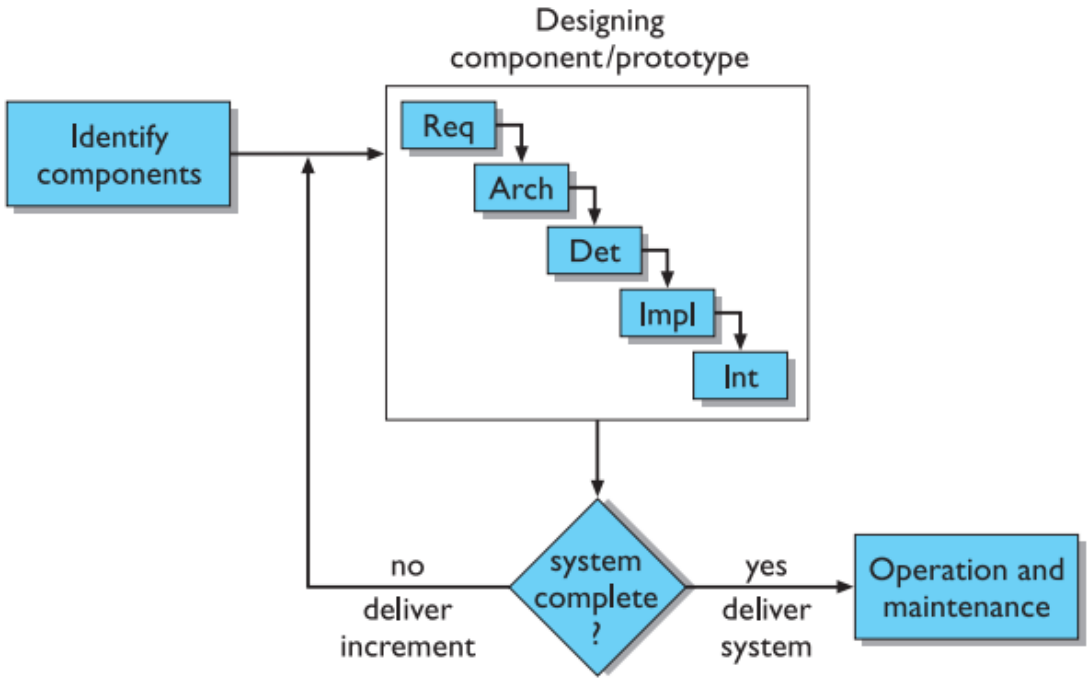

Figure 6.6 Incremental prototyping within the life cycle

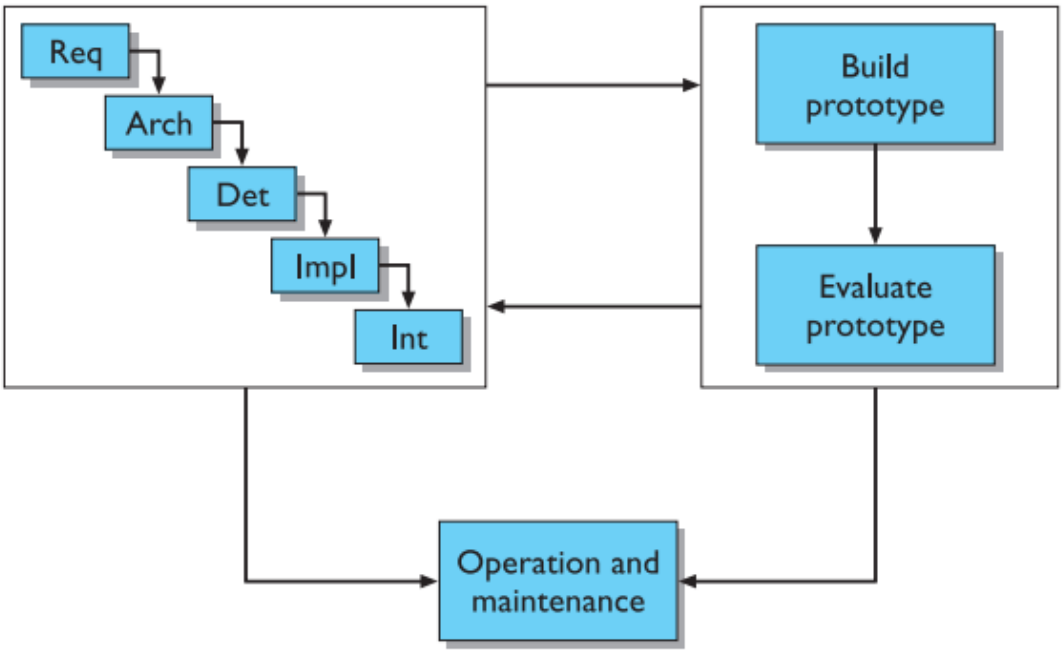

Figure 6.7 Evolutionary prototyping throughout the life cycle

On the management side, there are several potential problems

Time Building prototypes takes time and, if it is a throw-away prototype, it can be seen as precious time taken away from the real design task
Planning Most project managers do not have the experience necessary for adequately planning and costing a design process which involves prototyping
Non-functional features Often the most important features of a system will be non-functional ones, such as safety and reliability, and these are precisely the kinds of features which are sacrificed in developing a prototype.
Contracts The design process is often governed by contractual agreements between customer and designer which are affected by many of these managerial and technical issues.

6.4.1 Techniques for prototyping

Storyboards

Probably the simplest notion of a prototype is the storyboard, which is a graphical depiction of the outward appearance of the intended system, without any accompanying system functionality.

Limited functionality simulations:

For example, we might want to build a prototype for the VCR with undo described earlier using only a workstation display, keyboard and mouse. We could draw a picture of the VCR with its control panel using a graphics drawing package, but then we would want to allow a subject to use the mouse to position a finger cursor over one of the buttons to ‘press’ it and actuate some behavior of the VCR. In this way, we could

simulate the programming task and experiment with different options for undoing.

6.5:DESIGN RATIONALE

Design rationale is the information that explains why a computer system is the way it is, including its structural or architectural description and its functional or behavioral Description.

classification to describe various design rationale techniques

the first set of techniques(process oriented) concentrates on providing a historical record of design decisions and is very much tailored for use during actual design discussions. These techniques are referred to as process-oriented design rationale.

The structure-oriented approach does not capture historical information. Instead, it captures the complete story of the moment, as an analysis of the design space which has been considered so far.

The final category of design rationale concentrates on capturing the claims about the psychology of the user that are implied
by an interactive system and the tasks that are performed on them

6.5.1 Process-oriented design rationale

Rittel's issue-based information
system, or IBIS, a style for representing design and planning dialog developed in the 1970s [308].

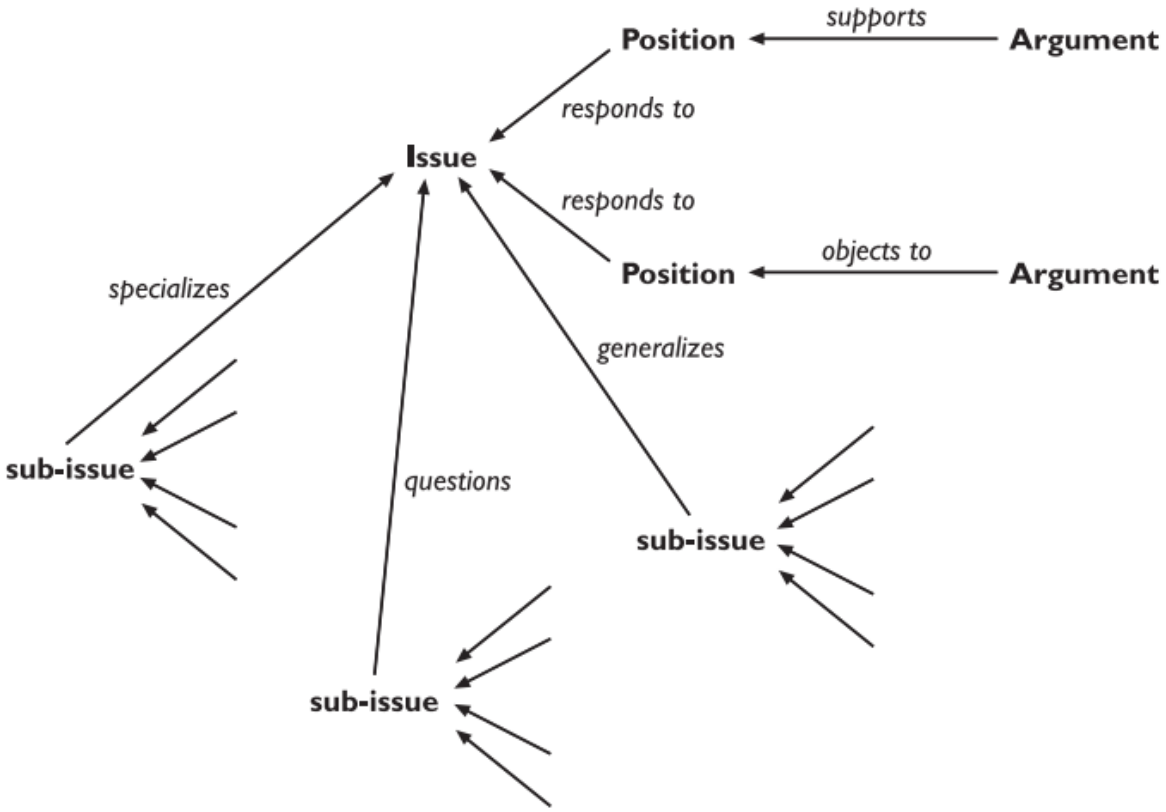

Figure 6.8 The structure of a gIBIS design rationale

6.5.2 Design space analysis

a more deliberative approach to design rationale which emphasizes a post hoc structuring of the space of design alternatives
that have been considered in a design project. Their approach, embodied in the
Questions, Options and Criteria (QOC) notation, is characterized as design space
analysis

The design space is initially structured by a set of questions representing the major issues of the design. Since design space analysis is structure oriented, it is not so important that the questions recorded are the actual questions asked during design meetings. Rather, these questions represent an agreed characterization of the

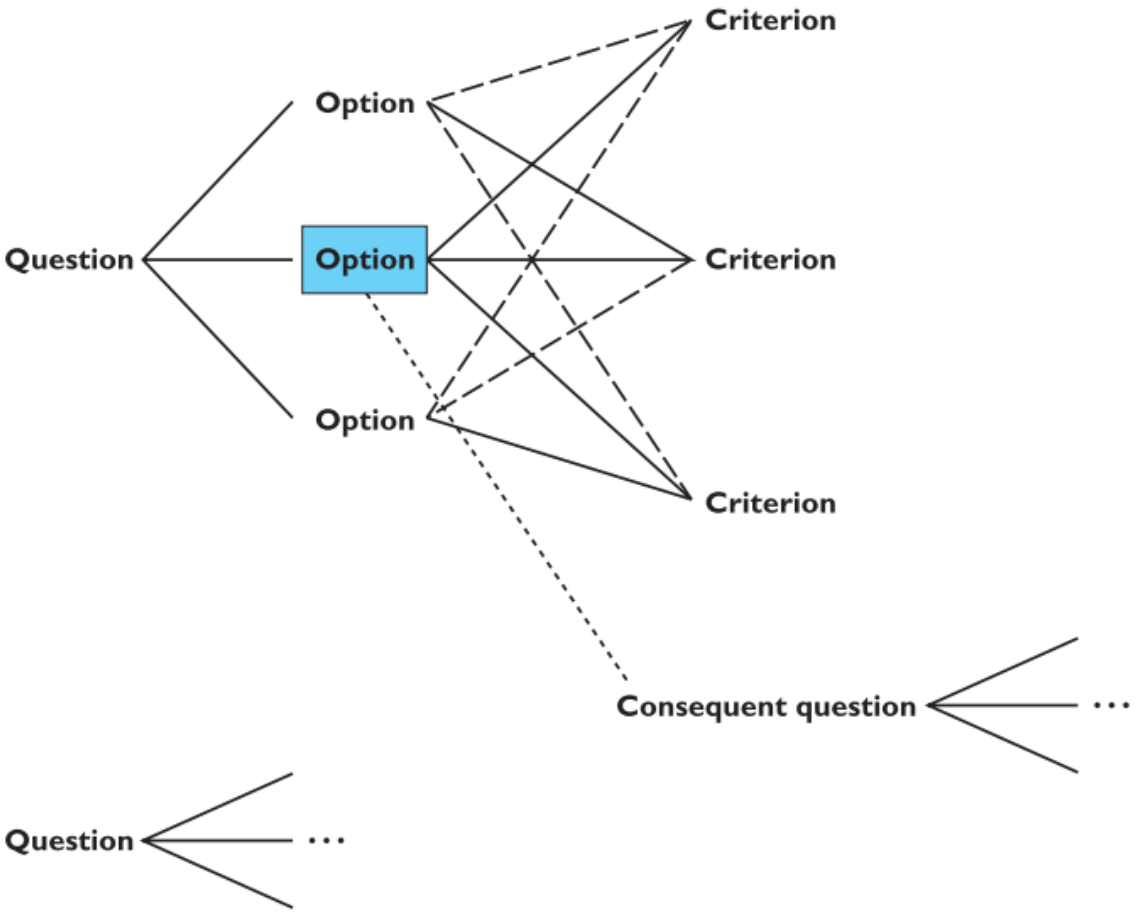

Figure 6.9 The QOC notation

Questions in a design space analysis are therefore similar to issues in IBIS except in the way they are captured. Options provide alternative solutions to the question. They are assessed according to some criteria in order to determine the most favorable option. In Figure 6.9 an option which is favorably assessed in terms of a criterion is linked with a solid line, whereas negative links have a dashed line. The most favorable option is boxed in the diagram.

Another structure-oriented technique, called Decision Representation Language (DRL), developed by Lee and Lai, structures the design space in a similar fashion

to QOC, though its language is somewhat larger and it has a formal semantics

The advantage of the formal semantics of DRL is that the design rationale can be used as a computational mechanism
to help manage the large volume of information. For example, DRL can track the dependencies between different decision problems, so that subsequent changes to the design rationale for one decision problem can be automatically propagated to other dependent problems.

6.5.3 Psychological design rationale

The final category of design rationale tries to make explicit the psychological claims of usability inherent in any interactive system in order better to suit a product for the tasks users have.

Worked exercise(page-255)

Worked exercise What is the distinction between a process-oriented and a structure-oriented design rationale technique? Would you classify psychological design rationale as process or structure oriented?

Why?

Answer **The distinction between a process- and structure-oriented design rationale resides in what information the design rationale attempts to capture. Process-oriented design rationale is interested in recording an historically accurate description of a design team making some decision on a particular issue for the design. In this sense, processoriented design rationale becomes an activity concurrent with the rest of the design process. Structure-oriented design rationale is less interested in preserving the historical evolution of the design. Rather, it is more interested in providing the conclusions of**
the design activity, so it can be done in a post hoc and reflective manner after the fact.
The purpose of psychological design rationale is to support the task–artifact cycle. Here, the tasks that the users perform are changed by the systems on which they perform the tasks. A psychological design rationale proceeds by having the designers of the system record what they believe are the tasks that the system should support and then building the system to support the tasks. The designers suggest scenarios for the tasks which will be used to observe new users of the system. Observations of the users provide the information needed for the actual design rationale of that version of the system. The consequences of the design's assumptions about the important tasks are then gauged against the actual use in an attempt to justify the design or suggest improvements. Psychological design rationale is mainly a process-oriented approach. The activity of a claims analysis is precisely about capturing what the designers assumed about the system at one point in time and how those assumptions compared with actual use. Therefore, the history of the psychological design rationale is important. The discipline involved in performing a psychological design rationale requires designers to perform the claims analysis during the actual design activity, and not as post hoc reconstruction.

Check exercise solution on drive(Aniruddha uploaded)
6.1 (a) How can design rationale benefit interface design and why might it be rejected by
design teams?
(b) Explain QOC design rationale using an example to illustrate

When design rationale is subjective, and there is no cohesive thinking within the desing team, **it may get rejected sometimes for lack of time to achieve the vision, sometimes due to lack of common skills among the executing desingers working on the same product.**

Explain QOC design rationale using an example to illustrate.

The QOC (Questions, Options, Criteria) desgin rationale is the attempt to design the best methods of tking surveys, test , etc. to get data usuable for whatever purpose . **Real world examples: census, ballots , periodic exams , petitions.** Trial lawyers design thier questions to get the answers taht are favourable of thier postions. The teachers of lawyers study the past knowlegde of lawyering to improve thier craft , both in the class room and in the courtroom. Its a method used to determine the characterstics of a group of people. The results of thses actions are useful for administrative purposes, marketing and sales, showing the need for assistance, improvement, passes of legislation.