Module II

Process Framework Models: Capability maturity model (CMM), ISO 9000. Phases in Software development – requirement analysis requirements elicitation for software, analysis principles, software prototyping, specification.

Q1. Explain CMM with suitable sketch

2.1 Capability Maturity Model

- Created by the Software Engineering Institute, a research center founded by Congress in 1984
- Describes an evolutionary improvement path for software organizations from an ad hoc, immature process to a mature, disciplined one.
- Provides guidance on how to gain control of processes for developing and maintaining software and how to evolve toward a culture of software engineering and management excellence.

Process Maturity Concepts

a.**Software Process**- it is a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products (e.g., project plans, design documents, code, test cases, user manuals).

b. Software Process Capability – it describes the range of expected results that can be achieved by following a software process. It is the means of predicting the most likely outcomes to be expected from the next software project the organization undertakes.

c. Software Process Performance- actual results achieved by following a software process

d.Software Process Maturity - Extents to which a specific process is explicitly defined, managed, measured, controlled and effective. It implies potential growth in capability and it indicates richness of process and consistency with which it is applied in projects throughout the organization

CMM Levels

Maturity level indicates level of process capability:There are five maturity levels in CMM. They are.

- Initial
- Repeatable
- Defined
- Managed
- Optimizing

a. Level 1- Initial- The software process is characterized as ad hoc, . Few processes are defined, and success depends on individual effort.

- At this level, frequently have difficulty making commitments that the staff can meet with an orderly process
- Products developed are often over budget and schedule
- Wide variations in cost, schedule, functionality and quality targets

- Capability is a characteristic of the individuals, not of the organization

b. Level2- Repeatable

- Basic process management processes are established to track cost, schedule, and functionality.
- The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
- Realistic project commitments based on results observed on previous projects
- Software project standards are defined and faithfully followed
- Processes may differ between projects
- Process is disciplined
- earlier successes can be repeated

c. Level 3- Defined

- The software process for both management and engineering activities . It is documented, standardized, and integrated into a standard software process for the organization.
- All projects use an approved, tailored version of the organization’s standard software process for developing an maintaining software.

d.Level 4- Managed

- Detailed measures of the software process and product quality are collected.
- Both the software process and products are quantitatively understood and controlled.
- Narrowing the variation in process performance - known limits are exceeded- corrective action can be taken
- Quantifiable and predictable
- predict trends in process and product quality

e.Level 5- Optimizing

- Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.
- Goal is to prevent the occurrence of defects
- Data on process effectiveness used for cost benefit analysis of new technologies and proposed process changes.

Key Processing Areas- Except for level 1, each level is decomposed into key process areas (KPA)

Each KPA identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing software capability.

a.Level 2 KPA

- Requirements Management
  - Establish common understanding of customer requirements between the customer and the software project
  - Requirements is basis for planning and managing the software project
  - Not working backwards from a given release date
- Software Project Planning
  - Establish reasonable plans for performing the software engineering activities and for managing the software project

- Software Project Tracking and Oversight
  - Establish adequate visibility into actual progress
  - Take effective actions when project's performance deviates significantly from planned
- Software Subcontract Management
  - Manage projects outsourced to subcontractors
- Software Quality Assurance
  - Provide management with appropriate visibility into process being used by the software projects and work products
- Software Configuration Management
  a. Establish and maintain the integrity of work products
  b. Product baseline
  c. Baseline authority

b. Level 3 KPA

- Organization Process Focus
  a. Establish organizational responsibility for software process activities that improve the organization's overall software process capability.
- Organization Process Definition
  a. Develop and maintain a usable set of software process assets which are stable foundation that can be institutionalized and a basis for defining meaningful data for quantitative process management
- Training Program
  - o Develop skills and knowledge so that individual can perform their roles effectively and efficiently
  - o Organizational responsibility
  - o Needs identified by project
- Integrated Software Management
  - o Integrated engineering and management activities
  - o Engineering and management processes are tailored from the organizational standard processes
  - o Tailoring based on business environment and project needs
- Software Product Engineering
  - o technical activities of the project are well defined (SDLC)
  - o correct, consistent work products
- Intergroup Coordination
  - o Software engineering groups participate actively with other groups
- Peer Reviews
  - o early defect detection and removal
  - o better understanding of the products
  - o implemented with inspections, walkthroughs, etc

c.Level 4 KPA

- Quantitative Process Management
  - o control process performance quantitatively
  - o actual results from following a software process
  - o focus on identifying and correcting special causes of variation with respect to a baseline process

- Software Quality Management
  quantitative understanding of software quality- products and process

d.Level 5 KPA

- Process Change Management
  - o continuous process improvement to improve quality, increase productivity, decrease cycle time
- Technology Change Management
  - o identify and transfer beneficial new technologies
    - tools
    - methods
    - processes
- Defect Prevention
  - o causal analysis of defects to prevent recurrence

Benefits

- Helps forge a shared vision of what software process improvement means for the organization
- Defines set of priorities for addressing software problems
- Supports measurement of process by providing framework for performing reliable and consistent appraisals
- Provides framework for consistency of processes and product

Maturity Level	**Description**
1 – Initial	The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics. **Key Process areas-None**
2- Repeatable	Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications. **KPA –Requirements management,Software project planning,Software pjt tracking and oversight,Software Quality assurance,software configuration management**
3 – Defined	The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

	KPA-Organization process focus,organization process definition,training programme,integrated software management,peer reviews,inter-group oordination etc
4 – Managed	Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled. KPA-Quantitative process management and software quality management.
5 – Optimized	Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies. KPA-Defect prevention,technology change management,process change management

Q2. Explain ISO 9000

<u>ISO 9000</u>

- ISO (International Standards Organization) is a consortium of 63 countries established to formulate and foster standardization.
- ISO published its 9000 series of standards in 1987.
- The ISO 9000 standard specifies the guidelines for maintaining a quality system.
- ISO 9000 specifies a set of guidelines for repeatable and high quality product development.
- ISO 9000 standard is a set of guidelines for the production process

<u>ISO 9001</u>

- ISO 9001 applies to the organizations engaged in design, development, production, and servicing of goods.
- This is the standard that is applicable to most software development organizations.

<u>ISO 9002</u>

- ISO 9002 applies to those organizations which do not design products but are only involved in production.
- Examples of these category industries include steel and car manufacturing industries that buy the product and plant designs from external sources and are involved in only manufacturing those products.
- Therefore, ISO 9002 is not applicable to software development organizations.

<u>ISO 9003</u>

- ISO 9003 applies to organizations that are involved only in installation and testing of the products

Need for obtaining ISO 9000 Certification

- Confidence of customers in an organization increases

- ISO 9000 requires a well-documented software production process to be in place. A well-documented software production process contributes to repeatable and higher quality of the developed software.
- Makes the development process focused, efficient, and cost-effective.
- Points out the weak points of an organization and recommends remedial action.
- Sets the basic framework for the development of an optimal process and Total Quality Management

Summary of ISO 9001

- Main requirements related to software industry

Management Responsibility (4.1)

- The management must have an effective quality policy.
- The responsibility and authority of all those whose work affects quality must be defined and documented.
- A management representative, independent of the development process, must be responsible for the quality system.
- The effectiveness of the quality system must be periodically reviewed by audits.

Quality System (4.2)

- A quality system must be maintained and documented.

Contract Reviews (4.3)

- Before entering into a contract, an organization must review the contract to ensure that it is understood, and that the organization has the necessary capability for carrying out its obligations.

Design Control(4.4)

- The design process must be properly controlled, this includes controlling coding also.-a good configuration control system must be in place.
- Design inputs must be verified as adequate.
- Design must be verified.
- Design output must be of required quality.
- Design changes must be controlled.

Document Control(4.5)

- There must be proper procedures for document approval, issue and removal.
- Document changes must be controlled. Thus, use of some configuration management tools is necessary.

Purchasing (4.6)

- Purchasing material, including bought-in software must be checked for conforming to requirements.

Purchaser Supplied Product (4.7)

- Material supplied by a purchaser, for example, client-provided software must be properly managed and checked.

Product Identification (4.8)

- The product must be identifiable at all stages of the process. In software terms this means configuration management.

Process Control (4.9)

- The development must be properly managed.
- Quality requirement must be identified in a quality plan.

Inspection and Testing (4.10)

- In software terms this requires effective testing i.e., unit testing, integration testing and system testing. Test records must be maintained.

Inspection, Measuring and Test Equipment (4.11)

- If integration, measuring, and test equipments are used, they must be properly maintained and calibrated.

Inspection and Test Status (4.12)

- The status of an item must be identified. In software terms this implies configuration management and release control.

Control of Nonconforming Product (4.13)

- In software terms, this means keeping untested or faulty software out of the released product, or other places whether it might cause damage.

Corrective Action(4.14)

- This requirement is both about correcting errors when found, and also investigating why the errors occurred and improving the process to prevent occurrences. If an error occurs despite the quality system, the system needs improvement.

Handling, (4.15)

- This clause deals with the storage, packing, and delivery of the software product.

Quality records (4.16)

- Recording the steps taken to control the quality of the process is essential in order to be able to confirm that they have actually taken place.

Quality Audits (4.17)

- Audits of the quality system must be carried out to ensure that it is effective.

Training (4.18)

- Training needs must be identified and met.

Salient Features of ISO 9001 Certification

- All documents concerned with the development of a software product should be properly managed, authorized, and controlled. This requires a configuration management system to be in place.
- Proper plans should be prepared and then progress against these plans should be monitored.
- Important documents should be independently checked and reviewed for effectiveness and correctness.
- The product should be tested against specification.
- Several organizational aspects should be addressed e.g., management reporting of the quality team.

Short Comings of ISO 9000 Certification

- ISO 9000 requires a software production process to be adhered to but does not guarantee the process to be of high quality.
- It also does not give any guideline for defining an appropriate process.
- ISO 9000 certification process has no international accreditation agency exists.
- Therefore it is likely that variations in the norms of awarding certificates can exist among the different accreditation agencies and also among the registrars.
- Organizations getting ISO 9000 certification often tend to downplay domain expertise.

- These organizations start to believe that since a good process is in place, any engineer is as effective as any other engineer in doing any particular activity relating to software development.
- ISO 9000 does not automatically lead to continuous process improvement,

3. Explain different types of Software Requirements?
   - Requirements Engineering:- It is the process of establishing the services
     - that the customer requires from a system
     - and the constraints under which it operates and is developed
   - The requirements themselves are the descriptions of the
     - system services
     - and constraints that are generated during the requirements engineering process

The types of requirements

- User requirements
  - Statements in natural language plus diagrams of the services the system provides and
  - its operational constraints. Written for customers
- System requirements
  - A structured document setting out detailed descriptions of the system services.
  - Written as a contract between client and contractor
- Software specification
  - A detailed software description which can serve as a basis for a design or implementation. Written for developers
- Functional requirements- Statements or services the system should provide
  - how the system should react to particular inputs and how the system should behave in particular situations.
- Non-functional requirements- Constraints on the services
  - functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- Domain requirements
- Define system properties and constraints
- e.g. reliability, response time and storage requirements.
- Constraints are I/O device capability, system representations, etc.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless

4. Compare functional and functional requirements

Non functional requirements-

- Define system properties and constraints
- e.g. reliability, response time and storage requirements.
- Constraints are I/O device capability, system representations, etc.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless
  - **Product requirements-** Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.

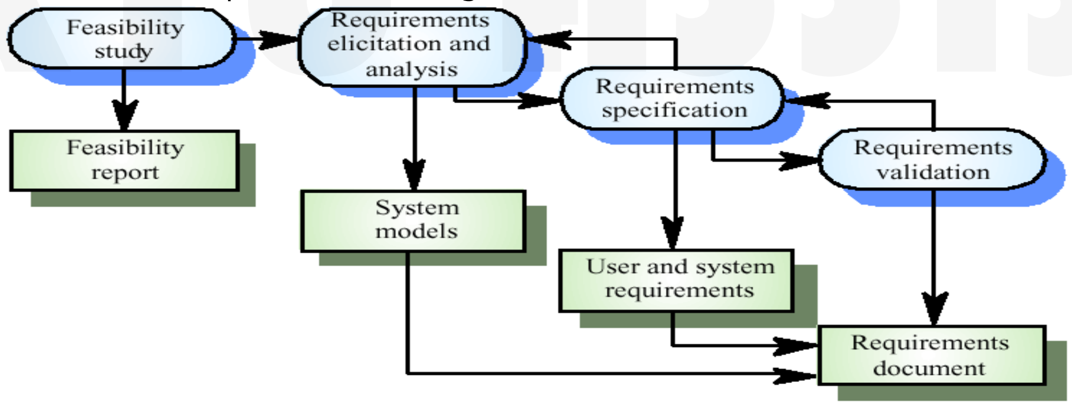
    - **Organisational requirements-** Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
    - **External requirements-** Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Functional Requirements

- Describe functionality or system services
- Depend on the type of software, expected users and the type of system where the software is used
- Functional user requirements may be high-level statements of what the system should do
- but functional system requirements should describe the system services in detail

5. Explain the activities in software requirement engineering process

- Requirement Engineering process- Processes used to discover, analyse and validate system requirements
- The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements
- RE processes are
    - Requirements elicitation
    - Requirements analysis
    - Requirements validation
    - Requirements management



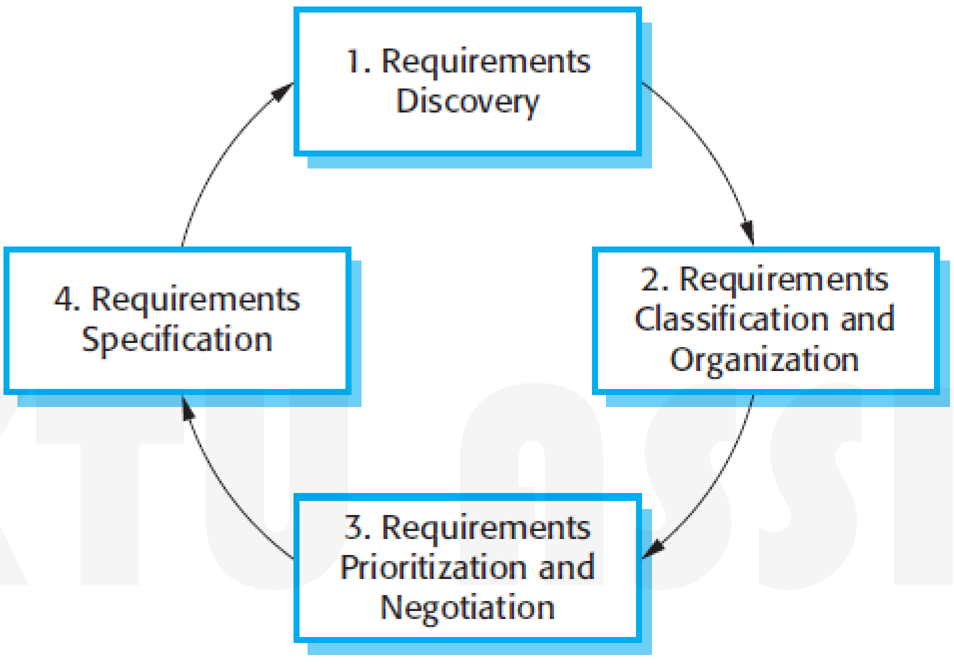
1. Feasibility study An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies. The study considers whether the proposed system will be cost-effective from a business point of view and if it can be developed within existing budgetary constraints. A feasibility study should be relatively cheap and quick. The result should inform the decision of whether or not to go ahead with a more detailed analysis.

2. Requirements elicitation and analysis This is the process of deriving the system requirements through observation of existing systems, discussions with potential users and procurers, task analysis, and so on. This may involve the development of one or more system models and prototypes. These help you understand the system to be specified.

3. Requirements specification Requirements specification is the activity of translating

the information gathered during the analysis activity into a document tha tdefines a set of requirements. Two types of requirements may be included in this
document. User requirements are abstract statements of the system requirements
for the customer and end-user of the system; system requirements are a
more detailed description of the functionality to be provided.
4. *Requirements validation* This activity checks the requirements for realism, consistency,
and completeness. During this process, errors in the requirements document
are inevitably discovered. It must then be modified to correct these problems.

6. Explain the activities involved in requirement elicitation and analysis


1. Requirements Discovery
2. Requirements Classification and Organization
3. Requirements Prioritization and Negotiation
4. Requirements Specification

The process activities are:
1. *Requirements discovery* This is the process of interacting with stakeholders of the
system to discover their requirements. Domain requirements from stakeholders and
documentation are also discovered during this activity. There are several complementary
techniques that can be used for requirements discovery, which I discuss
later in this section.
2. *Requirements classification and organization* This activity takes the unstructured
collection of requirements, groups related requirements, and organizes
them into coherent clusters. The most common way of grouping requirements is
to use a model of the system architecture to identify sub-systems and to associate
requirements with each sub-system. In practice, requirements engineering
and architectural design cannot be completely separate activities.
3. *Requirements prioritization and negotiation* Inevitably, when multiple stakeholders
are involved, requirements will conflict. This activity is concerned with
prioritizing requirements and finding and resolving requirements conflicts
through negotiation. Usually, stakeholders have to meet to resolve differences
and agree on compromise requirements.
4.Requirements specification The requirements are documented and input into the

next round of the spiral.

Eliciting and understanding requirements from system stakeholders is a difficult process for several reasons:
1. Stakeholders often don't know what they want from a computer system except in the most general terms; they may find it difficult to articulate what they want the system to do; they may make unrealistic demands because they don't know what is and isn't feasible.
2. Stakeholders in a system naturally express requirements in their own terms and with implicit knowledge of their own work. Requirements engineers, without experience in the customer's domain, may not understand these requirements.
3. Different stakeholders have different requirements and they may express these in different ways. Requirements engineers have to discover all potential sources of requirements and discover commonalities and conflict.
4. Political factors may influence the requirements of a system. Managers may demand specific system requirements because these will allow them to increase their influence in the organization.
5. The economic and business environment in which the analysis takes place is dynamic. It inevitably changes during the analysis process. The importance of particular requirements may change. New requirements may emerge from new stakeholders who were not originally consulted.

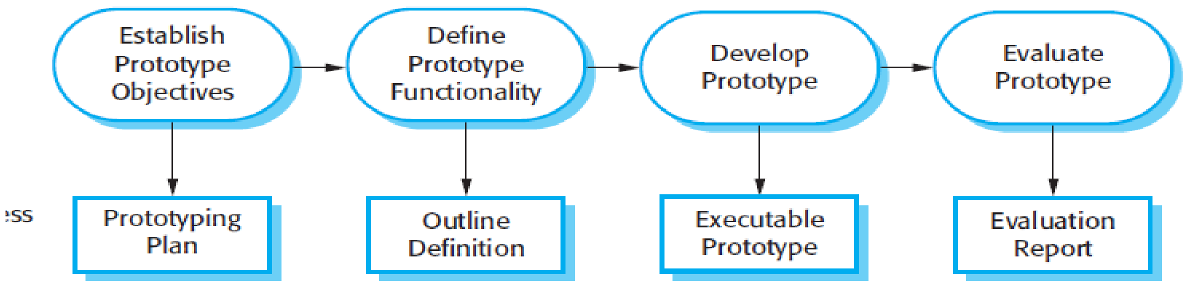
7. Explain Software Prototyping

- A prototype is an initial version of a software system that is used to demonstrate concepts, try out design options, and find out more about the problem and its possible solutions.
- Rapid, iterative development of the prototype is essential so that costs are controlled and system stakeholders can experiment with the prototype early in the software process.
- A software prototype can be used in a software development process to help anticipate changes that may be required:

1. In the requirements engineering process, a prototype can help with the elicitation and validation of system requirements.
2. In the system design process, a prototype can be used to explore particular software solutions and to support user interface design.

- System prototypes allow users to see how well the system supports their work. They may get new ideas for requirements, and find areas of strength and weakness in the software.
- Furthermore, as the prototype is developed, it may reveal errors and omissions in the requirements that have been proposed.
- Prototyping is also an essential part of the user interface design process. Because of the dynamic nature of user interfaces, textual descriptions and diagrams are not good enough for expressing the user interface requirements. Therefore, rapid prototyping with end-user involvement is the only sensible way to develop graphical user interfaces for software systems.
- A function described in a specification may seem useful and well defined. However, when that function is combined with other functions, users often find that their initial view was incorrect or incomplete. The system specification may then be modified to reflect their changed understanding of the requirements. A system prototype may be

used while the system is being designed to carry out design experiments to check the feasibility of a proposed design.



Activities in software prototyping

- A process model for prototype development is shown in Figure. The objectives of prototyping should be made explicit from the start of the process. These may be to develop a system to prototype the user interface, to develop a system to validate functional system requirements, or to develop a system to demonstrate the feasibility of the application to managers.
- The same prototype cannot meet all objectives. If the objectives are left unstated, management or end-users may misunderstand the function of the prototype. Consequently, they may not get the benefits that they expected from the prototype development.
- The next stage in the process is to decide what to put into and, perhaps more importantly, what to leave out of the prototype system. To reduce prototyping costs and accelerate the delivery schedule, you may leave some functionality out of the prototype. You may decide to relax non-functional requirements such as response time and memory utilization. Error handling and management may be ignored unless the objective of the prototype is to establish a user interface. Standards of reliability and program quality may be reduced.
- The final stage of the process is prototype evaluation. Provision must be made during this stage for user training and the prototype objectives should be used to derive a plan for evaluation. Users need time to become comfortable with a new system and to settle into a normal pattern of usage. Once they are using the system normally, they then discover requirements errors and omissions

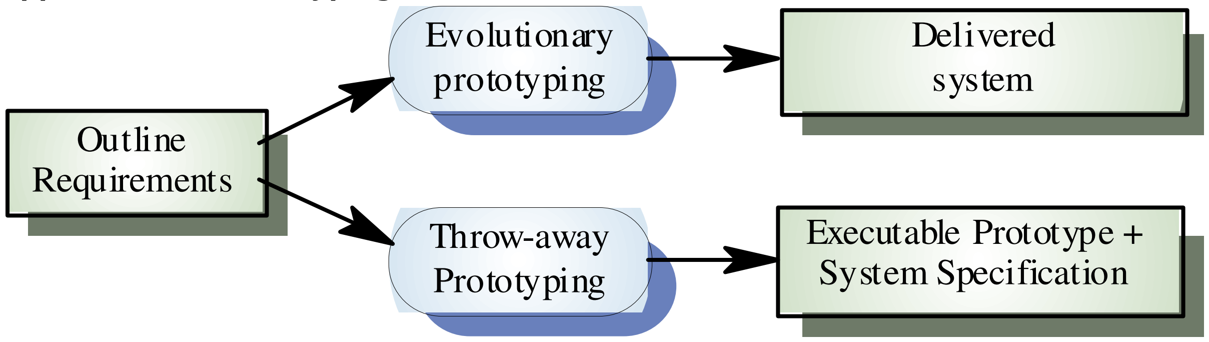
✓ A general problem with prototyping is that the prototype may not necessarily be used in the same way as the final system. The tester of the prototype may not be typical
of system users.

✓ The training time during prototype evaluation may be insufficient.
If the prototype is slow, the evaluators may adjust their way of working and
avoid those system features that have slow response times. When provided with better response in the final system, they may use it in a different way.

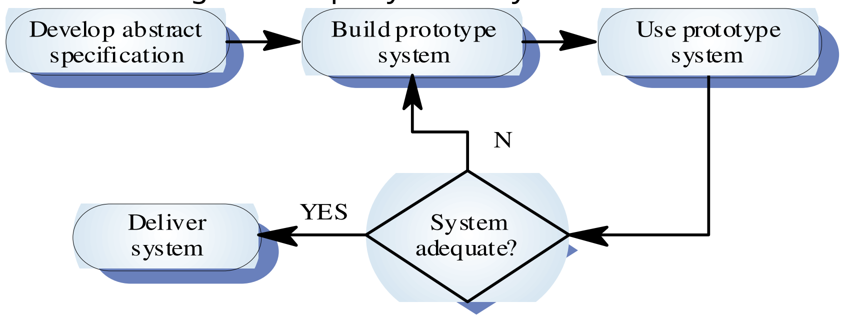
✓ Developers are sometimes pressured by managers to deliver throwaway prototypes,
particularly when there are delays in delivering the final version of the software.
However, this is usually unwise:
1. It may be impossible to tune the prototype to meet non-functional requirements,

such as performance, security, robustness, and reliability requirements, which were ignored during prototype development.
2. Rapid change during development inevitably means that the prototype is undocumented.
The only design specification is the prototype code. This is not good enough for long-term maintenance.
3. The changes made during prototype development will probably have degraded the system structure. The system will be difficult and expensive to maintain.
4. Organizational quality standards are normally relaxed for prototype development.
Prototypes do not have to be executable to be useful

Approaches to Prototyping



1. **Evolutionary prototyping**
   - An initial prototype is produced and refined through a number of stages to the final system
   - to deliver a working system to end-users. The development starts with those requirements which are best understood.
   - Must be used for systems where the specification cannot be developed in advance e.g. AI systems and user interface systems
   - Based on techniques which allow rapid system iterations
   - Verification is impossible as there is no specification. Validation means demonstrating the adequacy of the system



Advantages

- ➢ Accelerated delivery of the system
  - Rapid delivery and deployment are sometimes more important than functionality or long-term software maintainability
- ➢ User engagement with the system

- Not only is the system more likely to meet user requirements, they are more likely to commit to the use of the system

Problems with evolutionary Prototyping

Management problems

- Existing management processes assume a waterfall model of development
- Specialist skills are required which may not be available in all development teams

Maintenance problems

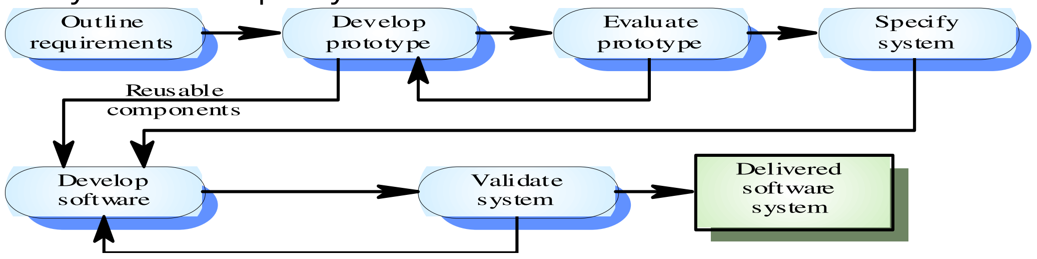
- Continual change tends to corrupt system structure so long-term maintenance is expensive.

Contractual problems

- Some parts of the requirements (e.g. safety-critical functions) may be impossible to prototype and so don't appear in the specification
- An implementation has no legal standing as a contract
- Non-functional requirements cannot be adequately tested in a system prototype

2. Throw-away prototyping

- A practical implementation of the system is produced to help discover requirements problems and then discarded. The system is then developed using some other development process.
- to validate or derive the system requirements. The prototyping process starts with those requirements which are poorly understood.
- Used to reduce requirements risk
- The prototype is developed from an initial specification, delivered for experiment then discarded
- The throw-away prototype should NOT be considered as a final system
- Some system characteristics may have been left out
- There is no specification for long-term maintenance
- The system will be poorly structured and difficult to maintain



- Developers may be pressurised to deliver a throw-away prototype as a final system
- This is not recommended
    - It may be impossible to tune the prototype to meet non-functional requirements
    - The prototype is inevitably undocumented

- The system structure will be degraded through changes made during development
- Normal organisational quality standards may not have been applied

8. Explain Software specification

Requirements specification is the activity of translating the information gathered during the analysis activity into a document that defines a set of requirements. Two types of requirements may be included in this document. User requirements are abstract statements of the system requirements
for the customer and end-user of the system; system requirements are more detailed description of the functionality to be provided.

Types of software system requirements:

- *Functional requirements,* describe the requested functionality/behaviour of the system: services (functions), reactions to inputs, exceptions, modes of operations
    - Depend on the system, the software, and the users
    - Can be expressed at different levels of detail (user/system requirements)
    - For a system, it is desirable to have a complete and consistent set of functional requirements
    - *Completeness*: all required system facilities are defined
    - *Consistency*: there are no contradictions in requirements
- *Non-functional requirements,* represent constraints on the system and its functionality: performance constraints, compliance with standards, constraints on the development process
    - Many apply to the system as a whole
    - More critical than individual functional requirements
    - More difficult to verify

  Kinds of non-functional requirements:
    - Product requirements
    - Organizational requirements
    - External requirements

- *Domain requirements,* can be either functional or non-functional and reflect the particularities of the application domain.
    - Domain requirements indicate specific computations, additional functionality, or constraints on other requirements

Software Requirement Specification

- This is the way of representing requirements in a consistent format
- It Serves as a contract between customer & developer.

It should

- Correctly define all requirements
- not describe any design details
- not impose any additional constraints

Characteristics of Good SRS

- Correct

    - An SRS is correct if and only if every requirement stated therein is one that the software shall meet.
- Unambiguous
    - An SRS is unambiguous if and only if, every requirement stated therein has only one interpretation.
- Complete
    - An SRS is complete if and only if, it includes the following elements
    - (i) All significant requirements, whether related to functionality, performance, design constraints, attributes or external interfaces.
    - (ii)Responses to both valid & invalid inputs.
    - (iii) Full Label and references to all figures, tables and diagrams in the SRS and definition of all terms and units of measure.
- Consistent
    - An SRS is consistent if and only if, no subset of individual requirements described in it conflict.
- Stability
    - An identifier is attached to every requirement to indicate either the importance or stability of that particular requirement
- Verifiable
    - An SRS is verifiable, if and only if, every requirement stated therein is verifiable.
- Modifiable
    - An SRS is modifiable, if and only if, its structure and style are such that any changes to the requirements can be made easily, completely, and consistently while retaining structure and style.
- Traceable
    - An SRS is traceable, if the origin of each of the requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation.

IEEE has published guidelines and standards to organize an SRS.

1. Introduction
    1.1 Purpose
    1.2 Scope
    1.3 Definition, Acronyms and abbreviations
    1.4 References
    1.5 Overview

2. The Overall Description
    2.1 Product Perspective
        2.1.1 System Interfaces
        2.1.2 Interfaces
        2.1.3 Hardware Interfaces
        2.1.4 Software Interfaces
        2.1.5 Communication Interfaces
        2.1.6 Memory Constraints
        2.1.7 Operations

2.1.8 Site Adaptation Requirements

2.2Product Functions

2.3 User Characteristics

2.4 Constraints

2.5 Assumptions for dependencies

2.6 Apportioning of requirements

3. Specific Requirements

3.1 External Interfaces

3.2 Functions

3.3 Performance requirements

3.4 Logical database requirements

3.5 Design Constraints

3.6 Software System attributes

3.7 Organization of specific requirements

3.8 Additional Comments.

4. Supporting information

4.1 Table of contents and index

4.2 Appendixes

END