Module-1 Software Engineering

→ Virtually all countries now depend on complex computer based systems.

→ National infrastructures and utilities are controlled by computer-based systems & most electrical products include a computer and controlling s/w.

→ Industrial manufacturing and distribution is completely computerized.

→ Entertainment, including the music industry, computer games & film and television, is s/w intensive.

→ Producing and maintaining s/w in a cost effective way is essential for the functioning of national & international societies.

→ S/w systems are abstract and intangible.

→ They are not constrained by the properties of materials, governed by physical laws, or by manufacturing processes.

→ This simplifies s/w engineering, as there are no natural limits to the potential of s/w.

→ Because of the lack of physical constraints, s/w sys. can quickly become extremely complex, difficult to understand, & expensive to change.

→ There are many different types of s/w sys., from simple embedded sys. to complex, worldwide information sys.

→ It is pointless to look for universal notations, methods, or techniques for SE becoz different types of s/w require different approaches.

Scanned with CamScanner

→ All of these applications need SE; they do not all need the same SE techniques.

→ There are still many reports of s/w projects going wrong & 'S/w failures'.

→ S/w engineering is criticized as inadequate for modern s/w development.

→ Many of these so-called s/w failures are consequence of two factors.

1. Increasing demands: As new SE techniques help us to build larger, more complex sys., the demands change.
   ↳ Sys. have to be built and delivered more quickly
   ↳ Sys. have to have new capabilities that were previously thought to be impossible.
   ↳ Existing SE methods cannot cope and new SE techniques have to be developed to meet new demands.

2. Low expectations: It is easy to write computer pgms without using SE methods and techniques.
   ↳ Many companies have drifted into s/w development as their products and services have evolved. They do not use SE methods in their every day work. Consequently, their s/w is often more expensive & less reliable than it should be.
   ↳ Need better SE eduction & training to address this problem.
   ↳ lack of best practices lead to s/w crisis.

Scanned with CamScanner

Professional s/w development

→ Lots of people write programs.

→ People in business write spreadsheet pgms to simplify their jobs, scientists and engineers write programs to process their experimental data, & hobbyists write pgms for their own interest and enjoyment.

→ The vast majority of s/w development is a professional activity where s/w is developed for specific business purposes, for inclusion in other devices, or as s/w products such as information systems, CAD systems, etc.

→ Professional s/w, intended for use by someone apart from its developer, is usually developed by teams rather than individuals.

→ It is maintained and changed throughout its life.

→ S/w engineering is intended to support professional s/w development, rather than individual programming.

→ It includes techniques that support program specification, design, and evolution.

Software : Software is not just the programs but also all associated documentation & configuration data which is needed to make the programs operate correctly.

→ S/w sys. usually consists of a no. of separate programs, configuration files, that are used to setup these programs, System documentation which discribes the structure of sys., User documentation, which explains how to use the sys.

Scanned with CamScanner

→ S/w engineers are concerned with developing s/w products i.e, s/w which can be sold to a customer.

→ There are two fundamental types of s/w products

↳ Generic products: These are stand-alone systems that are produced by a development organization and sold on the open market to any customer

Ex: PC s/ws such as databases, word processors, drawing packages etc.

↳ Customized (or Bespoke) products: These are systems that are commissiond by a particular customer. A s/w contractor develops the s/w especially for that customer.

Ex: S/w include control sys. for electronic devices, sys. for business process and air traffic control systems.

Essential attributes of good software:

Product Characteristic	Description
1. Maintainability	S/w should be written in such a way that it can evolve to meet the changing needs of customers. ↳ This is a critical attribute because s/w change is an inevitable requirement of a changing business environment.
2. Dependability and Security	It includes a range of characteristics including reliability, security, and safety. ↳ Dependable s/w should not cause physical or economic damage in the event of system failure. ↳ Malicious users should not be able to access or damage the system.
3. Efficiency	S/w should not make wasteful use of sys. resources such as memory and processor cycles. ↳ Efficiency ∴ includes responsiveness, processing time, memory utilization etc.

Scanned with CamScanner

4. Acceptability: S/w must be acceptable to the type of users for which it is designed.
↳ This means that it must be understandable, usable, & compatible with other systems that they use.

Software engineering

→ Software engineering is an engineering discipline, that is concerned with all aspects of s/w production from the system specification to maintaining the system.

↳ Engineering discipline:
↳ Engineers make things work. They use theories, methods & tools
↳ Engineers also recognize that they must work to organizational and financial constraints. They look solutions within constraints.

↳ All aspects of software production:
↳ S/w engineering is not just concerned with the technical processes of s/w development, but also with activities such as s/w project management and the development of tools, methods & theories to support s/w production.

→ S/w engineers adopt a systematic and organized approach to their work

→ Software engineering is important for two reasons:
↳ More & more, individuals & society rely on advanced s/w sys. That need to be able to produce reliable & trustworthy sys. economically and quickly.
↳ It is usually cheaper, in the long run, to use s/w engineering methods and techniques for s/w sys. rather than just write the programs as if it was a personal programming project.
↳ For most types of sys., the majority of costs are the costs of changing the s/w after it has gone into use.

Scanned with CamScanner

Software Process:

→ Software process is a sequence of activities that leads to the production of a s/w product.

→ The Systematic approach that is used in s/w engineering is sometimes called a s/w process.

There are 4 fundamental activities

- Software specification: where customers and engineers define the s/w that is to be produced and the constraints on its operation.
- Software development: Where the s/w is designed & programmed.
- Software Validation: Where the s/w is checked to ensure that it is what the customer requires.
- Software evolution: Where the s/w is modified to reflect changing customer and market requirements.

→ Different types of sys. need different development processes.

Ex: Real time s/w in an aircraft has to be completely specified before development begins.

E-commerce sys., the specification & the pgm are usually developed together.

→ S/w engineering is related to both computer science & sys. engineering:

↳ Computer science is concerned with theories & methods which underlie computers and s/w systems,

↳ S/w engineering is concerned with the practical problems of producing s/w.

↳ Some knowledge of Computer Science is essential for s/w engineers.

Scanned with CamScanner

↳ System engineering is concerned with all aspects of the development and evolution of complex systems, where s/w plays a major role.

↳ ∴ It is concerned with H/w development, policy & process design & system deployment, as well as SE.

↳ System engineers are involved in specifying the system, defining its overall architecture, and than integrating the different parts to create the finished system.

↳ System engineering is the older discipline than s/w engineering.

Key challenges facing software engineering

There are 3 general issues that affect many different types of software:

↳ Heterogeneity: Increasingly, systems are required to operate as distributed systems across networks that include different types of computer & mobile devices.

→ The challenge here is to develop techniques for building dependable software that is flexible enough to cope with this heterogeneity.

↳ Business and social change: Business and society are changing incredibly quickly as emerging economies develop and new technologies become available.

→ They need to be able to change their existing s/w & to rapidly develop new s/w. (delivers

→ Many traditional s/w engineering techniques are time consuming and delivery of new systems often takes longer than planned.

→ They need to evolve so that the time required for s/w to deliver value to its customers is reduced.

Scanned with CamScanner

↳ Security and trust: As s/w is intertwined with all aspects of our lives, it is essential that we can trust that s/w.
→ This is especially true for remote s/w systems accessed through a web page or web service interface.
→ We have to make sure that malicious users cannot attack our s/w and that information security is maintained.

Costs of s/w engineering:
→ Roughly 60% of s/w costs are development costs; 40% are testing costs.
→ For custom software, evolution costs often exceed development costs.

The best s/w engineering techniques and methods:
→ While all s/w projects have to be professionally managed and developed, different techniques are appropriate for different types of system.
Ex: Games should always be developed using a series of prototypes, whereas safety critical control systems require a complete & analyzable specification to be developed.
∴ can't say that one method is better than another.

Differences has the Web made to software engineering:
→ The Web has led to the availability of s/w services & the possibility of developing highly distributed service-based systems.
→ Web based systems development has led to important advances in programming languages and software reuse.

Scanned with CamScanner

S/w Engineering diversity

→ S/w engineering is a systematic approach to the production of S/w that takes into account practical cost, schedules and dependability issues, as well as the needs of S/w customers and producers.

→ There are no universal SE Methods and techniques that are suitable for all systems & all companies.

→ The most significant factor in determining which S/w Engg. methods & techniques are most important is the type of application that is being developed.

There are many different types of application including:

1. Stand-alone applications: These are application sys. that run on a local computer, such as a PC.
   ↳ They include all necessary functionality & do not need to be connected to a N/w.
   Ex: Office apps. on a PC, CAD Pgms, photo manipulation S/w etc.

2. Interactive transaction-based applications: These are applications that execute on a remote computer and that are accessed by users from their own PC's or terminals.
   ↳ These include web applications such as e-commerce applications where can interact with a remote sys. to buy goods & services, and business sys. etc.
   ↳ These often incorporate a large data store that is accessed and updated in each transaction.

3. Embedded control systems: These are S/w control sys. that control & manage H/w devices.
   Ex: S/w in mobile phone, S/w that controls anti lock braking in a car & S/w in microwave oven to control the cooking process.

Scanned with CamScanner

4. Batch processing systems: These are business systems that are designed to process data in large batches.
↳ They process large no. of individual i/ps to create corresponding o/ps
Ex: Periodic billing sys., such as phone billing sys. & salary payment systems.

5. Entertainment systems: These are sys. that are primarily for personal use & which are intended to entertain the user.
↳ The quality of the user interaction offered is the most imp. distinguishing characteristic of entertainment sys.

6. Systems for modeling and simulation: These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.
↳ These are often computationally intensive & require high-performance parallel systems for execution.

7. Data collection systems: These are systems that collect data from their environment using a set of sensors and send that data to other systems for processing.
↳ The s/w has to interact with sensors and often is installed in a hostile environment such as inside an engine or in a remote location.

8. Systems of systems: These are sys. that are composed of a no. of other s/w systems.
↳ Some of these may be generic s/w products, such as a spreadsheet programs
↳ Other systems in the assembly may be specially written for that environment.

Scanned with CamScanner

There are s/w engineering fundamentals that apply to all types of s/w systems:

1. They should be developed using a managed and understood development process. The organization developing s/w should plan the development process and have clear ideas of what will be produced & when it will be completed. Different processes are used for different types of s/w.
2. Dependability & performance are important for all types of systems. S/w should behave as expected, without failures and should be available for use when it is required. It should be safe in its operation and as far as possible, should be secure against external attack. The system should perform efficiently and should not waste resources.
3. Understanding & managing the s/w specification & requirements (what s/w should do) are important. Have to know what different customers & users of the system expect from it & Have to manage their expectations so that a useful system can be delivered within budget and to schedule.
4. Should make as effective use as possible of existing resources. This means that, where appropriate, should reuse s/w that has already been developed rather than write new software.

These are fundamental notions of process, dependability, requirements, management and reuse.

Scanned with CamScanner

Software engineering and the Web

→ The development of the World Wide Web has has a great profound effect on all of our lives.

→ Initially, the Web was primarily a universally accessible information store and it had little effect on s/w systems.

→ Around 2000, the Web started to evolve and more and more functionality was added to browsers.

→ This meant that web-based systems could be developed where, instead of a special purpose user interface, these systems could be accessed using a web browser.

→ This led to the development of a vast range of new sys products that delivered innovative services, accessed over Web.

→ These sys. products, the development of web browsers that could run small programs and do some local processing led to an evolution in business and organizational s/w.

→ Instead of writing s/w & deploying it on users PC's, the s/w was deployed on a web server.

→ This made it much cheaper to change & upgrade the s/w, as there was no need to install the s/w on every PC. It also reduced costs, as user interface development is expensive. So many businesses have moved to web based interaction with company s/w systems.

→ The next stage in the development of web based sys. was web services.

→ Web services are s/w components that deliver specific, useful functionality & which are accessed over the Web.

→ Applications are constructed by integrating these web services, which may be provided by different companies.

Scanned with CamScanner

→ In the last few years, 'Software as a service' has been developed.

→ It has been proposed that s/w will not normally run on local computers but will run on 'computing clouds' that are accessed over the Internet.

→ A computing cloud is a huge no. of linked computer sys. that is shared by many users.

→ Users do not buy s/w but pay according to how much the s/w is used or are given free access in return for watching adverts that are displayed on their screen.

→ Now s/w is highly distributed, sometimes across the world.

→ Business applications are not programmed from scratch but involve extensive reuse of components and programs.

The change in s/w organization has led to changes in the ways the web based systems are engineered. For ex.

→ S/w reuse has become the dominant approach for constructing web based systems. When building these systems, we think about how we can assemble them from pre-existing s/w components & sys.

→ It is now generally recognized that it is impractical to specify all the requirements for such systems in advance. Web-based systems should be developed & delivered incrementally.

→ User interfaces are constrained by the capabilities of web browsers.

Scanned with CamScanner

Software engineering ethics

→ S/w engineers must accept their job responsibilities.

→ They must behave in an ethical & morally responsible way, if they are to be respected as professionals.

→ S/w engineers should uphold normal standards of honesty and integrity.

→ They should not use their skills & abilities to behave in a dishonest way.

Professional responsibilities are:

↳ Confidentiality: Should respect the confidentiality of their employers or clients, irrespective of whether a formal confidentiality agreement has been signed.

↳ Competence: Should not misrepresent their level of competence.

→ They should not knowingly accept work that is outside their competence.

↳ Intellectual property rights: Should be aware of local laws governing, the use of intellectual property such as patents & copyright.

→ They should be careful to ensure that the intellectual property of employers & clients is protected.

↳ Computer misuse: Should not use their technical skills to misuse other people's computer. It ranges from trivial (game play) to extremely serious (dissemination of viruses or other malware).

Scanned with CamScanner

→ Organizations such as the ACM, the IEEE & BCS publish a code of ethics.

S/w engineering code of ethics & professional practice

ACM/ IEEE-CS code of ethics preamble:

S/w engineers shall commit themselves to making the analysis specification, design, development, testing & maintenance of S/w a beneficial & respected profession.

→ In accordance with their commitment to the health, safty & welfare of the public S/Er's shall adhere to the following 8 principles:

Code of ethics:

Public: S/Er's shall act consistently with the public interest.

Client and Employer: S/Er's shall act in a manner that is in the best interests of their client and employer consistant with the public interest.

Product: S/Er's shall ensure that their products & related modifications meet the highest professional standards possible.

Judgment: S/Er's shall maintain integrity & independence in their professional judgment.

Management: S/E managers & leaders shall subscribe to and promote an ethical approach to the management of S/w development & maintenance.

Profession: S/Er's shall advance the integrity & reputation of the profession consistent with the public interest.

Scanned with CamScanner

Colleagues: S/Er's shall be fair to and supportive of their colleagues.

→ ~~Virtually all com~~

Self: S/Er's shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Case studies

The 3 types of systems use as case studies

1. An embedded system: This is a system where the s/w controls a H/w device and is embedded in that device.
   → Issues in embedded systems typically include physical size, responsiveness, power management etc.
   Ex: A s/w system to control a medical device.
2. An information system: This is a system whose primary purpose is to manage & provide access to a database of information.
   → Issues in information systems include security, usability, privacy, & maintaining data integrity.
   Ex: A medical records system
3. A sensor-based data collection system: This is a system whose primary purpose is to collect data from a set of sensors & process that data in some way.
   → The key requirements of such systems are reliability & maintainability
   Ex: A wilderness weather station.

Scanned with CamScanner

→ An insuline pump Control System

→ An insuline pump is a medical system that simulates the operation of the pancreas (an internal organ).

→ The s/w controlling this system is an embedded sys., which collects info. from a sensor and controls a pump that delivers a controlled dose of insuline to a user.

→ People who suffer from diabetes use the system.

→ Diabetes is a relatively common condition where the human pancreas is unable to produce sufficient quantities of a hormone called insulin.

→ Insulin metabolises glucose in the blood. The conventional treatment of diabetes involved regular injections of insuline

→ Diabetics measure their blood sugar levels using an external meter & then calculate the dose of insulin that they should inject.

→ The problem with this treatment is that the level of insulin in the blood doesnot just depend on the blood glucose level but is a function of time when the insulin injection was taken.

→ This can lead to very low levels of blood glucose, if there is too much insulin or very high levels of blood sugar if there is too little insulin. These result serious problems.

→ Current advances in developing miniatured sensors have meant that it is now posible to develop automated insulin delivery systems.

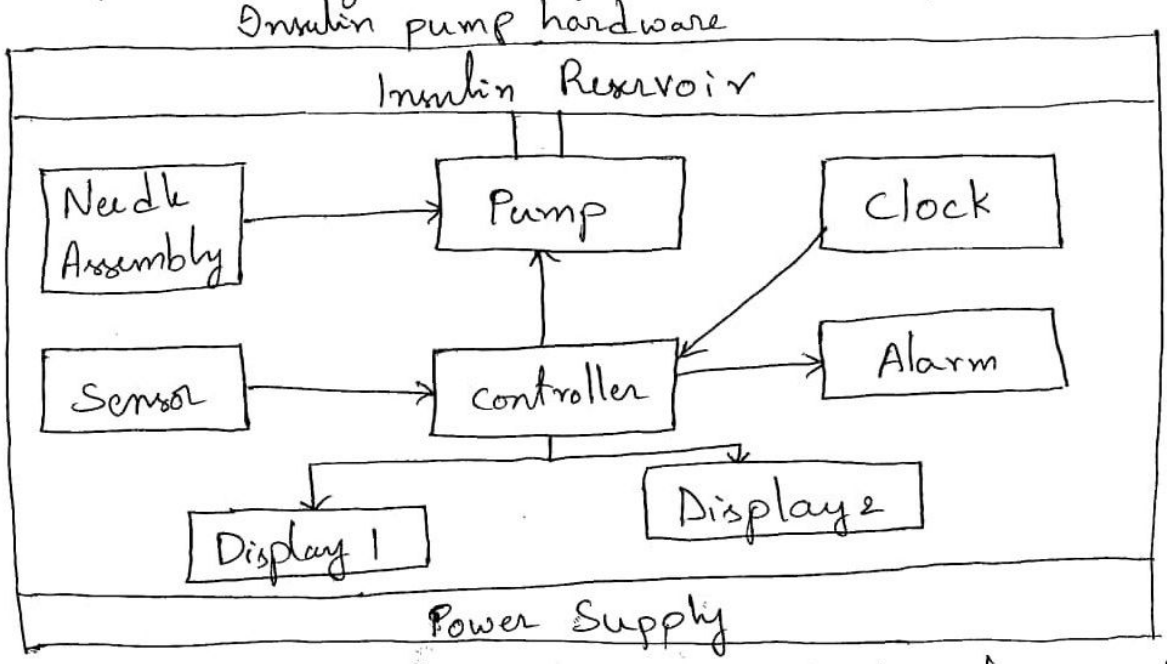
→ These systems monitor blood sugar levels and deliver an appropriate dose of insulin when required.

→ A s/w controlled insulin delivery system works by using a micro sensor embedded in the patient to measure some blood parameter that is proportional to the sugar level. This is then sent to the pump controller.

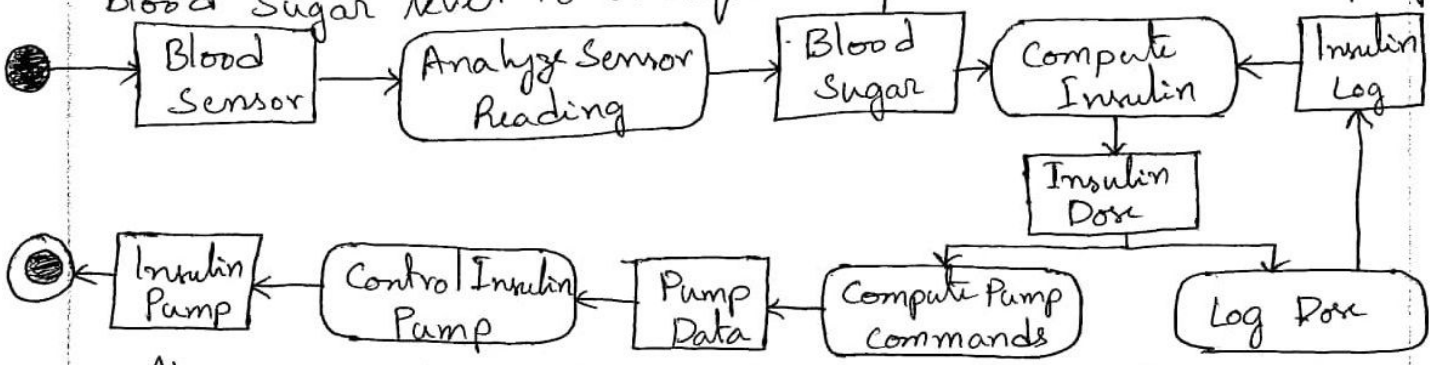
→ This controller computes the sugar level & the amount of insulin that is needed. It then sends signals to a miniaturised pump to deliver the insulin via a permanently attached needle.

Scanned with CamScanner

The components & organisation of the insulin pump shown below:



A UML activity model that shows how s/w transforms an i/p blood sugar level to a sequence of commands that drive pump.



→ It is a safety critical system. If the pump fails to operate or does not operate correctly, then the user's health may be damaged or they may fall into a coma becoz their blood sugar levels are too high/low.

There are 2 essential high level requirements this sys. must meet:

↳ The system shall be available to deliver insulin when required.

↳ The system shall perform reliably & deliver the correct amount of insulin to counteract the current level of blood sugar.

∴ The system must be designed & implemented to ensure that the system always meets these requirements.

Scanned with CamScanner

A patient information system for mental health care

→ A patient information system to support mental health care is a medical information system that maintains information about patients suffering from mental health problems & the treatments that they have received.

→ Most mental health patients do not require dedicated hospital treatment but need to attend specialist clinics regularly where they can meet a doctor who has detailed knowledge of their problems.

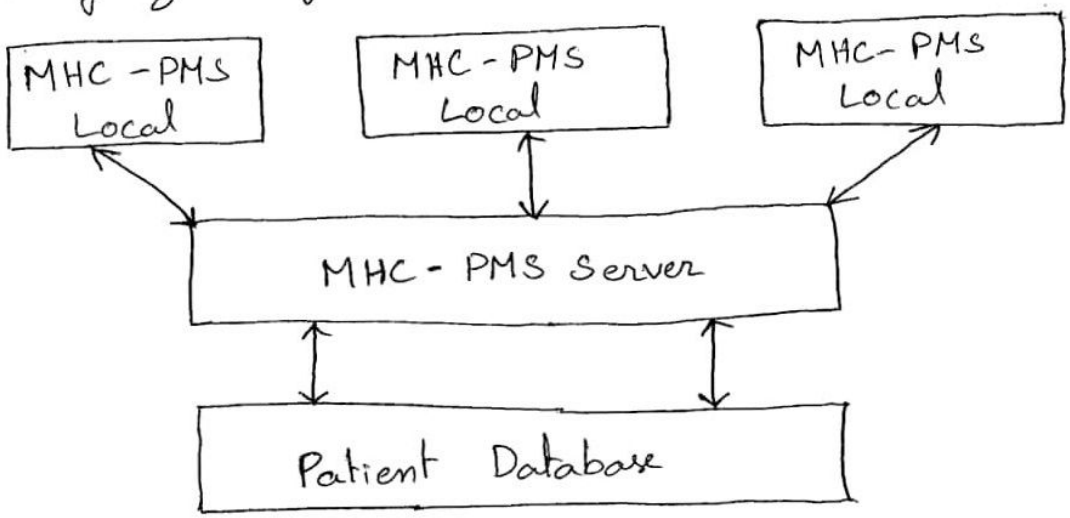
→ The MHC - PMS (Mental Health Care - Patient Management System) is an information system that is intended for use in clinics.

→ It makes use of a centralized database of patient info. but has also been designed to run on a PC, so that it may be accessed & used from sites that do not have secure n/w connectivity.

→ When the local systems have secure n/w access, they use patient info. in the database but they can download & use local copies of patient records when they are disconnected.

→ The system interact & exchange data with other clinical info. systems.

The organization of the MHC - PMS



Scanned with CamScanner

The MHC - PMS has 2 overall goals:

↳ To generate management info. that allows health service managers to assess performance against local & government targets.

↳ To provide medical staff with timely info. to support the treatment of patients.

→ The nature of mental health problems is such that patients are often disorganized, so may miss appointments, deliberately or accidentally lose prescriptions and medication, forget instructions & make unreasonable demands on medical staff.

→ They may drop in on clinics unexpectedly.

→ In some cases, they may be a danger to themselves or to other people.

→ They may regularly change address or may be homeless on a long term or short-term basis.

→ Where patients are dangerous, they may need to be Sectioned - confined to a secure hospital for treatment & observation.

→ Users of the system include clinical staff such as doctors, nurses, & health visitors (nurses who visit people at home to check on their treatment).

→ Nonmedical users include receptionists who make appointments medical records staff who maintain the records system, & administrative staff who generate reports.

→ The system is used to record info. about patients (name, address, age, next of kin, etc), consultations (data, doctor seen, subjective impressions of the patient, etc), conditions, & treatments.

→ Reports are generated at regular intervals for medical staff and health authority managers.

→ Reports for medical staff focus on info. about individual patients, management reports concerned with conditions, cost of treatment etc.

Scanned with CamScanner

The key features of the system are:

↳ Individual care management: Clinicians can create records for patients, edit the information in the system, view patient history, etc.

→ The system supports data summaries so that doctors who have not previously met a patient can quickly learn about the key problems & treatments that have been prescribed.

↳ Patient monitoring: The system regularly monitors the records of patients that are involved in treatment & issues warnings if possible problems are detected.

→ ∴ If a patient has not seen a doctor for some time, a warning may be issued.

→ One of the most important elements of the monitoring system is to keep track of patients who have been sectioned & to ensure that the legally required checks are carried out at the right time.

↳ Administrative reporting: The system generate monthly management reports showing the no. of patients treated at each clinic, the no. of patients who have entered & left the care system, no. of patients sectioned, the drugs prescribed & their costs, etc.

→ Two different laws affect the system.

→ These are laws on data protection that govern the confidentiality of personal info. & mental health laws that govern the compulsory detention of patients deemed to be a danger to themselves or others.

→ As in all medical sys., privacy is a critical system requirement. It is essential that patient info. is confidential & is never disclosed to anyone apart from authorized medical staff & the patient themselves.

Scanned with CamScanner

→ The MHC-PMS is also a safety-critical system.

→ Some mental illness cause patients to become suicidal or a danger to other people.

→ Whenever possible, the system should warn medical staff about potentially suicidal or dangerous patients.

→ The overall design of the system has to take into account privacy and safety requirements.

→ The system must be available when needed otherwise safety may be compromised & it may be impossible to prescribe the correct medication to patients.

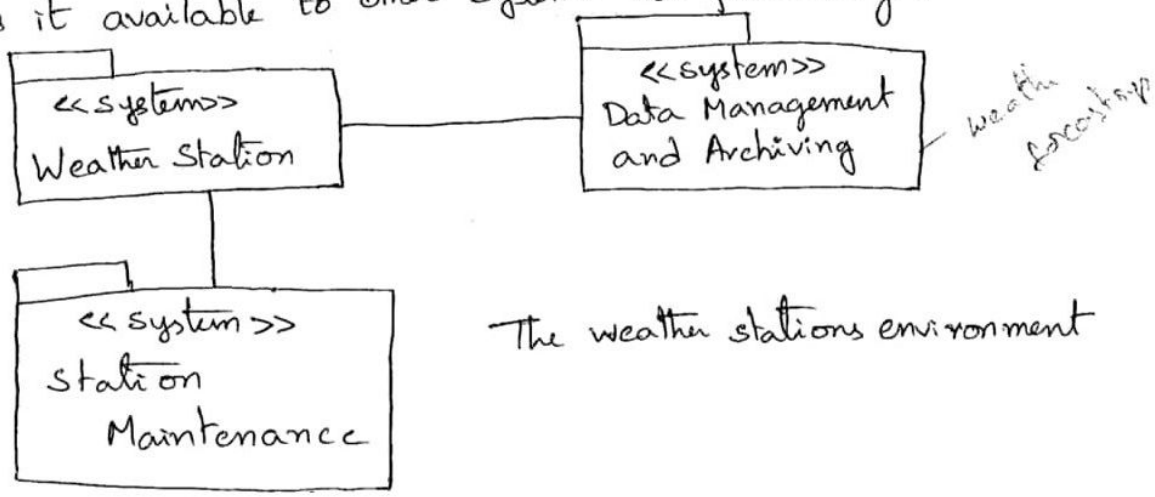
→ To ensure availability in the event of server failure or when disconnected from a N/w, multiple copies of the data should be maintained.

A Wilderness Weather Station.

→ To help monitor climate change & to improve the accuracy of weather forecasts in remote areas, the government of a country with large areas of wilderness decides to deploy several hundred weather stations in remote areas.

→ These weather stations collect data from a set of instruments that measure temperature & pressure, sunshine, rainfall, wind speed, & wind direction.

→ Wilderness weather stations are part of a large system, which is a weather info. system that collects data from weather stations & makes it available to other systems for processing.



The weather stations environment

Scanned with CamScanner

The systems in the weather stations are:

1. The weather station system: This is responsible for collecting weather data, carrying out some initial data processing and transmitting it to the data management system.

2. The data management and archiving system: This system collects the data from all of the wilderness weather stations, carries out data processing and analysis, and archives the data in a form that can be retrieved by other systems, such as weather forecasting systems.

3. The station maintenance system: This system can communicate by satellite with all wilderness weather stations to monitor the health of these systems & provide reports of problems.

→ It can update the embedded s/w in these systems. In the event of system problems, this system can also be used to remotely control a wilderness weather system.

→ The UML package symbol is used to indicate that each system is a collection of components & have identified the separate systems, using the UML stereotype <<system>>.

→ The associations between the packages indicate there is an exchange of information.

→ Each weather station includes a no. of instruments that measure weather parameters such as the wind speed & direction, the ground & air temparatures, the barometric pressure, & the rainfall over a 24 hour period.

Scanned with CamScanner

→ Each of these instruments is controlled by a s/w system that takes parameter readings periodically & manages the data collected from the instruments.

→ The weather station system operates by collecting weather observations at frequent intervals.

→ The bandwidth to the satellite is relatively narrow, the weather station carries out some local processing & aggregation of the data.

→ It then transmits this aggregated data when requested by the data collection system.

→ If, for whatever reason, it is impossible to make a connection, then the weather station maintains the data locally until communication can be resumed.

→ Each weather station is battery powered & must be entirely self-contained - there are no external power or n/w cables available.

→ All communications are through a relatively slow speed satellite link & the weather station must include some mechanism (solar or wind power) to charge its batteries.

→ Sys is deployed in wilderness areas, they exposed to severe environmental conditions & may be damaged by animals.

The station s/w is not just concerned with data collection but also

↳ Monitor the instruments, power & communication h/w & report faults to the management system.

↳ Manage the sys power, ensuring that batteries are charged whenever environmental conditions permit but also generators are shut down in damaging weather conditions, such as high wind.

↳ Allow for dynamic reconfiguration where parts of the s/w are replaced with new versions & where backup instruments are switched into the system in the event of system failure.

Scanned with CamScanner

Software Processes

→ A software process is a set of activities that leads to the production of a software product.

→ These activities may involve the development of software from scratch or modifying existing systems and by configuring & integrating off the shelf s/w or system components.

→ s/w processes are complex & rely on people making decisions & judgements.

→ There is no ideal process, and many organisation have developed their own approach to s/w development.

Eg: for critical system → Very structured development process is required.

for business system → flexible, agile process is likely to be more effective.

Fundamental activities are common to all s/w processes are:

↳ s/w specification: The functionality of the software and contraints on its operation must be defined.

↳ s/w design and implementation: The s/w to meet the specification must be produced.

↳ s/w validation: The s/w must be validated to ensure that it does what the customer wants.

↳ s/w evolution: The s/w must evolve to meet changing customer needs.

→ Software processes can be improved by process standardisation where the diversity in s/w process across an organisation is reduced. this leads to improved communication and reduction in training time.

→ standardisation is first step in introducing new s/w engineering methods & techniques.

Scanned with CamScanner

Software process models:

→ A software process model is an abstract representation of a software process.

→ Each process model represents a process from a particular perspective & provides only partial information about that process.

→ Generic models are abstractions of the process used to explain different approaches in s/w development.

Generic software process models are:

(1) The waterfall model: This takes the fundamental process activities of specification, development, validation & evolution and represents them as separate process phases such as requirements specification, software design, implementation, testing & so on.

(2) Evolutionary development: This approach interleaves the activities of specification, development and validation. An initial system is rapidly developed from abstract specifications. This is then refined with customer input to produce a system that satisfies the customer's needs.

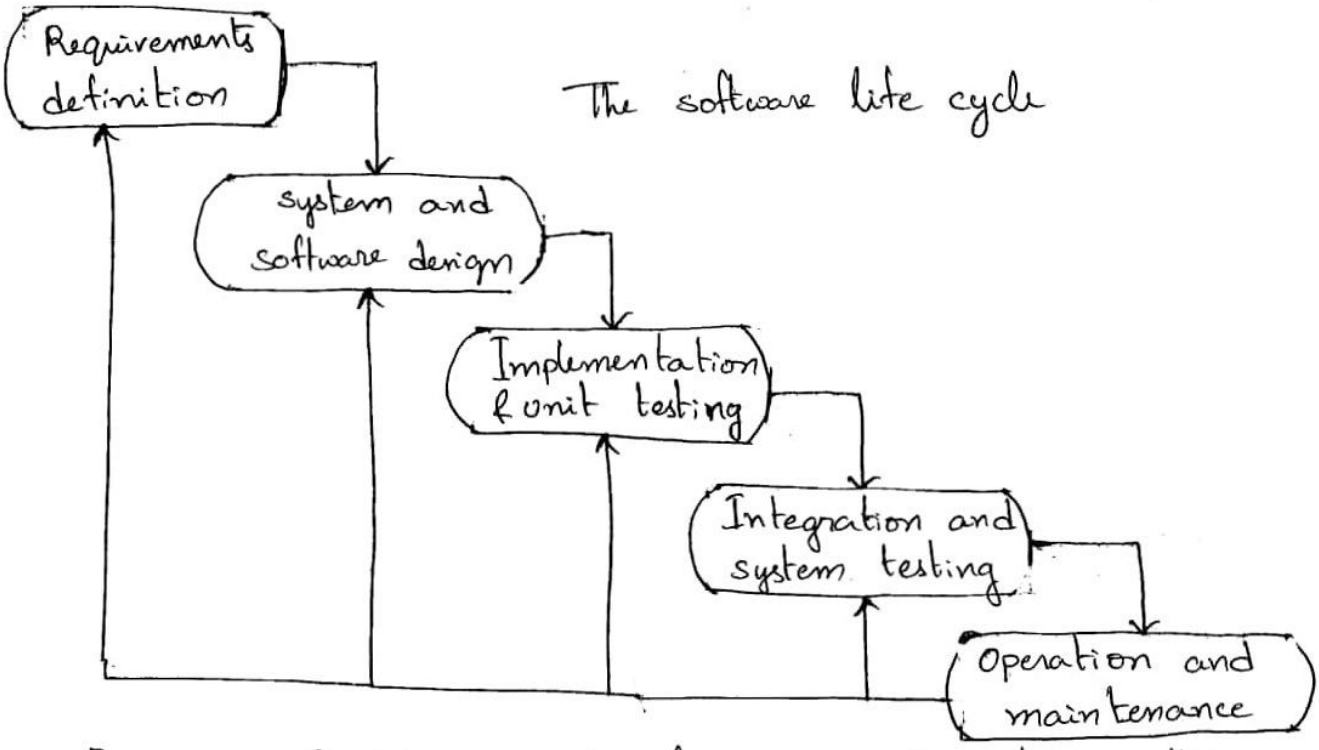
(3) Component-based software engineering: This approach is based on the existance of a significant number of reusable components. The system development process focuses on integrating these components into a system rather than developing them from scratch.

Eg: Formal development where a waterfall like process is used but the specification is a formal specification that is refined through several stages to an implementable design.

Scanned with CamScanner

The waterfall model:

→ The first published model of the software development process was derived from more general system engineering processes



The software life cycle

→ Because of the cascade from one phase to another, this model is known as the waterfall model or s/w life cycle.

Requirements analysis and definition: The system's services, constraints and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

System and software design: The systems design process partitions the requirements to either h/w or s/w systems.

→ It establishes an overall system architecture.

→ S/w design involves identifying and describing the fundamental software system abstractions & their relationships.

Implementation & Unit testing: In this stage, s/w design is ~~rela~~ realised as a set of programs or program units.

Unit testing involves verifying that each unit meets its specification.

Scanned with CamScanner

Integration and System testing: The individual program units or programs are integrated & tested as a complete system to ensure that the software requirements have been met. After testing, the s/w system is delivered to the customer.

Operation and Maintenance:

→ This is longest life cycle phase.

→ The system is installed & put into practical use.

→ Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system's services as new requirements are discovered.

→ The result of each phase is one or more documents that are approved.

→ The following phase will start as previous phase has finished.

→ In practise, these stages overlap & feed information to each other.

→ The s/w process is not a simple linear model but involves a sequence of iterations of the development activity.

→ Because of the costs of producing and approving documents iterations are costly & involve significant rework.
∴ after some iterations its normal to freeze parts of development. This premature freezing of requirement may mean that system won't do what the user wants.

→ During final life cycle phase, the s/w put into use errors & omissions in the original s/w requirements are covered.

→ System must therefore evolve to remain useful.
Making these changes may involve repeating previous process stages.

Scanned with CamScanner

→ Advantages of the waterfall model are documentation is produced at each phase & that it fits with other engineering process models.

→ Its major problem is its inflexible partitioning of the project into distinct stages.

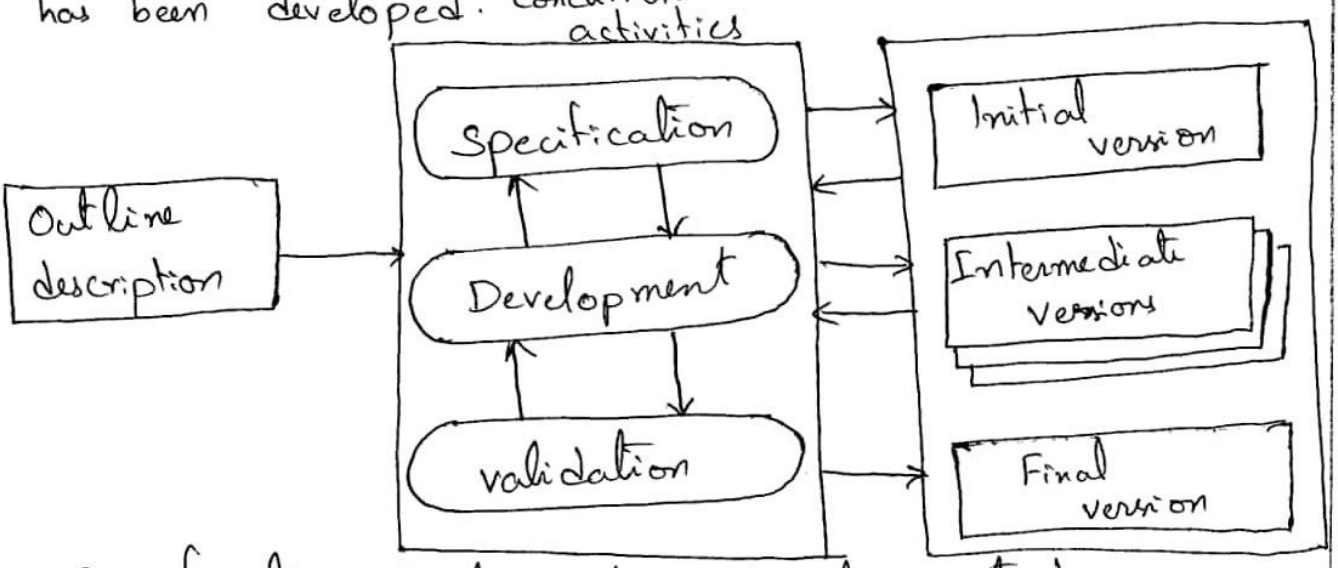
↳ Commitments must be made at an early stage in the process which makes it difficult to respond to changing user requirements.

→ Waterfall model should only be used when requirements are well understood & unlikely to change radically during system development.

→ This is used when the s/w project is part of a large system engineering project.

Incremental development

→ Incremental development is based on the idea of developing an initial implementation, exposing this to user comment & evolving it through several versions until an adequate system has been developed.



→ Specification, development & validation activities are interleaved rather than seperate, with rapid feedback across activities.

Scanned with CamScanner

→ Incremental s/w development, which is a fundamental part of agile approaches, is better than a waterfall approach for most business, e-commerce & personal systems.

→ Incremental development reflects the way that we solve problems.

→ We rarely work out a complete problem solution in advance but move toward a solution in a series of steps back tracking when we realize that we have made a mistake.

→ By developing the s/w incrementally, it is cheaper & easier to make changes in s/w as it is being developed.

→ Each increment or version of the system incorporates some of the functionality that is needed by the customer.

→ Generally, the early increments of the system include the most important or most urgently required functionality

→ This means that the customer can evaluate the system at a relatively early stage in the development to see if it delivers what is required.

→ If not, then only the current increment has to be changed & possibly, new functionality defined for later increments.

→ Incremental development has 3 benefits compared to waterfall

↳ The cost of accommodating changing customer requirements is reduced. The amount of analysis & documentation that has to be redone is much less than is required with the waterfall model.

Scanned with CamScanner

↳ It is easier to get customer feedback on the development work that has been done. Customers can comment on demonstrations of the s/w & see how much has been implemented. Customers find it difficult to judge progress from s/w design documents.

↳ More rapid delivery and development of useful s/w to the customer is possible, even if all of the functionality has not been included. Customers are able to use & gain value from the s/w earlier than is possible with a waterfall process.

→ Incremental development can be either plan driven, agile, or, more usually a mixture of these approaches.

→ In a plan-driven approach, the system increments are identified in advance.

→ If an agile approach is adopted, the early increments are identified but the development of later increments depends on progress & customer priorities.

In a management perspective, incremental approach has 2 problems.

↳ The process is not visible: Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost effective to produce documents that reflects every version of the system.

↳ System structure tends to degrade as new increments are added. Continual change tends to corrupt the s/w structure. Incorporating s/w changes becomes increasingly difficult and costly.

→ Small & medium size sys. incremental approach is best. For large sys. mixed process that incorporates the best features of waterfall & increment development models.

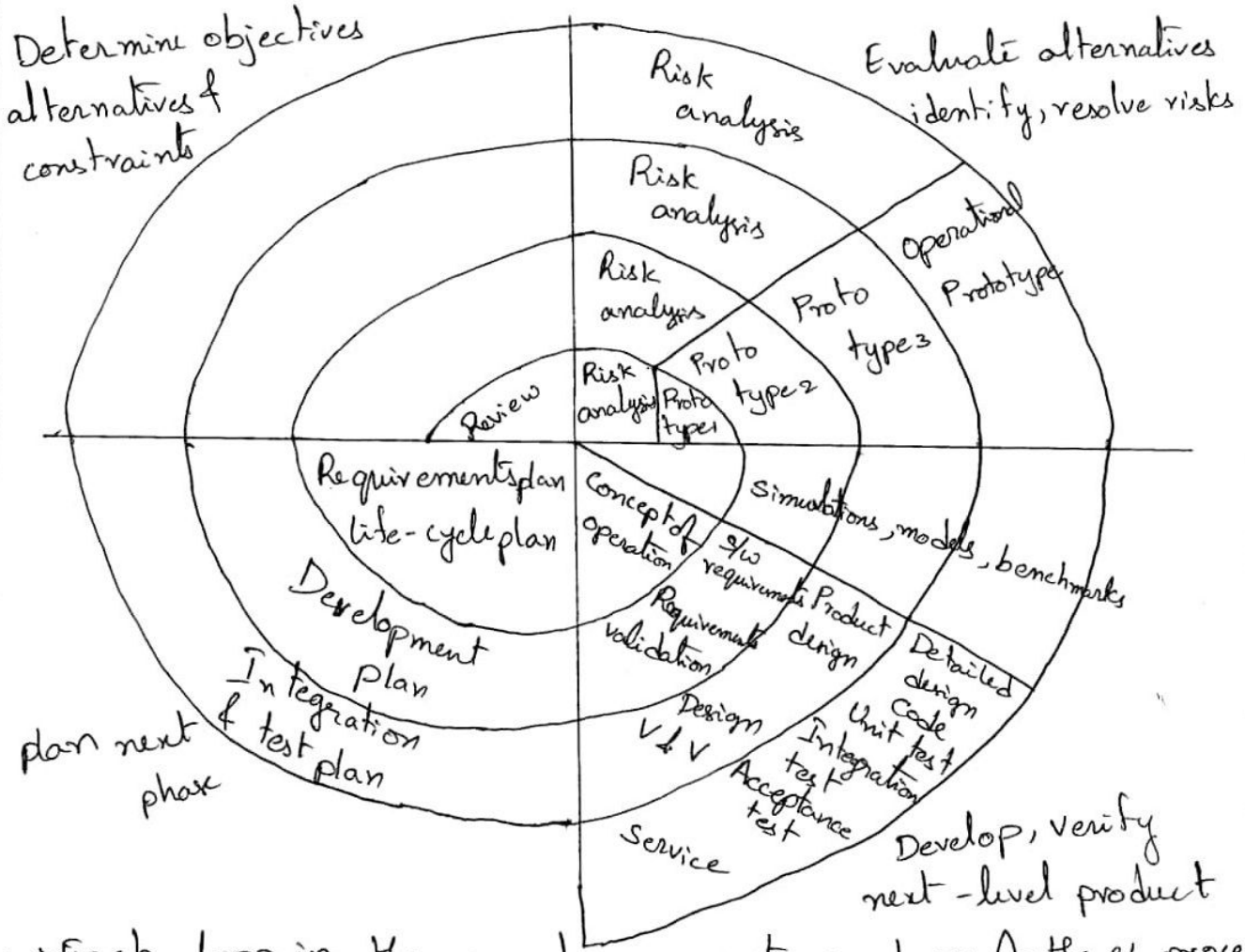
Scanned with CamScanner

Spiral Model

→ A risk-driven s/w process framework (the spiral model) was proposed by Boehm (1988).

→ The s/w process is represented as a spiral, rather than a sequence of activities with some backtracking from one activity to another.

Boehm's spiral model of the s/w process



→ Each loop in the spiral represents a phase of the s/w process.

→ The innermost loop might be concerned with system feasibility the next loop with requirements definition, the next loop with system design and, so on.

→ The spiral model combines change avoidance with change tolerance.

→ It assumes that changes are result of project risks and includes explicit risk management activities to reduce these risks.

Scanned with CamScanner

→ Each loop in the spiral is split into four sectors:

1. Objective setting:

→ Specific objectives for that phase of the project are defined

→ Constraints on the process & the product are identified & a detailed management plan is drawn up.

→ Project risks are identified. Alternative strategies, depending on these risks, may be planned.

2. Risk assessment and reduction:

→ For identified project risks, a detailed analysis is carried out.

→ steps are taken to reduce the risk.

Eg: if there is a risk that the requirements are inappropriate, a prototype system may be developed.

3. Development & validation:

→ After risk evaluation, a development model for the sys. is chosen.

Eg: If user interface risks are dominant, an appropriate development model might be evolutionary prototyping.

↳ If safety risks are main consideration, development based on formal transformations may be the most appropriate & so on.

↳ Waterfall model may be the most appropriate development model if the main identified risk is sub system integration.

4. Planning:

↳ The project is reviewed & a decision made whether to continue with a further loop of the spiral.

↳ If it is decided to continue, plans are drawn up for next phase of the project.

→ The difference b/w the spiral model & other s/w process models is the explicit recognition of risk in the spiral model.

→ Risk means something that can go wrong.

→ A cycle of spiral begines by elaborating objectives such as performance and functionality.

Scanned with CamScanner

→ Alternative ways of achieving these objectives & the constraints imposed on each of them are then enumerated.
→ Each alternative is assessed against each objective & sources of project risk are identified.
→ The next step is to resolve these risks by information gathering activities such as more detailed analysis, prototyping, and simulations
→ Once risks have been assessed, some development is caried out, followed by a planning activity for the next phase of the process.

Process Activities

→ Real s/w processes are interleaved sequences of technical, collaborative, and managerial activities with the overall goal of specifying, designing, implementing, and testing a s/w system.
→ The 4 basic process activities of specification, development, validation, & evolution are organized differently in different development processes.
→ In the waterfall model, they are organized in sequence, whereas in incremental development they are interleaved.
→ These activities are carried out depends on the type of s/w, people & organisational structures.
→ There is no right or wrong way to organise these activities.
Ex: In extreme programming, specifications are written on cards. Tests are executable & developed before the program itself
Evolution may involve substantial system restructuring or refactoring.

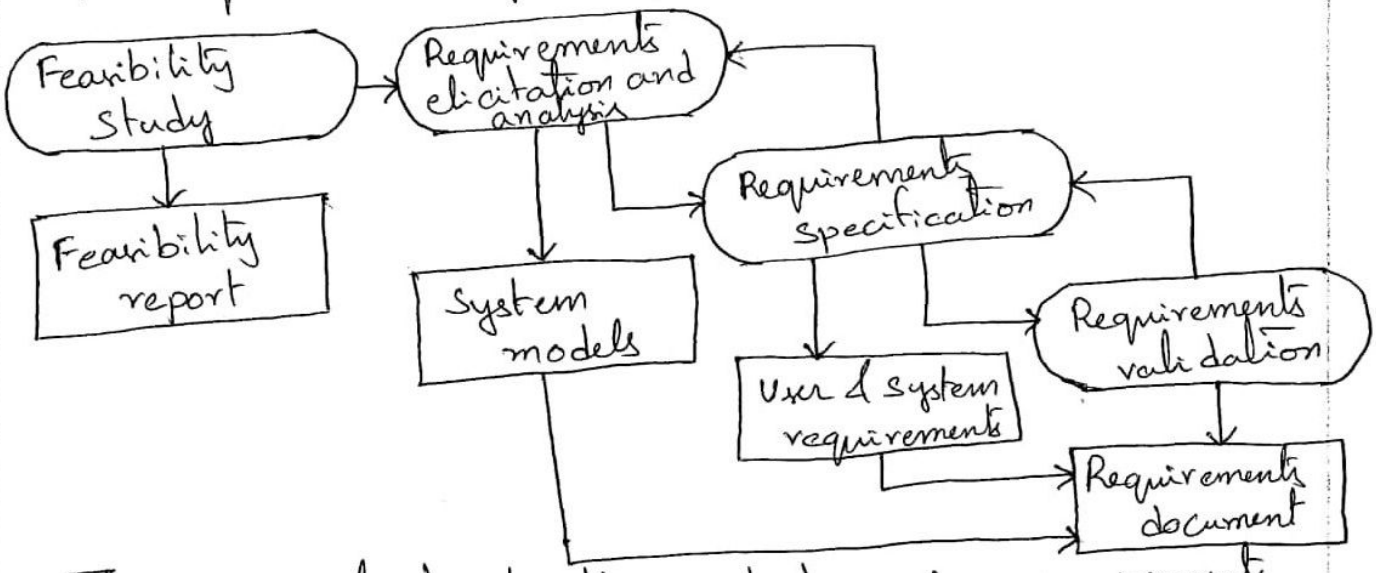
Scanned with CamScanner

Software Specification (or) Requirements engineering

→ It is the process of understanding & defining what services are required from the system & identifying the constraints on the system's operation & development.

→ It is a particularly critical stage of the s/w process as errors at this stage lead to later problems in the system design and implementation.

The requirements engineering process



→ This process leads to the production of a requirements document that is the specification for the system.

Requirements are presented at 2 levels of detail
- End users & customers need a high level statement of the requirements.
- System developers need a more detailed system specification.

There are 4 main activities in the requirements engineering process:

1. Feasibility study: An estimate is made of whether the identified user needs may be satisfied using current software & H/w technologies.
   - ↳ The study considers whether the proposed system will be cost-effective from a business point of view & also whether can be developed within the budget constraints.
   - ↳ It should be relatively cheap & quick. The result should inform the decision of whether to go ahead with more detailed analysis.

Scanned with CamScanner

2. <u>Requirements elicitation and analysis</u>: This is the process of deriving the system requirements through observation of existing systems, discussions with potential users & procurers, task analysis, & so on.

↳ This may involve the development of one or more system models & prototypes.

↳ These help the analyst understand the system to be specified.

3. <u>Requirements specification</u>: 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 documentation are

↳ User requirements – are abstract statements of the system requirements for the customer & end user of the system.

↳ system requirements – are more detailed description of the functionality to be provided.

4. <u>Requirements validation</u>: This activity checks the requirements for realism, consistancy, and completeness.

↳ During this process, errors in the requirements document are inevitably discovered.

↳ It must then be modified to correct these problems.

→ Requirements analysis continues during definition & specification & new requirements come to light throughout the process.

∴ The activities of analysis, definition & specification are interleaved.

Scanned with CamScanner

Software design and implementation

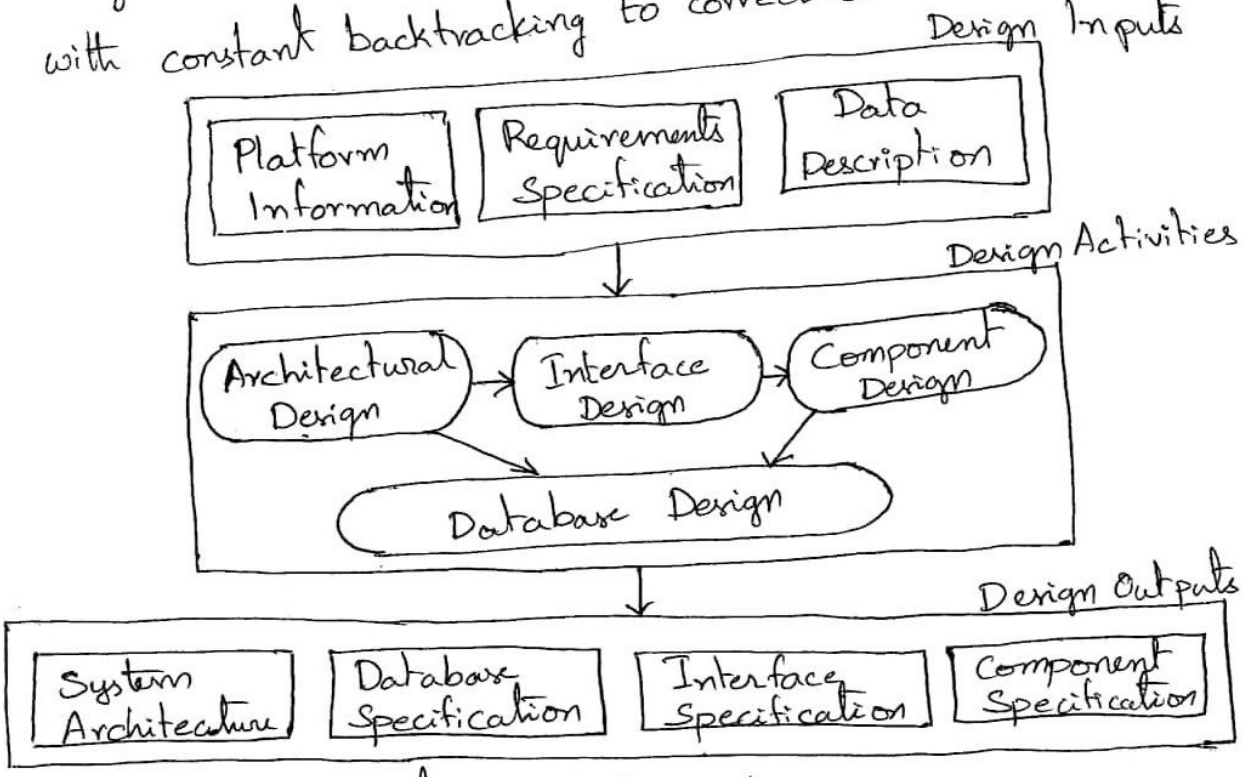
→ The implementation stage of s/w development is the process of converting a system specification into an executable system.

→ It involves processes of s/w design and programming

→ A software design is a description of the structure of the s/w to be implemented, the data models & structures used by the system, the interfaces b/w system components and, sometimes the algorithms used.

→ Designers do not arrive at a finished design immediately but develop the design iteratively.

→ They add formality and detail as they develop their design with constant backtracking to correct earlier designs.



A general model of the design process

→ It is an abstract model of this process showing the i/ps to design process, process activities, and the documents produced as o/ps from this process.

→ The diagram suggests that the stages of the design process are sequential. In practice design process activities are interleaved

Scanned with CamScanner

→ Feedback from one stage to another and consequent design rework is inevitable in all design processes.

→ Most s/w interfaces with other s/w systems. These include the operating system, database, middleware, & other application systems.

→ These makeup the 'S/w platform', the environment in which the s/w will execute.

→ Information about this platform is an essential i/p to the design process, as designers must decide how best to integrate it with the s/w's environment.

→ The requirements specification is a description of the functionality the s/w must provide and its performance & dependability requirements.

→ If the system is to process existing data, then the description of that data may be included in the platform specification;

→ otherwise, the data description must be an i/p to the design process so that the system data organization to be defined.

→ The activities in the design process vary, depending on the type of system being developed.

→ There are 4 activities that may be part of the design process for information systems:

1. Architectural design - where we identify the overall structure of the system, the principal components (subsystems or modules) their relationships, and how they are distributed.
2. Interface design - Where we define the interfaces b/w system components. This interface specification must be unambiguous.

Scanned with CamScanner

↳ With a precise interface, a component can be used without other components having to know how it is implemented. Once interface specifications are agreed the components can be designed and developed concurrently.

3. Component design - where we take each system component & design how it will operate. This may be a simple stmt of the expected functionality to be implemented, with the specific design left to the programmer. Alternatively, it may be a list of changes to be made to a reusable component or a detailed design model. The design model may be used to automatically generate an implementation.

4. Database design - Where we design the system data structures and how these are to be represented in a database. Again the work here depends on whether an existing database is to be reused or a new database is to be created.

→ These activities lead to a set of design outputs,

→ The detail & representation of these vary considerably.

↳ For critical systems, detailed design documents setting out precise and accurate descriptions of the system must be produced.

↳ If a model driven approach is used, these o/ps may mostly be diagrams.

↳ Where agile methods of development are used, the o/ps of the design process may not be seperate specification documents but may be represented in the code of the program.

→ Structured methods for design were developed in the 1970's & 1980's & were the precursor to the UML & object-oriented design.

→ They rely on producing graphical models of the system & in many cases, automatically generating code from these models.

Scanned with CamScanner

→ Model-driven development (MDD) or model-driven engg where models of the s/w are created at different levels of abstraction, is an evolution of structured methods.

→ In MDD, there is greater emphasis on architectural models with a separation b/w abstract implementation independent models & implementation-specific models.

→ The models are developed in sufficient detail so that the executable system can be generated from them.

→ The development of a program to implement the system follows from the system design processes.

→ S/w development tools may be used to generate a skeleton pgm from a design.

→ Programming is a personal activity & there is no general process

→ that is some programmers start with components that they understand, others take the opposite approach i.e, starts with which they didn't understand.

→ Programmers carry out some testing of the code they have developed. This often reveals program defects that must be removed from program. This is called debugging.

→ Defect testing & debugging are different processes.

→ Testing establishes the existence of defects.

→ Debugging is concerned with locating & correcting these defects.

Scanned with CamScanner

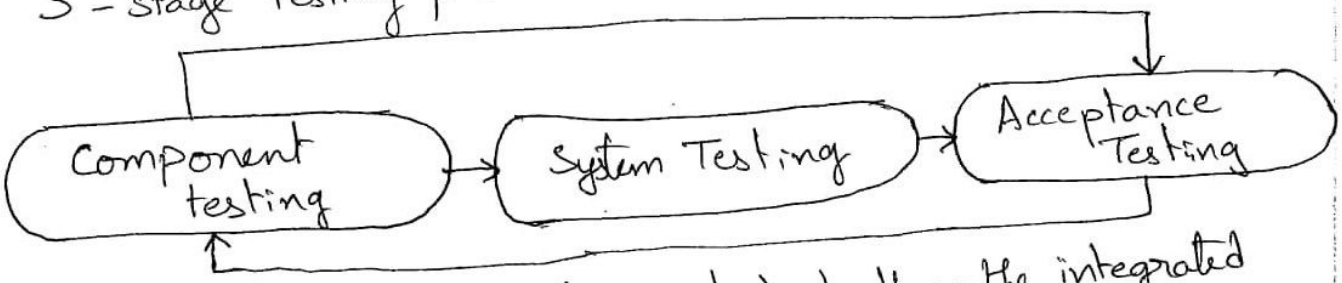
Software Validation or Verification and Validation (V&V)

→ It is intended to show that a system conforms to its specification & the system meets the expectations of the customer.

→ It involves checking processes, such as inspections & reviews, at each stage of the S/w process from user requirements definition to program development.

→ Except for small programs, systems should not be tested as a single.

3 - stage testing process



→ where system components are tested, then the integrated system is tested and finally, the system is tested with the customer's data.

→ The process is an iterative one with information being fed back from later stages to earlier parts of the process.

→ The stages in testing process are

1. Development testing: The components making up the system are tested by the people developing the system.

↳ Each component is tested independently, without other components.

↳ Components may be simple entities such as functions or object classes, or may be groupings of these entities.

↳ Test automation tools, such as JUnit, that can rerun component tests when new versions of the component are created, are commonly used.

Scanned with CamScanner

2. System testing: System components are integrated to create a complete system.
↳ This process is concerned to finding errors that result from unanticipated interactions b/w components & components interface problems.
↳ It is also concerned with validating that the sys. meets its functional & non functional requirements & testing the system emergent properties.

3. Acceptance testing: This is final stage in the testing process before the system is accepted for operational use.
↳ The system is tested with data supplied by the sys. customer rather than with simulated test data.
↳ Acceptance test may reveal errors & omissions in the system requirements definition because the real data exercise the system in different ways from the test data.
↳ It may also reveal requirements problems where the system's facilities do not really meet the user's needs or the system performance is unacceptable.

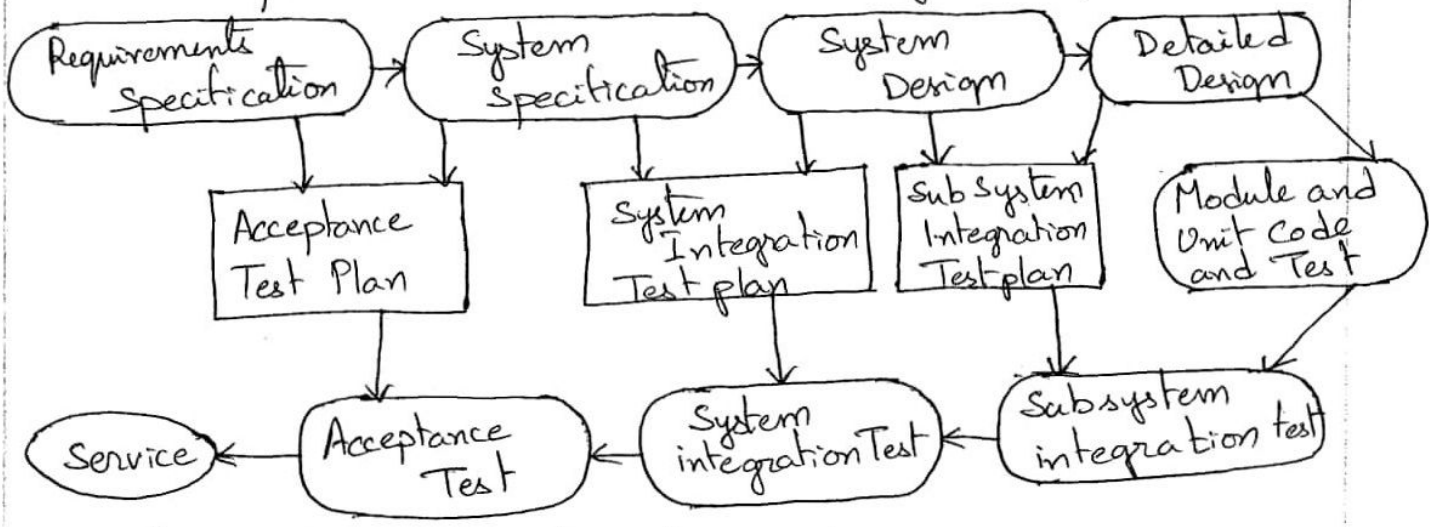
→ Component development & testing are interleaved.
→ Programmers makeup their own test data & incrementally test the code as it is developed.
→ If an incremental approach to development is used, each increment should be tested as it is developed, with these tests based on the requirements for that increment.
→ In extreme programming tests are developed along with the requirements before development starts. It helps testers & developers.

Scanned with CamScanner

→ Later stages of testing involve integrating work from a no. of programmers & must be planned in advance.

→ An independent team of testers should work from preformulated test plans that are developed from the system specification & design.

The test plans are the link b/w testing & development activities.



Testing phases in a plan-driven s/w process

→ Acceptance testing is sometimes called alpha testing. custom systems are developed for single client.

→ The alpha testing process continues until the system developer & the client agree that the delivered system is an acceptable implementation of the system requirements.

→ When a system is to be marketed as a s/w product, a testing process called beta testing is often used.

→ Beta testing involves delivering a system to a no. of potential customers who agree to use that system.

→ They report problems to the system developers.

→ This exposes the product to real use and detects errors that may not have been anticipated by system builders. After this feed back, the system is modified & released either for further beta testing or for general sale.

Scanned with CamScanner

Software Evolution

→ The flexibility of software systems is one of the main reasons why more & more s/w is being incorporated in large, complex systems.

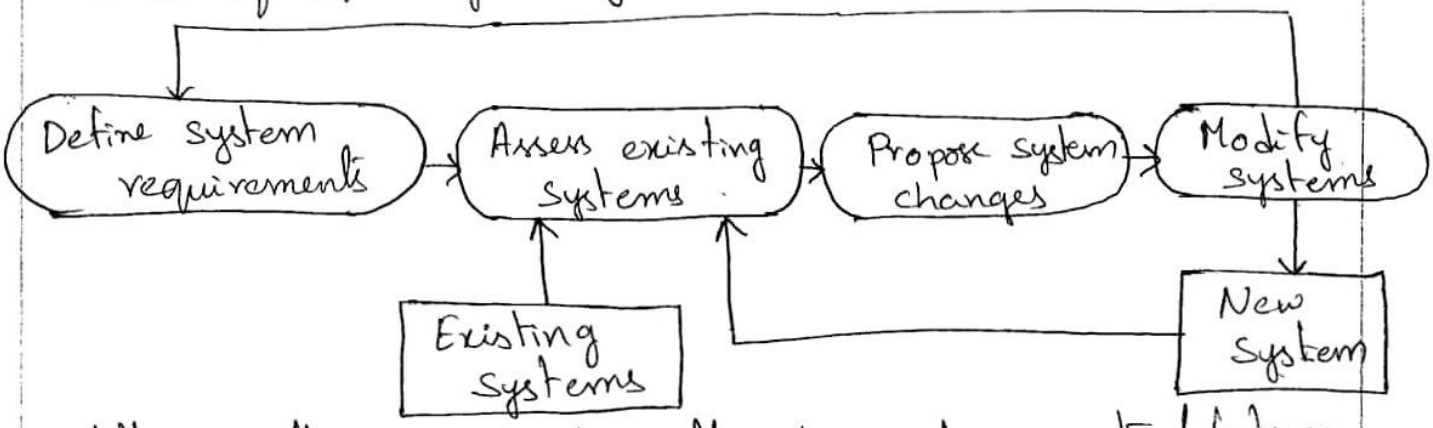
→ H/w is very expensive to make changes.

→ Changes can be made to s/w at any time during or after the system development.

→ Even extensive changes are still much cheaper than corresponding changes to system hardware.

→ There has a split between the process of s/w development & the process of s/w evolution (s/w maintenance)

→ Rather than two seperate processes, it is more realistic to think of s/w engineering as an evolutionary process.



→ Where software is continually changed over its lifetime in response to changing requirements and customer needs.

Scanned with CamScanner

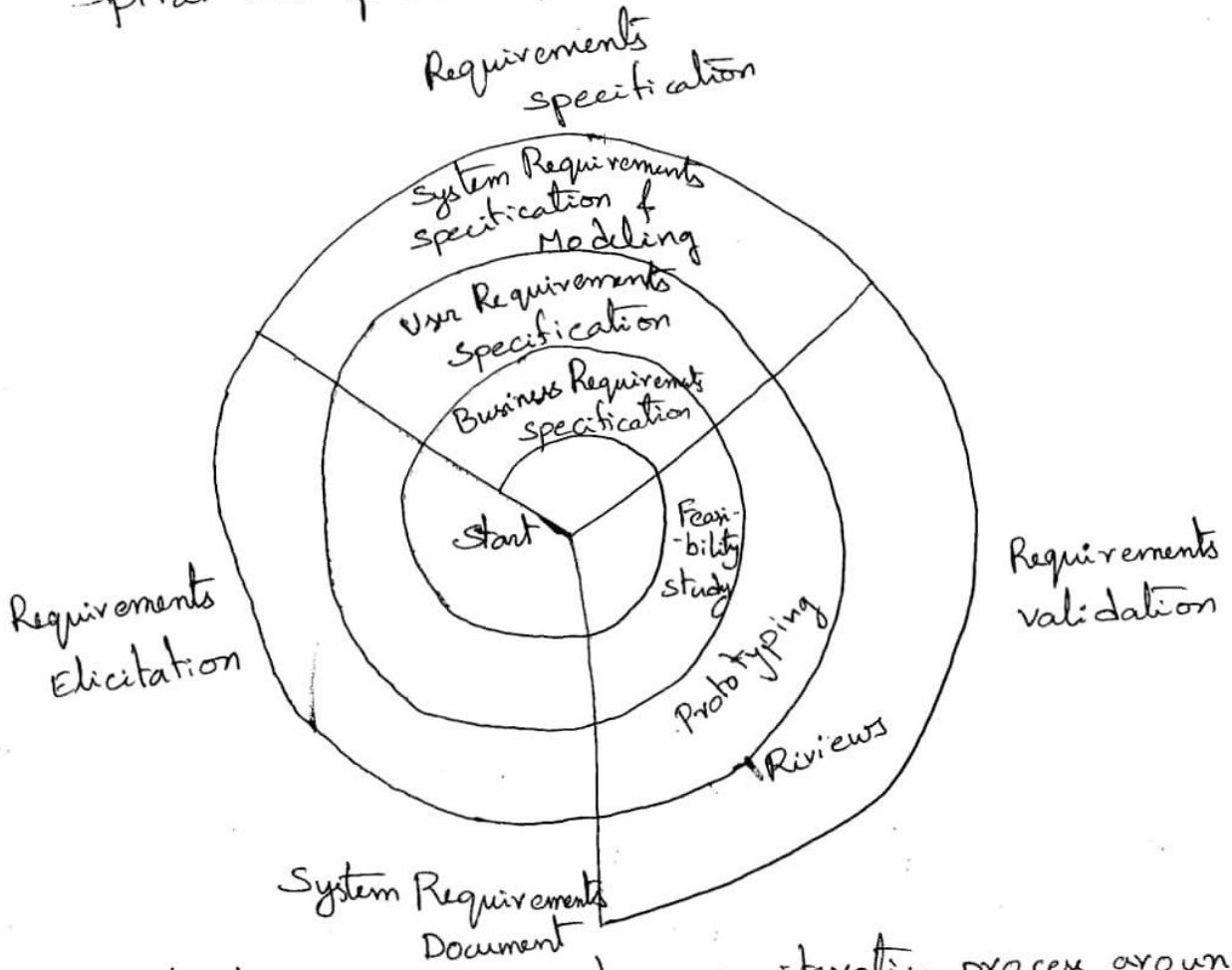
Requirements engineering processes

→ Requirements engineering process may include 4 high level activities

→ These focus on assessing if the system is useful to the business (feasibility study), discovering requirements (elicitation and analysis), converting these requirements into some standard form (specification), and checking that the requirements actually define the system that the customer wants (validation).

→ In practice, requirements engineering is an iterative process in which the activities are interleaved.

Spiral view of the requirements engineering process



→ The activities are organized as an iterative process around a spiral, with the o/p being a system requirements document.

Scanned with CamScanner

⇒ The amount of time & effort devoted to each activity in each iteration depends on the stage of the overall process & the type of system being developed.

→ This model accommodates approaches to development in which the requirements are developed to different levels of detail.

→ The no. of iterations around the spiral can vary.

→ This model allows the requirements & system implementation to be developed together.

→ Some people consider requirements engineering to be the process of applying a structured analysis method such as object oriented analysis.

→ This involves analysis the system & developing a set of graphical system models, such as use-case models, that than serve as a system specification.

Requirements elicitation and analysis

→ After an initial feasibility study, the next stage of the requirements engineering process is requirements elicitation & analysis.

→ In this activity, s/w engineers work with customers & sys. end users to find out application domain, services the system should provide, required performance of the system, H/w constraints and so on.

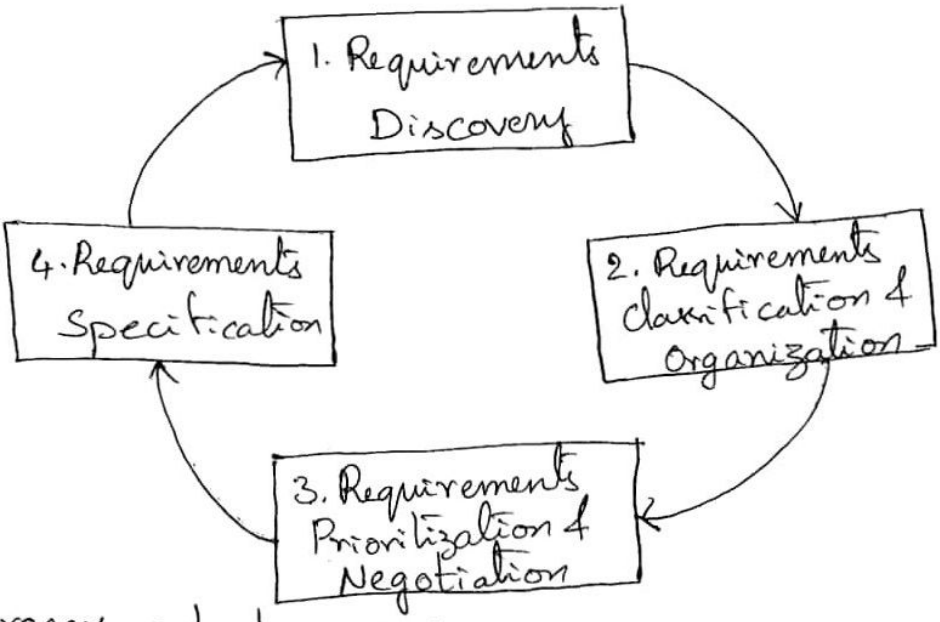
→ A system stakeholder is anyone who should have some direct or indirect influence on the system requirements & who will interact & affected by the system.

→ End users, managers, engineers involved in maintenance, domain experts, trade unions are called stakeholders.

→ Each organization will have its own version or instantiation of this general model depending on local factors such as the expertise of the staff, type of system being developed, standard use etc.

Scanned with CamScanner

A process model of requirements elicitation & analysis



The process activities are:

<u>Requirements discovery</u>: This is the process of interacting with stakeholders in the system to collect their requirements.

↳ Domain requirements from stakeholders & documentation are also discovered during this activity.

<u>Requirements classification & Organization</u>: This activity takes the unstructured collection of requirements, groups related requirements & organises them into coherent clusters.

<u>Requirements prioritization & negotiation</u>: Where multiple stakeholders are involved, requirements will conflict.

↳ This activity is concerned with prioritising requirements, & finding & resolving requirements conflicts through negotiation.

<u>Requirements specification</u>: The requirements are documented & input into the next round of the spiral.

↳ Formal or informal requirements documents may be provided.

→ The requirements elicitation & analysis is an iterative process with continual feedback from each activity to other activities.

→ The process cycle starts with requirements discovery & ends with requirements documentation.

Scanned with CamScanner

→ The analyst's understanding of the requirements improves with each round of the cycle. The cycle ends when the requirements document is complete.

Eliciting & understanding stakeholders requirements is difficult due to:

↳ Stakeholders don't know what they really want from computer sys.

↳ Stakeholders express requirements in their own terms.

↳ Different stakeholders have different requirements, which they may express in different ways.

↳ Organisational & political factors may influence the sys. requirements

↳ The requirements change during the analysis process. New stakeholders may emerge & the business environment change.

→ In the requirements specification stage, the requirements that have been elicited are documented in such a way that they can be used to help with further requirements discovery.

→ At this stage, an early version of the system requirements document may be provided, but it will have missing sections & incomplete requirements.

→ The requirements may be documented as tables in a document or on cards.

→ Writing requirements on cards is very effective, as these are easy for stakeholders to handle, change and organise.

Requirements discovery

→ It is the process of gathering information about the required system & existing systems, & distilling the user and system requirements from this information.

→ Sources of information during the requirements discovery phase include documentation, system stakeholders & specifications of similar systems.

Scanned with CamScanner

→ One may interact with stakeholders through interviews & observations & may use senarios & prototypes to help with the requirements discovery.

→ Stakeholders range from system end-users through managers & external stakeholders such as regulators who certify the acceptability of the system.

Ex: System stakeholders for the mental health care patient info sys include:

Patients: whose info is recorded in the system.

Doctors: who are responsible for assessing and treating patients.

Nurses: who coordinates the consultations with doctors & administer some treatments.

Medical receptionists: who manage patients' appointments.

IT staff: who are responsible for installing & maintaining the system.

A medical ethics manager: who must ensure that the system meets current ethical guidelines for patient care.

Healthcare managers: who obtain management info from sys.

Medical records staff: who are responsible for ensuring that system info can be maintained & preserved, and that record keeping procedures have been properly implemented.

→ Different requirements sources like stakeholders, domain, systems can all be represented as system viewpoints with each viewpoint showing a subset of the requirements for the system.

→ Different viewpoints on a problem see the problem in different ways.

Scanned with CamScanner

Interviewing

→ Formal or informal interviews with system stakeholders are part of most requirements engineering process.

→ The requirements engineering team puts questions to stakeholders about the system that they use & the system to be developed. Requirements are derived from the answers to these questions.

Interviews may be 2 types:

↳ Closed interviews: Where the stakeholders answers a predefined set of questions.

↳ Open interviews: Where there is no predefined agenda.

→ The requirements engineering team explores a range of issues with system stakeholders & hence develops a better understanding of their needs.

→ Interviews with stakeholders are normally a mix of these.

→ The answers to some questions may lead to other issues.

→ Completly open ended discussions rarely work well.

→ Interviews are good for getting an overall understanding of what stakeholders do like.

↳ how they might interact with the system.

↳ difficulties they face with current systems.

→ However Interviews are not good for understanding the requirements from the application domain.

→ It is hard to elicit domain knowledge during interviews for 2 reasons:

↳ All application specialist use teminology & jargon that is specific to a domain.

↳ Some domain knowledge is so familiar to stakeholders that they either find it difficult to explain or they think it is so fundamental that it isn't worth mentioning.

Scanned with CamScanner

→ Interviews are not an effective technique for eliciting knowledge about organisational requirements & constraints.

→ Effective interviews have two charateristics:

↳ They are open-minded, avoid preconceived ideas about the requirements & are willing to listen to stakeholders.

↳ They prompt the interviewee to start discussions with a question, a requirements proposal or by suggesting working together on a prototype system.

→ Information from interviews supplements other information about the system from documents, user observations & so on.

→ Some times, apart from information from documents, interviews may be the only source of information about the system requirements.

Scenarios

→ Scenarios are real life examples of how a system can be used.

→ Scenarios can be particularly useful for adding detail to an outline requirements description.

→ The scenarios starts with an outline of interaction, & during elicitation details are added to create a complete description.

General Scenario may include:

1. A description of what the system & user expect when scenario starts
2. A description of the normal flow of events in the scenario.
3. A description of what can go wrong & how this is handled.
4. Information about other activities that might be going on at the same time.
5. A description of the system state when the scenario finishes.

Scanned with CamScanner

→ Scenario-based elicitation can be carried out informally where the requirements engineering works with stakeholders to identify scenarios & to capture details of these scenarios.

→ Scenarios may be written as text, supplemented by diagrams, screen shots, & so on. Alternatively usecases may be adopted.

Ex: Scenario for collecting medical history in MHC-PMS

Initial Assumption: The patient has seen a medical receptionist who has created a record in the system and collected the patient's personal info. (name, add, age etc). A nurse is logged on to the system & is collecting medical history.

Normal: The nurse searches for the patient by family name. If there is more than one patient with the same surname, the given name and date of birth are used to identify the patient.

↳ The nurse chooses the menu option to add medical history.

↳ The nurse then follows a series of prompts from the system to enter info. about consultations elsewhere on mental health problems, existing medical conditions, medication currently taken, allergies, & home life.

What can go wrong: The patient's record does not exist or cannot be found. The nurse should create a new record and record personal info.

↳ Patient conditions or medication are not entered in the menu. The nurse should choose the 'other' option & enter free text describing the condition / medication.

Patient cannot/will not provide information on medical history. The nurse should enter free text recording the patient's inability / unwillingness to provide information. The system should print the standard exclusion form starting that the lack of information may mean that treatment will be limited or delayed. This should be signed and handed to the patient.

Other activities: Record may be consulted but not edited by other staff while information is being entered.

Scanned with CamScanner

System state on completion: User is logged on. The patient record including medical history is entered in the database, a record is added to the system log showing the start & end time of the session and the nurse involved.

Use Cases

→ Use cases are a requirements discovery technique that were first introduced in the objectory method

→ They have now become a fundamental feature of Unified Modeling Language (UML)

→ A Use-case identifies the actors involved in an interaction and names the type of interaction.

→ This is then supplemented by additional information describing the interaction with the system.

→ The additional info. may be a textual description or one or more graphical models such as UML sequence or state charts.

→ Use cases are documented using a high level use case diagram.

→ The set of use cases represents all of the possible interactions that will be described in the system requirements.

→ Actors in the process, who may be human or other systems are represented as stick figures.

→ Each class of interaction is represented as a named ellipse.

→ Lines link the actors with the interaction.

→ optionally, arrowheads may be added to lines to show how the interaction is initiated.

In the next page shown some of the use cases for the patient information system.

Scanned with CamScanner

→ There is no hard and fast distinction b/w scenarios & use cases.

→ Some consider that each usecase is a single senario; others an encapsulate a set of senarios in a single usecase.

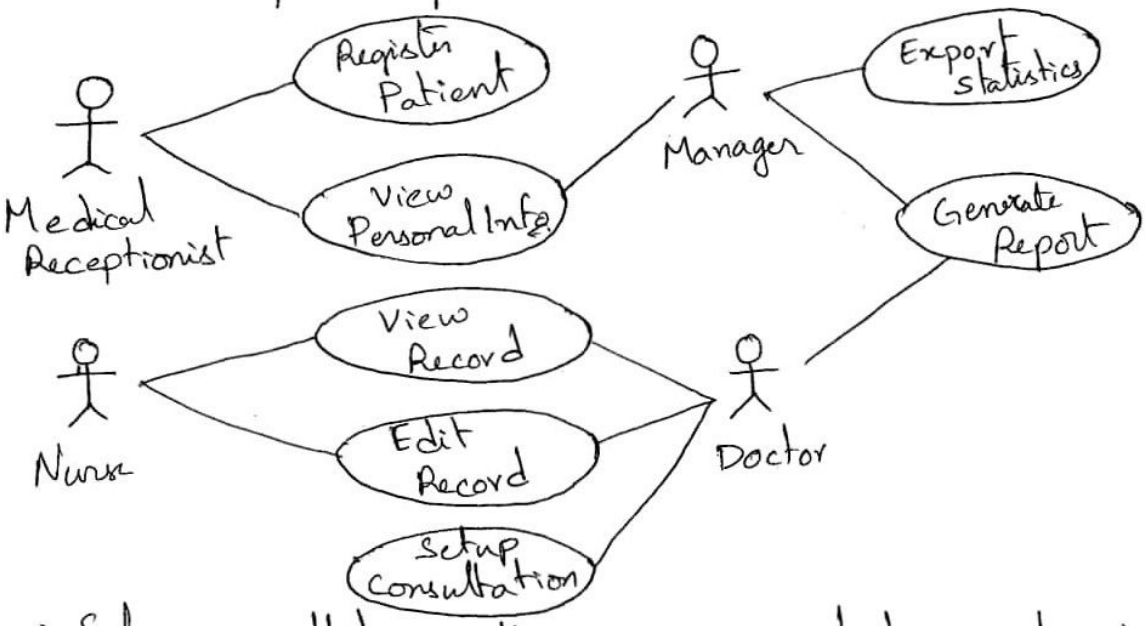
→ Each senario is a single thread through the usecase.

→ ∴ there should be a senario for the normal interaction + senarios for each possible exception.

→ Use cases identify the individual interactions b/w sys. & its users or other systems.

→ Each use case should be documented with a textual description.

Ex: Description of setup consultation usecase from fig.



→ Setup consultation allows 2 or more doctors, working in different offices, to view the same record at the same time. One doctor initiates the consultation by choosing the people involved from a menu of doctors who are online. The patient record is then displayed on their screens but only the initiating doctor can edit the record.

→ Secenorios & use cases are effective for eliciting requirements from stakeholders who interact directly with system. Each type of interaction can represented as use case.

→ The UML is de facto standard for OO Modeling, so use cases & use case based elicitation are now widely used for reqmts elicitation.

Scanned with CamScanner

Ethnography: (Observation)

→ Software systems do not exist in isolation - they are used in a social & organisational context, & s/w system requirements may be derived or constrained by that context.

→ Satisfying these social and organisational requirements is often critical for the success of the system.

→ One reason why many s/w systems are derived but never used is that they do not take proper account of the importance of these requirements.

→ Ethnography: is an observational technique that can be used to understand social & organisational requirements.

→ Analyst observes the day to day work & notes made of the actual tasks in which participants are involved.

→ The value of ethnography is that it helps analysts discover implicit system requirements that reflect the actual rather than the formal processes in which people are involved.

→ People often find it very difficult to articulate details of their work.

→ They understand their own work but may not understand its relationship with other work in the organisation.

→ Social & organisational factors that affect the work but that are not obvious to individuals may only become clear when noticed by an unbiased observer.

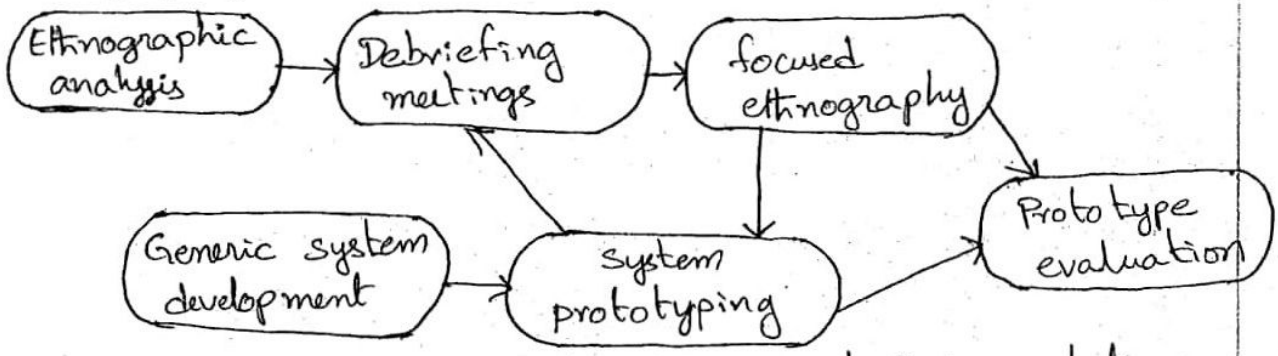
Ethnography is particularly effective at discovering 2 types of requirements:

1. Requirements that are derived from the way in which people actually work rather then the way in which process definitions say they ought to work.

Scanned with CamScanner

2. Requirements that are derived from cooperation & awareness of other people's activities.

→ Ethnography may be combined with prototyping.



→ The ethnography informs the development of the prototype so that fewer prototype refinement cycles are required.

→ The prototyping focuses the ethnography by identifying problems & questions that can then be discussed with the ethnographer.

→ Then ethnographer should look for the answers to these questions during the next phase of the system study.

→ Ethnographic studies can reveal critical process details that are often missed by other requirements elicitation techniques.

→ As its focus on the end-user, this approach is not appropriate for discovering organisational or domain requirements.

→ Ethnographic studies cannot always identify new features that should be added to a system.

→ Ethnography is not, therefore, a complete approach to elicitation on its own, & it should be used to complement other approaches, such as usecase analysis.

Scanned with CamScanner

Functional and Non-functional requirements

→ Software system requirements are often classified as functional, or non-functional requirements:

Functional requirements: These are statements of services the system should provide, how the system should react to particular i/ps, and how the system should behave in particular situations.

↳ The functional requirements may also explicitly state what the system should not do.

Non-functional requirements: These are constraints on the services or functions offered by the system.

↳ They include timing constraints, constraints on the development process and standards.

↳ Non-functional requirements often apply to the system as whole.

↳ They do not usually just apply to individual system features or services.

Functional requirements

→ The functional requirements for a system describe what the system should do.

→ These requirements depend on the type of s/w being developed, the expected users of the s/w and the general approach taken by the organisation when writing requirements.

→ Functional user requirements are described in an abstract way that can be understood by system users.

→ Functional system requirements are describe the system functions, its i/ps & o/ps, exceptions, etc. in detail.

Scanned with CamScanner

Ex: Functional requirements for the MHC-PMS, used to maintain information about patients receiving treatment for mental health problems:

1. A user shall be able to search the appointments lists for all clinics.
2. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
3. Each staff member using the system shall be uniquely identified by his or her eight-digit employee number.

→ These functional user requirements define specific facilities to be provided by the system.

→ These have been taken from the user requirements document and they show that functional requirements may be written at different levels of detail.

→ The functional requirements specification of a system should be both complete and consistent.

→ Completeness means that all services required by the user should be defined.

→ Consistency means that requirements should not have contradictory definitions.

→ For large, complex systems, it is practically impossible to achive requirements consistency and completeness.

→ Becoz, it is easy to make mistakes & omissions when writing specifications for large, complex systems and different system stakeholders have different and often inconsistent needs in large, complex systems.

Scanned with CamScanner

Non-functional requirements

→ Non-functional requirements are requirements that are not directly concerned with the specific functions delivered by the system to its users.

→ They may relate to emergent system properties such as reliability, response time and store occupancy.

→ Non-functional requirements, such as performance, security, or availability, usually specify or constrain characteristics of the system as whole.

→ These are more critical than individual functional requirements.

→ Failing to meet a non functional requirement can mean that whole system is unusable.

Ex: Aircraft system does not meet its reliability requirements it will not be certified as safe for operation.

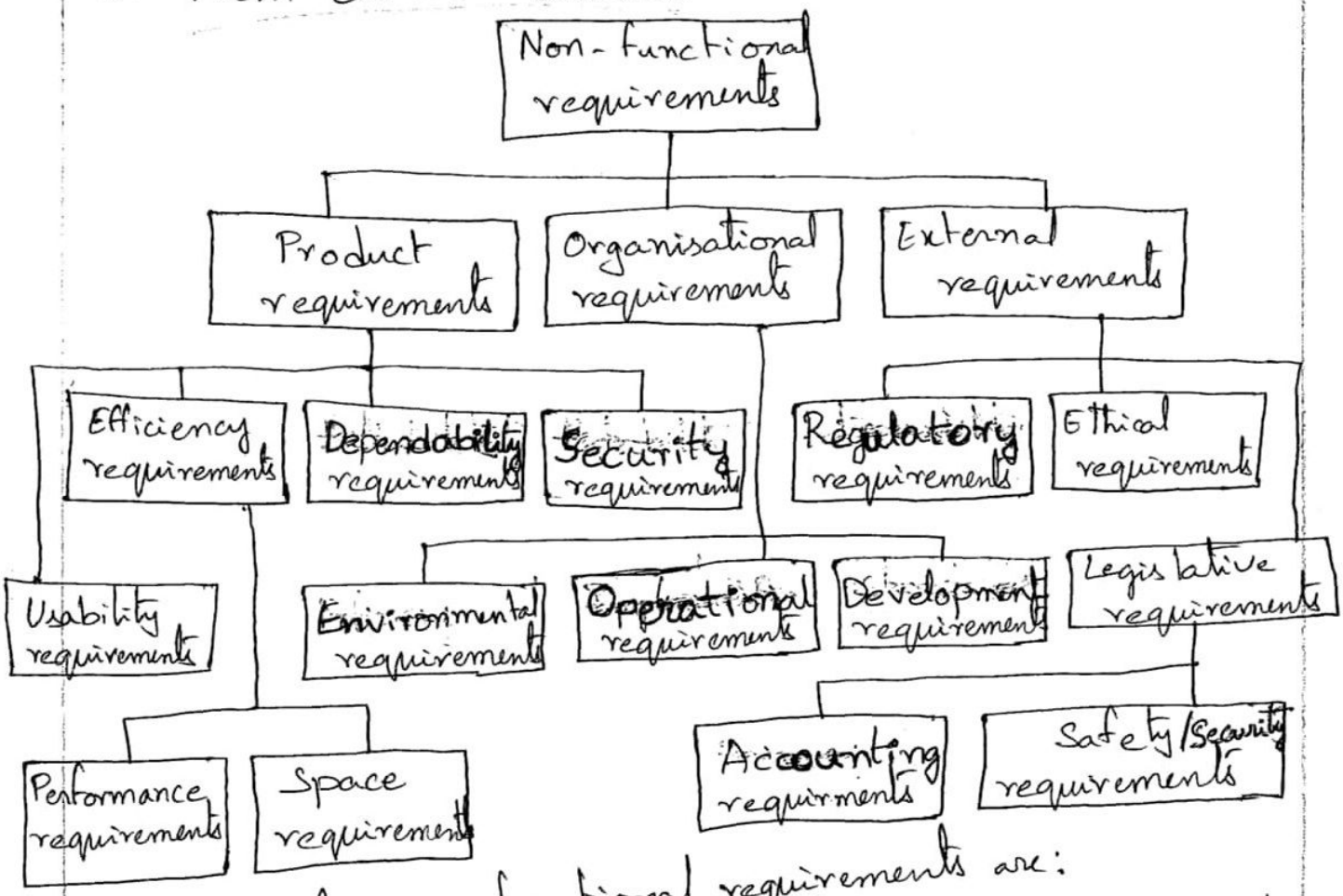
→ The implementation of these requirements may be diffused throughout the system. There are 2 reasons for this

1. Non functional requirements may affect the overall architecture of a system rather than the individual components.
2. A single non-functional requirement, such as a security requirement, may generate a no. of related functional requirements that define new system services that are required. In addition it may also generate requirements that restrict existing requirements.

→ Non functional requirements arise through user needs becoz of budget constraint, organizational policies, the need for interoperability with other s/w or H/w systems, External factors such as safety regulations or privacy legislation.

Scanned with CamScanner

→ classification of non functional requirements into requirement characteristics of the s/w (product requirements) the organization developing the s/w (Organizational requirements) or from external sources.



The types of non functional requirements are:

Product requirements: These requirements specify s/w product behaviour.

Ex: performance requirements: how fast system must execute & how much memory requires - space

reliability requirements that set out the acceptable failure rate.

Security requirements & usability requirements.

Organizational requirements: These requirements are derived from policies & procedures in the customer's & developers organization.

Ex: operational process requirements that define how the system will be used.

development process requirements that specify the programming language the development environment or process standards to be used, &

environmental requirements that specify the operating environment of sys.

Scanned with CamScanner

External requirements: This broad heading covers all requirements that are derived from factors external to the system & its development process

Ex: Regularity requirements that set out what must be done for the system to be approved for use by a regulator,

↳ Legislative requirements that must be follwed to ensure that the system operates within the law;

↳ Ethical requirements that ensure that the system will be acceptable to its users & the general public.

Ex. of non-functional requirements in MHC-PMS

Product Requirement

The MHC-PMS shall be available to all clinics during normal working hours (Mon-Fri 8:30-17:30). Downtime with in normal working hours shall not exceed five seconds in any one day.

Organizational Requirement

Users of the MHC-PMS system shall authenticate themselves using their health authority identity card.

External Requirement

The system shall implement patient privacy provisions as set out in Hstan-03-2006-priv.

→ A common problem with non-functional requirements is that users or customers often propose these requirements as general goals such as ease of use, the ability of the system to recover from failure or rapid user response.

→ should write non-functional requirements quantitatively so that they can be objectively tested.

Scanned with CamScanner

Metrics for specifying non-functional requirements

Property	Measure
Speed	Processed transactions/second, User/event response time, screen refresh time
Size	Mbytes, Number of ROM chips
Ease of use	Training time, Number of help frames
Reliability	Mean time to failure, Probability of unavailability, Rate of failure occurrence, Availability.
Robustness	Time to restart after failure, Percentage of events causing failure, Probability of data corruption on failure
Portability	Percentage of target dependent statements, Number of target systems.

→ Non functional requirements often conflict & interact with other functional or non-functional requirements.

→ It is difficult to seperate functional & non-functional requirements in the requirements document.

→ If the non-functional requirements are stated separately from the functional requirements, the relationships between them may be hard to understand.

The Software requirements document

→ It is also called as software requirements specification or SRS.

→ S/w requirements document is the official statement of what the system developers should implement.

→ It should include both the user requirements for a system & a detailed specification of the system requirements.

→ Sometimes, the user & system requirements are integrated into a single description.

Scanned with CamScanner

→ In other cases, the user requirements are defined in an introduction to the system requirements specification.

→ If there are a large no. of requirements, the detailed sys. requirements may be presented in a seperate document.

→ Requirements documents are essential when an outside contractor is developing the s/w system.

→ The requirements document has a diverse set of users, ranging from the senior management of the organization that is paying for the system to the engineers responsible for developing the s/w.

System Customers	Specify the requirements & read them to check that they meet their needs. Customers specify changes to the requirements.
Managers	Use the requirements document to plan a bid for the system & to plan the system development process
System Engineer	Use the requirements to understand what system is to be developed.
System Test Engineer	Use the requirements to develop validation tests for the system.
System Maintenance Engineer	Use the requirements to understand the sys. & relationships between its parts

→ One possible organization for requirements document that is based on an IEEE standard for requirements documents.

→ The info. included in a requirements document depends on type of s/w being developed & approach to development that is to be used.

Scanned with CamScanner

Structure of requirements document

chapter	Description

Preface: This should define the expected readership of the document & describe its version history, including a rationale for the creation of a new version & a summery of the changes made in each version.

Introduction: This should describe the need for the system. it should briefly describe its functions & explain how it will work with other sys. It should describe how the sys. fits into the overall business or strategic objectives of the organization commissioning the s/w.

Glossary: This should define the technical terms used in the document. We should not make assumptions about experience or expertise of reader.

User requirements definition: The services provided for the user & the non-functional sys. requirements should be described in this section. This description may use natural language, diagrams or other notations that are understandable by customers. Product & Process standards which must be followed should be specified.

System architecture: It present a high level overview of the anticipated system architecture showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.

System requirements specification: This describe the functional & non functional requirements in more detail. If needed, further detail may also be added to the non functional requirements, eg: interfaces to other sys. may be defined.

System models: This should set out one or more sys. models showing the relationships b/w the system components & the system & its environment. These might be object models, dataflow models & semantic data models.

System evolution: This should describe the fundamental assumptions on which the system is based & anticipated changes due to h/w evolution, changing user needs etc.

Appendices: These should provide detailed, specific information which is related to the application which is being developed.
Ex. It may included are h/w & database descriptions.

Index: Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, etc.

Scanned with CamScanner

Requirements specification

→ Requirements specification is the process of writing down the user and system requirements in a requirements document.

→ The user and system requirements should be clear, unambiguous easy to understand, complete, & consistent.

→ In practice, this is difficult to achieve as stakeholders interpret the requirements in different ways.

→ The user requirements for a system should describe the functional and non functional requirements so that they are understandable by system users who don't have detailed technical knowledge.

→ They should specify only the external behavior of the system.

→ While writing user requirements, should not use s/w jargon, structured notations, or formal notations.

→ User requirements should write in natural language, with simple tables, forms, and intuitive diagrams.

→ System requirements are expanded versions of the user requirements that are used by s/w engineers as starting point for the system design.

→ They add detail & explain how the user requirements should be provided by the system.

→ They may be used as part of the contract for the implementation of the system & should therefore be a complete & detailed specification of the whole system.

→ The system requirements should simply describe the external behavior of the system & its operational constraints.

→ They should not be concerned with how the system should be designed or implemented.

→ At the level of detail required to completely specify a complex s/w system, it is practically impossible to exclude all design information. There are several reasons:

↳ A system architecture may be designed to structure the requirements.

Scanned with CamScanner

↳ The system may interoperate with other existing system that generate design requirements.

↳ The use of specific architecture to specify non functional requirements may be domain requirements.

→ User requirements are almost always written in natural language supplemented by appropriate diagrams & tables in the requirements doc.

→ System requirements may also be written in natural language but other notations based on forms, graphical system models, or mathematical system models can also be used.

Ways of writing a system requirements specification:

Notations	Description
Natural language sentences	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements written in natual language on a standard form or template.
Design description language	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML usecase & sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finit state machines or sets. These unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification.

→ when need to show how state changes or to describe sequence of actions graphical models are used. Formal mathematical specifications are sometimes used to describe the requirements for safety-or security-critical systems, but are rarely used in other circumstances.

Scanned with CamScanner

Natural language specification

→ Natural language has been used to write requirements for s/w since the beginning of s/w engineering.

→ It is expensive, intuitive, universal, vague, ambiguous and its meaning depends on the background of the reader.

To minimize misunderstandings when writing natural language requirements:

↳ Invent a standard format & ensure that all requirement definitions adhere to this format. standardizing the format makes omissions less likely & requirements easier to check.

↳ Use language consistently. Mandatory requirements are requirements that the system must support and are usually written using 'shall'. Desirable requirements are not essential & are written using 'should'.

↳ Use text highlighting (bold, italic, or color) to pick out key parts of the requirements.

↳ Avoid, as far as possible, the use of computer jargon. Inevitably detail technical terms will creep into the user requirements.

↳ Whenever possible, should try to associate a rationale with each user requirement. The rationale should explain why the requirement has been included.

Ex: 2 requirements for the embedded s/w for the automated insulinpump.

1. The system shall measure the blood sugar and deliver insulin if required, every 10 minutes.
2. The system shall run a self test routine every minute with the conditions to be tested & the associated actions defined.

Scanned with CamScanner

Structured specifications

→ Structured natural language is a way of writing system requirements, where the freedom of the requirements writer is limited & all requirements are written in a standard way.

→ The advantage of this approach is that it maintains most of the expressiveness & understandability of natural language.

→ Structured language notations use templates to specify system requirements.

→ To use a structured approach to specifying system requirements, need to define one or more standard forms or templates for requirements & represent these templates as structured forms.

→ When a standard form is used for specifying functional requirements the following information should be included.

1. A description of the function or entity being specified.
2. A description of its inputs & where these come from.
3. A description of its outputs & where these go to.
4. Info. about Info. that is needed for the computation or other entities in the system that are used.
5. A description of the action to be taken.
6. If functional approach is used, pre, post condition setting out what must be true before the function is called & after function called.
7. Description of the side effects of the operation.

Ex: A structured specification of a requirement for a insulin pump

Function: Compute insulin dose : safe sugar level

Description: Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone b/w 3 & 7 units.

I/ps: Current sugar reading (r_2) the previous two readings (r_0, r_1)

Source: Current sugar readings from sensors other readings from memory.

O/ps: CompDose - the dose in insulin to be delived.

Destination: Main control loop.

Scanned with CamScanner

Action: CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing & the rate of increase is increasing, then CompDose is computed by dividing the difference b/w the current sugar level & the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.

Requirements: Two previous readings so that the rate of change of sugar level can be computed.

Pre-condition: The insulin reservoir contains at least the maximum allowed single dose of insulin.

Post-condition: r_0 is replaced by r_1 then r_1 is replaced by r_2.

Side effects: None.

→ Using structured specifications removes some of the problems of natural language specification.

→ Variability in the specification is reduced & requirements are organized more effectively.

→ It is sometimes difficult to write requirements in a clear & un-ambiguous way, particularly when complex computations are to be specified.

→ To solve this problem, can add extra information to natural language requirements using tables or graphical models of the system.

→ Tables are particularly useful when there are a no. of possible alternative situations & need to describe the actions to be taken for each of these.

Ex:

Condition	Action
Sugar level falling ($r_2 < r_1$)	CompDose = 0
Sugar level stable ($r_2 = r_1$)	CompDose = 0
Sugar level increasing & rate of increase decreasing ($(r_2 - r_1) < (r_1 - r_0)$)	CompDose = 0
Sugar level increase & rate of increase is stable or increasing ($(r_2 - r_1) \geq (r_1 - r_0)$)	CompDose = round($(r_2 - r_1)/4$) if rounded result = 0 then CompDose = Minimum Dose

Tabular specification of computation for an insulin pump

Scanned with CamScanner

Requirements Validation

→ Requirements validation is the process of checking that requirements actually define the system that the customer really wants.

→ Requirements validation is important because errors in a requirements document can lead to extensive rework costs when when they are discovered during development or after the system is in service.

→ The cost of fixing a requirements problem by making a system change is much greater than repairing design or coding errors.

→ The reason for this is that a change to the requirements usually means that the system design & implementation must also be changed. & then the system must be tested again.

During the requirements validation process, checks should be carried out on the requirements in the requirements document. They are

↳ Validity checks: A user may think that a system is needed to perform certain functions.

→ Further thought & analysis may identify additional or different functions that are required.

→ Systems have diverse stakeholders with different needs, & any set of requirements is inevitably a compromise across the stakeholder community.

↳ Consistency checks: Requirements in the document should not conflict.

→ That is, there should not be contradictory constraints or different descriptions of the same system function.

↳ Completeness checks: The requirements document should include requirements, that define all functions, & the constraints intended by the system user.

↳ Realism checks: Using knowledge of existing technology, the requirements should be checked to ensure that they could be actually implemented.

→ These checks should also take account of the budget and schedule for the system development.

Scanned with CamScanner

↳ Verifiability: To reduce the potential for dispute b/w customer & contractor, system requirements should always be written so that they are verifiable.

→ This means that we should be able to write a set of tests that can demonstrate that the delivered system meets each specified requirement.

No. of requirements validation techniques:

1. Requirements reviews: The requirements are analyzed systematically by a team of reviewers.

2. Prototyping: In this approach to validation, an executable model of the system is demonstrated to end users & customers.

→ They can experiment with this model to see if it meets their real needs.

3. Test-case generation: Requirements should be testable.

→ If the tests for the requirements are devised as part of the validation process, this often reveals requirements problems.

→ If a test is difficult or impossible to design, this usually means that the requirements will be difficult to implement & should be reconsidered.

→ It is difficult to show that a set of requirements meets a user's needs.

Requirements management

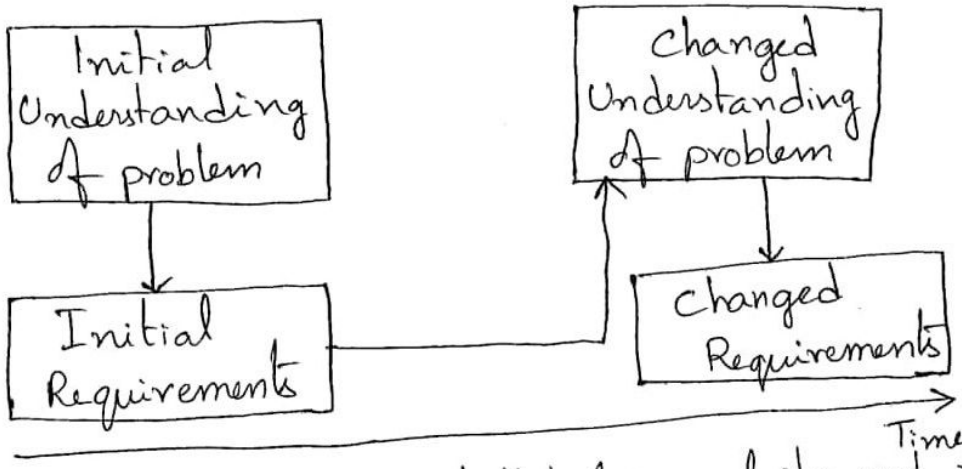
→ The requirements for large s/w systems are always changing.

→ One reason for this is that these systems are usually developed to address 'wicked' problems - problems that cannot be completely defined.

→ Because the problem cannot be fully defined, the software requirements are bound to be incomplete.

→ During the s/w process, the stakeholders understanding of the problem is constantly changing.

Scanned with CamScanner



→ Once a system has been installed & is regularly used, new requirements inevitably emerge.

→ It is hard for users and system customers to anticipate what effects the new system will have on their business processes & the way that work is done.

→ Once end users have experience of a system, they will discover new needs & priorities.

There are several reasons why change is inevitable:

1. The business & technical environment of the system always changes after installation. New H/w may be introduced, it may be necessary to interface the system with other systems, business priorities may change, and new legislation & regulations may be introduced that the system must necessarily abide by.

2. The people who pay for a system & the users of that system are rarely the same people. System customers impose requirements because of organizational & budgetary constaints. These may conflict with end user requirements & , after delivery, new features may have to be added for user support if the system is to meet its goals.

3. Large systems usually have a diverse user community, with many users having different requirements & priorities that may be conflicting or contradictory. The final system requirements are inevitably a compromise b/w them and, with experience, it is often discovered that the balance of support given to different users has to be changed.

→ Requirements management is process of understanding & controlling changes to system requirements.

Scanned with CamScanner

↳ Requirements management planning

→ Planning is an essential first stage in the requirements management process.

→ Requirements management is very expensive.

→ The planning stage establishes the level of requirements management detail that is required.

During this stage need to decide on:

Requirements identification: Each requirement must be uniquely identified so that it can be cross referenced with other requirements & used in traceability assessments.

A change management process: This is the set of activities that assess the impact & cost of changes.

Traceability policies: These policies define the relationships b/w each requirement & b/w the requirements & the system design that should be recorded & how these records should be maintained.

Tool support: Requirements management involves the processing of large amounts of information about the requirements. Tools that may be used range from specialist requirements management systems to spreadsheets & simple database systems.

The requirements management needs automated support & s/w tools for this should be chosen during the planning phase.

We need tools support for:

1. Requirements storage: The requirements should be maintained in a secure, managed data store that is accessible to everyone involved in the requirements engineering process.

Scanned with CamScanner

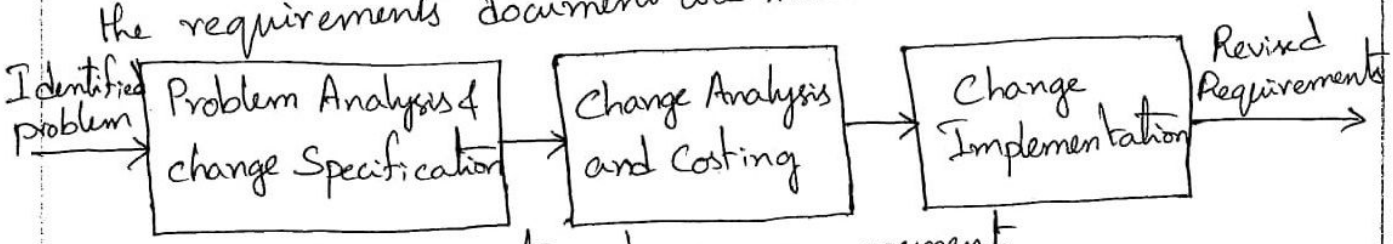
2. Change management: The process of change management is simplified if active tool support is available.

3. Traceability management: Tool support for traceability allows related requirements to be discovered. some tools use natural language processing technique to help discover possible relationships between requirements.

↳ Requirements change management

→ Requirements change management should be applied to all proposed changes to a systems requirements after the requirements document has been approved.

→ The advantage of using formal process for change management is that all change proposals are treated consistently & changes to the requirements document are made in controlled way.



Requirements change management

→ There are 3 principle stages to a change management process:

1. Problem analysis & change specification: The process starts with an identified requirements problem or sometimes, with a specific change proposal.

↳ During this stage, the problem or the change proposal is analysed to check that it is valid.

↳ The results of the analysis are feedback to the change requestor, & some times a more specific requirements change proposal is than made.

Scanned with CamScanner

2. Change analysis and costing: The effect of the proposed change is assessed using traceability information & general knowledge of the system requirements.
↳ The cost of making the change is estimated in terms of modifications to the requirements document & if appropriate to the system design & implementation.
↳ Once this analysis is completed, a decision is made whether to proceed with the requirements change.

3. Change implementation: The requirements document & where necessary, the system design & implementation are modified.
↳ We should organise the requirements document so that we can make changes to it without extensive rewriting or reorganization.
↳ As with programs, changeability in documents is achieved by minimising external references & making the document sections as modular as possible.
↳ Thus individual sections can be changed & replaced without affecting other parts of the document.

→ Agile development processes, such as extreme programming, have been designed to cope with requirements that change during the development process.
→ In these processes, when a user proposes a requirements change, this does not go through a formal change management process.
→ Rather, the user has to prioritise that change & if it is high priority, decide what system features that were planned for the next iteration should be dropped.

Scanned with CamScanner