- Nuja Joshi

- Software Engineering

Unit 1

1.1 # Software Engineering :

- Software engineering is concerned with theories, methods and tools for professional software development.
- Software engineering is concerned with cost-effective software development.
- Software costs more to maintain than it does to develop. For systems with a long life, maintainence costs may be several times development costs.

FAQ's about software engineering :

1. What are the costs of software engineering?
2. What are software engineering methods?
3. What is CASE (Computer Aided Software Engineering)?
4. What are attributes of good software?
5. What are the key challanges facing software engineering?
6. What is software?
7. What is software engineering?
8. What is difference between software engineering & computer science?
9. What is difference between software engineering & system engineering?

10 What is a software process?
11 What is a software process model?

What is professional & ethical responsibility of a software engineer?
→ Confidentiality
→ Ability
→ Punctuality
→ Good behaviour.

What is a software?

- Computer programs and associated documentation.
- Software products may be developed for a particular customer or may be developed for a general market.
- Software products may be:
  * Generic - developed to sold to a range of different customer.
  * Bespoke (custom) - developed for a single customer accordingly to their specification.

What is software engineering?

- is an engineering discipline which is concerned with all

aspects of software development

- Software engineers should adopt a systematic & organised approach to their work & use appropriate tools & techniques depending on the problem to be solved, the development constraints & resources available.

What is the difference between software engineering and system engineering.

Q no 12 (a) Module 3.

→ System engineering is concerned with all aspects of computer based system development including hardware, software and process engineering. Software engineering is part of this process.

→ System engineers are involved in system specification, architectural design integration & deployment.

Note:

Q: # Main task of software engineering.

1. Modify the existing software to correct errors, to adopt it to new hardware or to upgrade interfaces and improve performance.
2. Consult with engineering staffs to evaluate interface between h/d & s/w, develope specification

and performance requirements and resolve customer problems.

3. Develope and direct s/w system testing and validation procedures.
4. Store, retrieve and manipulate data for analysis of system capabilities and requirements.
5. Supervise and assign work to programmers, designers, technologists and technicians and other engineering and scientific personel.

Main task of a system engineer :-

1. Provide system operational training who required.
2. Design and develope computer software.
3. Cordinating the implementation of s/w systems and consulting with clients about technical issues and s/w needs.

What is the difference between software engineering and computer science.

→ Computer science is concerned with theory and fundamentals, software engineering is concerned with practica-

lities of ~~deploying~~ developing and delivering useful software.

- Computer science theories are currently insufficient to act as a complete development for software engineering.

What is software process?

→ A set of activities whose goal is the development or evolution of software.

→ Generic activities in all software processes are :

1) Specification - what the system should do and its development constraints.
2) Development - production of software system.
3) Validation - checking the softwares what the customer wants.
4) Evolution - Changing the software in response to changing demands.

What is a software process model?

→ A simplified representation of a software process, presented from a specific perspective.

→ Examples of process perspective are:
- workflow perspective - sequence of activities
- Data-flow perspective - information flow
- role / action perspective - who does what

→ Generic process models:
- Waterfall
- Evolutionary development
- Formal transformation
- Integration from reusable components. (Reuse based development).

What are the costs of software engineering?

→ Roughly 60% of costs are development costs, 40% are testing costs. For custom software, evolution cost often exceed development costs.
→ Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.
→ Distribution of cost depends on the development model that is used.

from book

What is CASE (Computer Aided Software Engineering)

→ Software systems which are intended to provide automated support for software activities.
→ CASE systems are often used for method support.
1. Upper CASE - Tools to support the early activities of of requirements and design.
2. Lower CASE - Tools to support later activities such as programming, testing and debugging.

What are attributes of good software?

→ Maintainability : Software must evolve to meet changing needs
→ Dependability : Software must be trustworthy.
→ Efficiency : Software should not make wasteful use of system resources.
→ Usability : Software must be usable by the users for which it was designed.

Professional and ethical responsibility :

→ Software Engineering involves wider responsibilities than simply the application of technical skills.

- SE must behave in an honest and ethically respon-sible way if they are to be respected as professional
- Ethical behaviour is more than simply upholding the law

What are the Key challanges facing software engineering?

→ Coping with legacy systems, coping with increasing diversity and coping with demands for reduced delivery time.

→ Legacy system : Old valuable systems must be maintained and obtained

→ Heterogenity : Systems are distributed and include a mix of hardware and software.

→ Delivery : There is increasing pressure for faster delivery of software.

Issues of professional responsibility :

→ Confidentiality : Engineers should normally respect the confidentiality of their employeers / clients

→ Competence : Engineers should not misrepresent their level of competence.

→ Intellectual property rights : Engineers should be aware of local laws governing the use of intellectual property. Such as copyright.

→ Computer misuse - Software Engineers should not use their technical skills to misuse other people's computers.

IEEE code of ethics principle :

→ Public : Software Engineers shall act consistently with the public interest.

→ Client and employer : Software engineer shall act in a manner that is in the best interest of their client and employer.

→ Product : Software engineers shall ensure that their products and related modification meet the highest professional standard possible.

→ Judgement : Shall maintain integrity and independence in their professional judgement.

→ Management : Managers and leaders shall promote an ethical approach to the management of software development and maintainence.

→ Profession : shall advance the reputation of profession consistent with public issues.

PCPJMP

→ Colleagues : Shall be fair to and supportive of their colleagues.

→ Self : shall participate in lifelong learning regarding the practice of their progession.

1.2 System Engineering :

What is a system ?

- A purposeful collection of inter-related components working together towards some common objective.
- A system may include software, electrical, electronic & mechanical hardware & be operated by people.
- System components are dependent on other system components.
- The properties & behaviour of system components are inter-mingled. (Combine)

Problems of System Engineering :

- System engineering requires a great deal of co-ordination across disciplines.
  - Almost infinite possibilities for design across components.
  - Mutual distrust & lack of understanding

across engineering discipline.
- System must be designed to last many years in changing environment.

Emergent properties :-

- Properties of the system as a whole rather than properties that can be derived from the properties of components of a system.
- Emergent properties are a consequence of the relationships between system components.
- Therefore they can only be assessed and measured once the components have been integrated into a system.

Examples of emergent properties :-

1. The overall weight of the system
2. The reliability of a system.
3. The usability of a system.

Types of emergent properties :-

1 Functional properties :-

These appear when all the parts of a system work together to achieve some objective.
Eg: साइकलका Parts

2. Non-functional emergent properties:
These relate to the behaviour of the system in its operational environment. Eg: reliability, performance, safety & security.

The system engineering process:

1. Usually follows a 'waterfall' model because of the need for parallel development of different part of the system.
→ little scope for iteration between phases, because hardware changes are very expensive. Software may have to compensate for hardware problems.

2. Inevitably involves engineers from different discipline who must work together. Much scope of misunderstanding here. Diff. disciplines use a different vocabulary and much negotiation is required. Engineers may have personal agendas to fulfill.

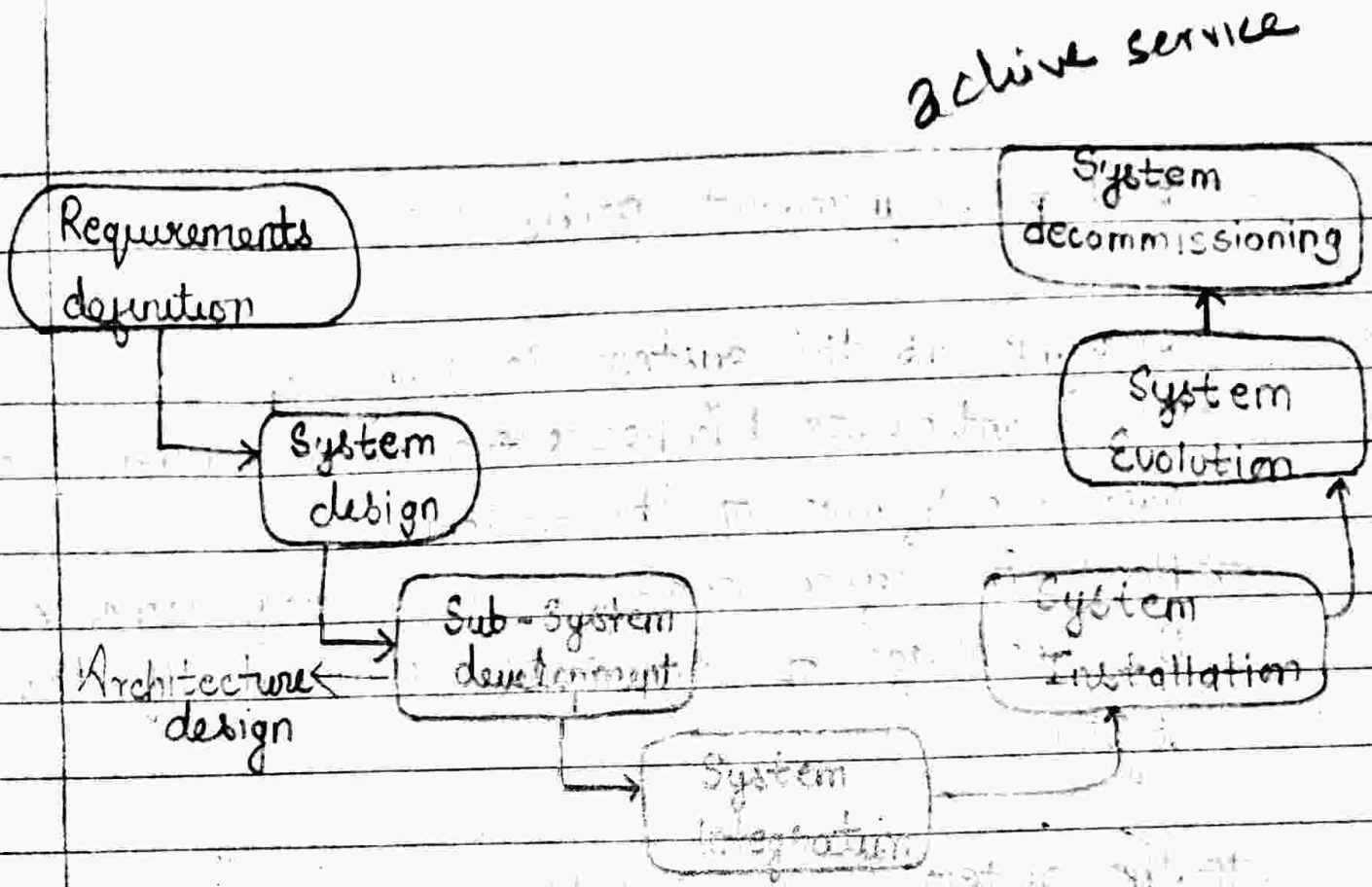


Fig. System Engineering

9th Nov. # System requirements definition :-

→ Three types of requirement defined at this stage :
- Abstract functional requirements. System functions are defined in an abstract way.
- System properties → non-functional requirements for the system in general are defined.
- Undesirable characteristics → Unacceptable system behaviour is specified.

→ Should also define overall organisational objectives for the system.

System requirement problems :

(Change हुन सक्ने requirement)

→ Changing as the system is being specified.

→ Must anticipate hardware/& communications development over the lifetime of the system.

→ Hard to define non-functional requirements without the knowledge of component structure of the system.

The system design process :

Q no 3

1. → Partition requirements :

2068 Organise requirements into related groups.

Same same nature हरु

2. → Identify subsystems :

Identify a set of subsystems which collectively can meet the system requirements.

3. → Assign requirements to sub systems :

Cause particular problems when COST are integrated.

4. → Specify sub-system functionality.

5. → Define sub-system interfaces :

critical

Critical activity for parallel sob-system development.

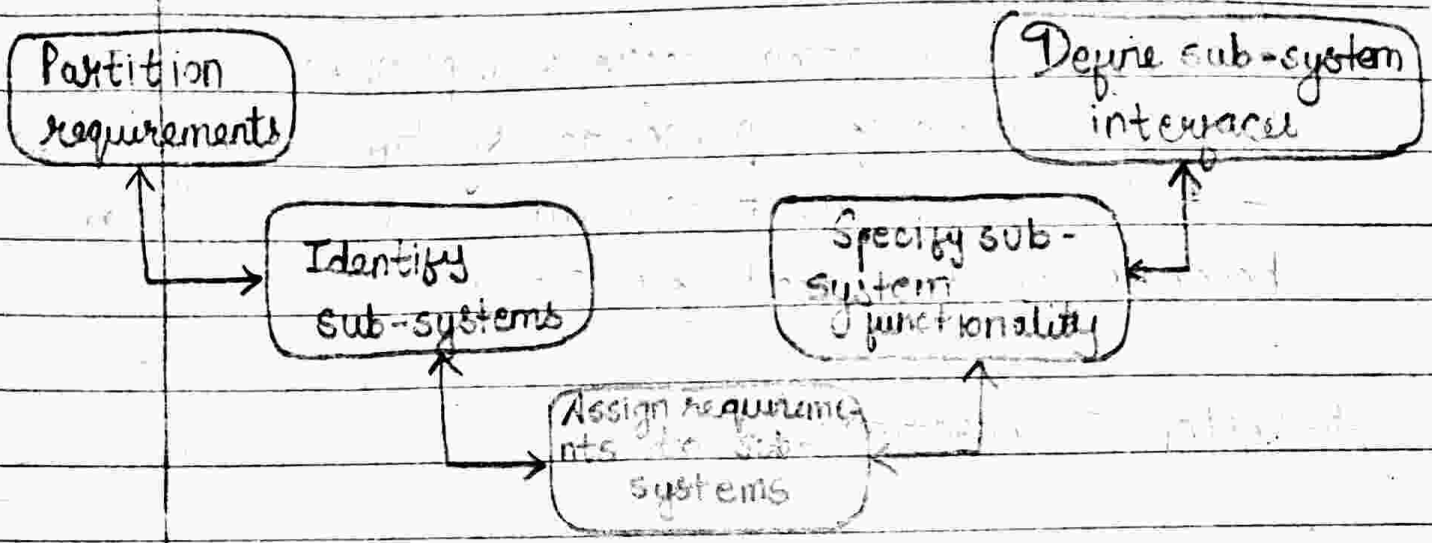


Fig. System design process

System design problems :

→ Requirements partitioning to hardware, software & human components may involve a lot of negotiation.

→ Hardware platforms may be inappropriate for software requirements so software must compensate for this.

Sub-system development :

→ Typically parallel projects developing the hardware, software and communications.

shelf)

→ May involve some COTS (Commercial Off-the-Shelf) system, procurement.

→ Lack of communication accross implementation teams.

→ Slow mechanism for proposing system changes means that the development schedule may be extended because of the need for rework.
rework

System Integration:- (Book + Note)

2 → The process of putting hardware, software & people together to make a system.

→ Should be tackled incrementally so that sub-systems are integrated one at a time.

3 → Interface problems between sub-systems are usually found at this stage.

4 → May be problems with uncoordinated deliveries of system components.

11th Nov. # System Installation:-

→ Environmental assumptions may be incorrect.

→ May be human resistance to the introduction of a new system. (road झमेला interference)
new

→ System may have to coexist with alternative system

- mainly in legacy system

for some time. (पुरानो machine सगो coordinate गर्नुपर्छ)

→ May be physical installation problems (eg:- cabling problems)

→ Operator training has to be identified.
(नयाँ system सिक्दा त्यो बुझ्नु training दिनुपर्छ)

System Operation :

requirement ता तदेखेको कुरा

→ Will bring unforeseen requirements to light.

→ Users may use the system in a way which is not anticipated by system designers.

→ May reveal problems in the interaction with other systems.

- physical problems of incompatibility.
- increased operator error rate because of inconsistent interfaces. (Do you want to save?)

धेरैं सिक्नु पर्छ।

System evolution : (modify गर्ने)

→ large systems have a long lifetime. They must evolve to meet changing requirements.

→ Evolution is inherently costly.

→ Existing systems which must be maintained are

COTS - System બનાવનારનું પહેલા, engineer સાથે ચર્ચા discuss કરવું પડેલ।

બનાવો | કિંમત જ્યારેય કે modify કરીને વાપરે અનુસાર, cost effective, requirement ના modify કરે

Sometimes called legacy system.

22nd Nov, 2012.

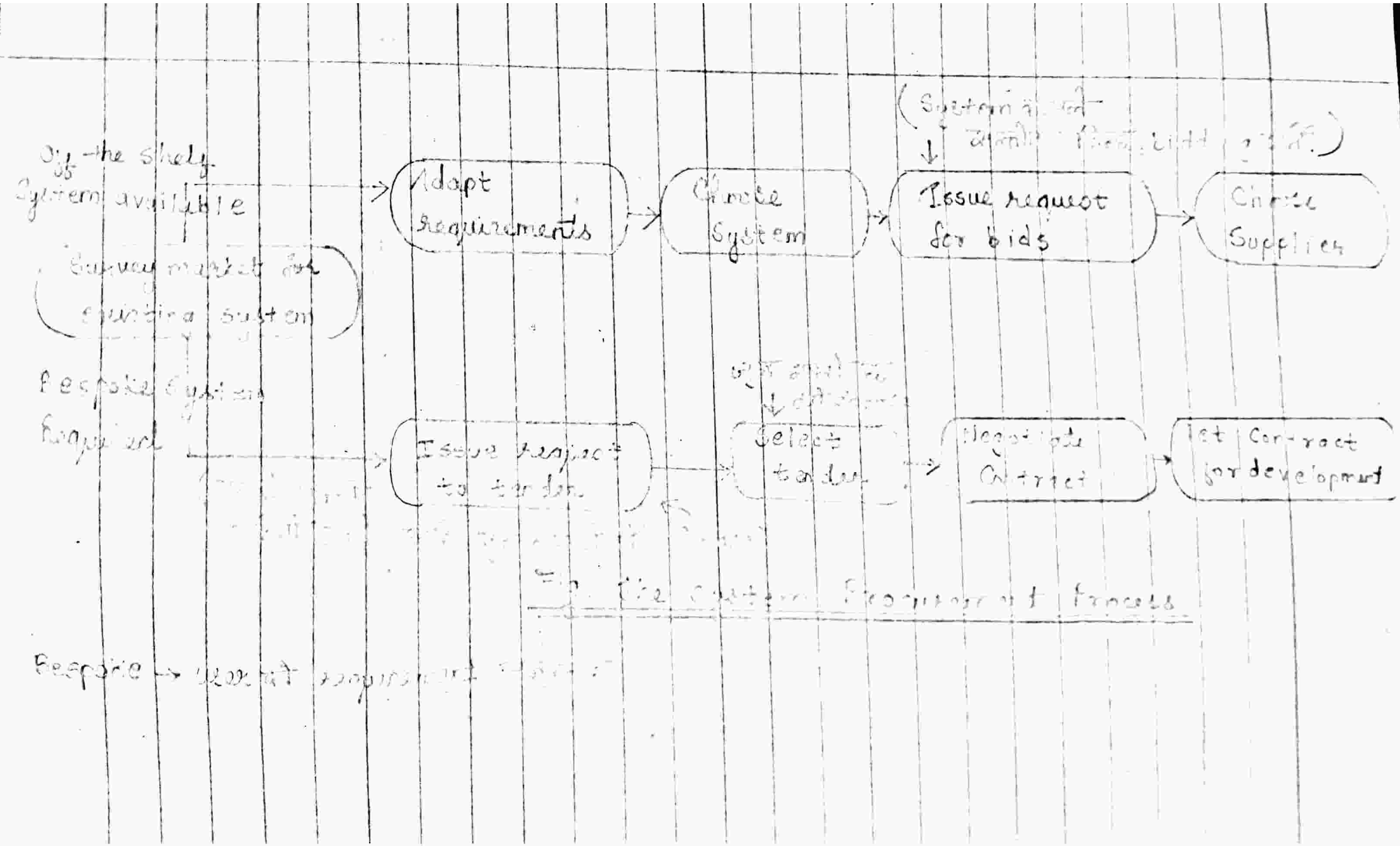
System decommissioning :

- Tacking the system out of service after its useful lifetime.
- May require removal of materials (eg: dangerous chemicals) which pollute the environment.
- May require data to be restructured and converted to be used in some other system.

System procurement :

↳ by buying goods & services

- Acquiring a system for an organization to meet some need.
- Some system specification & architectural design is usually necessary before procurement.
  - you need a specification to let a contract for system development.
  - the specification may allow you to buy a commercial off-the-shelf (COTS) system. Almost always cheaper than developing a system from scratch.


Off-the shelf System available
Survey market for existing system
Adapt requirements
Choose System
Issue request for bids
Choose Supplier
Bespoke System required
Issue request to tender
Select tender
Negotiate Contract
Let Contract for development

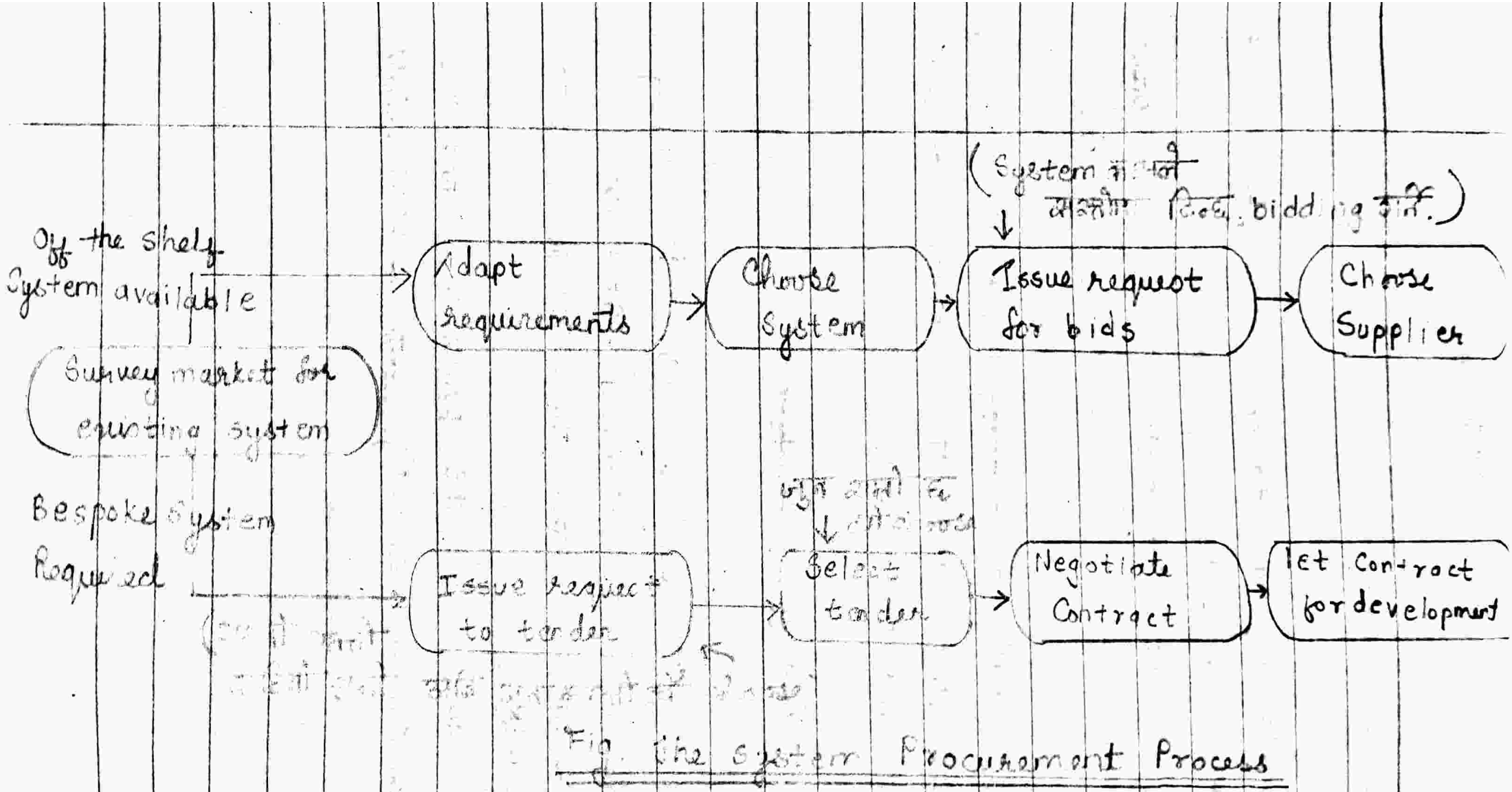


Fig the system Procurement Process

Bespoke → User of requirement

Procurement issues :

- Requirements may have to be modified to match the capabilities of off-the-shelf components.
- The requirement specification may be the part of the contract for the development of the system.
- There is usually a contract negotiation period to agree changes after the contractor to build a system has been selected.

Contractor and sub-contractors :

- The procurement of large system is usually based around some principal contractor.
- Sub contractor are issued to other suppliers to supply parts of the system.
- Customer deals with the principal contractor and does not deal directly with sub contractors.

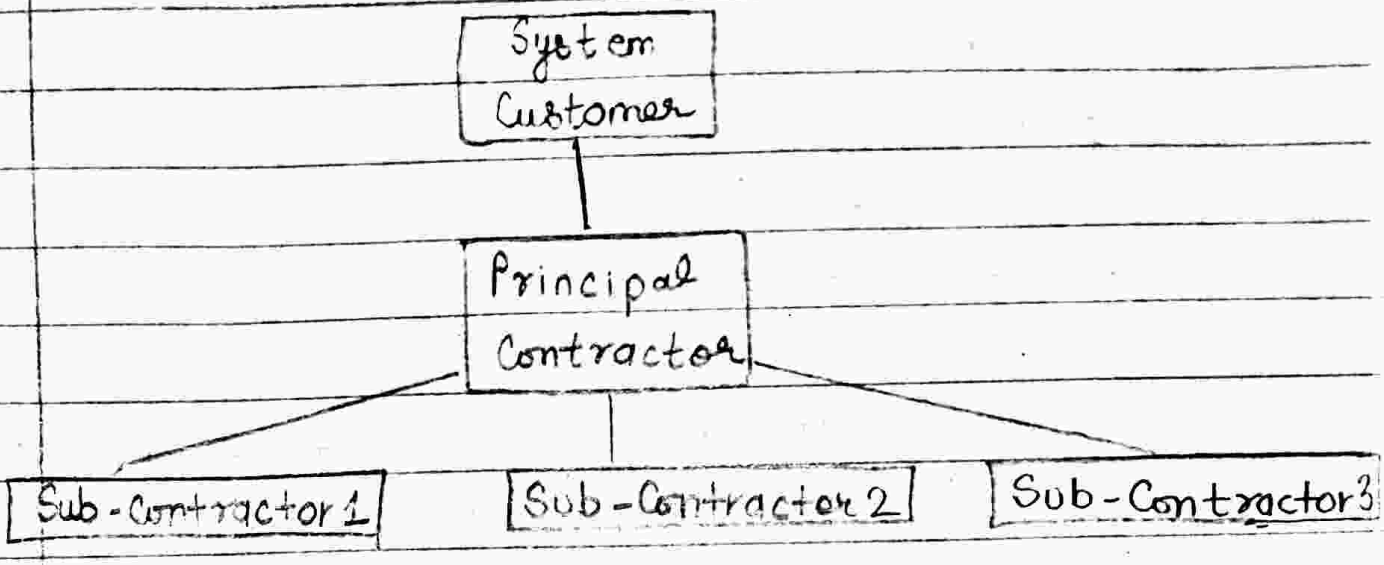


Fig. Contractor / Sub-contractor Model

System reliability engineering :-

- Because of the component inter-dependencies faults can be propagated through the system.
- System failures often occur because of unforseen inter-relationships between components.
- It is probably impossible to anticipate all possible component relationships.
- Software reliability measures may give a false picture of the system reliability.

1. # Hardware reliability :-

- What is the probability of a hardware component

failing and how long does it take to repair that component?

2. # Software reliability :-

- How likely is it that a software component will produce an incorrect output. Software failure is usually distinct from hardware failure in that software does not wear out.

↑ To carry production

3. # Operator reliability :-

How likely is that the operator of a system will make an error?

23rd Nov, 2012. # Software Processes :- (1.3)

Software process is a structured set of activities required to develop a software system.

1) Specification -
2) Design
3) Validation - Testing phase
4) Evolution - modify

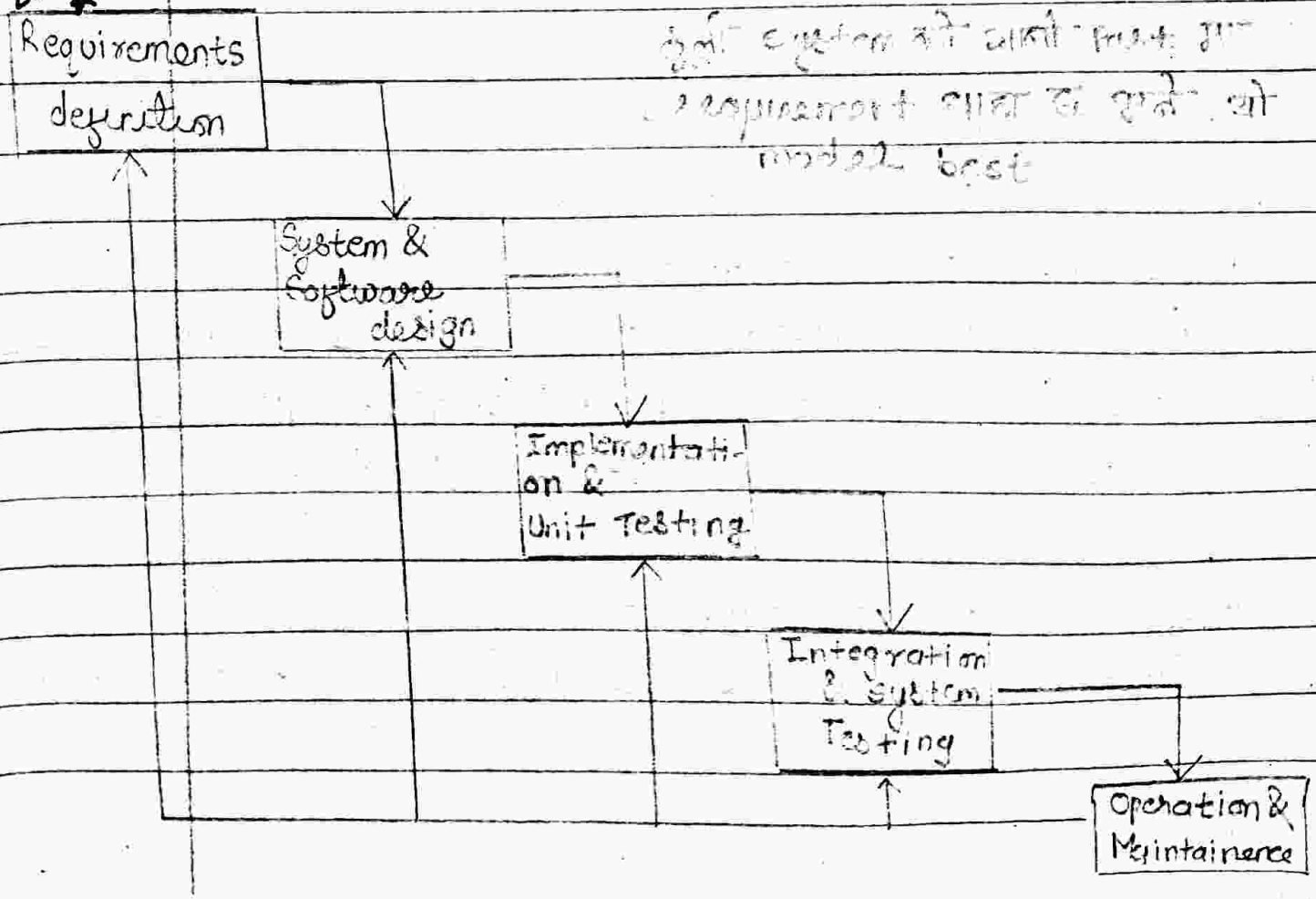
Critical system - hamper पार्ने system, cheap budget, कोही long life system, sensitive, fail हुने chance Nil हुनुपर्ने coz dherai hamper गर्दैन।

- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

Generic Software Process models

1) The waterfall model
2) Evolutionary development
3) Formal systems development
4) Re-use based development.

1) The waterfall model :



Advatages :

- This model is only appropriate when the requirements are well understood.

Disadvantages :

- difficulty of accomodating change after the process is underway.
- Inflexible partitioning of project into distinct stages.
- difficult to respond to changing customer requirements.

Evolutionary development - Customer interaction करके feedback लेते हुए so better than WF.

i) Exploratory development : (working की process)
Objective is to work with customer & to evolve a final system from an initial outline specification. Should start with well-understood requirements.

ii) Throw-away prototyping :
Objective is to understand the system requirement

- Small type system is appropriate for evolutionary development. (Not 4 large system)

Should start with poorly understood requirements.

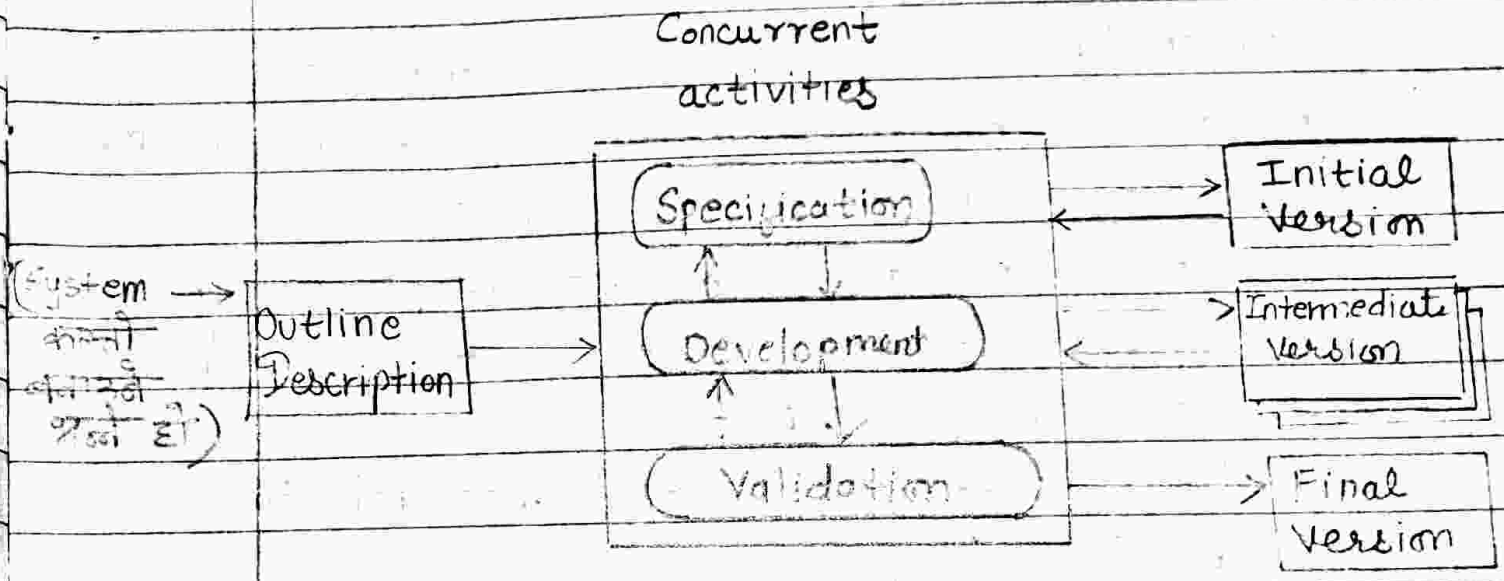


Fig. Evolutionary development model

Problems

- lack of process visibility
- Systems are often poorly structured.
- Special skills (eg ÷ in languages for rapid prototyping may be required.

Applicability ÷

- For small or medium-size interactive systems.
- For parts of large systems.
- For short-lifetime systems.

cleanroom - white box testing

Formal Systems development :

- Based on the transformation of a mathematical specification through different representations to an executable program.
- Transformations are 'correctness-preserving so it is straight forward to show the program conforms to its specification.
- Embodied in the 'cleanroom' approach to software development.

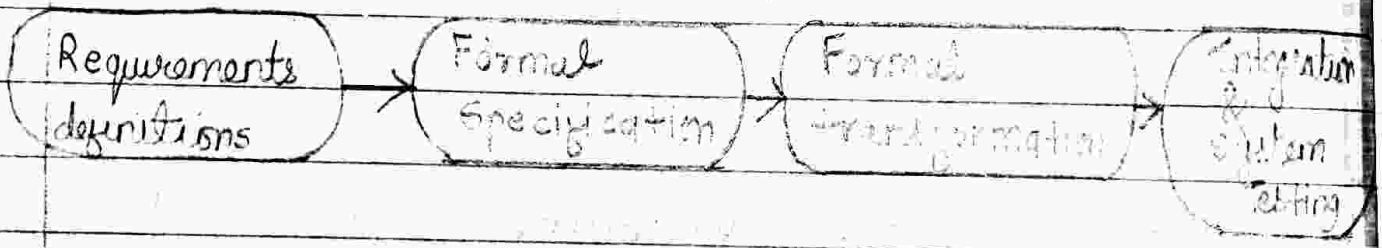


Fig. Formal System Development

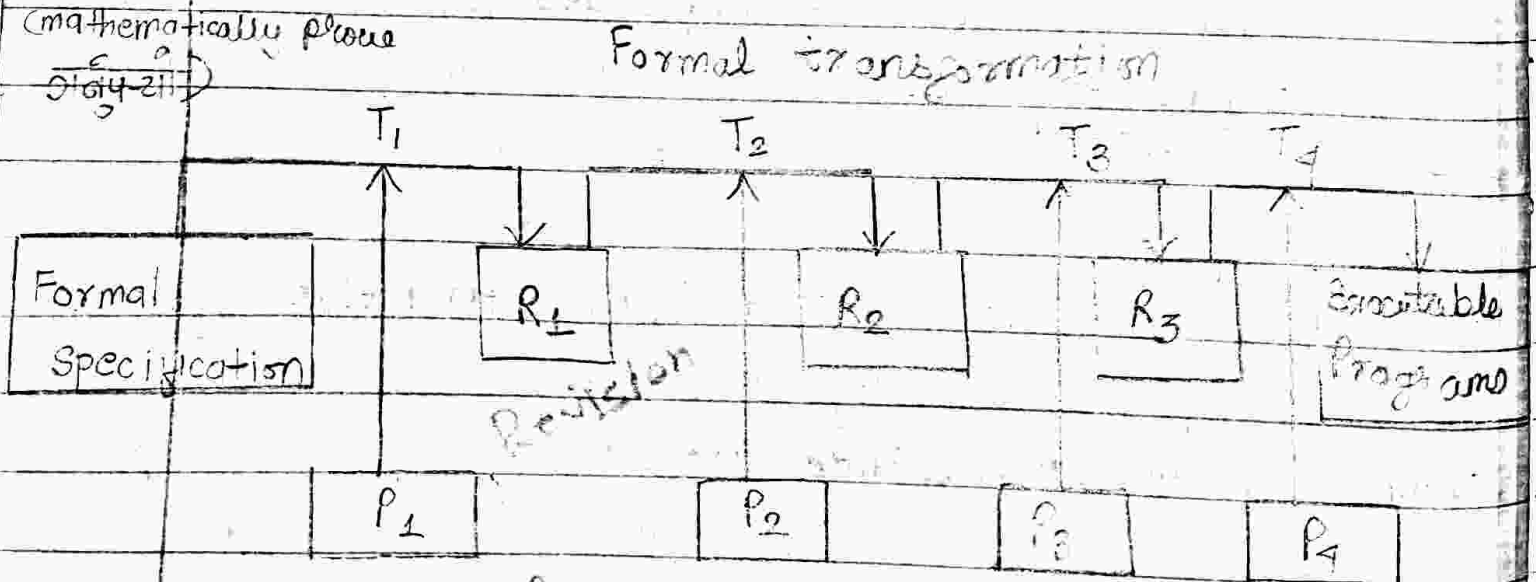


Fig. Formal transformation

हरेक कुरालाई mathematically solve गरेरमात्र पर्ने यसको disadvantage हो।

Problems :

- Need for specialised skills & training to apply the technique.
- Difficult to formally specify some aspects of the system such as user interface.

Applicability :

- Critical systems especially those where a safety or security case must be made before the system is put into operation.

Reuse Oriented development : (पुरानोलाई नै reuse गर्ने)
/ Component Based Software Engineering.

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial - Off - the - Shelf) systems.
- This approach is becoming more important but still limited experience with it.

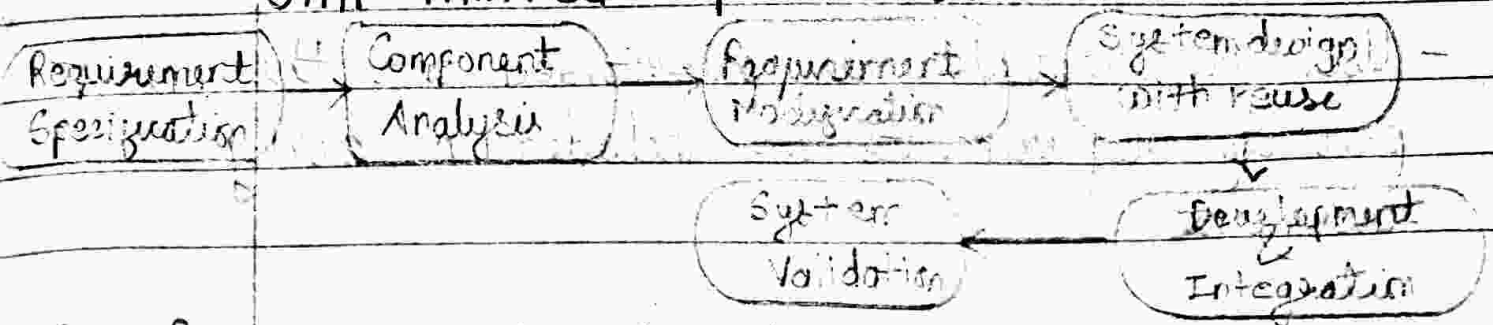


Fig Reuse oriented development

WF & Evolutionary difference

- # Process iteration ÷ (सब process जहाँ फिर Checking

- Systems requirements always evolve in the course of a project so process iteration where earlier stages are reworked is always part of the process for large systems.

- Iteration can be applied to any of the generic process model.

- Two related approaches:
  - Incremental development
  - Spiral development (Specification, design, validation, evolution)

Incremental development ÷

- Rather than deliver the system as a single delivery, the development & delivery is broken down into increments with each increment delivering part of the required functionality.

- User requirements are prioritised and the highest priority requirements are included in early increments.

जुन priority छ त्यसलाई अगाडि राख्छौं,
Part, part गर्दै integrate गर्दै अनि अगाडिको सँग validate
Most priority sub system लाई max^m time check so error कम हुन्छ

- Once the development of an increment is started, the requirements are frozen through requirements for later increments can continue to evolve.

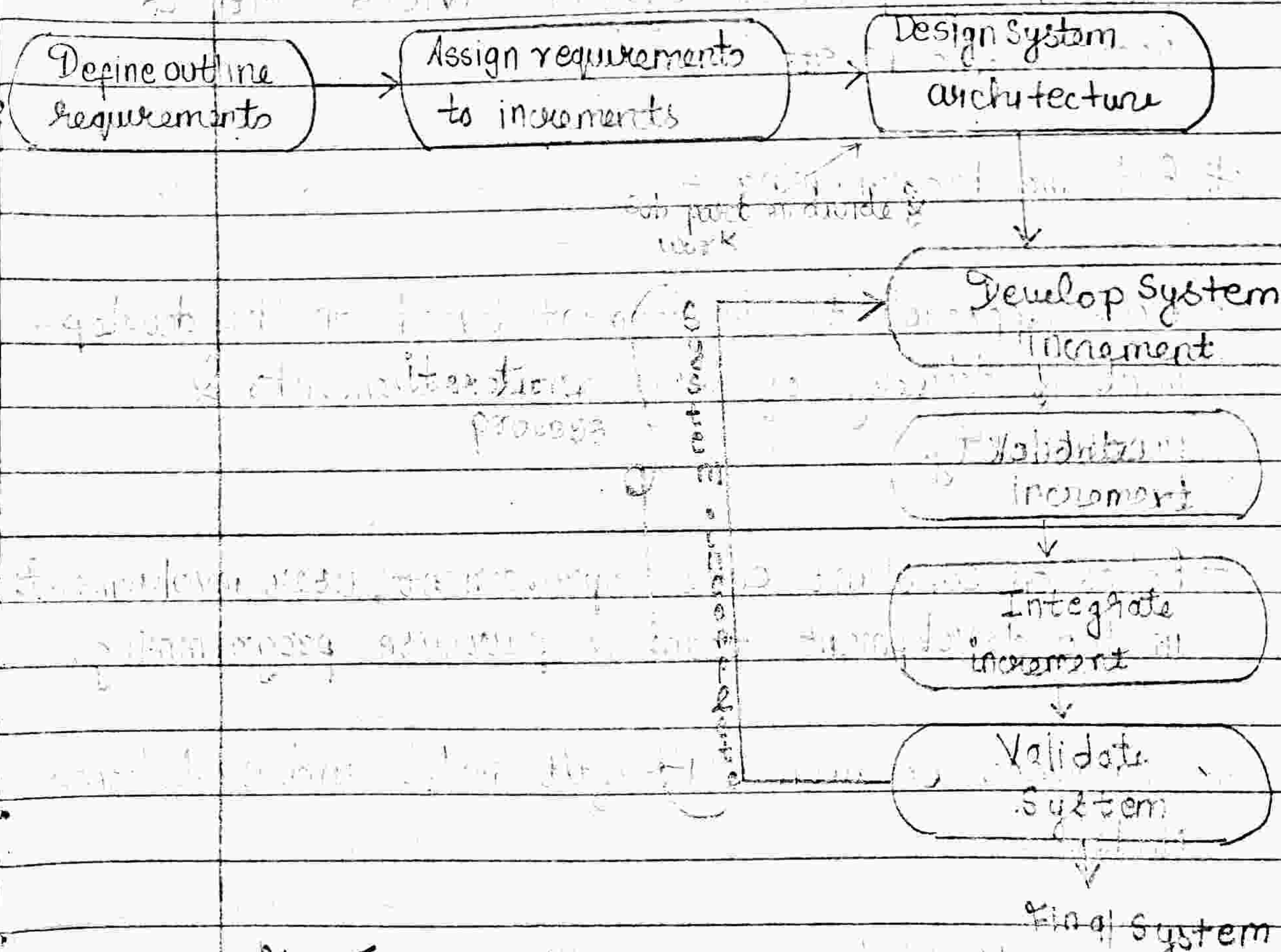


Fig: Incremental Development

Advantages :-

- Customer value can be delivered with each increment so system functionality is available earlier.

(यो system बाट न हुनै chance कम हुन्छ)
- early use of system (पहिला सकाई transaction मोड्ने) यहि यस पाउँछ
- Early increments act as a prototype to help elicit requirements for later increments.
- lower risk of overall project failure.
- The highest priority system services tend to receive the most testing.

Extreme Programming ÷ (priority अनुसार कुन आदि critical आदि अत्यन्तको सँग integrate (Rapid development System) यस्तो आदि कम हुन्छ)
- New approach to development based on the development & delivery of very small increments & functionality.

- Relies on constant code improvement, user involvement in the development team & pairwise programming.

Difference between waterfall model and evolutionary Model:

Waterfall Model	Evolutionary Model.
1. It is expensive and time consuming as it was designed to minimize the use of expensive resources.	1. It works with client & to evolve a final system from an initial outline specification.

2. It describes the process of stepwise refinement.	2. It starts with well understood requirements and system evolves by adding new features.
3. The model is slow & uneconomical as there is difficulty of accomodating change after development has been stored.	3. lacks process visibility and offers poor structure to the project because of adhoc manner of development.
4. Separate & distinct phases of specification & development.	4. Specification & development phases are interleaved.
5. Advantages	
i. Reflects engineering practices.	i. Meets the intermediate needs of customer.
ii. Stays in place long-term.	ii) Specification can be developed incrementally.
iii Straightforward, simple.	
6. Disadvantages	
i. Iteration becomes costly	i. Process is not visible.

- further process ईती

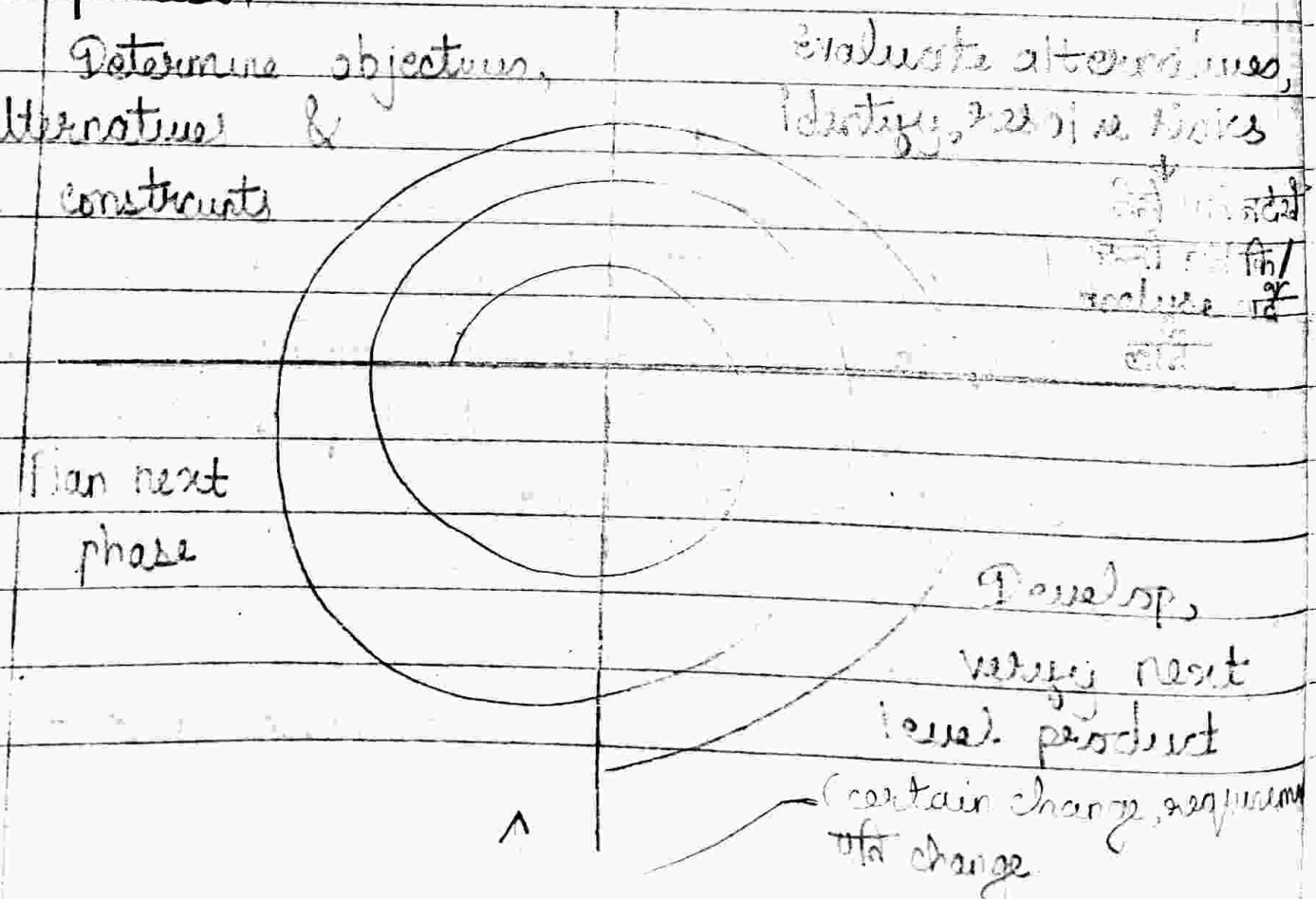
ii. Inflexible	ii. Systems are often poorly structured. iii. Special tools & techniques may be required.

Spiral development :-

Q no 2 model ques.

Rather than represent the software process as a sequence of activities with some backtracking from one activity to another, the process is represented as spiral.

- Each loop represents a phase of the software process.



SE - management पनि हुन्छ, कसरी काम गर्ने, tools (CASE कुन use गर्ने i.e difference from other engineers.
Suppose bridge बनाउनलाई यदि बनाईन्छ तर SE ले आफैं मा सोच्नुपर्छ। कसरी use गर्ने पर्छ, code

Software Specification :

- The process of establishing what services are required and the constraints on the system's operation & development.

- Requirements engineering process :
    - Feasibility study
    - Requirements elicitation & analysis (requirements बनाउने, सोध्ने कि के चाहिन्छ, धेरै खाली)
    - Requirements specification
    - Requirements validation

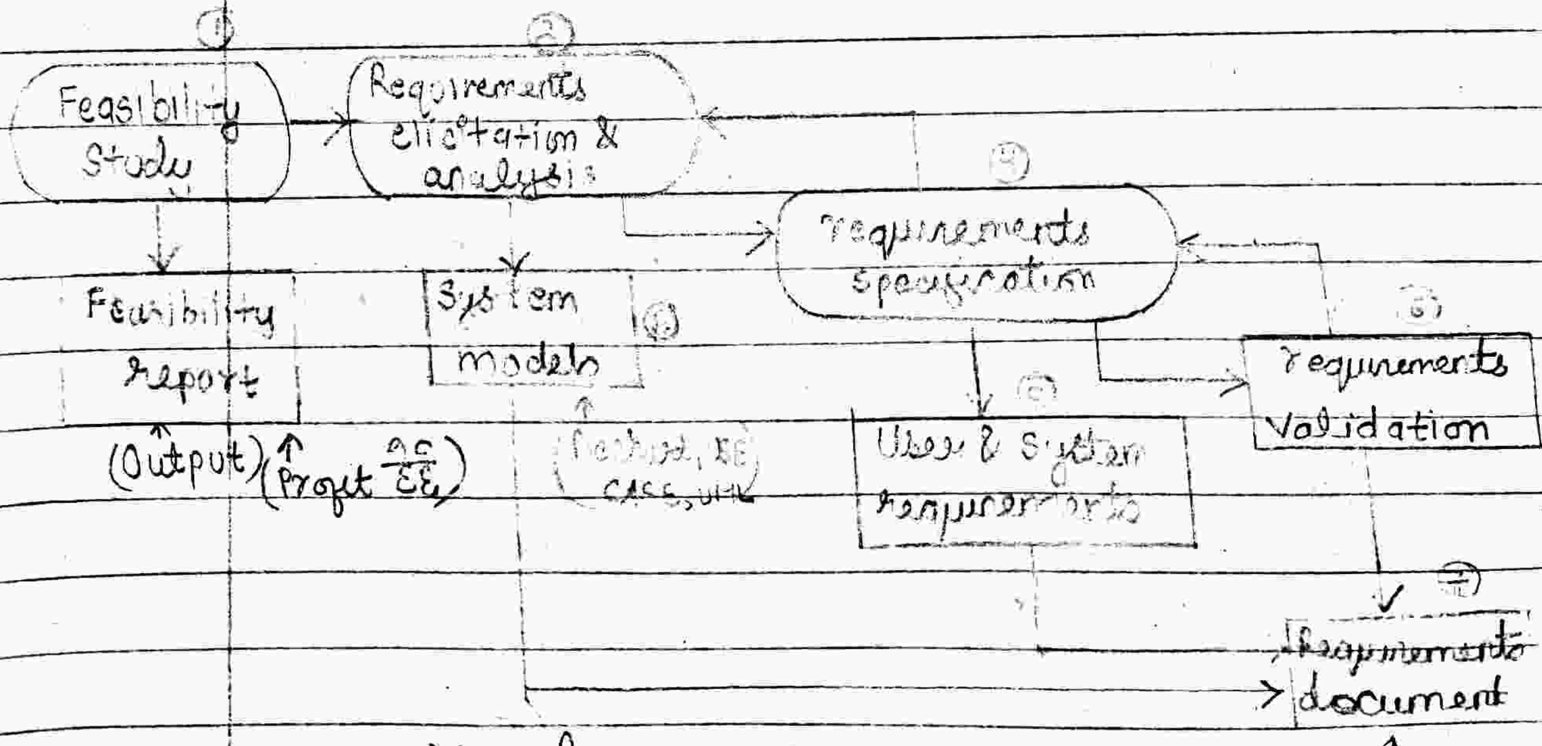


Fig Requirements engineering process
(User लाई बुझाउने किसिमले, abstract गरेर)

→ CASE tools, जस्तो design, program लेख्ने पनि धेरै यन्त्र
" " use गर्न आनि executable file बनाउने

Software design and implementation:

- The process of converting the system specification into an executable system.
- Software design means design a software structure that realises the specification.
- Implement means translate this structure to an executable program.
- The activities of design & implementation are closely related & may be interleaved.

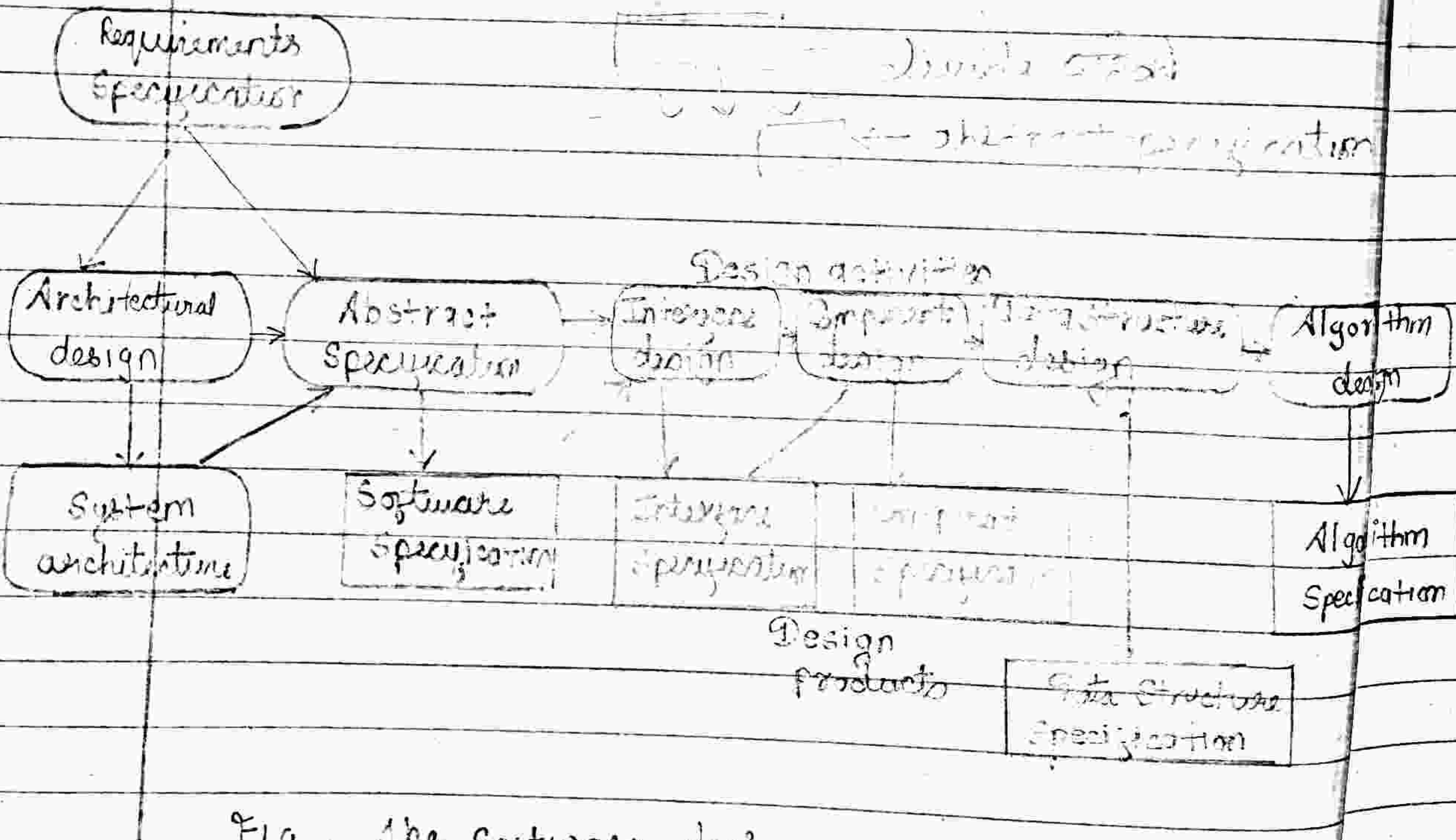


Fig. The Software design process

Test case – before running ni case [illegible] ??
Convert → Test data – after, data real data पठाउँछ ।

Design models :/ Structured method

- Systematic approaches to developing a software design.
- The design is usually documented as a set of graphical models.
- Possible models :
  - Data-Flow model
  - Entity-Relation-Attribute model
  - Structural model
  - Object model. (CLASS, USE-CASE)

Programming & debugging :

- Translating a design into a program & removing errors from that program.
- Programming is a personal activity - there is no generic programming process.
- Programmers carry out some program testing to discover faults in the program & remove these faults in the debugging process.

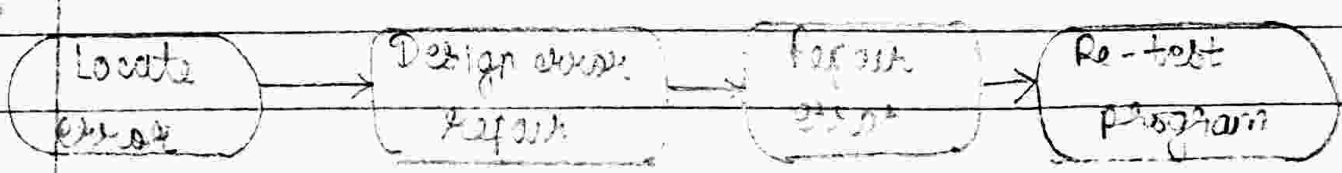


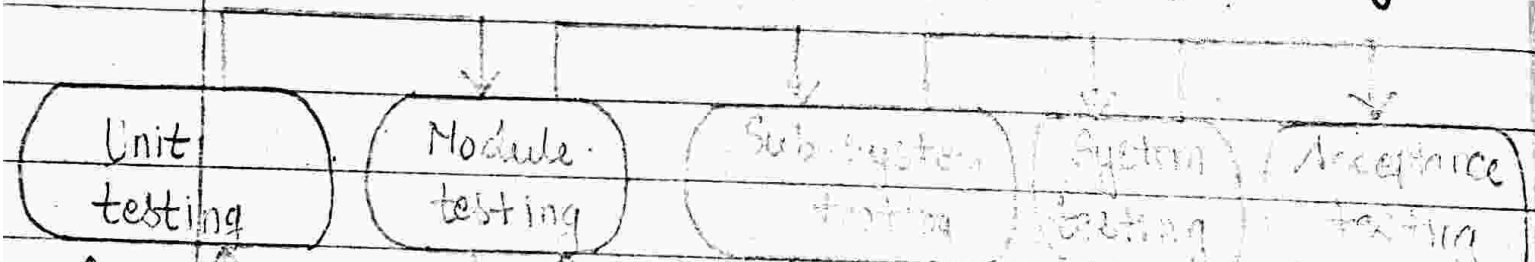
Fig. The debugging process

Software Validation :

- Verification & Validation is intended to show that a system conforms to its specification & meets the requirements & the system customer.

- Involves checking & review processes & system testing.

- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.



Component Testing

Integration Testing

Fig. The testing process.

V-model

इसको test plan बनाकर हुए

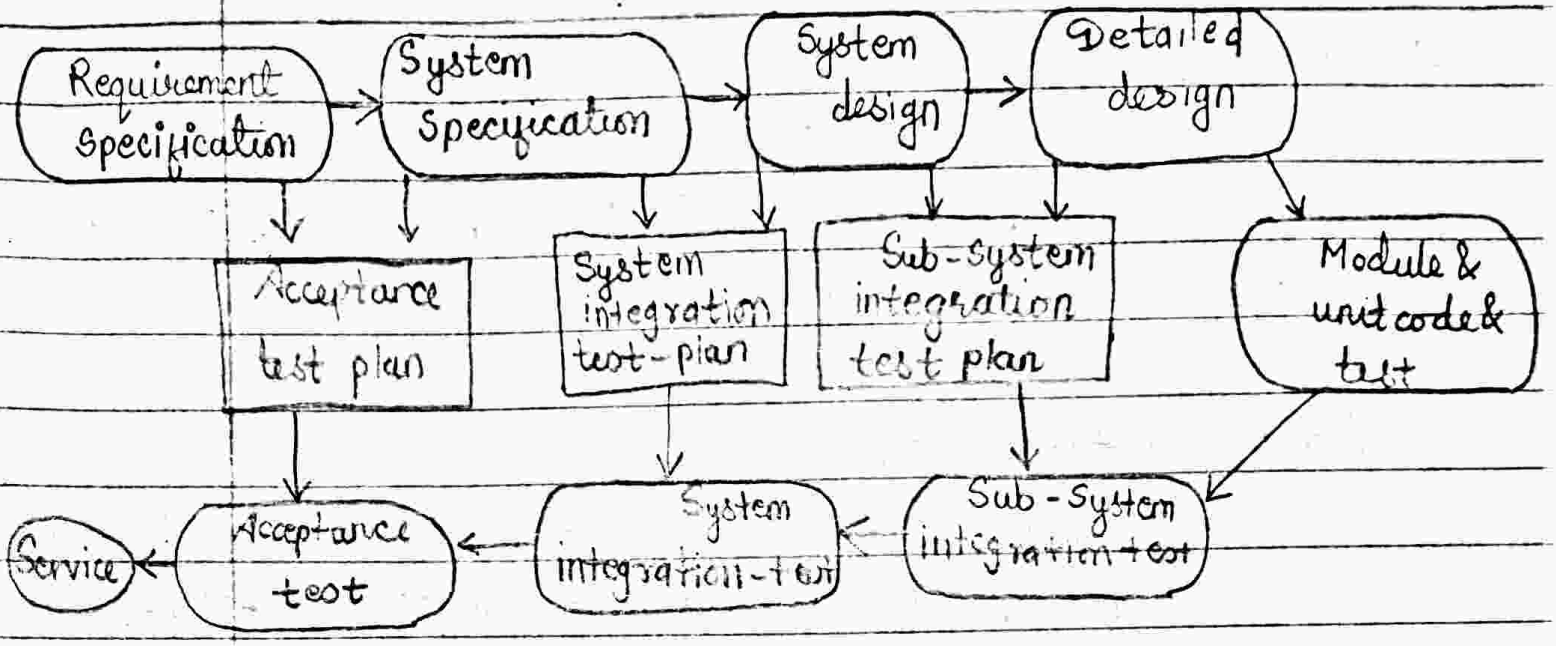


Fig. Testing Phases

Software Evolution :-

फेरि use गरेर बनाउने, Flexibility - part बढाउन / हटाउन सक्ने ।

- Software is inherently flexible & can change.
- As requirements change through changing business circumstances, the software that supports the business must also evolve & change.

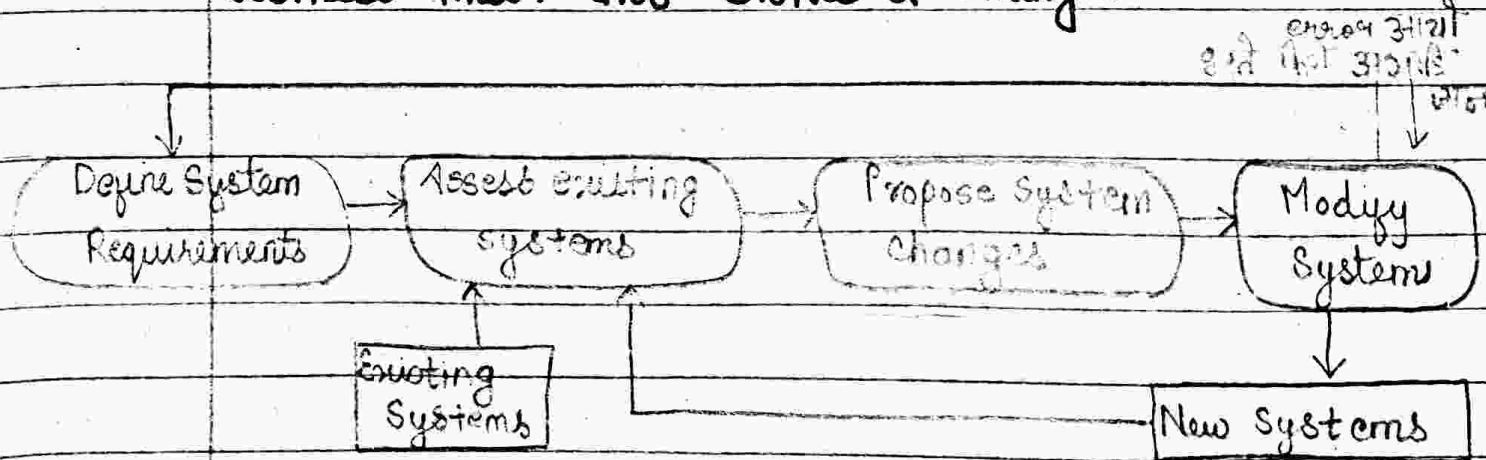


Fig. System Evolution

CASE ÷ (Automated Support tools)

(जसैं case या CASE tools use हुन्छ गर्ने हैन, कोई engineer ले आफै use गर)

- Computer Aided Software Engineering (CASE) is software to support software development & evolution processes.

- Activity automation:
  - Graphical editors for system model development
  - Data dictionary to manage design entities.
  - Graphical UI builder for user interface construction.
  - Debuggers to support program fault finding
  - Automated translators to generate new versions of a program.

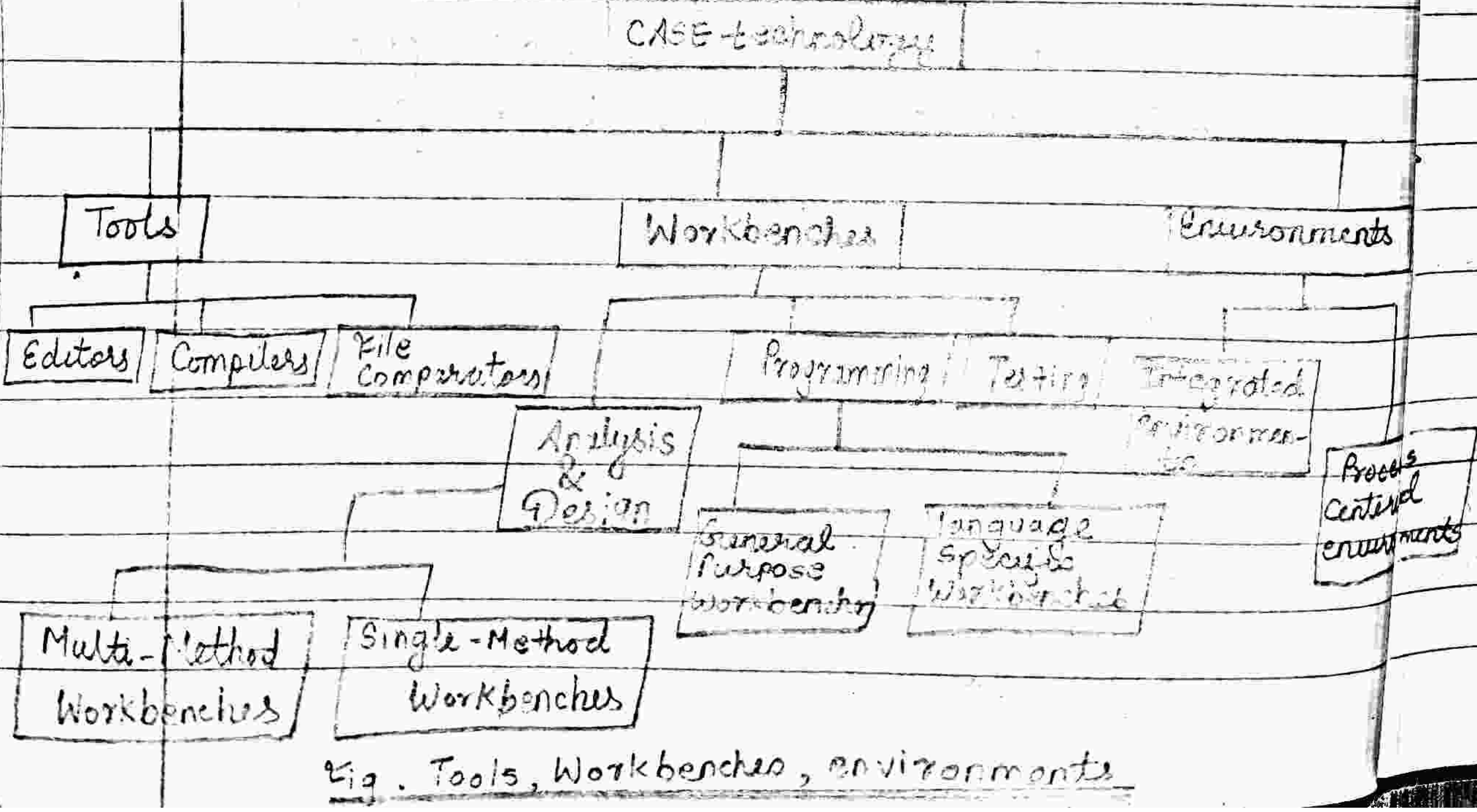


Fig. Tools, Workbenches, environments

Class Assignment :-

Q no 1) Explain why programs that are developed using evolutionary development are likely to be difficult to maintain?

→ As software process includes four main process, they are specification, design, validation & evolution. After all the processes are done evolution is the last process carried on the development of a program.

In an evolutionary development, first the system requirements are defined, after this, the changes are proposed in a system & we again modify the old system into a new system and whenever the system doesn't meet any requirements, it is again modified hence it becomes very hard for the engineers to re-design the system again & again.

Finally, we can say that as the requirement

of changes through the different changing business trends in the field, the whole software must also change from the existing system. Hence, evolutionary development seems to be difficult to maintain.

Q no 2) Suggest why it is important to make a distinction between developing the user requirements & developing system requirements in the requirements engineering process.

→ To develop any software there must be a good interaction between the user and the engineers, if it is done, then only we can have a good system. Before the system are made user proposes all the inputs to the engineers. Whenever the inputs are produced, all the inputs are combined & the final system is produced.

We can say that user requirements determine the system. Simple i/p's produces ~~con~~ simple o/p's

and hence, if there is simple interface & if the cost is also low than the system is also built.

Checked 2nd Dec 2021

Unit 1.1 # Project Management ÷ { Organising, planning and scheduling s/w proj

Unit 1.4.

- Software Project Management
- Concerned with activities involved in ensuring that software is delivered on time & on schedule & in accordance with the requirements of the organisations developing & processing the software.

- Project management is needed because software development is always subject to budget & schedule constraints that are set by the organisation developing the software.

Software management distinctions

- The product is intangible.
- The product is uniquely flexible.
- Software engineering is not recognised as an engineering discipline same as mechanical, electrical

engineering etc.
- The software development process is not standarised
- Many software projects are "one-off" projects

Management Activities :

- Proposal Writing
- Project Planning & Scheduling
- Project Costing
- Project monitoring & reviews
- Personel selection & evaluation
- Report Writing & presentation

Project Staffing :

- Project budget may not allow for the use of highly-paid staff.
- Staff with the appropriate experience may not be available.
- An organisation may wish to develope employee skills on a software projects.
- Managers have to work within these constraints especially when there is is an international shortage of skilled IT staff.

Project Planning :

- Probably the most time-consuming project management activity.
- Continous activity from initial concept through to system delivery. Plans must be regularly revised as new information becomes available.
- Various types of plan may be developed to support the main s/w project plan that is concerned with schedule & budget.

Types Of Plan

1) Quality plan
2) Validation plan
3) Configuration Management plan
4) Maintainence plan (time + cost)
5) Staff development plan

Project Plan Structure

- Introduction
- Project organisation
- Risk analysis
- Hardware & s/w resource requirements

- Work breakdown
- Project schedule
- Monitoring & reporting mechanism.

Lab 5: What are the four different attributes which all s/w product should have? Suggest four other attributes that may sometimes be significan

2) Reliability & safety are related but distinct dependability attributes. Describe the most importan distinction between these attributes & explain why it is possible for a reliable system to be unsafe & viceversa.

Ans: Whenever a software is developed the softwar should deliver the required functionality & better performance to the user. The four attributes of software are:

a. Maintainability
b. Dependability
c. Efficiency
d. Usability.

a. Maintainability :-
Whenever a software is built, it should evolve to meet the changing needs of customers. This is the critical attribute because software change is an inevitable consequence of a changing business environment.

b. Dependability :-
All the software must be trustworthy. It also includes reliability, security & safety. Whenever a software is developed, it should not cause physical or economic damage in the event of system failure.

c. Efficiency :-
Software should have a proper processing time, memory utilisation. It should be efficient to the user. Also software should not make wasteful use of system resources such as memory & processor cycles.

d. Usability :-
Software must be easy to use to whom it is designed. Also, a good software should have

an appropriate user interface and adequate documentation.

Any other four attributes are :

1. Quality :
2. Secure
3. More interactive with the user
4. Properly structured.

Q no 2) The dependability of a computer system is a property of the system that equates to its trustworthies. Trustworthy means that the system will operate as they expect and the system will not 'fail' in normal use.

As reliability & safety are the attributes of dependability they are different from each other. Reliability of a system is the probability, over a given period of a time, that a system will correctly deliver services as expected by the user and safety of a system is a judgement of how likely it is that the system will cause

damage to people or its environment.

Even if the system is reliable, if the system causes problems or always cause a damage to the people then we cannot say a system is a reliable system. A system should be reliable in that it should conform to its specification & also operate without failures. It may incorporate fault tolerance features so that it can provide continous service even if fault occur. Software may still malfunction & cause system behaviour, which results in an accident.

safety

Checked by
Aruna
7th Dec 2012

Activity Organisation :

- Activities in a project should be organised to produce tangible outputs for management to judge progress.
- Milestones are the end point of a process activity.

diff type of report - milestone & deliverable
each phase report customer

- Deliverables are project results delivered to customers.
- The waterfall process allows for the straight forward defⁿ of progress milestones.

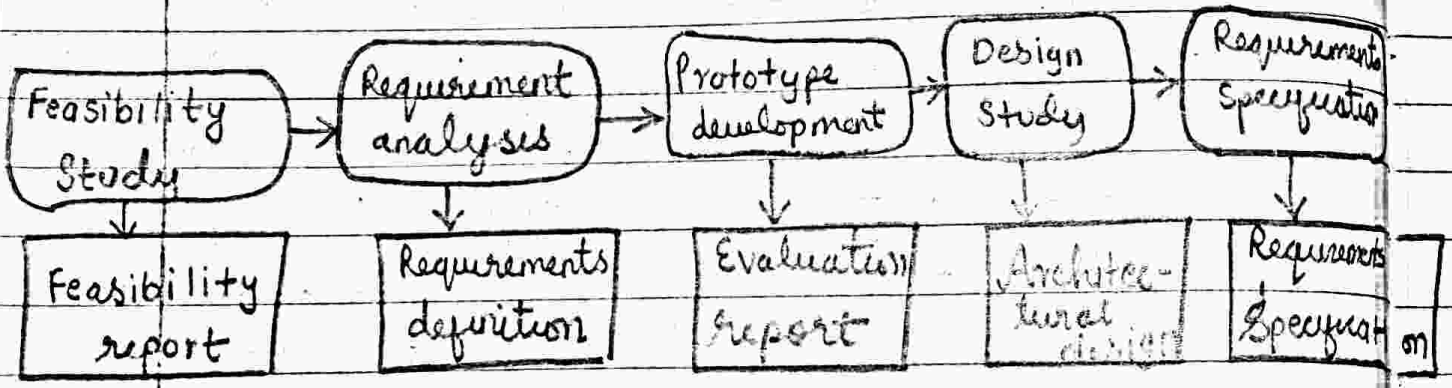


Fig. Milestones in Requirements Engineering Process.

Project Scheduling:

- Split project into task & estimate time & resources required to complete each task.
- Organise tasks concurrently to make optimal use of workforce.
- Minimize task dependencies to avoid delays caused by one task waiting for another to complete.
- Dependent on project managers intuition & experience.

The ability to understand something immediately

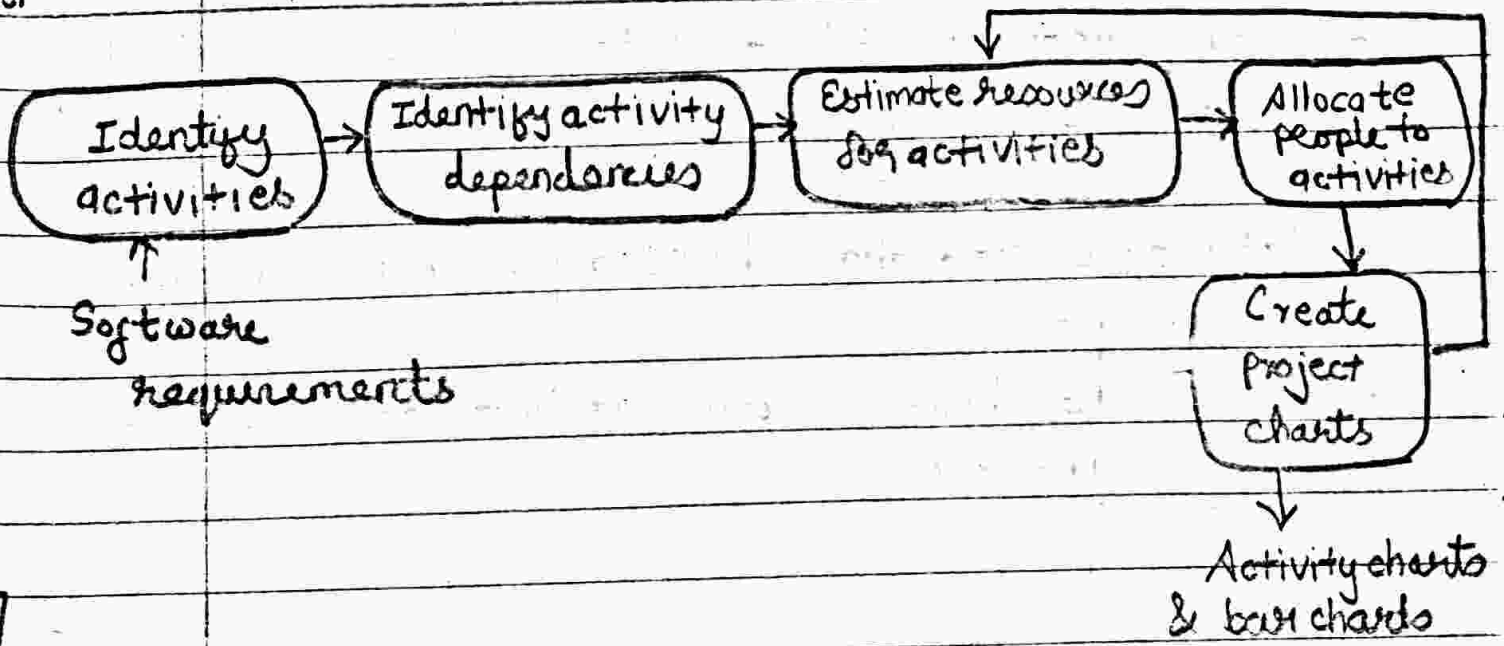


Fig. The project scheduling process

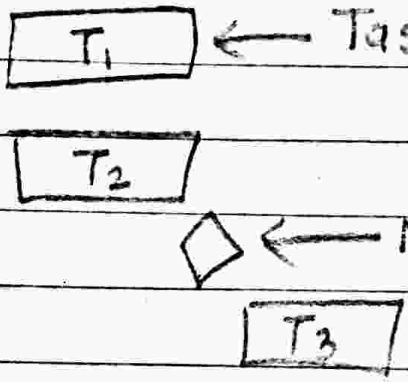
Scheduling Problems :

- Estimating the difficulty of problems & hence the cost of developing a solution is hard.
- Productivity is not proportional to the no of people working on a task.
- Adding people to a late project makes it later because of communication overheads.
- The unexpected always happens. Alway allow contigency in planning.

Bar charts and activity networks :

- Graphical notation used to illustrate the project schedule.

- Show project breakdown into tasks. Tasks should not be too small. They should take about a week or two.
- Activity charts show task dependencies & the critical path.
- Bar charts show schedule against calendar time.



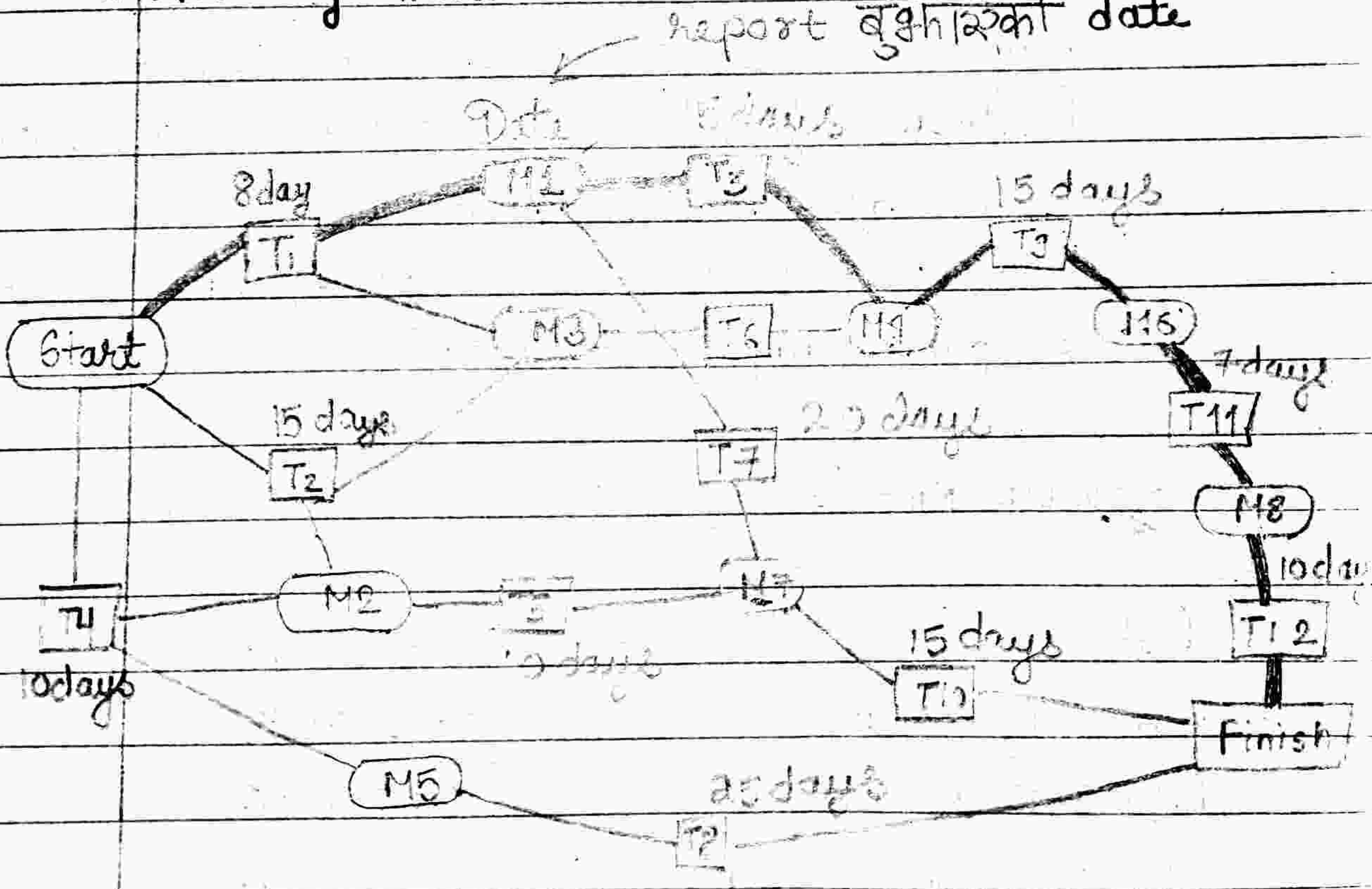
Task durations & dependencies

Task	Duration (days)	Dependencies
T1	8	
T2	15	
T3	15	T1 (M1)
T4	10	T2, T4 (M2)
T5	10	T1, T2 (M3)
T6	5	T1 (M1)
T7	20	T4 (M5)
T8	25	T3, T6 (M4)
T9	15	T5, T7 (M7)

T10	15	T5, T7(M7)
T11	7	T9(M6)
T12	10	T11(M8)

T2, T4 को काम सक्के पछि मात्र T5 start हुन्छ।

Activity network:

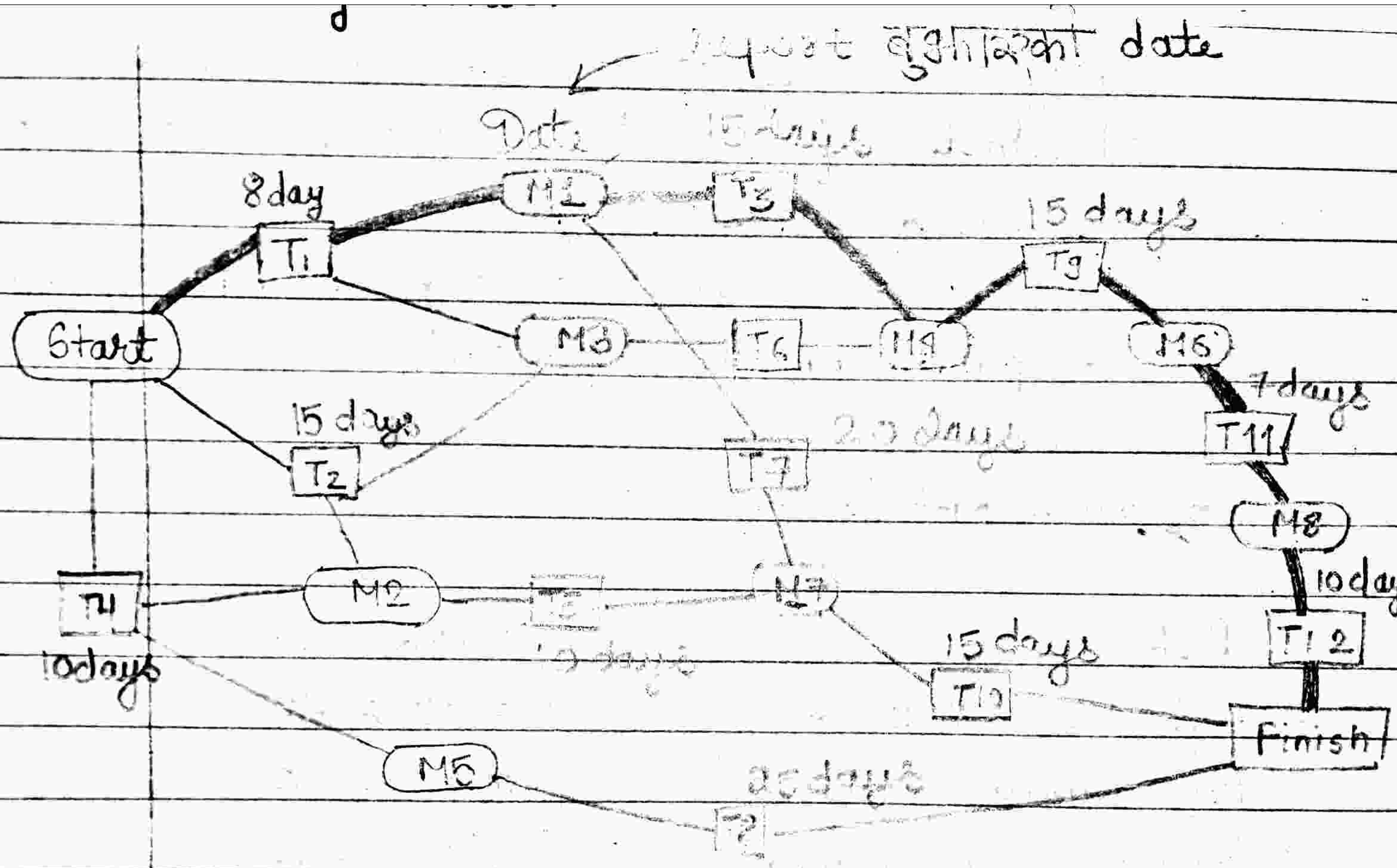


← critical path (longest path from start to finish)

→ if late produced results then we can delay as well.

non critical path मा ... critical path जस्तै लाग्न सक्छ।

ncies



← critical path (longest path from start to finish)

→ if late produced result then we can deny at will.

non critical path नै बढेमा critical path आउन बन सक्छ।

Risk Management :

- Risk Management is concerned with identifying risks and drawing up plans to minimize their effect on a project.

- A risk is a probability that some adverse circumstance will occur.
  - Project risks affect schedule or resources.
  - Product risks affect the quality or performance of the software being developed.
  - Business risks affect the organization developing or purchasing the software.

Software Risks

	Risk	Risk type
1.	Staff turnover	Project risk
2.	Management change	Project risk
3.	Hardware unavailability	Project risk
4.	Requirement change	Project risk & product risk
5.	Specification delays	Product risk & project risk
6.	Size underestimate	Project risk & product risk

7) CASE tool underperformance — Product risk
8) Technology change — Business risk
9) Product competition — Business risk

The risk management process :-

Q no 3 Model

Steps of risk mgmt

1. Risk identification :-
Identify project, product and business risks.

2. Risk analysis :-
Access the likelihood and consequences of these risks.

3. Risk planning :-
Draw up plans to avoid or minimize the effects of the risk.

4. Risk Monitoring :-
Monitor the risks throughout the project.

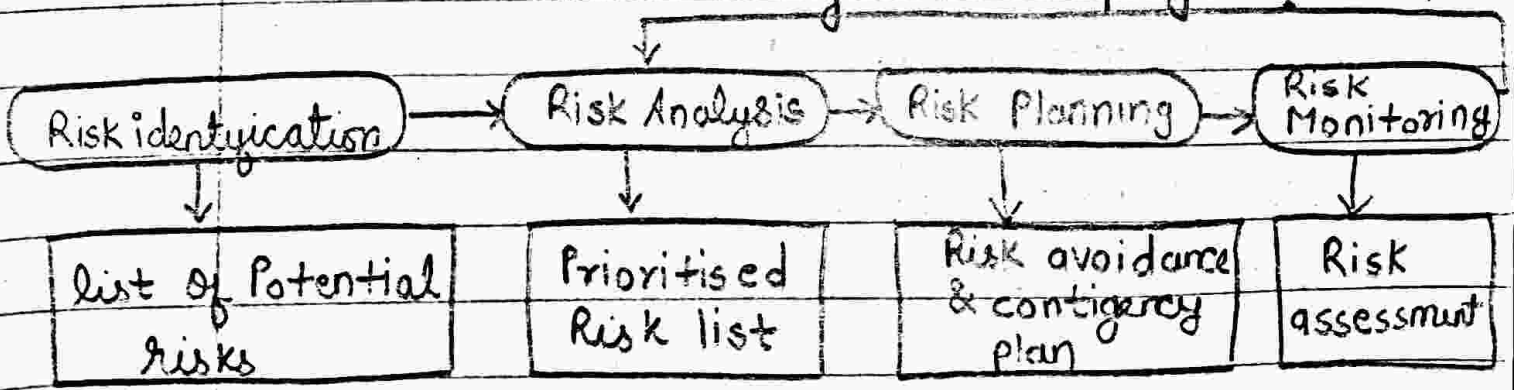


Fig. Risk Management Process

Risk identification

- Technology risks
- People risks
- Organisational risk
- Requirements risk
- Estimation risk.

Risk analysis :

- Assess probability and seriousness of each risk.
- Probability may be very low, low, moderate, high or very high.
- Risk effects might be catastrophic, serious, tolerable or insignificant.

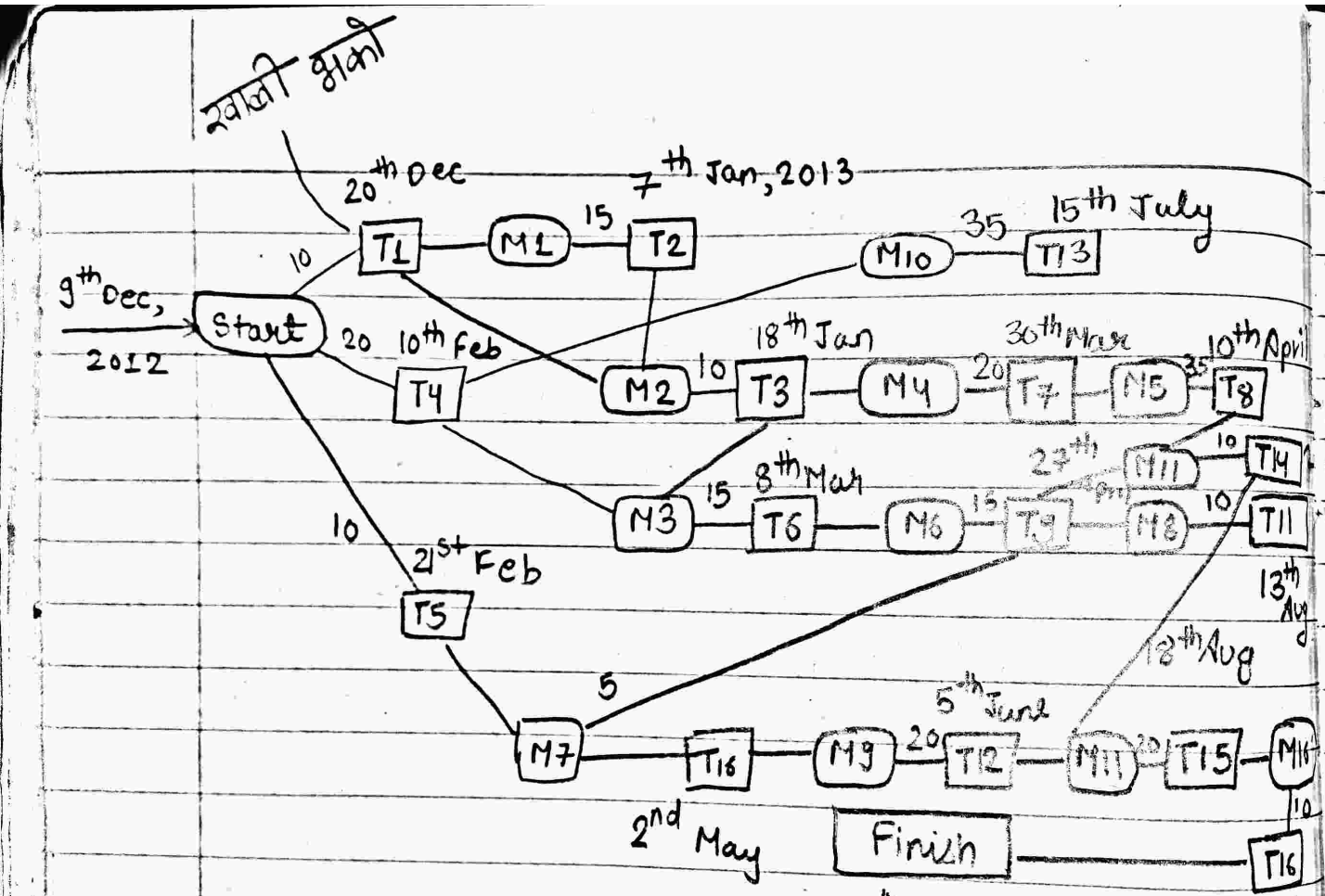
Risk Planning :

- Consider each risk and develop a strategy to manage that risk.
- Avoidance strategies
- Minimisation strategies
- Contigency plans

Risk Monitoring :

- Assess each identified risks regularly to decide whether or it is becoming less or more probable.
- Also assess whether the effects of the risk have changed.
- Each key risk should be discussed at management process meeting.

Task	Duration (days)	Dependencies	
T_1	10		
T_2	19	T1	M1
T_3	10	T1, T2	M2
T_4	20		
T_5	10		
T_6	15	T3, T4	M3
T_7	20	T3	M4
T_8	35	T7	M5
T_9	15	T6	M6
T_{10}	5	T5, T9	M7
T_{11}	10	T9	8
T_{12}	20	T10	9
T_{13}	35	T13, T9	10
T_{14}	10	T8, T9	11
T_{15}	20	T12, T14	12
T_{16}	10	T15	13

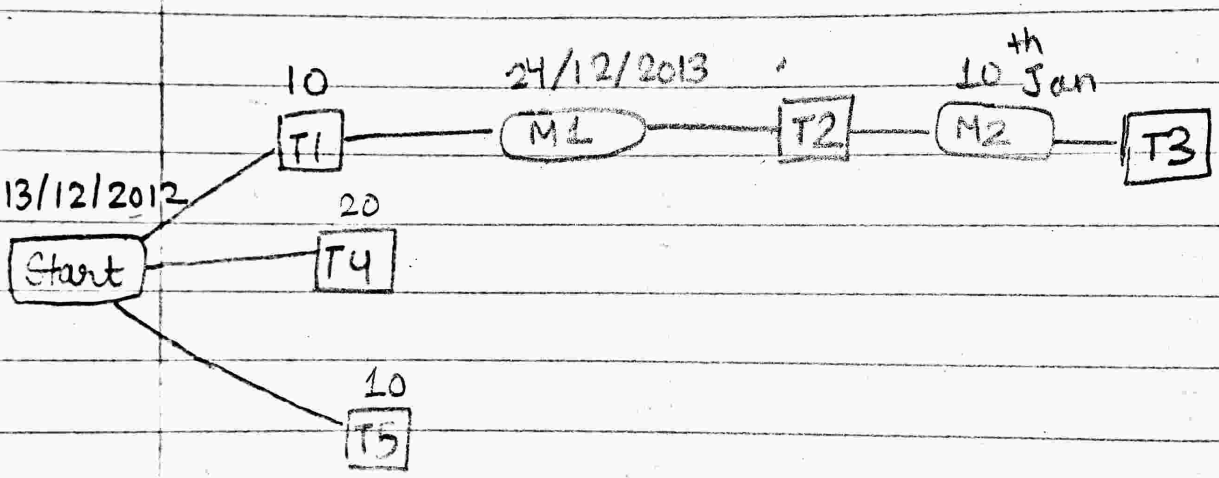

खाली धाको
20th Dec
7th Jan, 2013
15th July
9th Dec, 2012
Start
10
20
10th Feb
18th Jan
30th March
10th April
35
10
20
15
8th Mar
27th
10
6th July
10
21st Feb
13th Aug
18th Aug
5
5th June
20
20
2nd May
Finish
29th Aug, 2013
Working day =

2nd May

Finish — Tk

29th Aug, 2013

Working day =



Unit 2

Functional — User, System
Non-Functional — User, System

Simple sentence (system model बनाउने)
bar / pie chart

2.1 # Software Requirements ÷

6th July

- The process of establishing the services that the customer requires from a system & the constraint under which it operates & is developed.

- The requirement themselves are the description of the system services and constraints that are generated during the requirements engineering process.

What is a requirement?

→ It may range from a high-level abstract (user को इच्छा) statement of a service of a system constraint to a detailed mathematical functional specification.

→ This is inevitable (unavoidable) as a requirements may serve a dual function.

- May be the basis for a bid for a contract - therefore must be open to interpretation. requirement अनुसार धेरै बुझ्ने खालको हुनुपर्छ।

- May be basis for the contract - itself therefore must be defined in detail.
- Both these statements may be called requirements

Types of requirements :

* User requirement :
Statements in natural language plus diagram of the services the system provides & its operational constraints. written for customers.

* System requirement :
A structured document setting out detailed description of the system services. Written as a contract between client & contractor.

* Software specification :
A detailed software description which can serve as a basis for a design or implementation. Written for developers.

Requirement imprecision :

- Problems arise when requirements are not precisely

stated.

- Ambiguous requirements may be interpreted in different ways by developers & users.
- Consider the term 'appropriate viewers'.
  - User intention: Special purpose viewers for each different document type.
  - Developers interpretation: Provide a text viewer that shows the contents of the documents. (library of author art, date) এর মত

Requirements Completeness and consistency :

- In principle requirement should be both complete and consistent.
- Complete : They should include descriptions of all facilities required.
- Consistent : There should be no conflicts or contradiction in the descriptions of the system facilities.
- In practice, it is impossible to produce a complete & consistent requirements document.

Requirements Readers :

Requirements Readers :

User requirements →	Client managers System end-user Client engineer Contractor managers System architects
System requirements →	System end users Client engineers System architects Software developers
Software design specification →	Client engineers System architects Software developers

Functional and non-functional requirements :

- Functional requirements :
  Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.

- Non-functional requirements :

Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process etc.

- Domain requirements :
Requirements that come from the application domain of the system & that reflect characteristics of that domain.
Domain यदि आरम्भीलाई functional वि non-functional req. भनिन्छ।

Lab Assignment #06

1. Explain why the intangibility of a s/w poses special problems for s/w project management?
2. Explain why the best programmers do not always make the best s/w managers.
3. What is the critical distinction between milestones & deliverables?

Functional requirements:

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

Examples:

- The user shall be able to search either all of the initial set of databases or select a subset from it.
- Every order shall be allocated a unique identifier which the user shall be able to copy to the account's permanent storage area.

Non-functional requirements :-

- Define system properties and constraints eg :- reliability, response time & storage requirements.
- Process requirements may also be specified mandating a particular CASE system, programming

S/w इस standard हुनुपर्ने org^n z के चाहिन्छ भन्ने पहिलाको s/w ठिक छ कि छैन भन्ने हो।

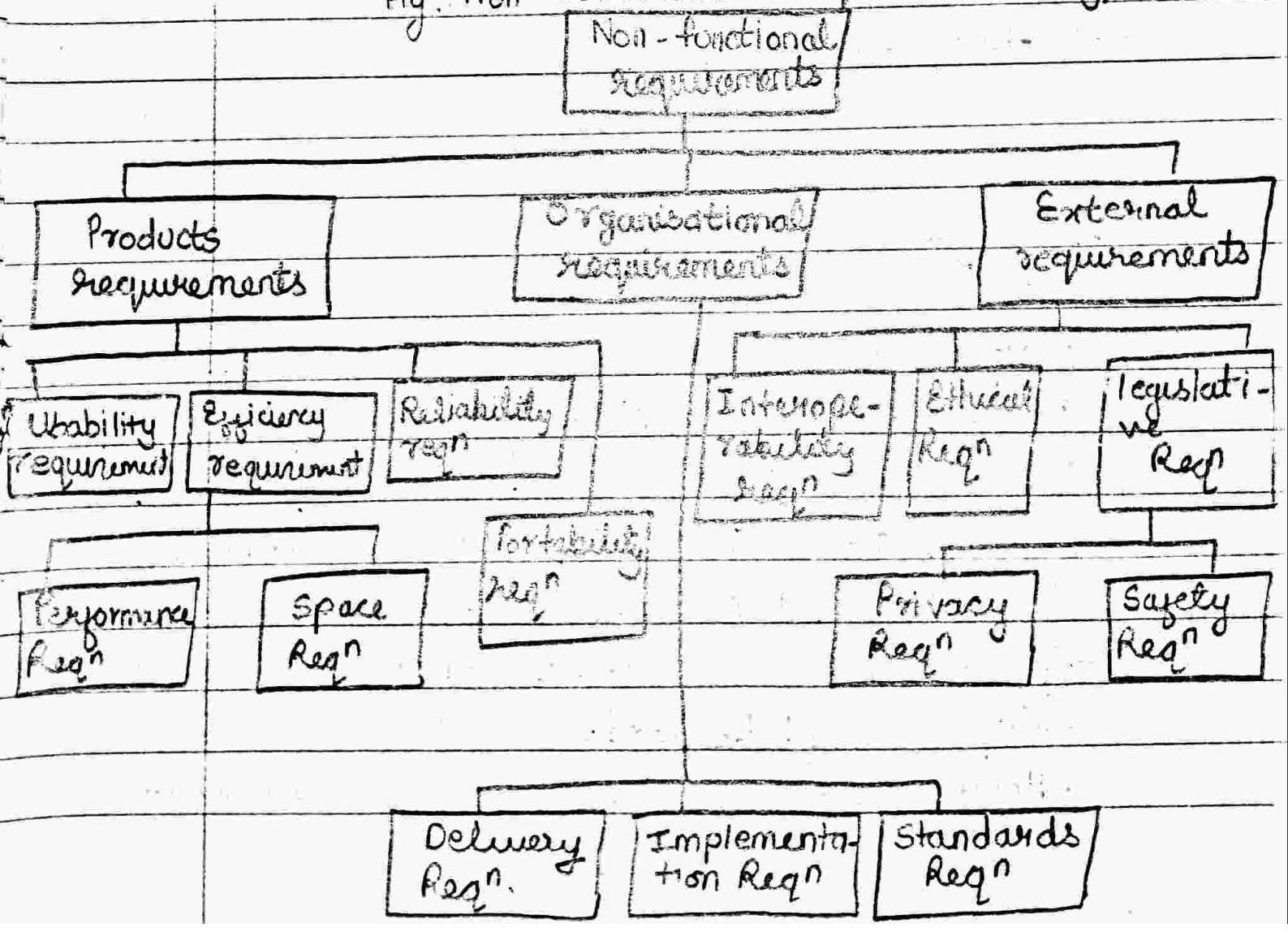
Interoperability - अरु सँग interaction हुनुपर्ने language or development method.

- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.

Classification:

1) Product requirements
2) Organisational requirements
3) External requirements

Fig. Non-functional requirements types



46
V2
9/20

Goals & requirements :

- Non-functional requirements may be very difficult to state precisely & imprecise requirements may be difficult to verify.
- Goals are helpful to developers as they convey the intentions of the system users. (prespecified)
- Goal ÷ A general intention of the user such as ease of use. (Goal usable उद्देश्य)
- Verifiable non-functional requirements ÷ A statement using some measure that can be objectively tested.
(Usability, Safety all are non-functional req^n)

Requirements interaction ÷

- Conflict between different non-functional requirements are common in complex systems.
- Spacecraft system :
  - To minimise (space) weight, the number of separate chips in the system should be minimised.
  - To minimise power consumption, lower power chips should be used.
  - However, using low power chips may mean

that more chips have to be used.
- which is the most critical requirements?

Domain requirements :

- Derived from the application domain & describe system characteristics & features that reflect the domain.
- May be new functional requirements, constraints on existing requirements or define specification computation.
- If domain requirements are not satisfied, the system may be unworkable.

Problems:
1. Understandability
2. Implicitness

User requirements :

- Should describe functional & non-functional requirements so that they are understandable by system users who don't have detailed technical knowledge.

- User requirements are defined using natural language, tables & diagrams.

Problems with natural language :

- lack of clarity
- Requirements confusion
- Requirements amalgamation.

• Guidelines :

1. Use diagramatic, grammar ইত্যাদি correct
2. Algorithm follow

The Requirements document :

→ The requirements document is the official statement of what is required by the system developers.
→ Should include both a definition and a specification of requirements.
→ It is now a design document. As far as possible, it should set of WHAT, the system should do rather than HOW it should be done.

Uses of a requirements document

System customers	Specify the requirements and read them to check that they meet their needs. They specify changes to the requirements.
Managers	Use the requirements document to plan a bid for the system & to plan the system development process.
System engineers	Use the requirements to understand what system is to be developed.
System test engineers	Use the requirements to develope validation test for the system.
System maintainence engineers	Use the requirements to help understand the system and the relationship between its parts.

IEEE Requirements Standard

VIP

- Introduction
- General description
- Specific requirements
- Appendices
- Index
- This is a generic structure that must be instantiated for specific systems.

Requirements document structure :

- Introduction
- Glossary
- User requirements definition
- System architecture
- System requirements specification
- System Models
- System evolution
- Appendices (इसका meaning)
- Index

relation मा पर्ने मान्छेहरु लाई stakeholder भनिन्छ (end users, engineers)

Requirements Engineering Process:
(requirement refine गर्नुपर्ने)

- The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements.

Q no 4. Model Q. However, there a number of generic activities common to all processes.

- Requirement elicitation (information gather)
- Requirements analysis के फाइदा, कमजोरी
- Requirements validation Requirement अनुसार valid छ भनी निश्चित
- Requirements management

Project लाई effect पार्ने

are describe

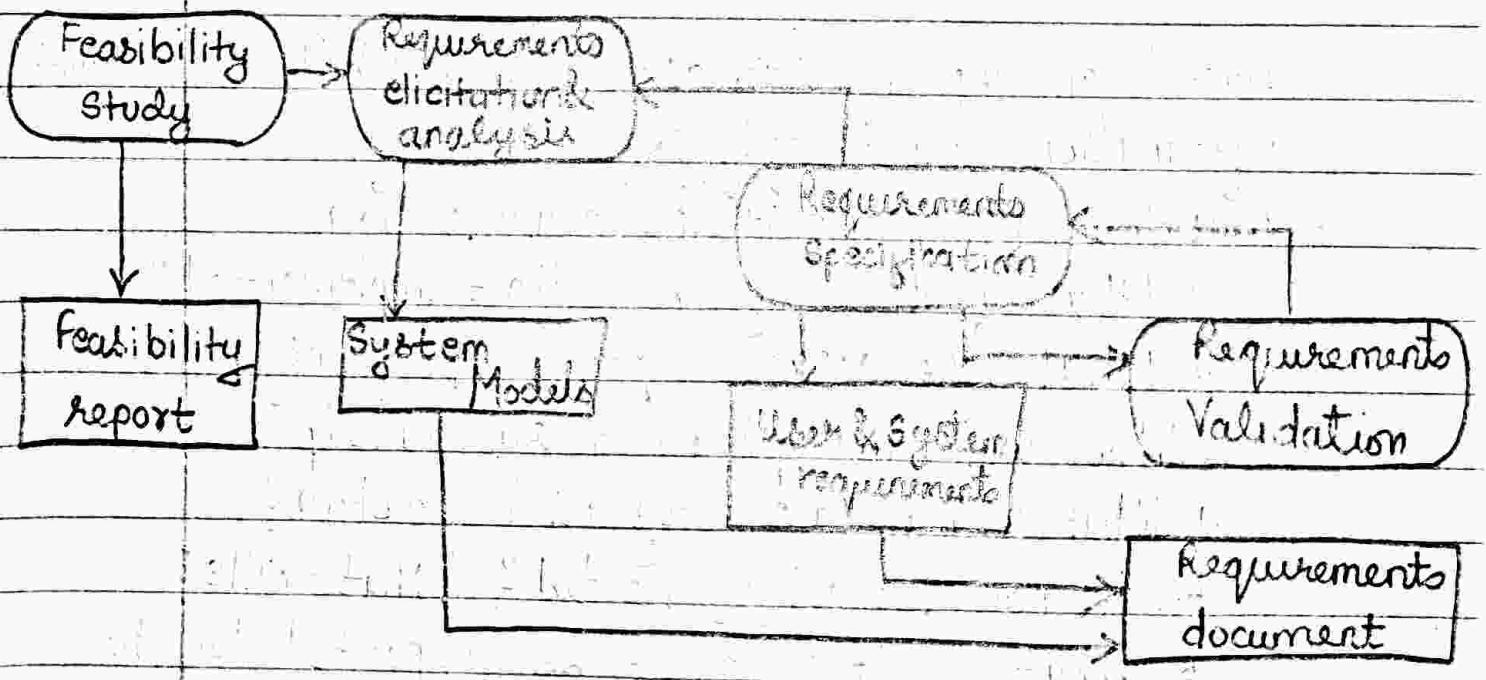


Fig. Requirements Engineering Process

Study phase

Feasibility Studies ÷ (पहिला हुन्छ कि दुईवटा हुनेछ)

Q no 3 2068 *

- A feasibility study decides whether or not the proposed system is worthwhile.
- A short focused study that checks
  - If the system contributes to organisational objectives.
  - If the system can be engineered using current technology and within budget.
  - If the system can be integrated with other systems that are used.

Requirement ela

Feasibility study implementation ÷

- Based on information assessment (what is required) information collection & report writing.
- Questions for people in the organisation.

question सोधेर

1. What if the system wasn't implemented?
2. What are current process problems?
3. How will the proposed system help?
4. What will be the integration problem?
5. Is new technology needed? What skills?
6. What facilities must be supported by the proposed system?

answer पाउने गरी feasible कि कि छ भनेर पाउँछौं।

Domain मा interact गरेको requirement functional / nonfunctional हुन सक्छ ।

Elicitation and analysis ÷ (यो Phase मा stakeholder को information collect गरिन्छ)

- Sometimes called requirements elicitation or requirements discovery.
- Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.
- May involve end-users, managers, engineers involved in maintainence, domain experts, trade unions etc. These are called stakeholders.

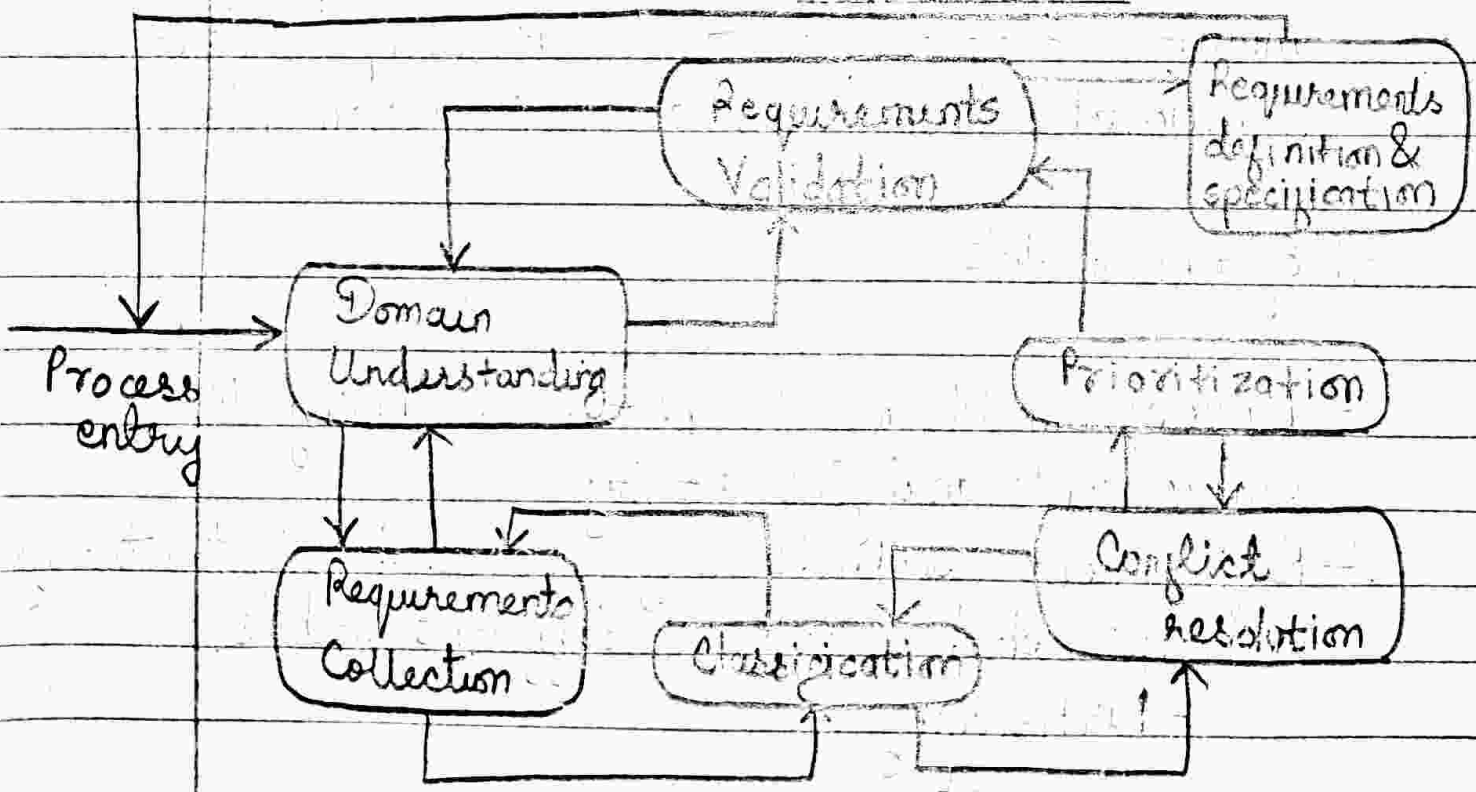


Fig. The requirements analysis process

Assignment

1 Explain ~~w~~
Discuss the problems of using natural language for defining user & system requirements & show using small examples, how structuring natural languages into forms can help avoid some of these difficulties.

2 Suggest how an engineers are responsible for drawing up a system requirements specification might keep track of relationship between functional & non-functional requirements.

System models : (Req^n elicitation & analysis)

- Different models may be produced during the requirement analysis activity.
- Requirement analysis may involve three structuring activities which result in different models
  - Partitioning - [illegible] dependencies
  - Abstraction - interrelated part ~~of~~ is group
  - Projection - [illegible]

Requirement discovery / elicitation

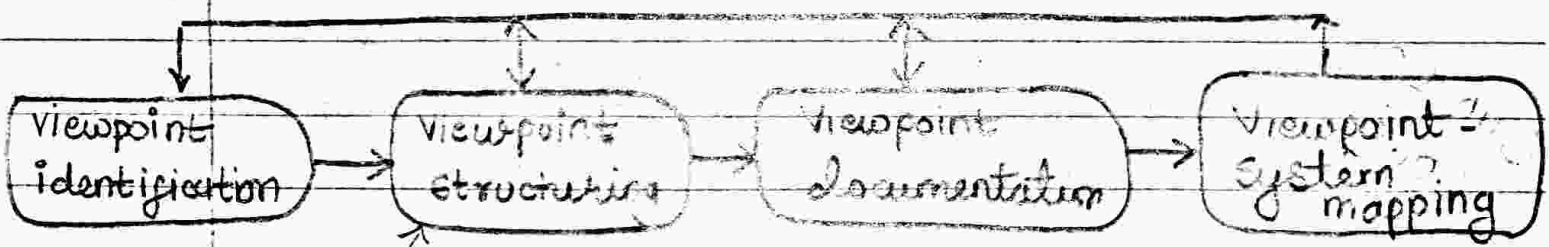
Viewpoint-oriented elicitation

- Stakeholders represent different ways of looking at a problem or problem viewpoints.
- This multi-perspective analysis is important as there is no single correct way to analyse system requirements.

Types of viewpoints :

1) Interactor viewpoints - directly impact पड़ता है i.e. enduser
2) Indirect viewpoints - d mgmt + security staff
3) Domain viewpoints - standard maintain होते हैं

People इन्हें data नहीं interact होते हैं और information gather होते हैं



well-structure form — The VORD method

Interviewing :

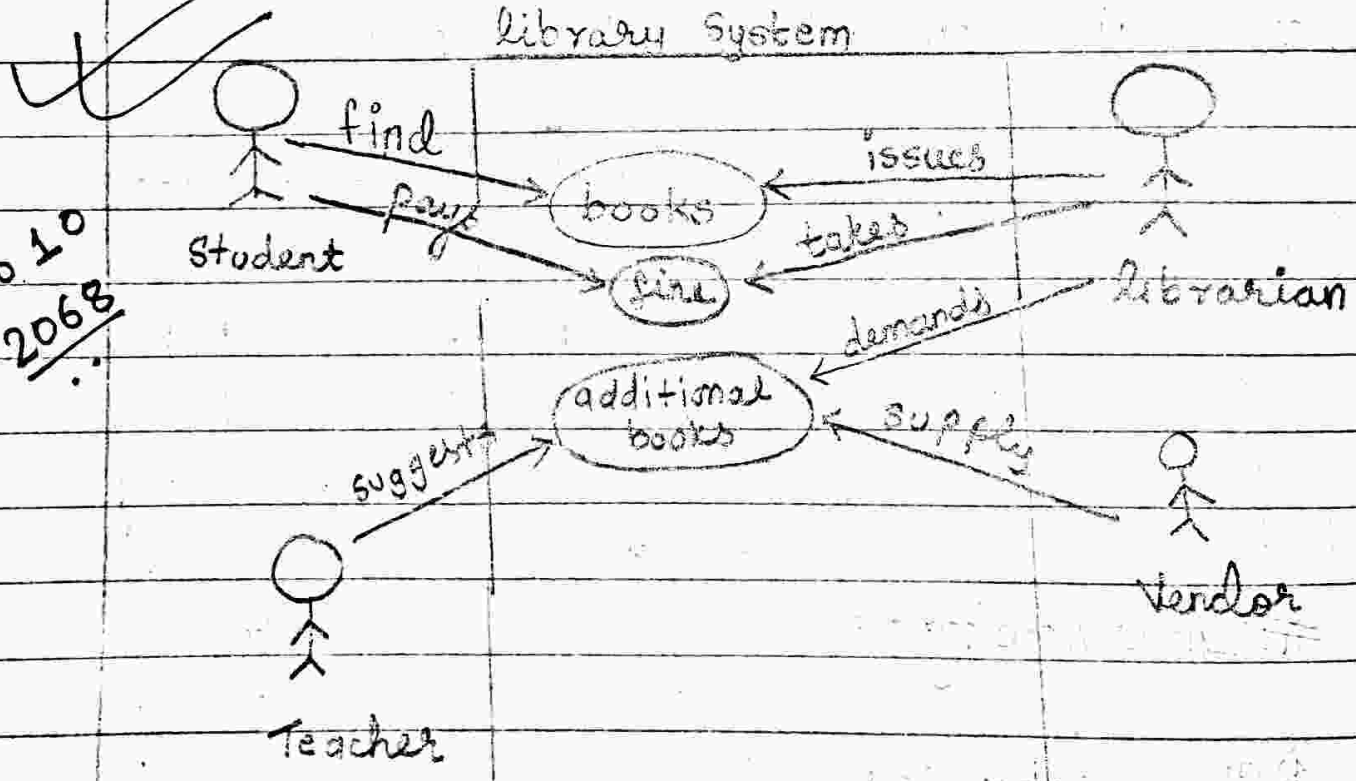
1) Closed interview (predefined - form)
2) Open interview

Scenarios :

- Scenarios are descriptions of how a system is used in practice.
- They are helped requirements elicitation as people can relate to these more rapidly than abstract statement of what they require from a system.
- Scenarios are particularly useful for adding detail to an online requirements description.

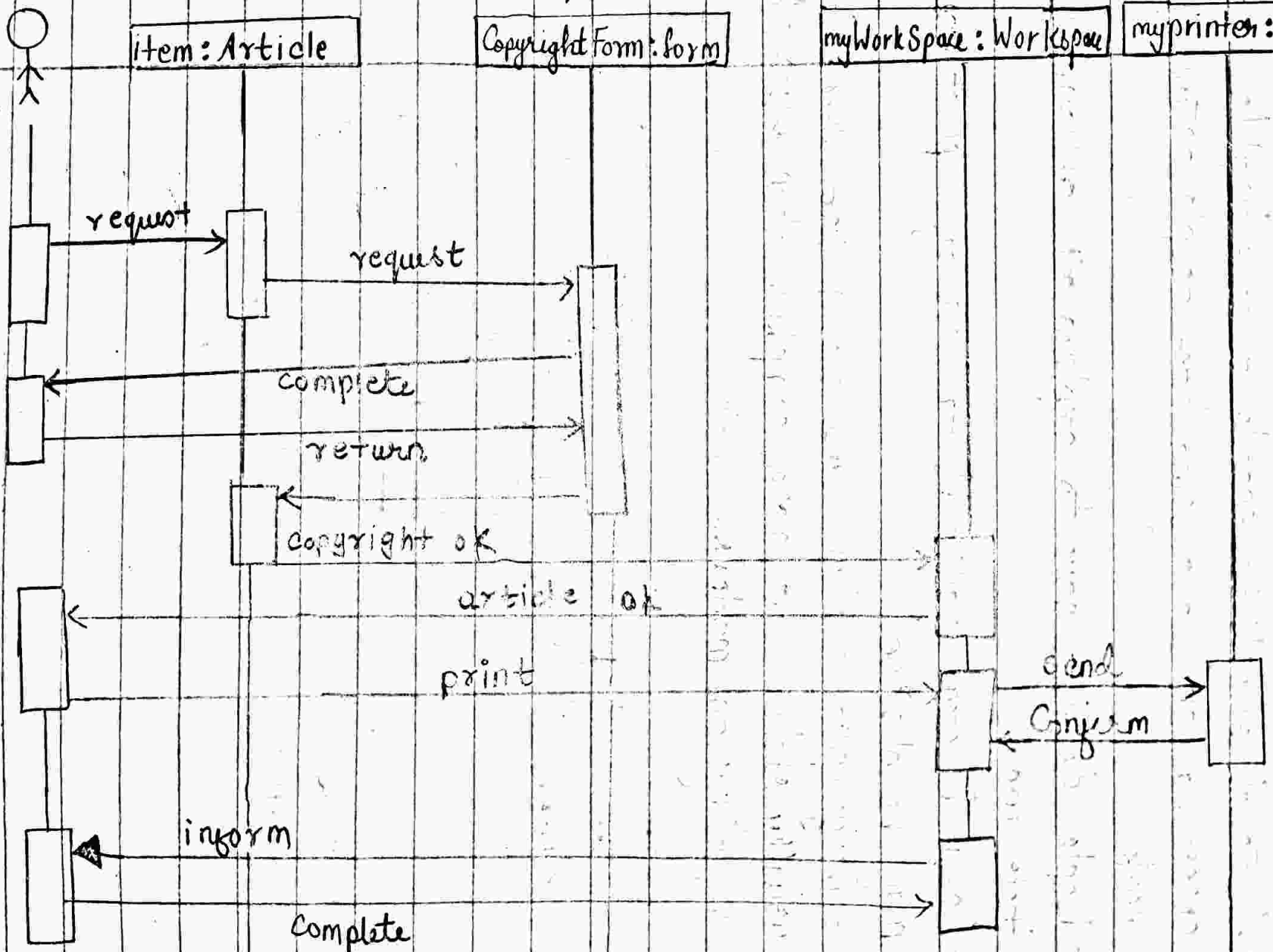
Use Case :

Q no 10
2068



Library मा book download गर्न पाइने कि नै

Sequence diagram:

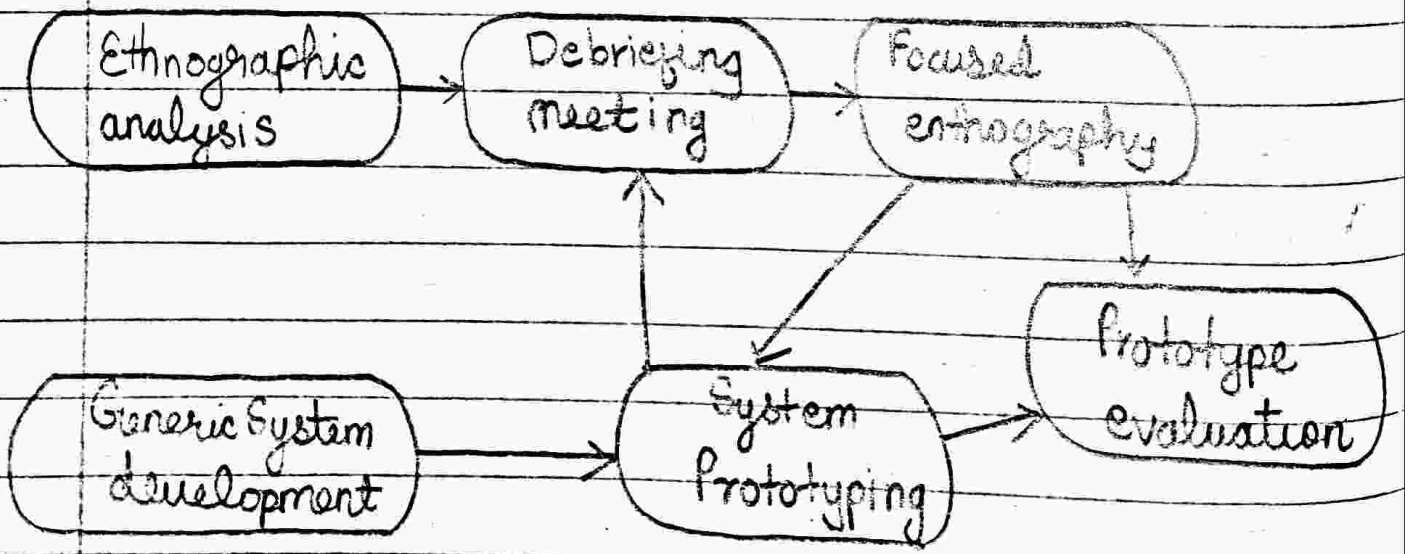


observation

& prototyping

Ethnography ÷ (आपण really collect करायचा data).
(prototype मी help करते)

- A social scientists spends a considerable time observing and analysing how people actually work.
- People do not have to explain or articulate their work.
- Social & organisational factors of importance may be observed.
- Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.



viewpoint, scenario, interview → Requirements

Requirements Validation :

- concerned with demonstrating that the requirements define the system that the customer really wants.
- Requirements error costs are high so validation is very important.

Requirement reviews :

- Regular reviews should be held while the requirements definition is being formulated.
- Both client & contractor staff should be involved in reviews.
- Reviews may be formal or informal. Good communications between developers, customers and users can resolve problems at an early stage.

Requirements Checking :

- Validation :
  Does the system provides the functions which best support the customer's needs?

- Consistency : Are there any requirements conflicts?

Data चाहिँ true हुनु test data मा पर्ने Binary search मा, real d
Test case - Algo implement गर्नु अगाडि assumption गर्नु (only theoritical.

- Completeness :
  Are all functions required by the customer included ?

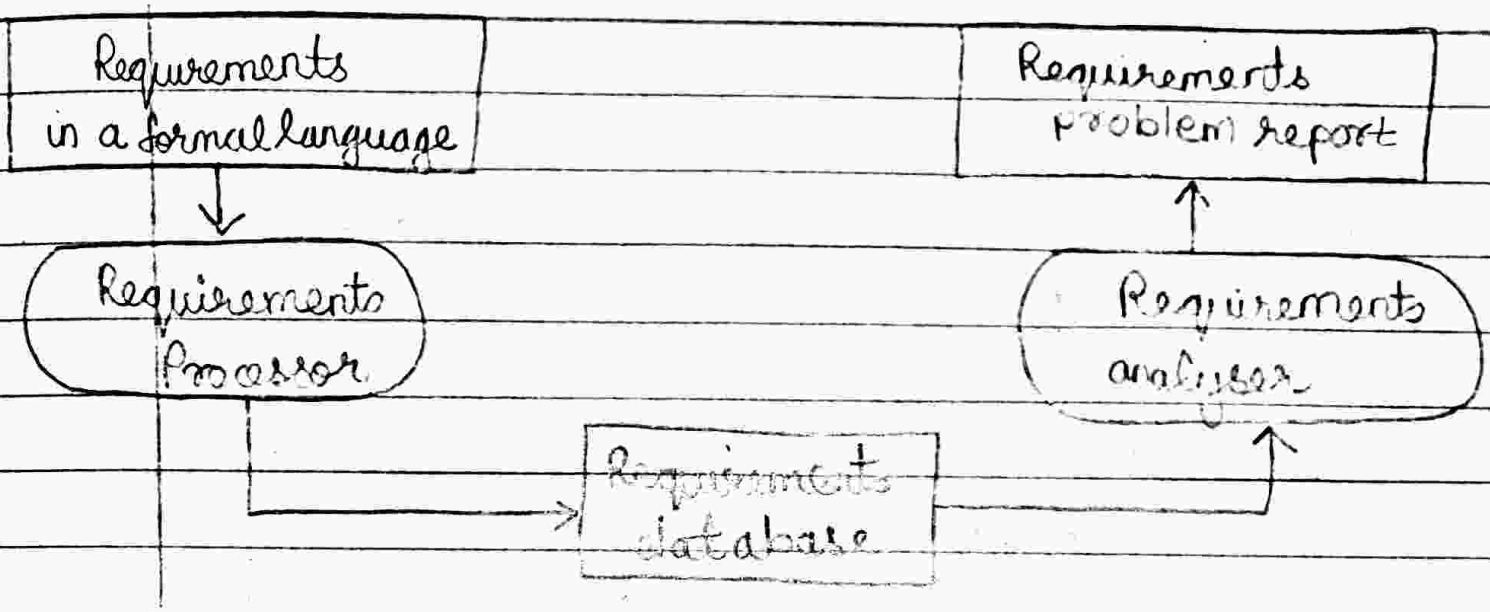
- Realism :
  Can the requirements be ~~elicited~~ implemented given available budget and technology.

- Verifiability :
  Can the requirements be checked ?

Requirements validation techniques :

1. Requirement review
2. Prototyping (executable code ~~बिर्दै~~)
3. Test-case generation
4. Automated Consistency Analysis.

Automated Consistency Checking



Lab 9

Draw sequence diagram and use case diagram ATM represting library system :

Requirements management :

- Requirements management is the process of managing changing requirements during the requirements engineering process & system development.
- Requirements are inevitably incomplete & inconsistent

Requirements evolution :

Req' change — Technology change, new rules & understanding, environment factor

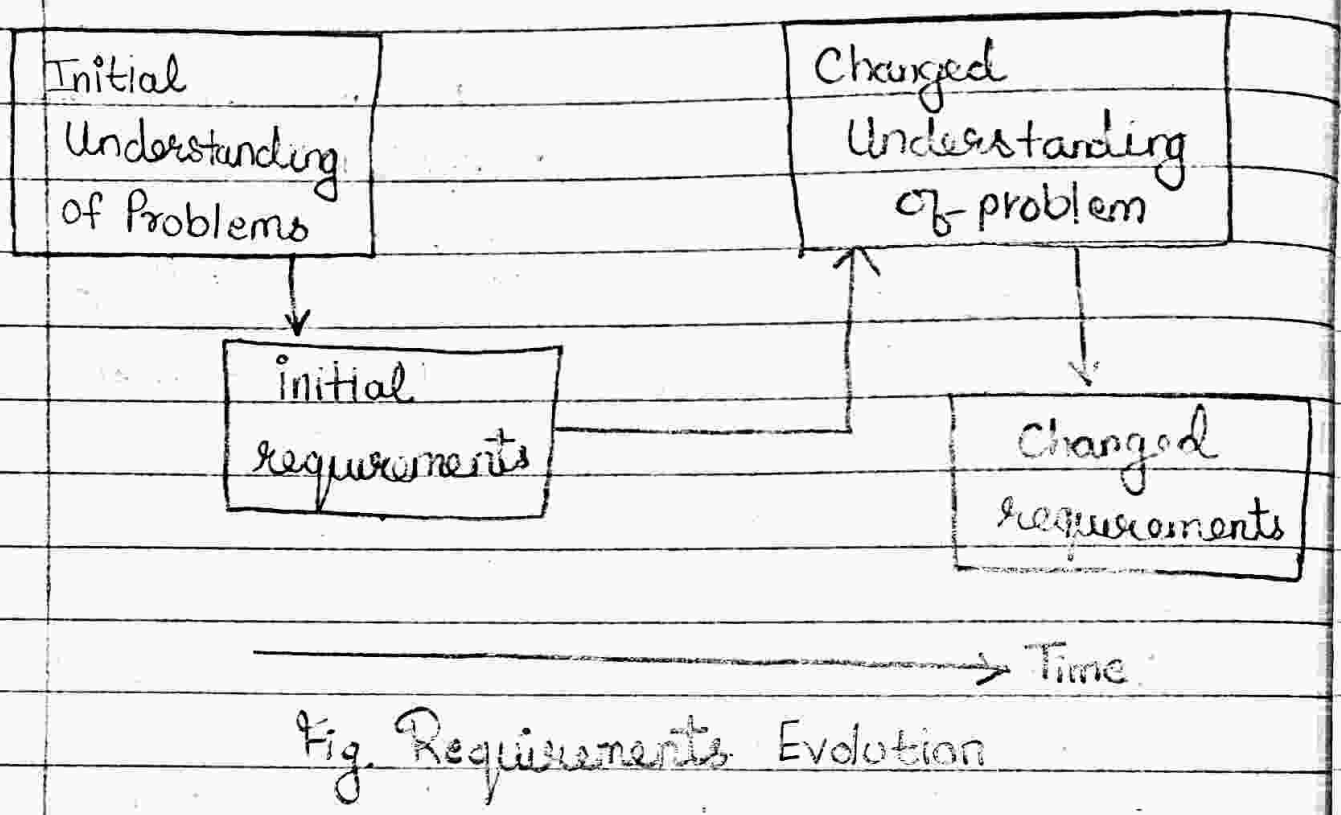


Fig. Requirements Evolution

Enduring and volatile requirements :
(change हुने, नहुने आधारमा req^m)

→ Enduring requirements :
Stable requirements derived from the core activity of the customer organisation. May be derived from domain models.
Hospital को doctor, nurse.

→ Volatile requirements :
Requirements which change during development or when the system is in use.
Rules & Policy of hospital, mgmt को policy change.

Classification of requirements :-

1) Mutable requirements (system को org^n change करने change हुने)
2) Emergent requirements
3) Consequential requirements - नयाँ technology आउँदा change
4) Compatibility requirements - rel^n ship with other companies

Requirements management planning

- Requirements identification
- A change management process
- Traceability policies - sub task बिचको interdependability
- CASE tool support

Traceability :-

- Traceability is concerned with the relationships between requirements, their sources and the system design.

- Source traceability:
Links from requirements to stakeholders who proposed these requirements.

1.1 & 1.2 Architectural design (sub part)

- Requirements traceability:
links between dependent requirements.

- Design traceability:
links from the requirements to design

Traceability Matrix

Reqd id	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2
1.1		D	R					
1.2			D			R		D
1.3	R			R				
2.1			R		D			D
2.2								D
2.3		R		D				
3.1								R
3.2							R	

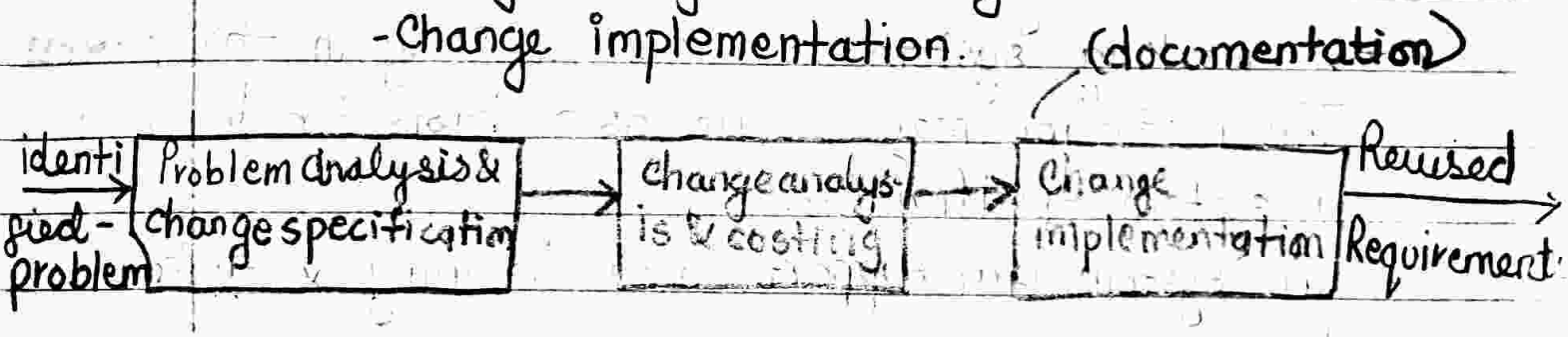
blank ← no rel^n

D – 1.1 is dep^n on 1.2

R – Weaker ~~dependent~~ rel^{n}ship.

Requirement Change management:

- Should apply to all proposed changes to the requirements.
- Principal stages :
  - Problem analysis
  - Change analysis & costing
  - Change implementation



Unit 2.2 SOFTWARE PROTOTYPING

- Prototying is the rapid development of a system.
- The principal use is to help customers & developers understand the requirements for the system.
- Prototyping can be considered as a risk reduction activity which reduces requirements risks.

Throw-away - req^n રાખી બાકી ના ભાગ ફેંકી દેવા

Evolutionary - integrate બનાવ્યા કરવું

Prototyping benefits :

- Misunderstandings between software users and developers are exposed.
- Missing services may be detected & confusing services may be identified.
- A working system is available early in the process.
- The prototype may serve as a basis for deriving a system specification.
- The system can support user training & system testing.

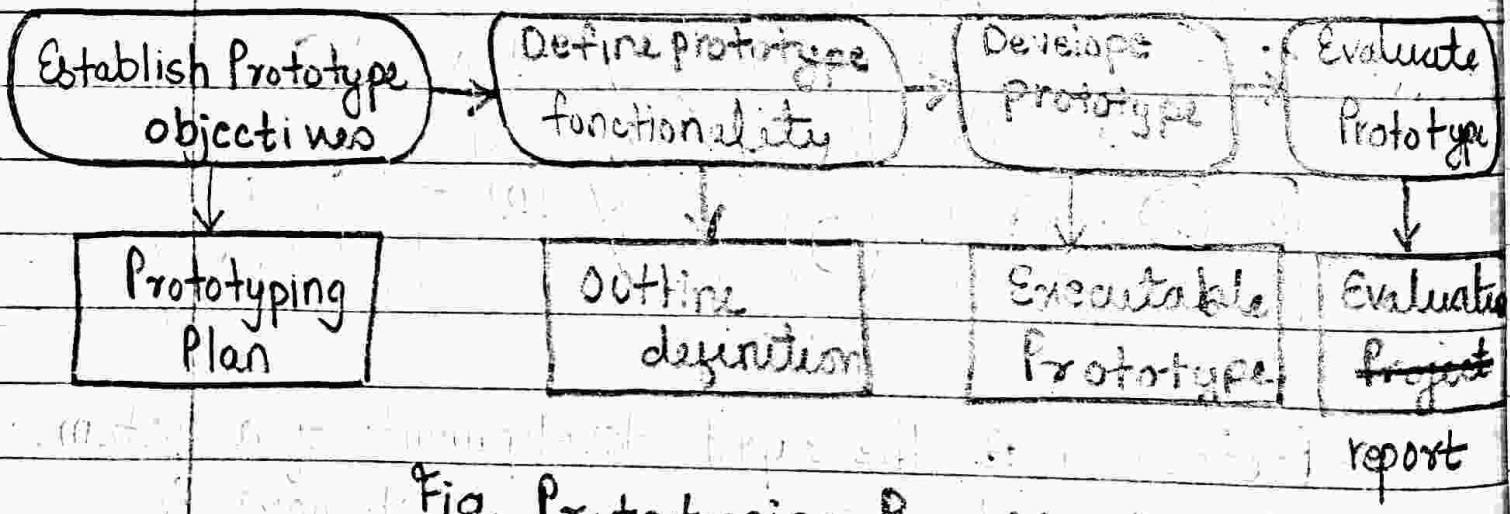


Fig. Prototyping Process

* Formal Specification
ATC मा use coz critical, normal मा use हुँदैन.
- Time / cost बढि हुने हरुमा यो use हुन्छ

* Formal Methods :
* Acceptance of formal method
Use of formal method
Specification technique
Formal specification language
use of formal specification
Development costs
Structure of an algebraic specification
Specification Component
Systematic algeb speci
Specification operation
Opn on a list ADT
List Specification
Primitive opn
Sector specification
Behavioural specification
Schema of a Z schema
An insulin pump

Modelling insulin pump
Schema invariant
dosage schema

Lab assignment 10.

1. Based on your experience with a Bank ATM, draw a data flow diagram modelling the data processing involved when a customer withdraws cash from the machine.

2. Develope a sequence diagram showing the interaction involved when a student registers for a course in a university courses may have limited enrollment, so the registration process must include checks that places are available. Assume that the student accesses an electronic catalog to find out about available courses.

Unit 3: Architectural Design

Introduction

- The design process identifying the sub-system making up a system and the framework for sub system control and communication is architectural design.

- The output of this design process is a description of the software architecture.

Architectural design :

- An early stage of the system design process.
- Represents the link between specification and design process.
- Often carried out in parallel with some specification activities.
- It involves identifying major system component and their communication.

Advantages of architectural design :

- Stakeholder communication
- System analysis
- Large scale reuse.

Imp

Architectural design process :

- System structuring
- Control modelling
- Modular decomposition

Sub-systems and modules

- A sub system is a system in its own right whose operation is independent of the services provided by other sub system.

- A module is a system component that provides services to other components but would not normally be considered as a separate system.

Control model

✓Architectural model / Distribution
✓Static structural model
✓Dynamic process model
✓Interface model
✓Relationship model

Architecture attributes

- Performance
- Security

- Safety
- Availability
- Maintainability

→ # System structuring :

- concerned with decomposing the system into interacting subsystems.

- The architectural design is normally expressed as a block diagram presenting an overview of the system structure.

- more specific models showing how subsystems share data, are distributed and interface with each other may also be developed.

→ # Repository Model :

Q no 6 Model Q.

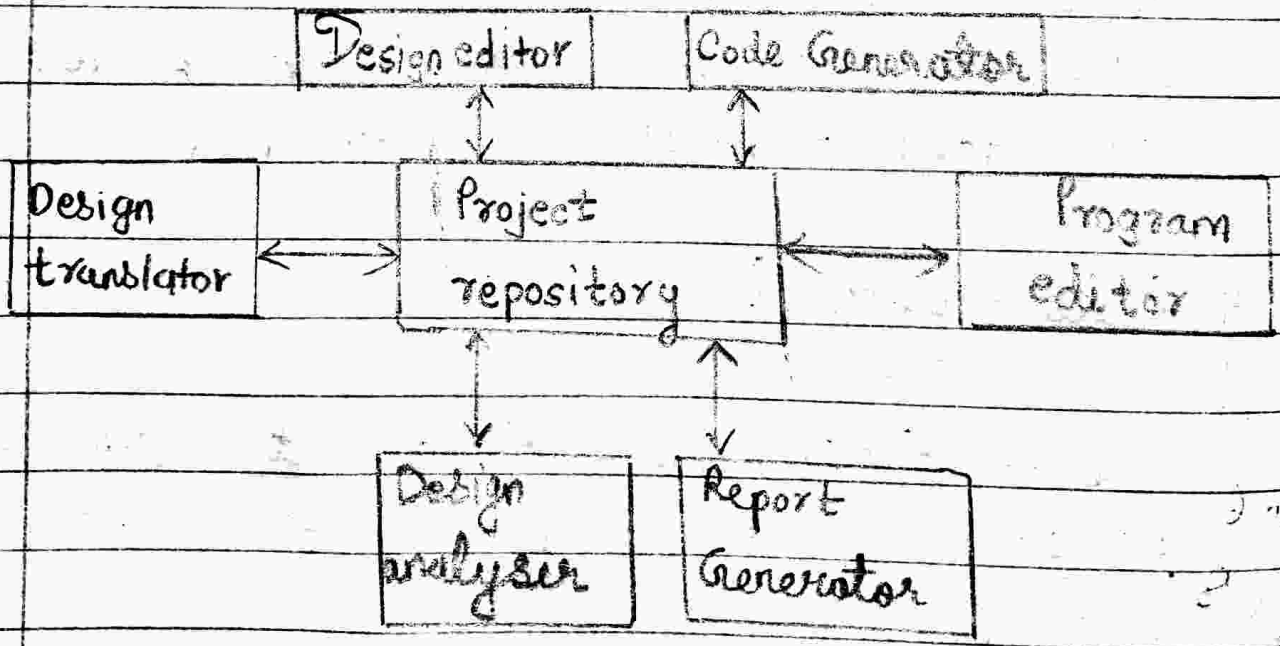
- Sub-systems must exchange data. This may be done in two ways:

- shared data is held in a central database or repository and may be accessed

by all sub-systems.

- Each sub-systems maintain its own database and passes data explicitly to other sub-systems.

- When large amounts of data are to be shared, the repository model of sharing is most commonly used.



CASE toolset architecture

Advantages :

- Efficient way to share large amounts of data.
- Sub-systems need not be concerned with how data is produced.
- Centralised management eg : backup, security etc.
- Sharing model is published as the repository schema.

Disadvantages :

- Sub systems must agree on a repository data model. Inevitably a compromise
- Data evolution is difficult & expensive.
- No scope for specific management policies.
- Difficult to distribute efficiently.

Client - Server Architecture :

- Distributed system model which shows how data and processing is distributed across a range of components.
- Set of stand alone servers which provide specific services such as printing, data management etc.
- Set of clients which call on these services.

- Network which allows clients to access servers.

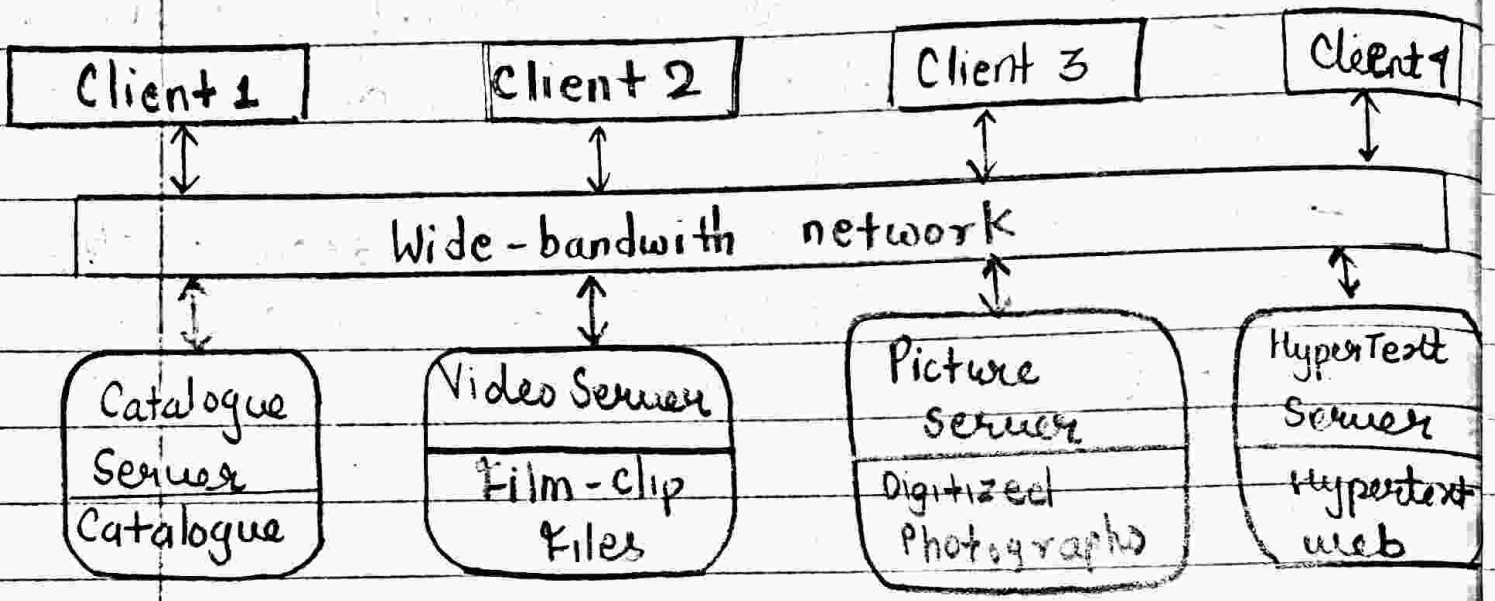


Fig. Film & Picture library

Advantages :

- Distribution of data is straightforward.
- Makes effective use of networked systems. May require cheaper hardware.
- Easy to add new servers or upgrade existing servers.

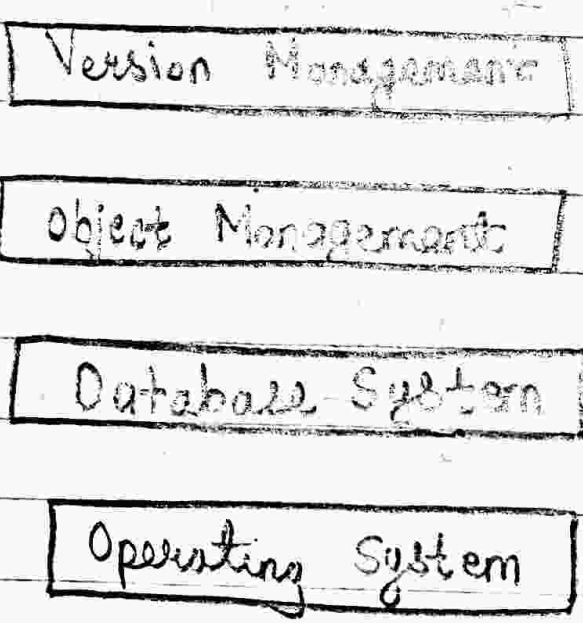
Disadvantages :

- No shared data model so sub-system use different data organisation, data interchange may be Inefficient.
- Redundant management in each server.

- No central register of names and services - it may be hard to find out what servers and services are available.

Abstract Machine Model:

- Organises the system into a set of layers each of which provide a set of services.
- Supports the incremental development of sub-systems in different layers. When a layer interface changes, only the adjacent layer is affected.
- However, often difficult to structure systems in this way.



Version Management System

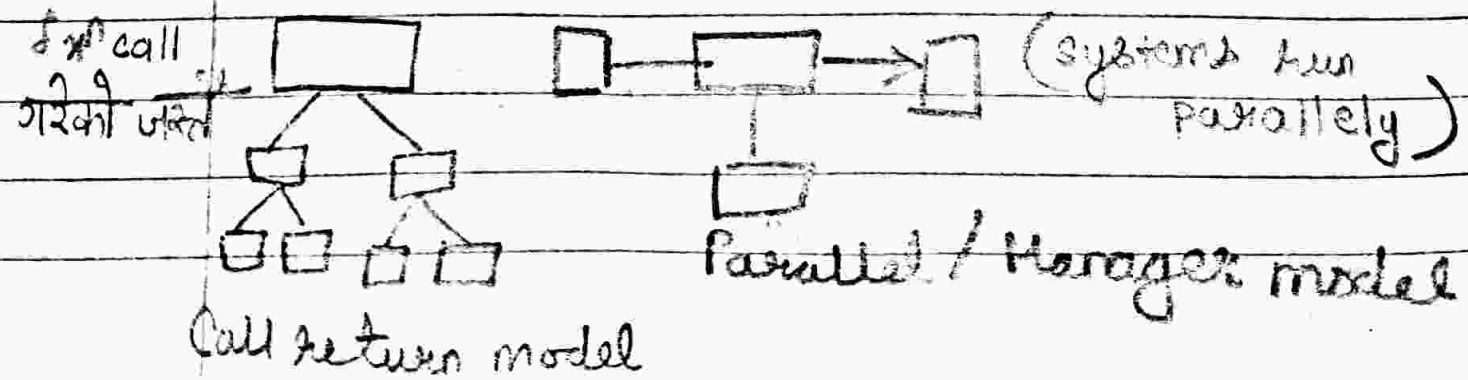
Control Models :

- Are concerned with the control floco between sub systems.
- Centralised control
- Event-based control

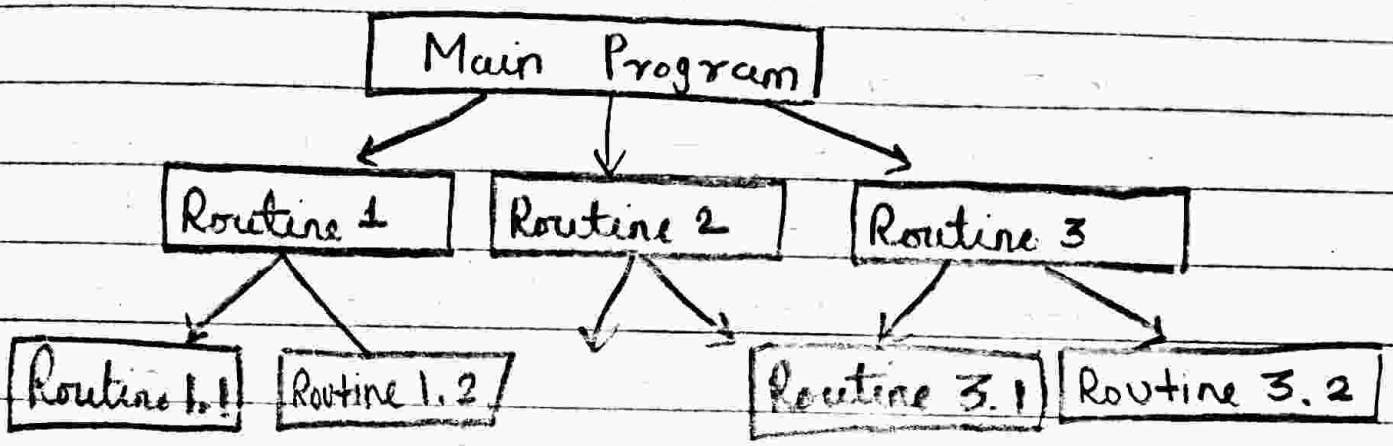
• Centralised control :

- A control sub-system takes responsibility for managing the execution of other sub-systems.
- Call return model : Top down subroutine model where control starts at the top of the subroutine hierarichy and moves downwards. Applicable to sequential system.

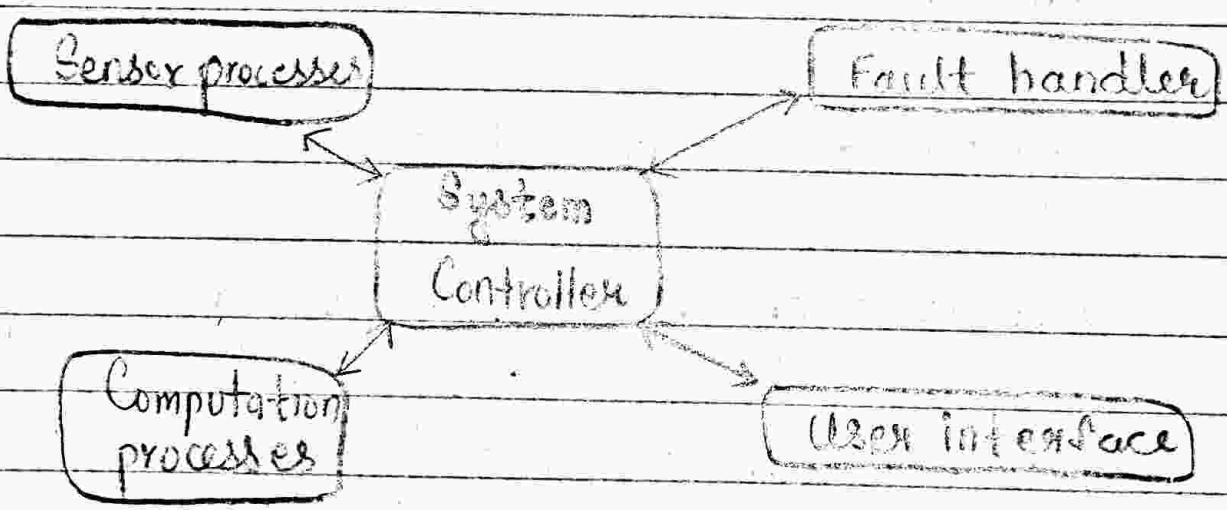
- Manager model :
Applicable to concurrent systems. One system components controls the stopping, starting & coordination of other system processes.



Call - return model.



Real time System Control :



Event - driven model

- driven by externally generated events.
- Two principle event drive models:
1. Broadcast model
2. Interrupt driven model

1. Broadcast model

- effective in integrating subsystems on different computers in a networks.
- subsystems register an interest in specific events. When these occur, control is transferred to the subsystem which can handle the event.
- control policy is not embedded in the event and message handler. Subsystems decide on events of interest to them.
- Subsystems dont know if or when an event will be handled.

2. Interrupt driven model :

- used in real systems where fast response to an event is essential.
- There are unknown interrupt types with a handler defined for each type.
- each type is associated with a memory location and a hardware switch causes transfer to its handler.
- Allows fast response but complex to program and difficult to validate.

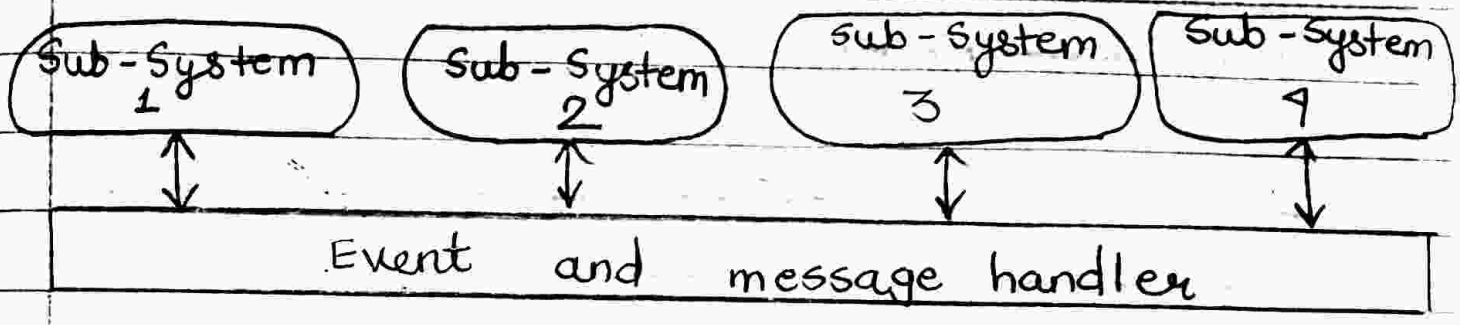


Fig. Broadcast Model

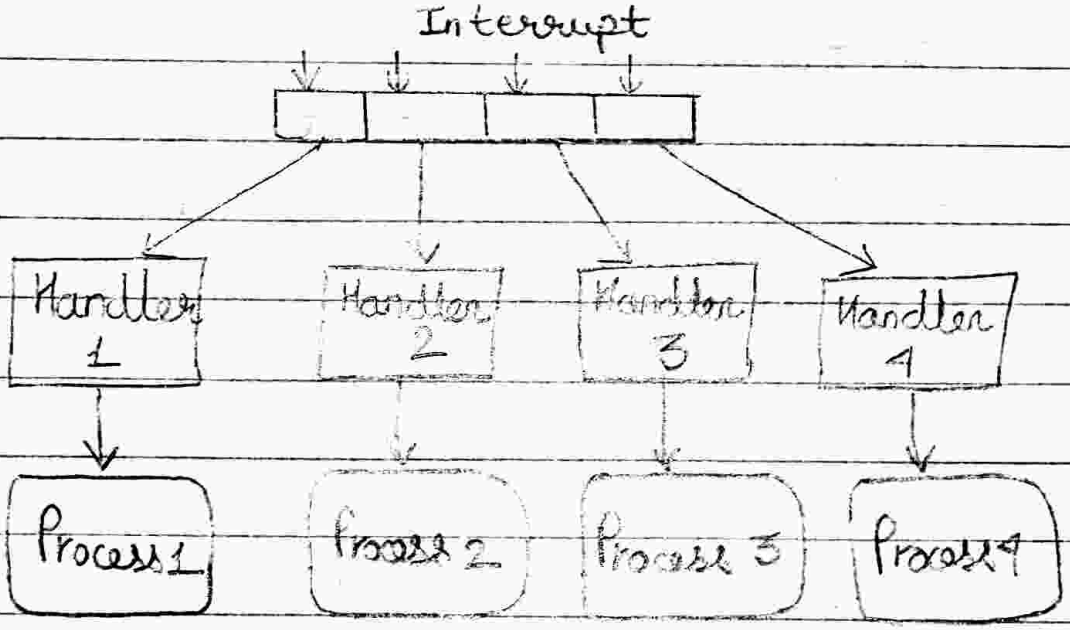


Fig. Interrupt-driven model

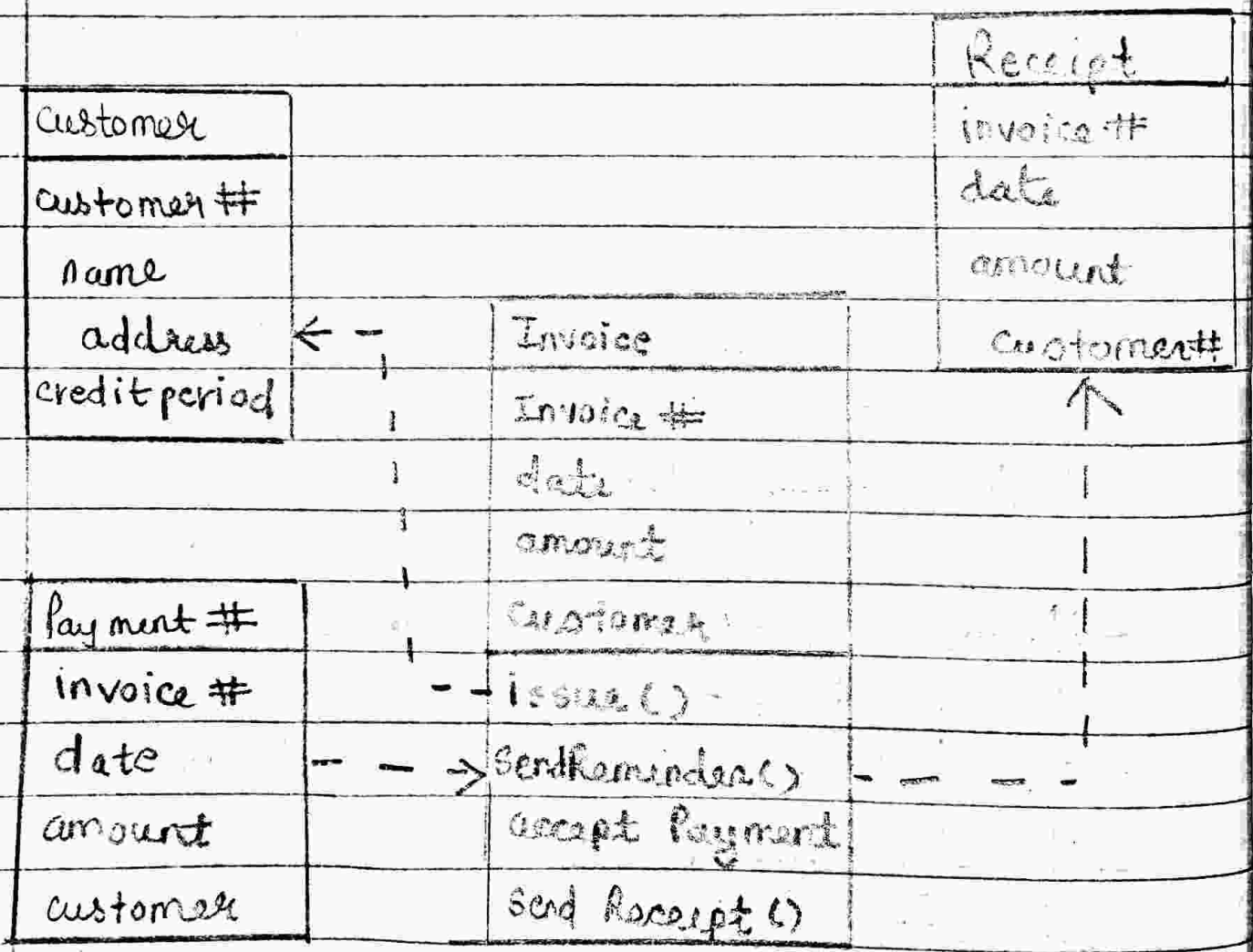
Modular decomposition :

- Another structural level where subsystems are decomposed into modules.
- Two modular decomposition models are

1. An object model
2. A data flow model.

1. An object model :

- Structure the system into a set of losely coupled objects with well defined interfaces.
- Object oriented decomposition is concerned with identifying object classes, their attributes & operations.
- When implemented objects are created from these classes & some contrd model used to co-ordinate object operation.



Invoice Processing System

- - →

↑ dotted line denotes, कोई attribute जिसको.

Advantages :

1. The implementation of objects can be modified without affecting other objects because objects are losely coupled.
2. Objects are often representations of real world entities so the structure of system is readily understandable.
3. Because these real world entities are used in different systems, objects can be reused.

Disadvantages :

1. To use services, objects must explicitly reference the name and interface of other objects.
2. If an interface change is required to satisfy proposed system changes the effect of that change on all users of the changed object must be evaluated.
3. More complex entities are sometimes difficult to represent as objects.

Function oriented

2. Data Flow Model : (filter)
(backtrack हुँदैन)

2068 Q.no 12(a)

- In a data flow model / function oriented pipeline functional transformations process their inputs & produce outputs.
- Data flows from one to another & is transferred as it moves through the sequence.
- The transformation may execute sequentially or in parallel and each processing step is implemented as a transform.

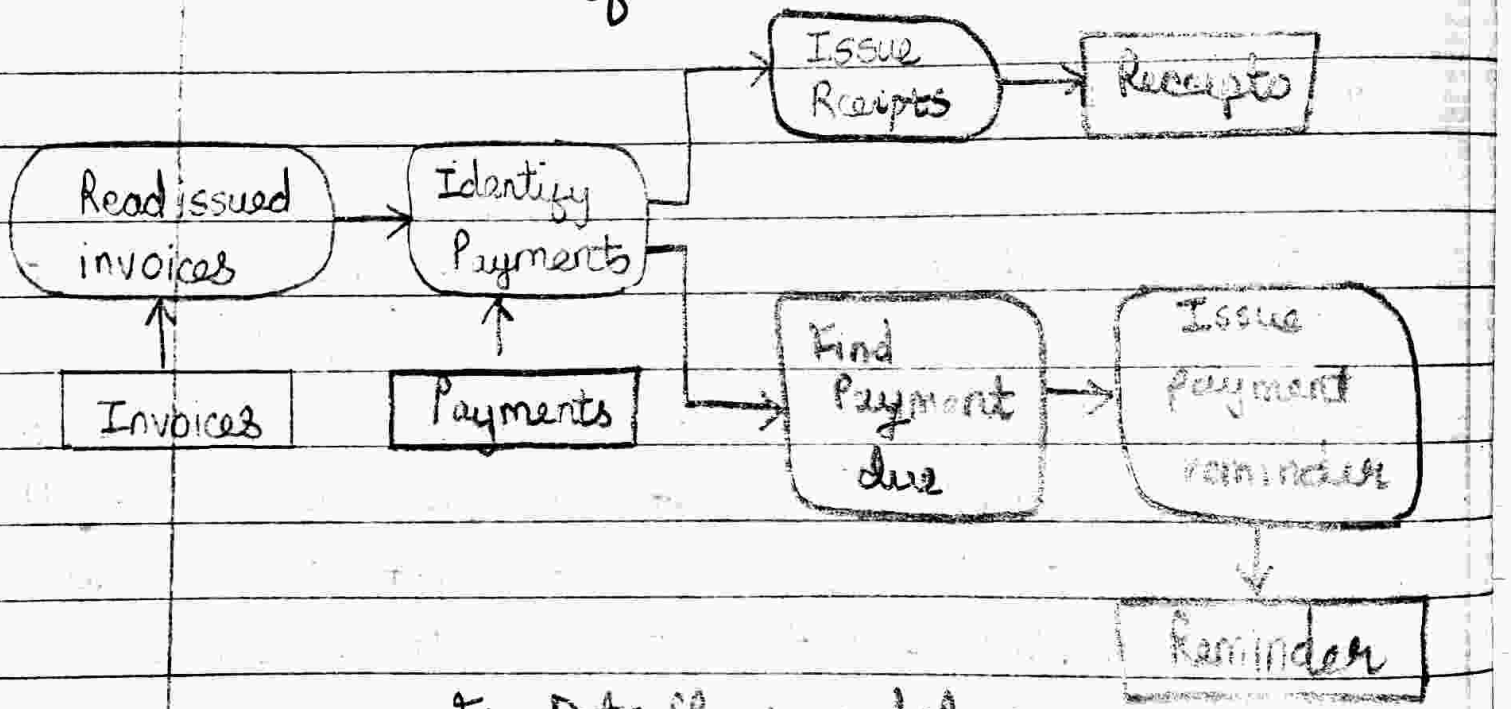


Fig. Data flow model for invoice processing system

Advantages :

- It supports the reuse of transformation.
- It is intuiting in that many people think of their work in terms of input and output processing.
- Evolving the system by adding new transformations is usually straightforward.
- It is simple to implement either as concurrent or sequential system.

Disadvantages :

- There has to be a common format for data transfer that can be recognized by all transformation.
- Each transformation must either agree with its communicating transformation the format of the data that will be processed or with a standard format for all data communicated must be imposed.

Domain-Specification Architecture :

- Architectural models which are specific to some

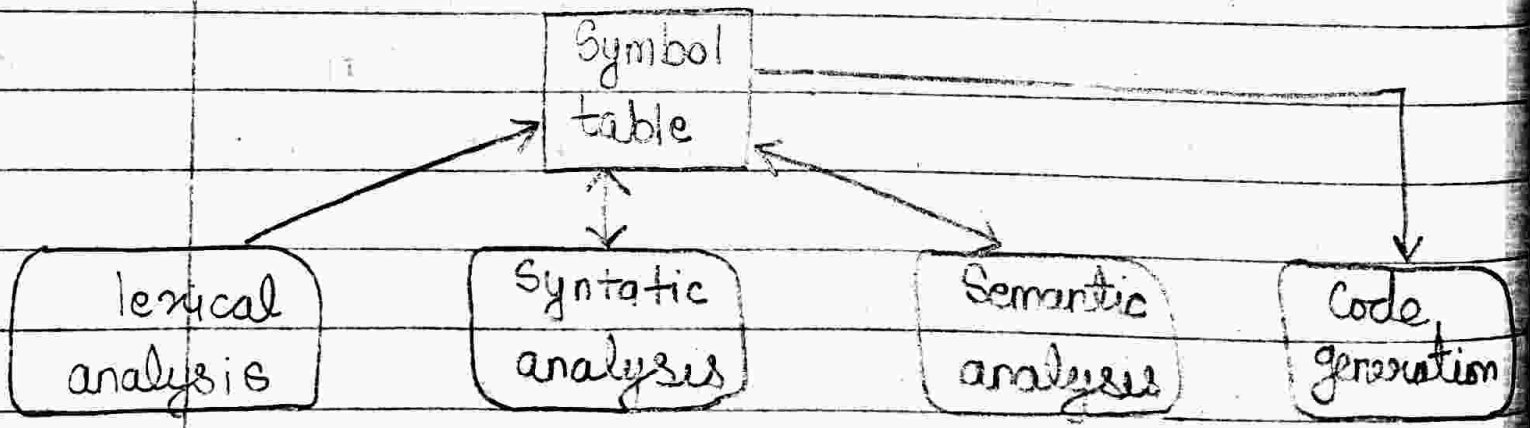
application domain.

Two types of domain specific models are

1. Generic model
2. Reference model

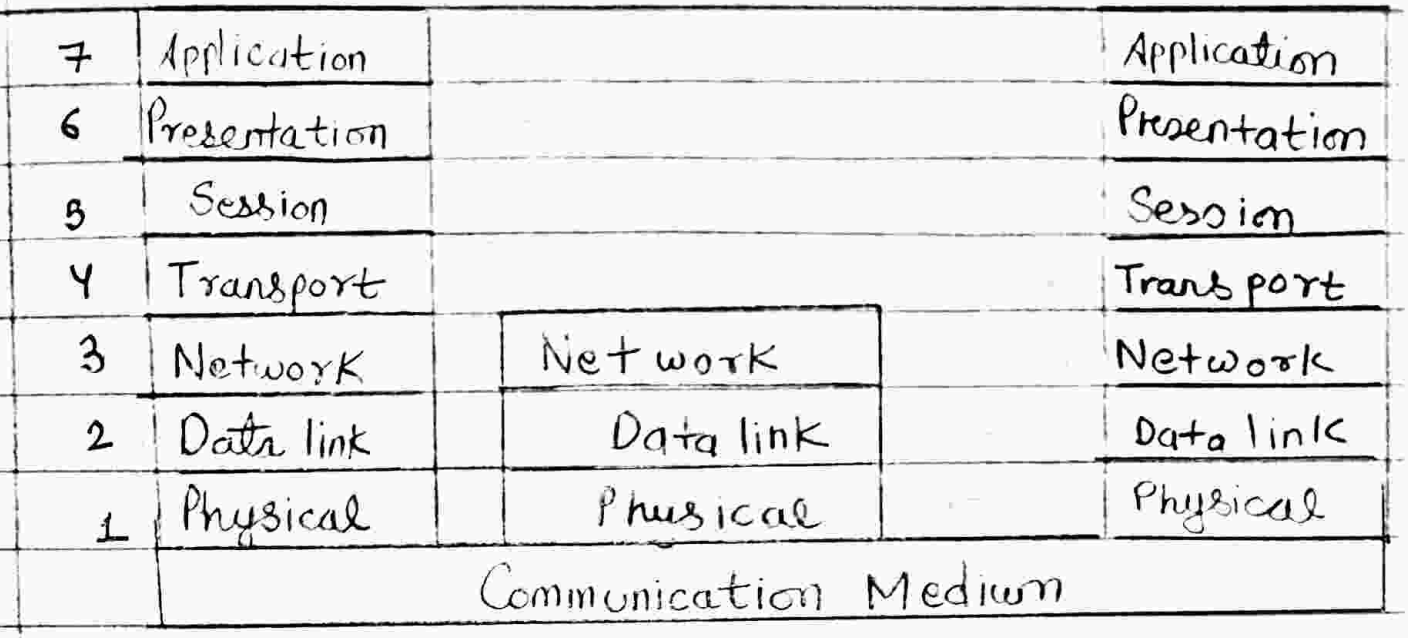
1. Generic model :
   - Are abstractions from a number of real systems.
   - they encapsulate the principle characteristics of these systems.

2. Reference model.
   - are more abstract and describe a larger classes of systems.
   - They are a way of informing designers about the general structure of that class of system.



Compiler model representing generic models

OSI reference model



Unit 3.

3.2 # Object Oriented Design :-

∴ Characteristics of OOD :-

→ Objects are abstractions of real world or system entities and manage themselves.

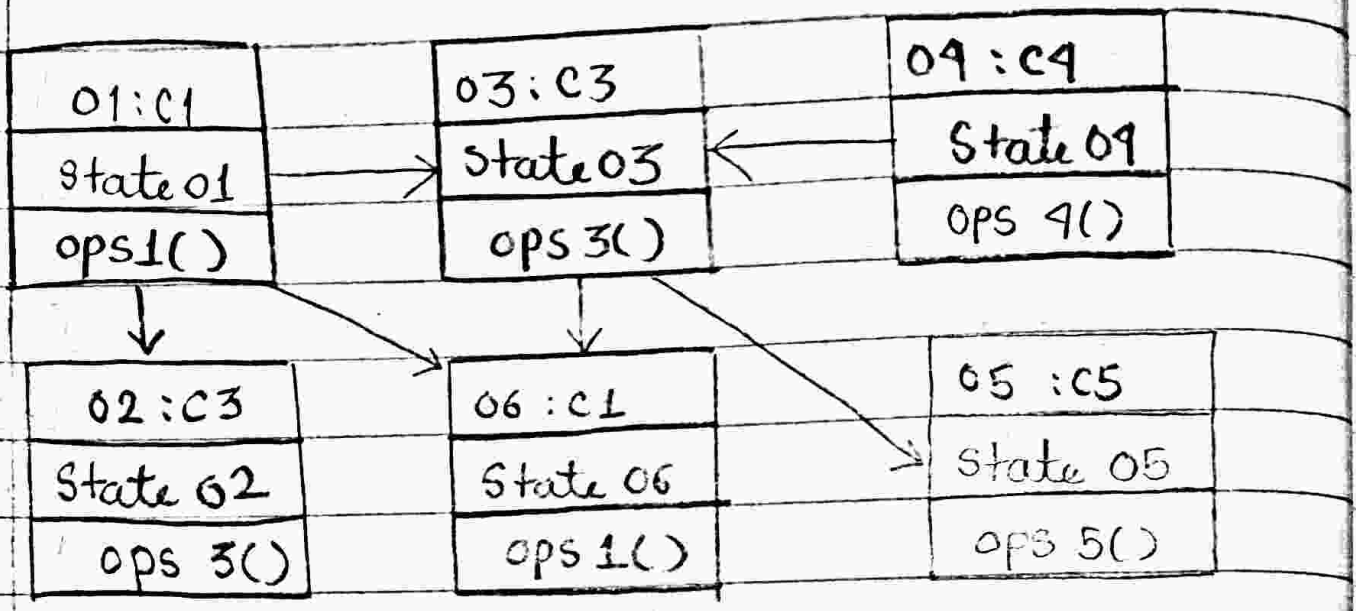
→ Objects are independent and encapsulate state & representation information.

→ System functionality is expressed in terms of object services.

→ Shared data areas are eliminated. Objects communicate by message passing.

→ Objects may be distributed & may execute sequentially or in parallel.

Class এই reuse করে বিভিন্ন।



Interacting objects

Advantage of OOD :

1. Easier Maintainence: Objects may be understood as standalone entities.
2. Objects are appropriate reusable components.
3. For some systems, there maybe an obvious mapping from real world entities to system objects.

Object Oriented Development :

- Object Oriented Analysis, Design & programming are related but distinct.

- OOA is concerned with developing an object model of the application domain. The objects in that model reflect the entities and operations associated with the problem to be solved.

- OOD is concerned with developing an object oriented system model to implement requirements.

- OOP is concerned with releasing and object oriented design using OOP language. Such as C++ / Java.

Object & Object classes :

→ Objects are entities in a software system which represent instances of real world and system entities.

→ Object classes are templates for objects. They may be used to create objects.

→ Object classes may inherit attributes & services from other object classes.

Unified Modelling Language (UML)

→ Several different notations for describing object oriented designs were proposed in 1980's & 1990's
→ UML is an integration of these notations.
→ It describes notations for a number of different models that may be produced during OOA and D

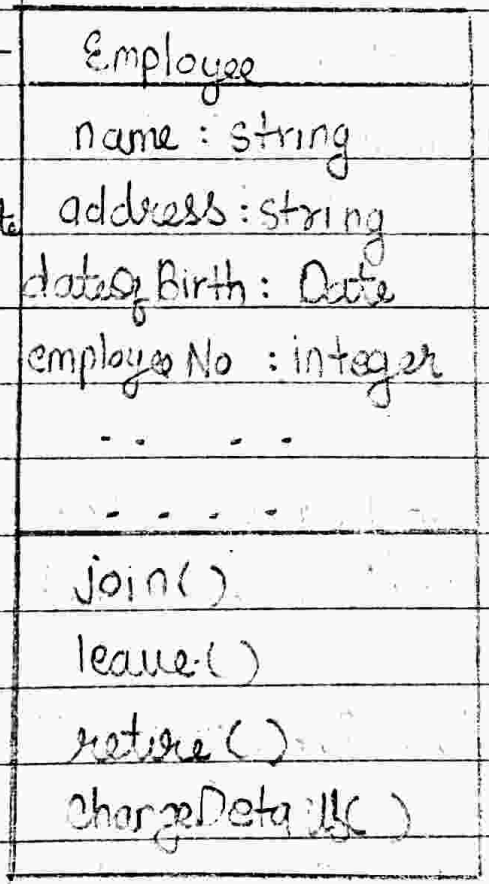


Fig. Employee Object Class (UML)

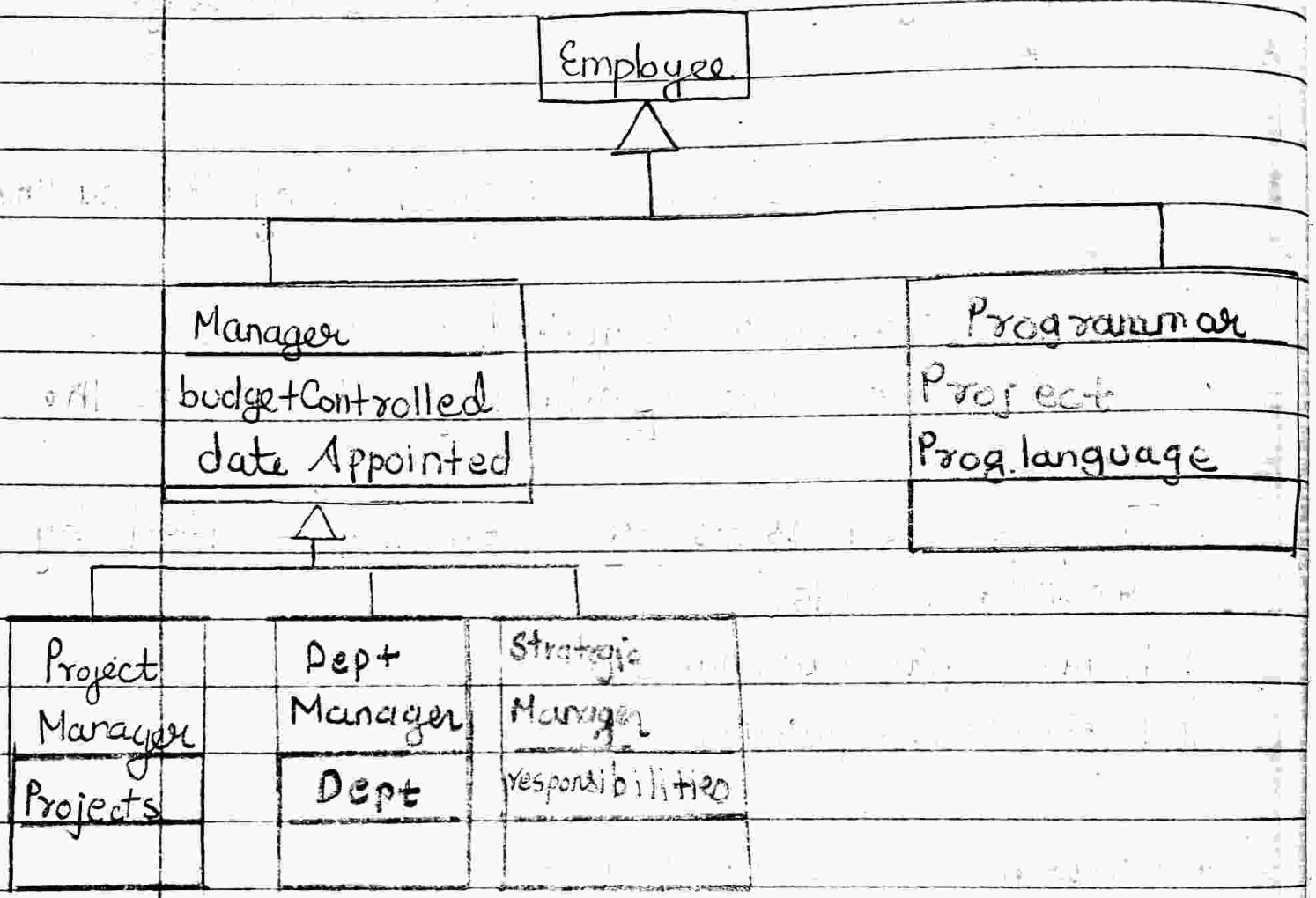
Object Communication :

1. Conceptually objects communicate by message passing.
2. Messages
   i) The name of the service requested by the calling object.
   ii) Copies of information required to execute the service & the name of holder for result of the service
3. In practice, messages are often implemented by procedure calls.
   i) Name = procedurename
   ii) information = parameterlist

Generalization & Inheritance :

1. Objects are member of classes which define attribute types & operations.
2. Classes may be arranged in a class hierarchy where one class i.e a super class is a generalization of one or more other classes i.e sub classes.
3. A sub class inherits the attributes & op^n's from its super class & may add new methods / attributes in its own.

4. Generalisation in the UML is implemented as Inheritance in OOP language.



A generalisation hierarchy

Advantage of inheritance :

1. It is an abstraction mechanism which may be used to classify entities.
2. It is a reuse mechanism at both design & the programming level.

3. The inheritance grap is a source of organizational knowledge about domains & systems.

Problems with inheritance :-

1. Object# classes are not self-contained, they cannot reunderstood without reference to their super-classes.
2. Designers have a tendency to reuse the inheritance graph created during analysis which can lead to significant inefficiency.
3. The inheritance graphs of analysis, design & implementation have different fx^ns & should be maintained separately.

UML Association :-

1. Objects & object classes participate in relationships with other objects & object classes.
2. In the UML, a generalised relationship is indicated by an association.
3. Associations may be annotated with information that describes the association.
4. Associations are general but may indicate that

an attribute of an object is associated object or that a method relies on an associated object.

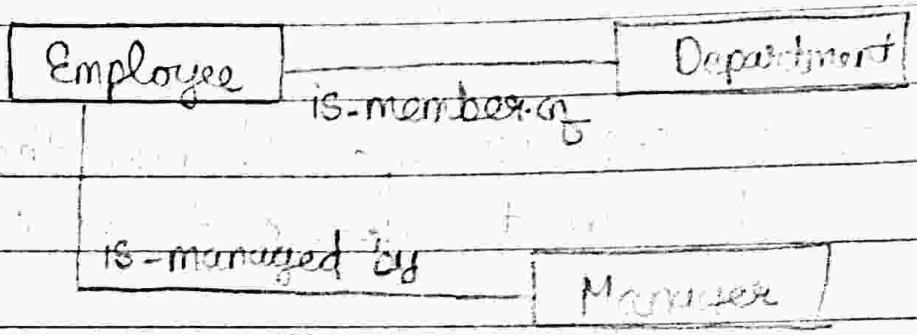


Fig: An association model

Lab Assignment :

1. Using examples explain the difference between object and object classes.
2. Under what circumstances might you develop a design where objects execute concurrently?
3. Using the UML graphical notation for object classes, design following object classes identifying attributes & operations:

- A telephone
- A printer for PC
- A bank account
- A library catalogue

Concurrent objects

1. The nature of objects as self obtained entities make them suitable for concurrent implementation.
2. The message passing model of object communication can be implemented directly if objects are running on separate processors in a distributed system

object oriented design process:-

1. Understand and define the context and the models of use of the system.
2. Design the system architecture.
3. Identify the principal objects in the system.
4. Develop designed models.
5. Specify object interfaces.

1 ⇒ i) Develop an understanding of the relationship between the software being design and its external environment.

System Context:

→ A system model that describes other systems in the environment.

Modes of system use :

→ A dynamic model that describes how the system interacts with its environment.

2. ⇒ i) Once interactions between the system and its environment have been understood, this information is used for designing the system architecture.

ii) There should be no more than seven entities in an architectural model.

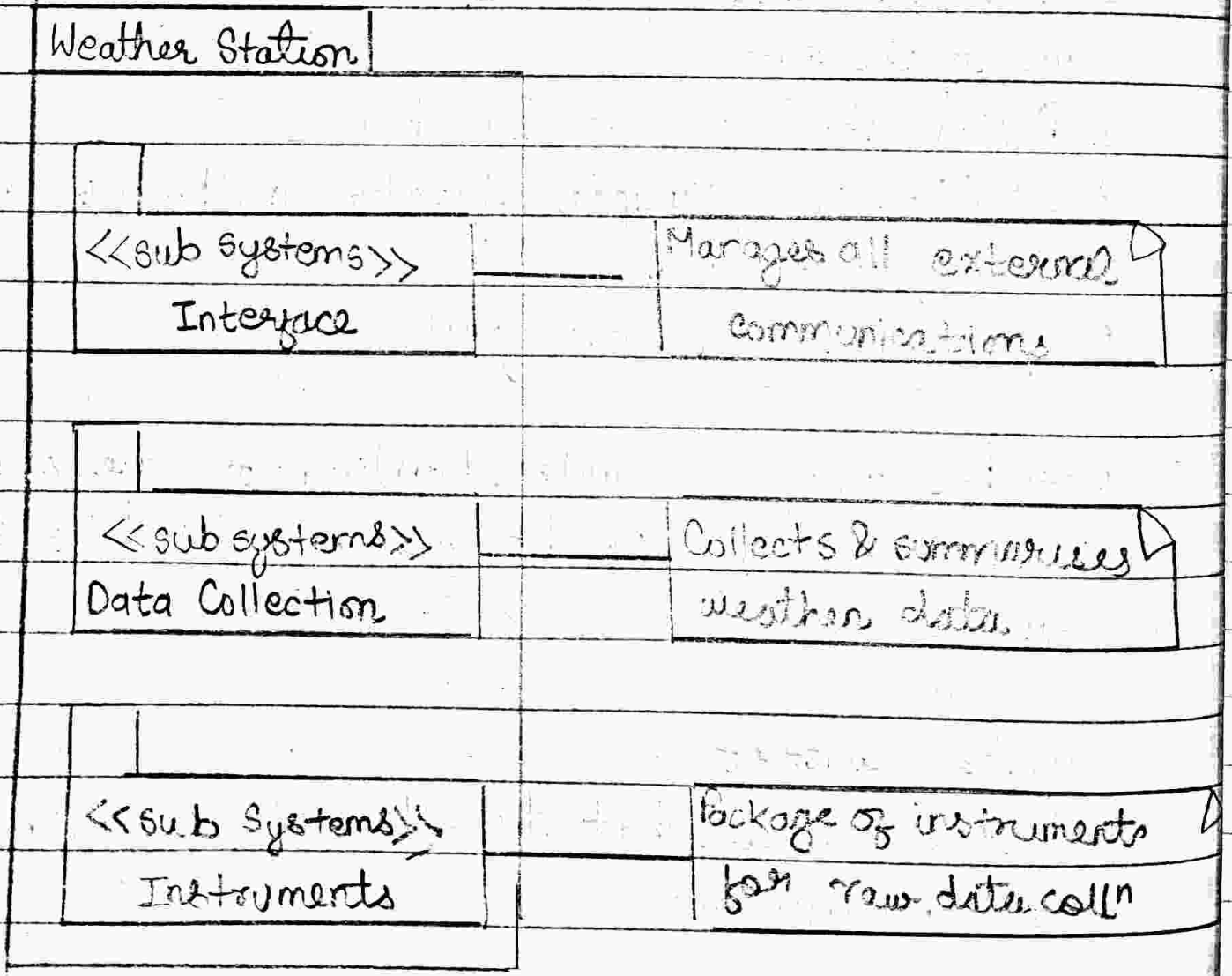


Fig. Weather station architecture

3 ⇒ i) Identifying objects or object classes is the most difficult part of OOD.

ii) There is no magic formula for object identification. It relies on the skill, experience and domain knowledge of the system designer.

iii) Object identification is an iterative process. You are unlikely to get it right first time.

➡ Approaches to object identification

i) Use a grammatical approach based on a natural language description of the system. (used in HOOD method). Objects and attributes are, operations or services are verbs.

ii) Base the identification on tangible things in the application domain.

iii) Use a behavioural approach and identify object based on what participates in what behaviour.

iv) Use a scenario based analysis. The objects, attribute and methods in each scenario are identified.

4) ⇒ i) Design models show the objects and object classes and relationships between these entities.

ii) Static models show the objects and objects classes and relationships between these entities.

iii) Static models describe the static structure of the system in terms of object classes and relationships.

iii) Dynamic models describe the dynamic interactions between objects.

Examples of design models:

i) Sub-system models :
Show logical grouping objects into coherent sub-system.

ii) Sequence model :
Show the sequence of object interaction.

iii) State machine models :
Show how individed object change their state in response to events.

iv) Other models includes usecase models, aggregation models, generalisation model etc.

5⇒ i) Object interfaces had to be specified so that the objects and other components can be designed in parallel.

ii) Designers should avoid designing the interface representations but should hide this in the object itself.

iii) The UML uses class diagrams for interface specification but Java may also be used.

• Design Evolution :-

1. Hiding information inside objects means that changes made to an object don't affect other objects in an unpredictable way.

2. Assume pollution monitoring facilities are to be added to weather stations. These same the air and compute the amount of different pollutants in the atmosphere.

Unit 4: Verification & Validation

4.1. Verification :

→ are we building the product right ?
i.e the software should confirm to its specificati

Validation : (customer ले use गर्दा proper o/p आउँदछ कि नाइ)
→ are we building the right product ?
i.e the software should do what user really requires.

BSc no 12 (b)
2068 *

V& V Confidence :

→ Depends on system purpose, user expectations and marketing environment.

i) Software ~~environment~~ function :
The level of confidence depends on how critical the software is to an organization.

ii) User expectations :
User may have low expectations of certain kinds of software.

iii) Marketing environment :
getting a product to market early may be

more important than finding defects in the program.

V&V Process:

1. Is a whole life cycle process that is V&V must be applied at each stage in the software process.
2. Has two principle objectives:
   a) The discovery of defects in the system.
   b) The assessment of whether or not the system is usable in an operational situation.

Static & Dynamic Verification:

i) Software inspections: (manually not computerized)
Concerned with analysis of the static system representation to discover problems, i.e static verification.
May be supplemented by tool based document & code analysis.

ii) Software Testing: (Dynamic)
Concerned with exercising (practice) and observing product behaviour, i.e dynamic verification.
The system is executed with test data & its

Sno 8
model

operational behaviour is observed.

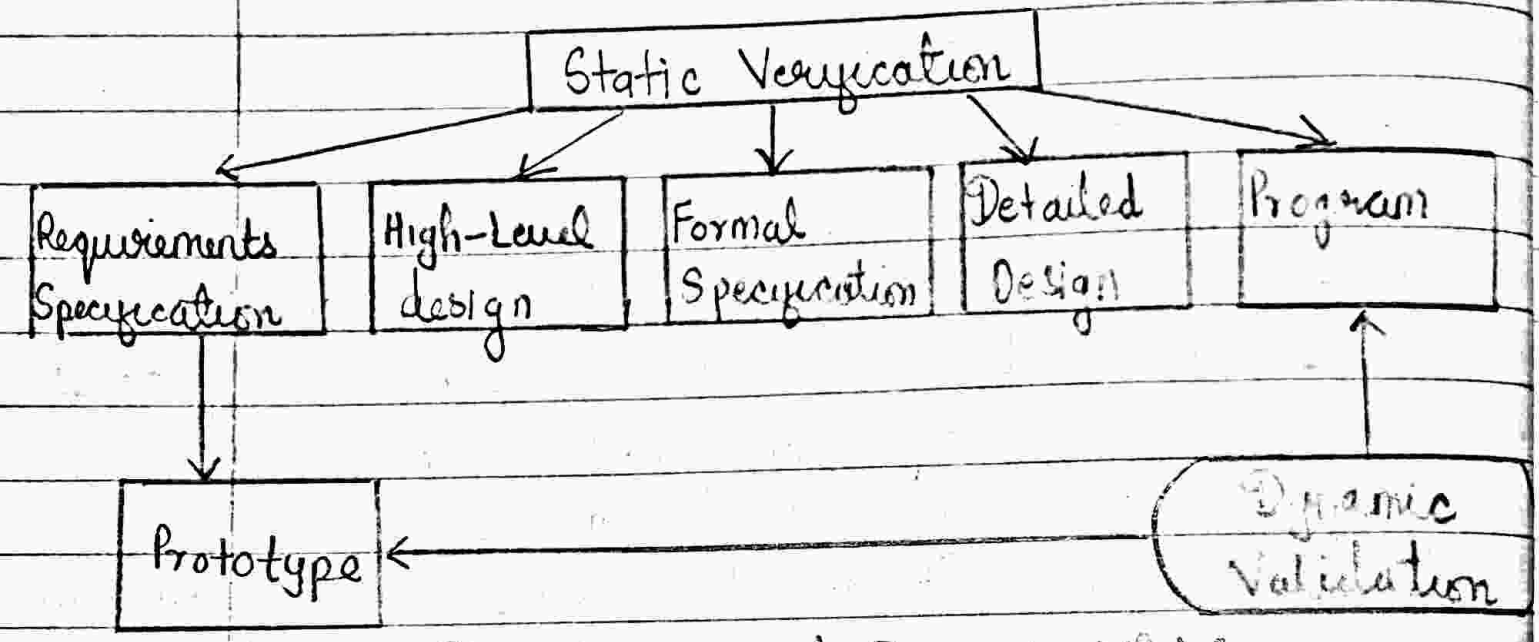


Fig. Static and Dynamic V&V

Program Testing :

1. Can reveal the presence of errors, not their absence.
2. A successful test is a test which discovers one or more errors.
3. The only validation technique for non-functional requirements.
4. Should be used in conjuction with static verification to provide full V&V coverage.

Types Of Testing :-

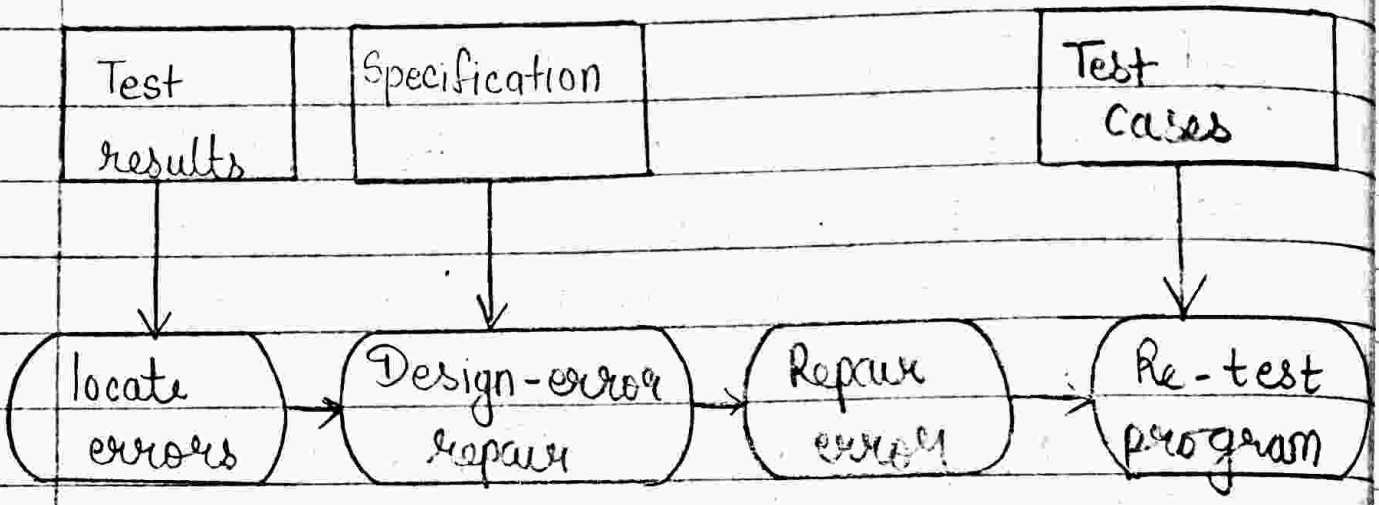
1) Defect Testing :-
→ Test designed to discover system defects.
→ A successful defect test is one which reveals the presence of defects in a system.

2) Statistical Testing :- / Validation Testing
Test designed to reflect the frequents of user inputs. Used for reliability estimation.

Testing & Debugging :-

1. Defect Testing & Debugging are distinct processess.
2. V & V is concerned with establishing the existence of defects in a program.
3. Debugging is concerned with locating & repairing these errors.
4. Debugging involves formulating hypothesis about program behaviour then testing these hypothesis to find the system errors.

The debugging process :



V& V Planning :

Q no 11 2068

1. Carefull planning is required to get the most, out of testing & inspection processess.
2. Planning should start early in the development process.
3. The plan should identify the balance between static verification & testing.
4. Test planning is about defining standards for the testing process rather than describing product test.

(V model बाहिर)

Q no 12 (a) 2068 *

Software inspections :

→ Involve people examining the source representation with the aim of discovering anomalies & defects.
→ Donot require execution of a system so may be used before implementation.
→ May be applied to any representation of the system i.e requirements, design, test data etc.
→ Very ~~exp~~ effective technique for discovering errors.

Inspection success :

1. Many different defects may be discovered in a single inspection. In testing one defect may mask another, so several execution are required.
2. The reuse domain and programming knowledge so reviewers are likely to have seen the types of errors that commonly arise.

Inspections & Testing :

1. Inspection & Testing, both should be used during V& V process.

2. Inspections can check conformance with a specification but not conformance with the customreal requirements.
3. Inspections cannot check non-functional characteristics such as performance, usablility etc.

Inspections ~~fix~~ pre-conditions :

exact thing
1. A precise specification must be available.
2. Team members must be familiar with the organisation Standards.
3. Syntatically correct code must be available.
4. An error checklist should be prepared.
5. Management must accept that the ~~specification~~ inspection will increase cost early in Software process.
6. Management must not use inspection for staff appraisal.

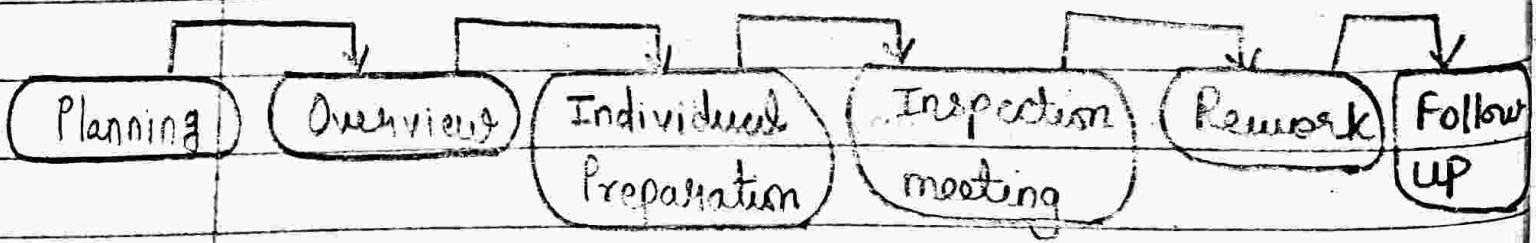


Fig. The inspection process.

Inspection Procedure :

- System overview presented to inspection team.
- Code and associated documents are distributed to inspection team in advance.
- Inspection takes place & discovered errors are noted.
- Modifications are made to repair discovered errors.
- Reinspection may or may not be required.

Inspection Team :

- Made up of atleast 4 members.
- Author of the code being inspected.
- Inspector who finds errors, omission & inconsistency
- Reader who reads the code to the team.
- Moderator ~~which~~ who cheers the meeting & notes discovered errors.
- Other roles are scribe & chief moderator.

wrote copies of things such as letters / documents

Inspection Checklist :

1. Checklist of common error should be used to drive the inspection.

2. Error checklist is programming language dependent
3. The weaker the typechecking, the larger the checklist
Eg: Initialization, loop termination, constant naming, array bound.

Inspection Rate :

1. 500 statements per hour during overview.
2. 125 source statement / hour during individual preparation.
3. 90-125 statement / hour can be inspected during inspection meeting.
4. Inspection is therefore an expensive process.

Automated Static Analysis :

1. The static analysers are s/w tools for source, text processing.
2. They parse the program text & try to discover potentially erroneous conditions & bring these to the attention of V&V team.
3. Very effective as an aid to inspections. A supplement to but not a replacement for inspections.

Stages of static Analysis :

1. Control Flow Analysis :
Checks for loops with multiple exit and entry points, finds unreachable code etc.

2. Data Use Analysis :
Detects ~~an~~uninitialized variables, variables ~~returns~~ written twice without an intervening assignment, variables which are declared but ~~not~~ never used etc.

3. Interface Analysis :
Checks the consistency of routine and procedure declaration & their use.

4. Information Flow Analysis :
Identifies the dependencies ~~of~~ between input & output variables. While it doesn't detect anomalies, it shows how the value of each program variable is derived from other variable values

5. Path Analysis :
Identifies paths through the program & sets out

the statement executed in that path.

Use of Static Analysis :

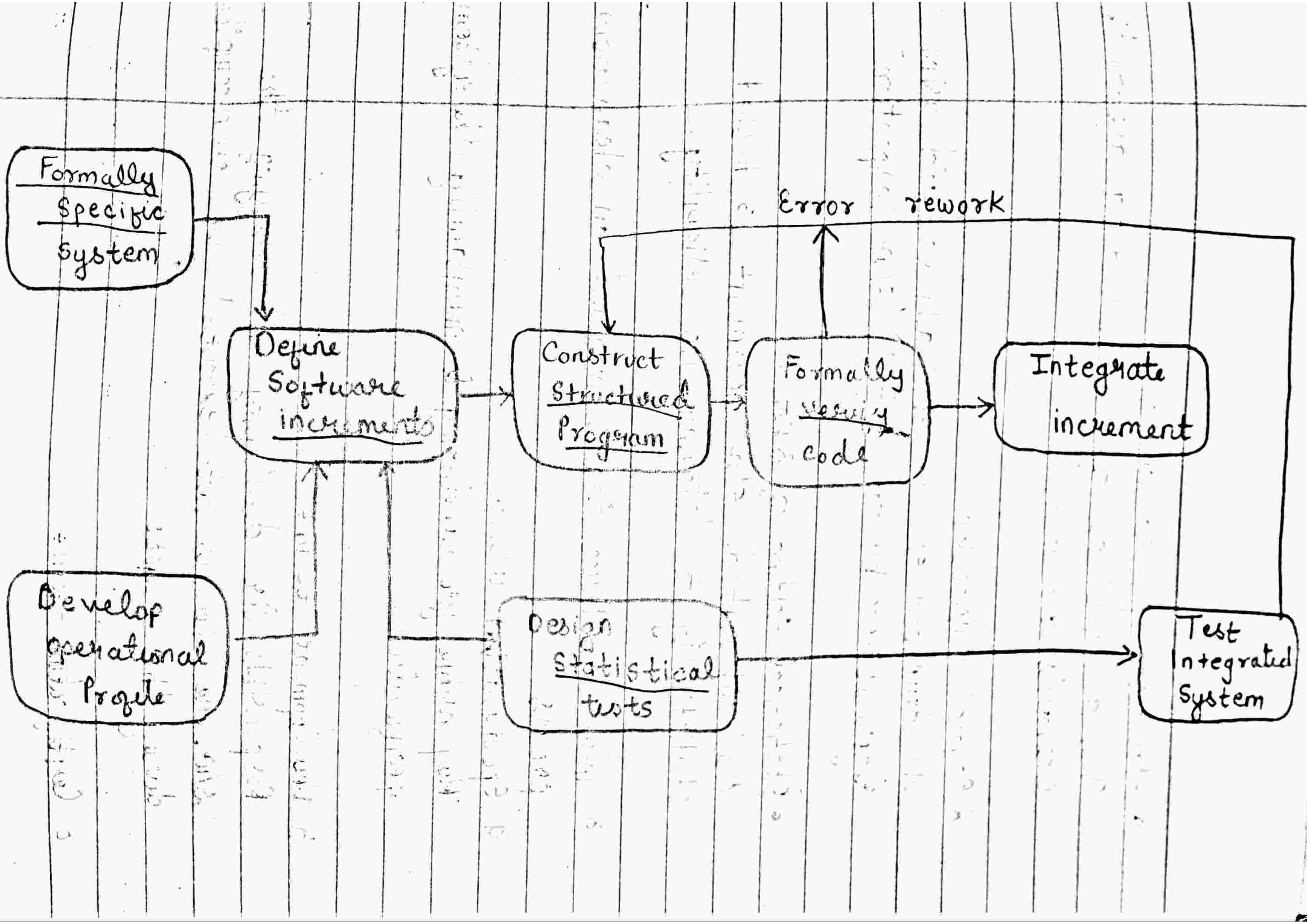
1. Valuable when a language such as C is used which has weak typing & hence many errors are undetected by the compiler.
2. Less cost effective for languages like Java that have strong typechecking & can detects many error during compilation.

Clean room Software Development : / Formal V&V

1. The objective of this approach to software development is zero defect s/w.
2. The cleanroom approach to s/w development is based on five key strategies :

a. Formal Specification :
The s/w to be developed is formally specified.

b. Incremental development :
The s/w is partitioned into increments that are developed & validated separately using clean room process.


Formally Specific System
Define Software increments
Construct Structured Program
Formally verify code
Error rework
Integrate increment
Develop operational Profile
Design Statistical tests
Test Integrated System

c. Structured programming:
Only a limited no of control & data abstraction constructs are used.

d Static verification:
The developed s/w is statically verified using rigorous (one by one detail) s/w inspection.

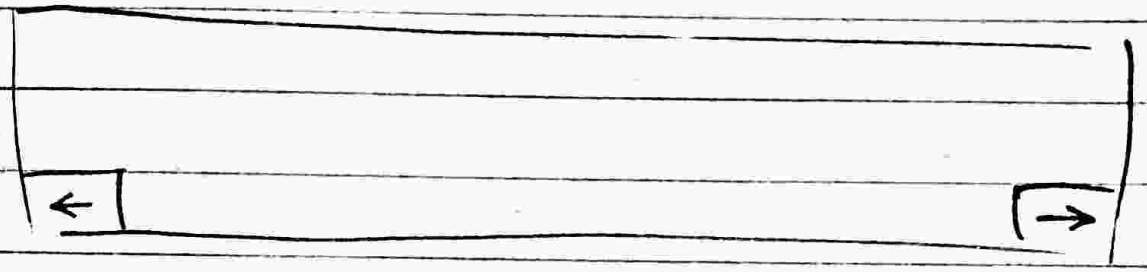
e. Statistical testing of system:
The integrated s/w increment is tested statistically to determine its reliability.

3. There are three teams involved in cleanroom process

a Specification team:
Responsible for developing & maintaining the system specification.

b Development team:
Responsible for developing and verifying the s/w. The s/w is not executed or even compiled during this process.

c Certification team:



Responsible for developing a set of statistical test to exercise the s/w after development.

Cleanroom Process Evaluation :-

1. Results in IBM have been very impressive with few discovered faults in delivered system.
2. Independent assessment shows that the process is no more expensive than other approaches.
3. Fewer errors than in a traditional development process.
4. Not clear how this approach can be transfered to an env. with less skill or less highly motivated engineers.