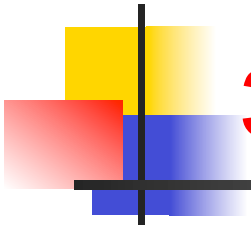
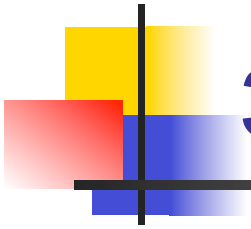


3. Software Requirements

Overview

- 3.1 Issues
- 3.2 Functional Requirements
- 3.3 Non-functional Requirements
- 3.4 Significance

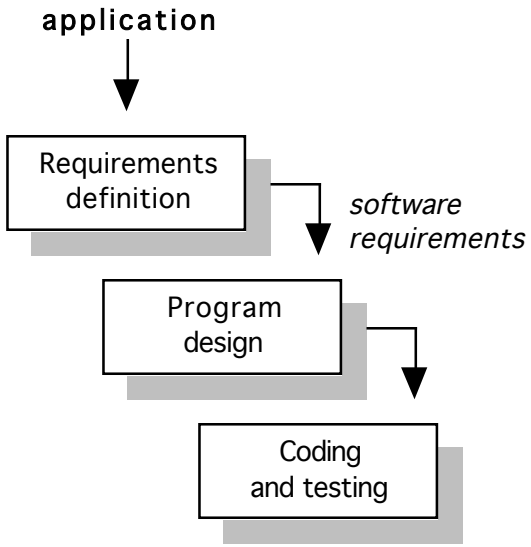


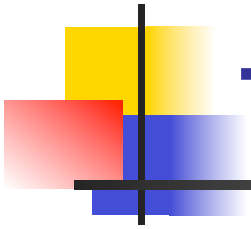
3.1 Issues

- What are requirements?
- Why are they significant?
- When are they generated?
- How are they generated?
- How are they documented?
- How are they managed?
- When are they discarded?
- Can requirements be implicit?

Simplifying Assumptions

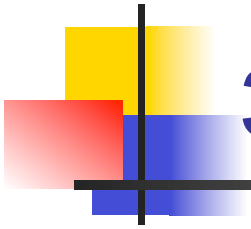
- Ignore the process by which requirements are generated from customer needs
- Consider a very simple life cycle model





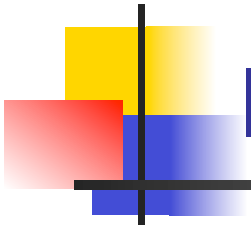
Terminology

- A requirement is a technical objective which is imposed upon the software, i.e., anything that might affect the kind of software that is produced
- A requirement may be imposed by
  - the customer
  - the developer
  - the operating environment
- The source, rationale, and nature of the requirement must be documented
- Requirements fall into two broad categories
  - functional
  - non-functional



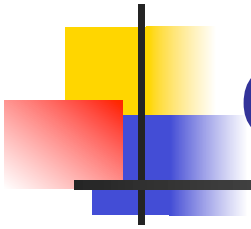
3.2 Functional Requirements

- Functional requirements are concerned with what the software must do
  - capabilities, services, or operations
- Functional requirements are not concerned with how the software does things, i.e., they must be free of design considerations
- Functional requirements are incomplete unless they capture all relevant aspects of the software's environment
  - they define the interactions between the software and the environment
  - the environment may consist of users, other systems, support hardware, operating system, etc.
  - the system/environment boundary must be defined



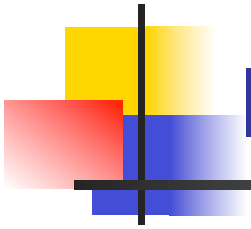
Important Messages

- Requirements are difficult to identify
- Requirements are difficult to write
- Requirements change over time
- Discrepancies between needs and capabilities may render the software useless
- Life-cycle considerations must be documented since they have design implications



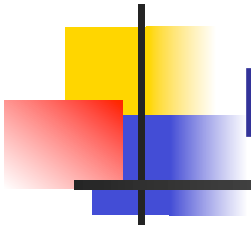
Communication Issues

- Missing or vague requirements are a recipe for disaster
- Anything that is not made explicit may not be implemented or considered
- Anything that is ambiguous is likely to get implemented in the least desirable way
- Standard trade practices may be omitted (in principle!)
- Cultural settings do affect the way in which requirements are written and read
- Multiple views may be required to formulate a reasonably complete set of requirements



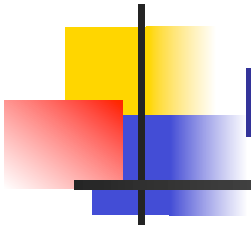
Implicit Requirements

- An interface specification can become a requirement definition only if
  - it is the only processing obligation
  - its semantics are well defined
- A product cannot be its own requirements definition because
  - the rationale for the design decisions is lost
  - there is no verification criterion



Non-Functional Requirements

- Non-functional requirements place restrictions on the range of acceptable solutions
- Non-functional requirements cover a broad range of issues
  - interface constraints
  - performance constraints
  - operating constraints
  - life-cycle constraints
  - economic constraints
  - political constraints
  - manufacturing

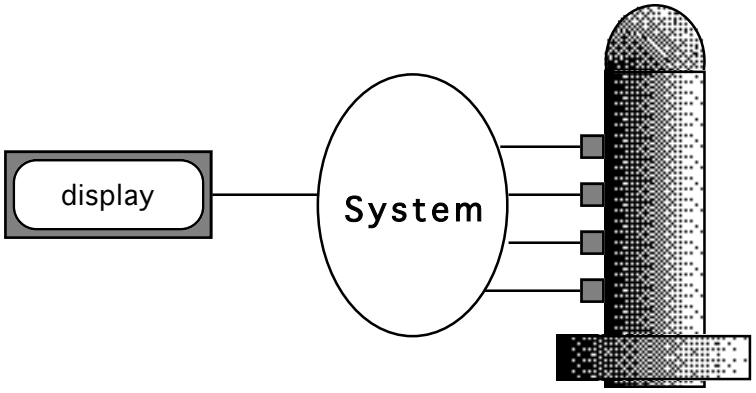


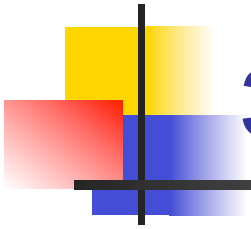
Important Messages

- Constraints are the main source of design difficulties
- No formal foundation on which to base the treatment of most non-functional requirements is available today
- Non-functional requirements are at least as dynamic as the functional ones

Case Study: Sensor Display

- Consider a small system that displays the status of several sensors (e.g., pressure, temperature, rate of change, etc.) on a limited-size screen.





3.4 Significance

- Requirements are the foundation for the software development process
- Requirements impact the life cycle throughout its phases
  - customer/developer interactions
  - contractual agreements
  - feasibility studies
  - quality assurance
  - project planning
  - risk analysis
  - testing
  - user documentation