Software Process Model

Software Process

- For success in large Software development, it is important to follow an engineering approach, consisting of a well-defined process.
- Process consists of activities/steps to be carried out in a particular order.
- Software process deals with both technical & management issues.
- Consists of different types of process

Process Types…

- Process for Software development:
  - Produces Software as end result.
- Multiple such processes may exist.
- A project followes a particulor process.
- Process for managing the project :
  - Defines project plaunning & control.
  - Effect estimates made & schedule prepared.
  - Resources provided.
  - Feedback taken for quality assurance.
  - Monitoring done.
- Process for managing the above processes themselves:
- Improving the processes based on new techniques,tools etc.
- Standardizations & certification(ISO,CMM)

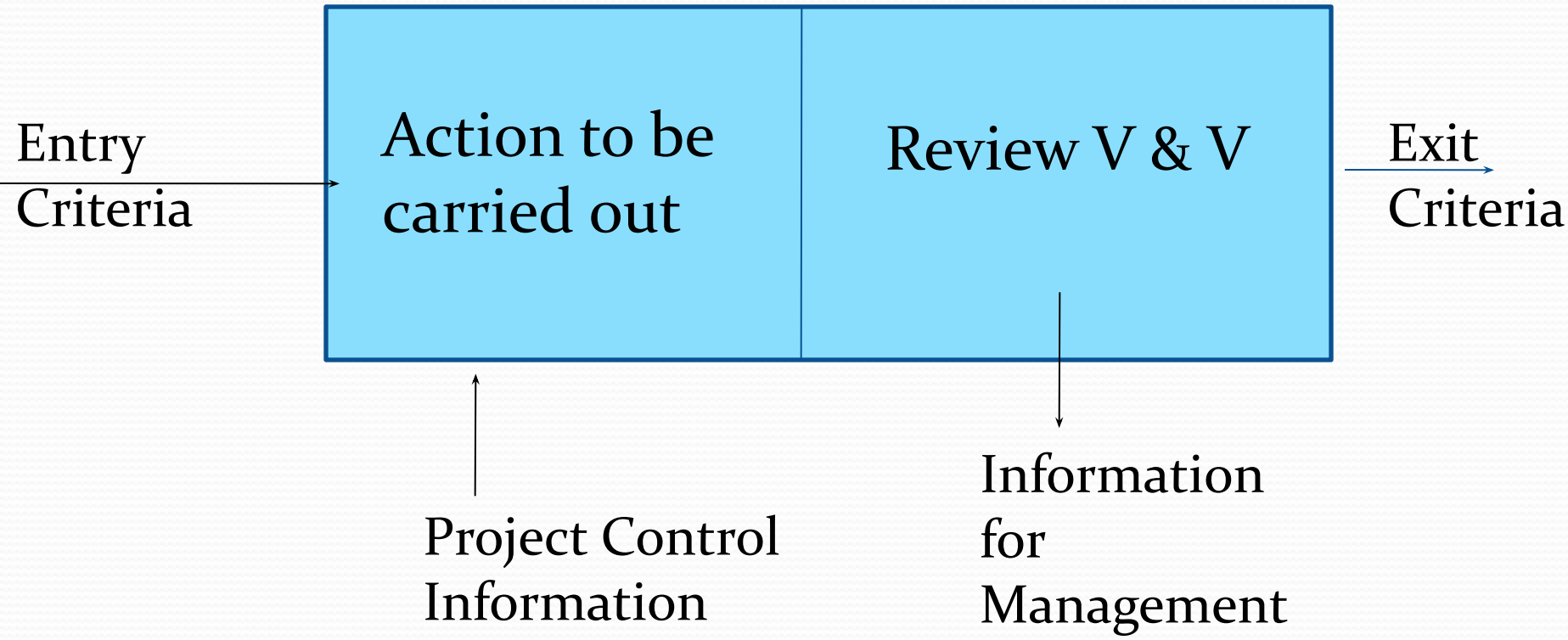
Multiple Processes

- A large development company may have multiple development processes.
  1. For client server based conventional application(sales processing,payroll).
  2. For enterprise-level project based on pachages & customerization
  3. For web based e-commerce Type
  4. For data warehousing/decision type.
- Company may have many projects in each category.

Steps in Process

- Each step has a well-defined objective
- Requires people with specific skills.
- Uses specific techniques ,tools ,guidelines.
- Takes specific inputs & produces well defined output.
- Step define when it may begin(Entry criteria) & when it ends(exit criteria)
- Step must be executed as per project plan that gives duration ,effort,resource,constrains.
- It must produces informations for management so that corrective action can be taken.
  - E.g adding more resources.
- A step ends in a review(V &V)
  - Verification :
    - Check consistency of outputs with inputs(of the step)
  - Validation:
    - Check consistency with user needs.

Process Steps



Software Process

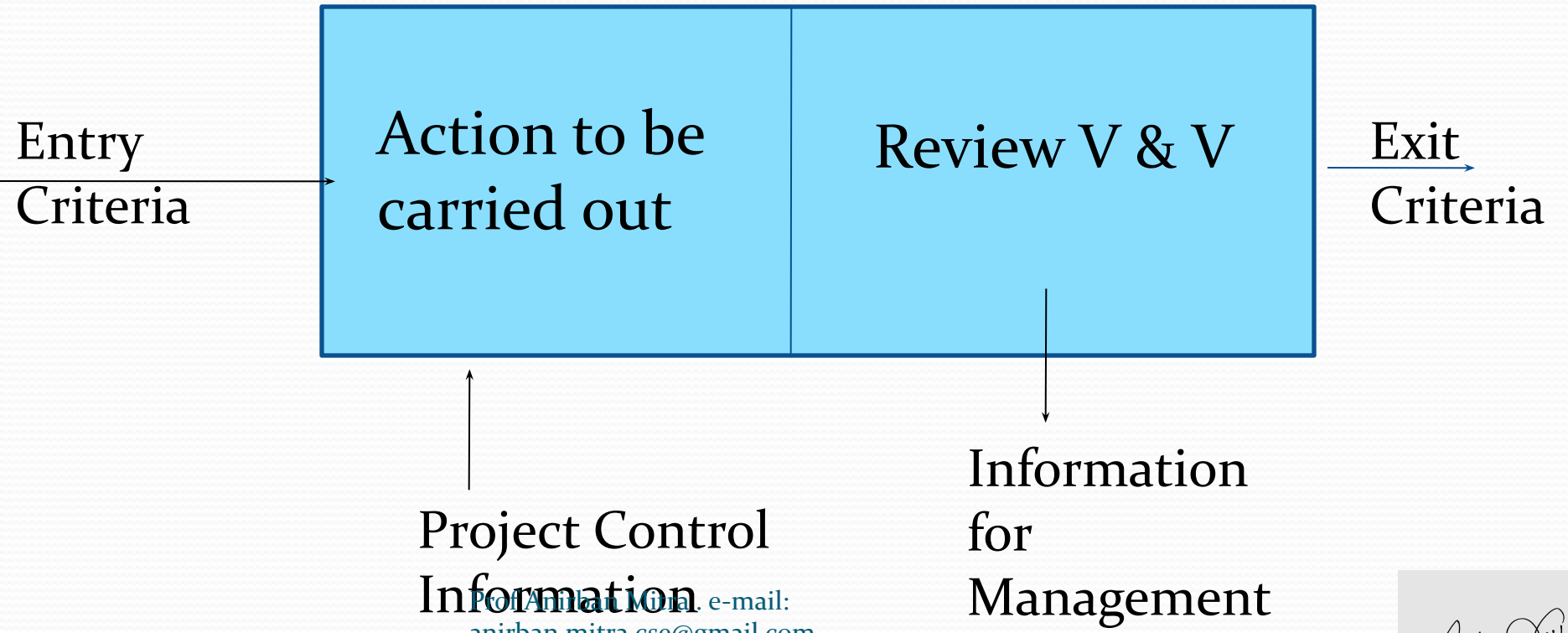
- For success in large Software development, it is important to follow an engineering approach, consisting of a well-defined process.
- Process consists of activities/steps to be carried out in a particular order.
- Software process deals with both technical & management issues.
- Consists of different types of proces

Steps in Process

- Each step has a well-defined objective
- Requires people with specific skills.
- Uses specific techniques ,tools ,guidelines.
- Takes specific inputs & produces well defined output.
- Step define when it may begin(Entry criteria) & when it ends(exit criteria)
- Step must be executed as per project plan that gives duration ,effort,resource,constrains.
- It must produces informations for management so that corrective action can be taken.
  - E.g adding more resources.
- A step ends in a review(V &V)
  - Verification :
    - Check consistency of outputs with inputs(of the step)
  - Validation:
    - Check consistency with user needs.

Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Process Steps



Prof Anirban Mitra. e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Characteristics of a Good Proecss

- Should be precisely defined-:
  - No ambiguity about what is to be done ,when,how,etc.
- It must be predictable-
  - Can be repeated in other projects with confidence about its outcome
  - Predictable with respect to effort,cost
  - Predictable for quality with respect to number and type of defects performance.

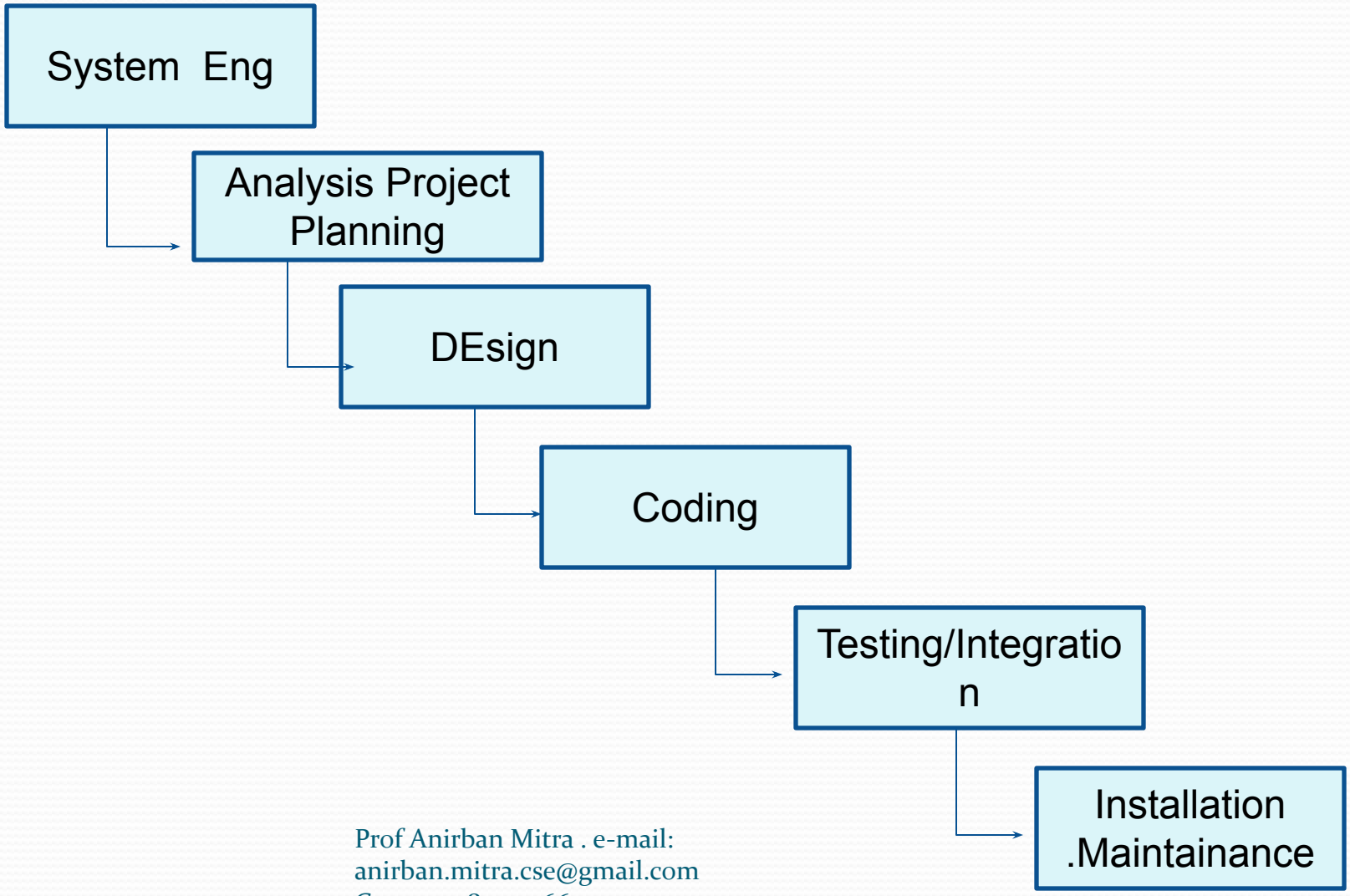
Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Characteristics of a Good Proecss..

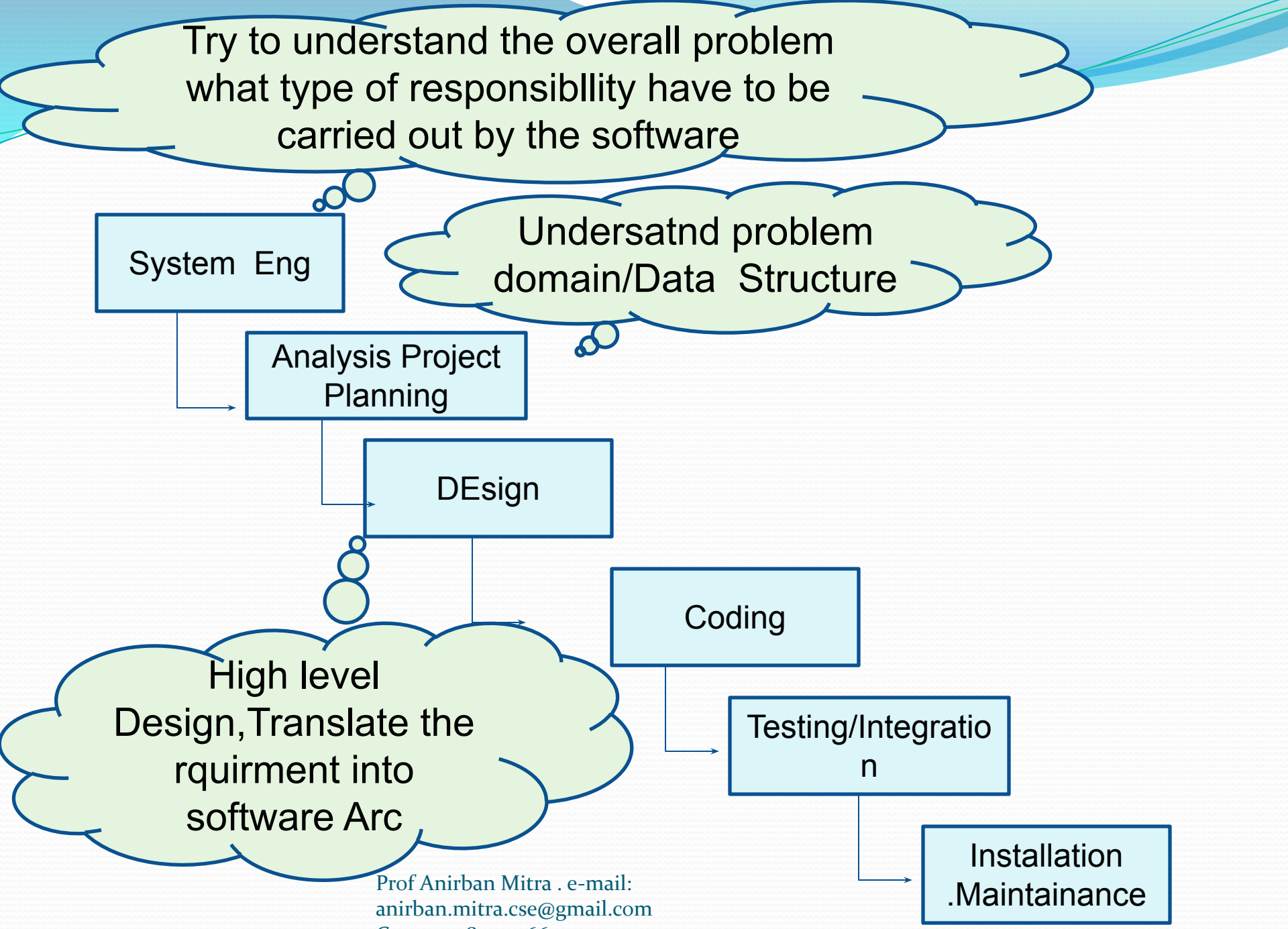
- Predictable process is said to be “under Statistical control” where actula values are close to expected values.
- It supports testing & maintainabilty
  - Maintenance by third party.
  - Follow standars,provide necessary documentation
  - This charecteristic differentates between prototype & product
  - Facilitates early detection of and removeal of defects.
    - Dfect add to project cost.
    - Late detection & correction is costly
- It should facilitate monitoring & improvement
  - Based on feedback
  - Permit use of new tools,technology
  - Permit measurments

Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Waterfall Model for development



Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

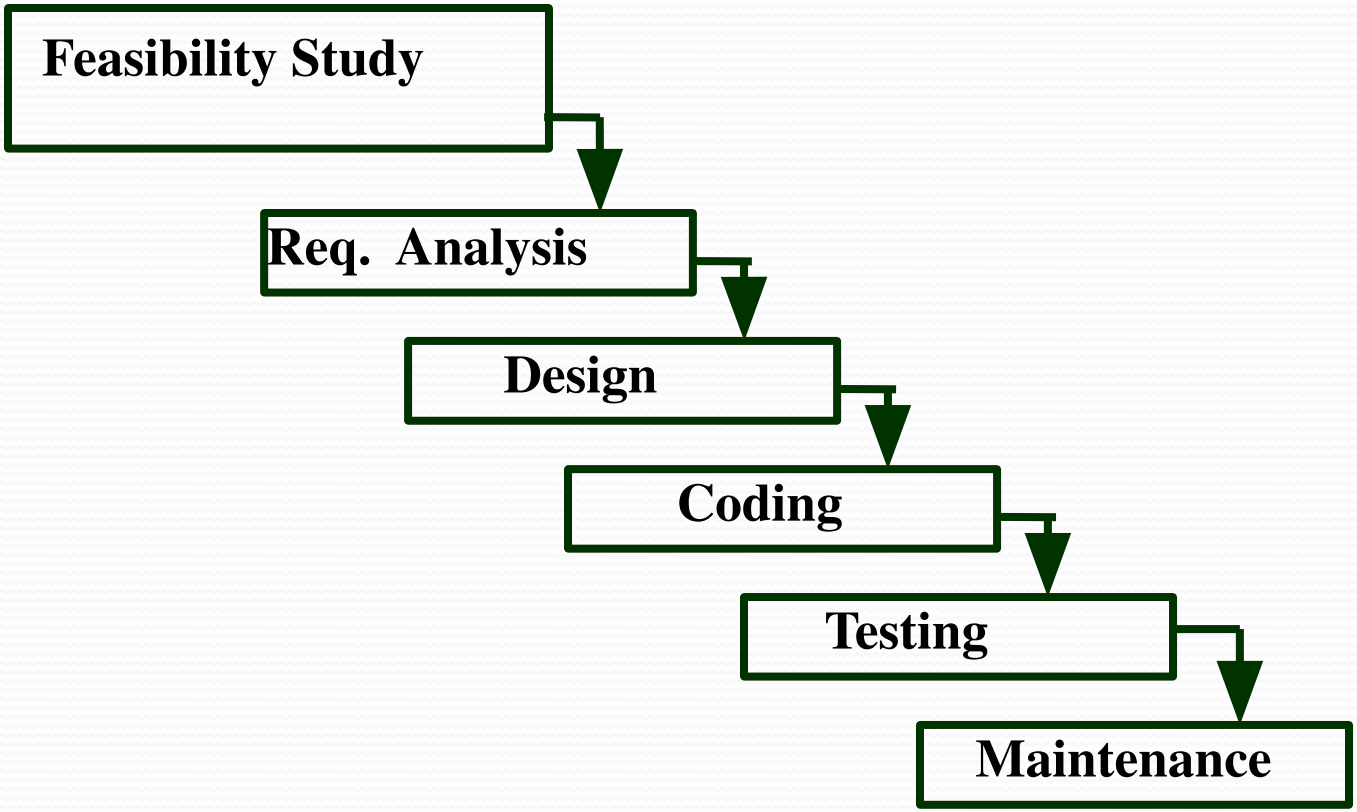


Prof Anirban Mitra . e-mail: anirban.mitra.cse@gmail.com
Contact:9830312660

Classical Waterfall Model

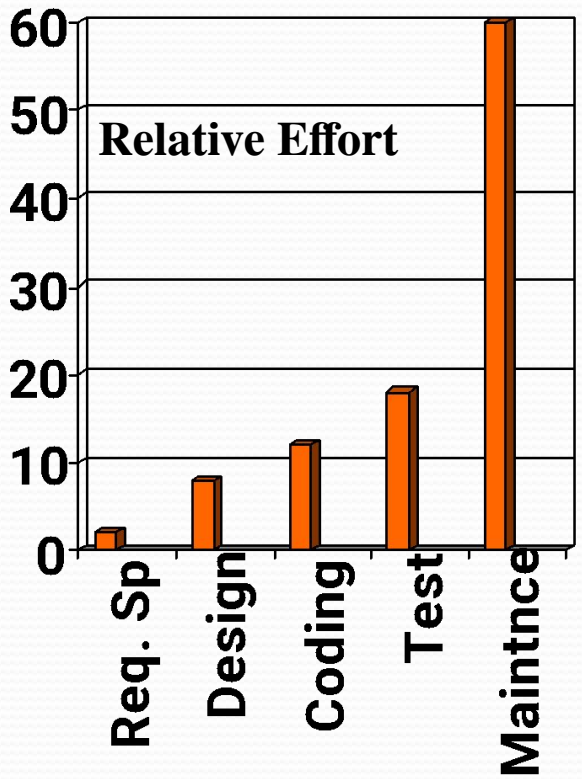
- Classical waterfall model divides life cycle into phases:
  - feasibility study,
  - requirements analysis and specification,
  - design,
  - coding and unit testing,
  - integration and system testing,
  - maintenance.

Classical Waterfall Model



Relative Effort for Phases

- Phases from requirement specification to testing
  - known as development phases.
- Among all life cycle phases
  - maintenance phase consumes maximum effort.
- Among development phases,
  - testing phase consumes the maximum effort.



Classical Waterfall Model (CONT.)

- Most organizations usually define:
  - standards on the outputs (deliverables) produced at the end of every phase
  - entry and exit criteria for every phase.
- They also prescribe specific methodologies for:
  - specification,
  - design,
  - coding,
  - testing, etc.

Feasibility Study

- Main aim of feasibility study: determine whether developing the product
  - financially worthwhile
  - technically feasible.
- Types of Feasibility study:
  - Technical Feasibility,
  - Operational Feasibility,
  - Economical Feasibility.

TECHNICAL Feasibility

- Checks whether the required resources to develop the system can be accessed within the allocated time and budget.

OPERATIONAL Feasibility

- Checks whether the new system performs all the intended operations.

ECONOMICAL Feasibility

- Determines whether the required system is capable of generating financial gains for the organization.

Requirements Analysis and Specification

- Aim of this phase:
  - understand the exact requirements of the customer,
  - document them properly.
- Consists of two distinct activities:
  - requirements gathering and analysis
  - requirements specification.

Requirements Gathering

- Gathering relevant data:
  - usually collected from the end-users through interviews and discussions.
  - For example, for a business accounting software:
    - interview all the accountants of the organization to find out their requirements.

Requirements Analysis (CONT.)

- The data you initially collect from the users:
  - would usually contain several contradictions and ambiguities:
  - each user typically has only a partial and incomplete view of the system.

Requirements Analysis (CONT.)

- Collect all related data from the customer:
  - analyze the collected data to clearly understand what the customer wants,
  - find out any inconsistencies and incompleteness in the requirements,
  - resolve all inconsistencies and incompleteness.

Requirements Analysis (CONT.)

- Ambiguities and contradictions:
  - must be identified
  - resolved by discussions with the customers.
- Next, requirements are organized:
  - into a Software Requirements Specification (SRS) document.

Requirements Analysis (CONT.)

- Engineers doing requirements analysis and specification:
  - are designated as analysts.

Design

- Design phase transforms requirements specification:
  - into a form suitable for implementation in some programming language.

Design

- In technical terms:
  - during design phase, **software architecture** is derived from the SRS document.
- Two design approaches:
  - function oriented approach,
  - object oriented approach.

FUNCTION ORIENTED Design Approach

- Consists of two activities:
  - Structured Analysis
  - Structured Design

Structured Analysis Activity

- Identify all the functions to be performed.
- Identify data flow among the functions.
- Decompose each function recursively into sub-functions.
  - Identify data flow among the sub-functions as well.

Structured Analysis (CONT.)

- Carried out using Data flow diagrams (DFDs).
- After structured analysis, carry out structured design:
  - architectural design (or high-level design)
  - detailed design (or low-level design).

Structured Design

- High-level design:
  - decompose the system into *modules*,
  - represent invocation relationships among the modules.
- Detailed design:
  - different modules designed in greater detail:
    - data structures and algorithms for each module are designed.

Object Oriented Design

- First identify various objects (real world entities) occurring in the problem:
  - identify the relationships among the objects.
  - For example, the objects in a pay-roll software may be:
    - employees,
    - managers,
    - pay-roll register,
    - Departments, etc.

Implementation

- Purpose of implementation phase (aka coding and unit testing phase):
  - translate software design into source code.

Implementation

- During the implementation phase:
  - each module of the design is coded,
  - each module is unit tested
    - tested independently as a stand alone unit, and debugged,
  - each module is documented.

Implementation (CONT.)

- The purpose of unit testing:
  - test whether individual modules work correctly.
- The end product of implementation phase:
  - a set of program modules that have been tested individually.

Integration and System Testing

- Different modules are integrated in a planned manner:
  - Modules should never integrated in one shot.
  - Normally integration is carried out through a number of steps.
- During each integration step,
  - the partially integrated system is tested.

Integration and System Testing

System Testing

- After all the modules have been successfully integrated and tested:
  - system testing is carried out.
- Goal of system testing:
  - **ensure that the developed system functions according to its requirements as specified in the SRS document**.

Maintenance

- Maintenance of any software product:
  - requires much more effort than the effort to develop the product itself.
  - development effort to maintenance effort is typically 40:60.

Maintenance (CONT.)

- Corrective maintenance:
  - Correct errors which were not discovered during the product development phases.
- Perfective maintenance:
  - Improve implementation of the system
  - enhance functionalities of the system.
- Adaptive maintenance:
  - Port software to a new environment,
    - e.g. to a new computer or to a new operating system.

Deliverables in Waterfall Model

- Project plan & feasibilty Report: Cost & time
- Requirement document:SRS
- System Design Document
- Detailed Design Document
- Test plans & test Report
- Source Cose : EXE form
- Software manuals:User Manual,Installation manual
- Review Report

Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Shortcoming Of WM

- Requirements may not be clearly known, Especially for applications not having existing counterpart
  - Railway reservation: Manual system existed, so SRS can Be defined
  - On-line container management for railways
  - Knowledge management for central bank
- Requirement change with time during project life cycle itself…

Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Shortcoming Of WM…

- Requirement change with time during project life cycle itself…
  - User may find solution of little use
  - Better to develop in parts in smaller increments
  - This is common for packages, System software
- Consider documentation heavy: so much documentation may not be required for all types of project.

Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Iterative Waterfall Model

- Classical waterfall model is idealistic:
  - assumes that no defect is introduced during any development activity.
  - in practice:
    - defects get introduced in almost every phase of the life cycle.

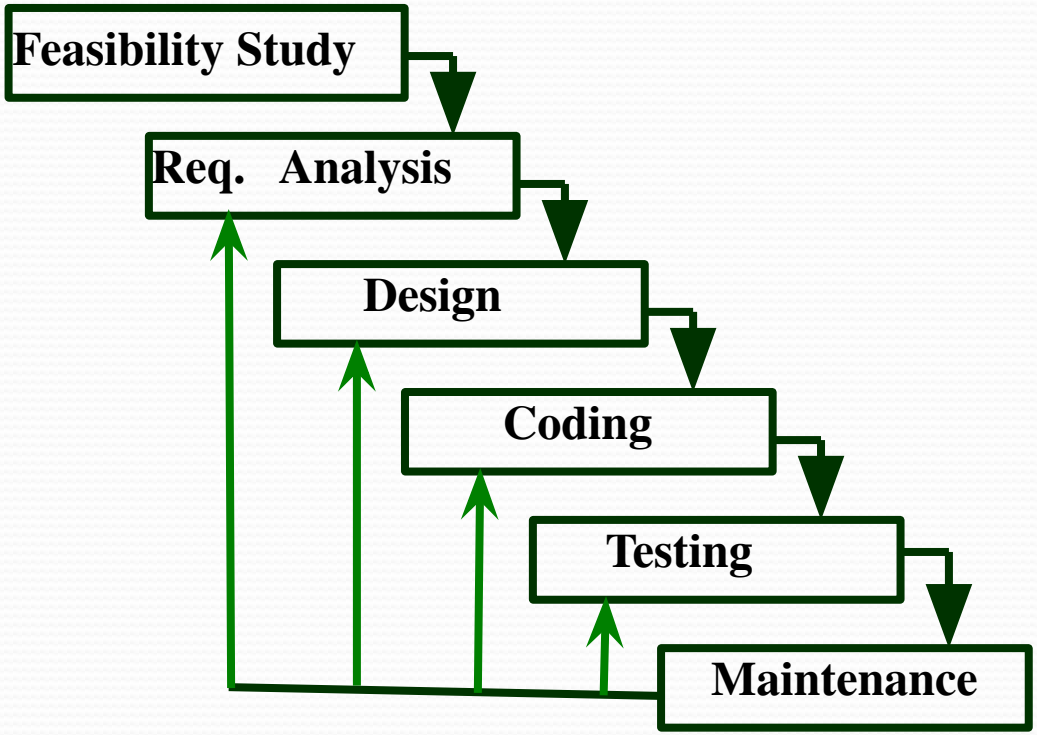
Iterative Waterfall Model (CONT.)

- Defects usually get detected much later in the life cycle:
  - For example, a design defect might go unnoticed till the coding or testing phase.

Iterative Waterfall Model (CONT.)

- Once a defect is detected:
  - we need to go back to the phase where it was introduced
  - redo some of the work done during that and all subsequent phases.
- Therefore we need feedback paths in the classical waterfall model.

Iterative Waterfall Model (CONT.)



Iterative Waterfall Model (CONT.)

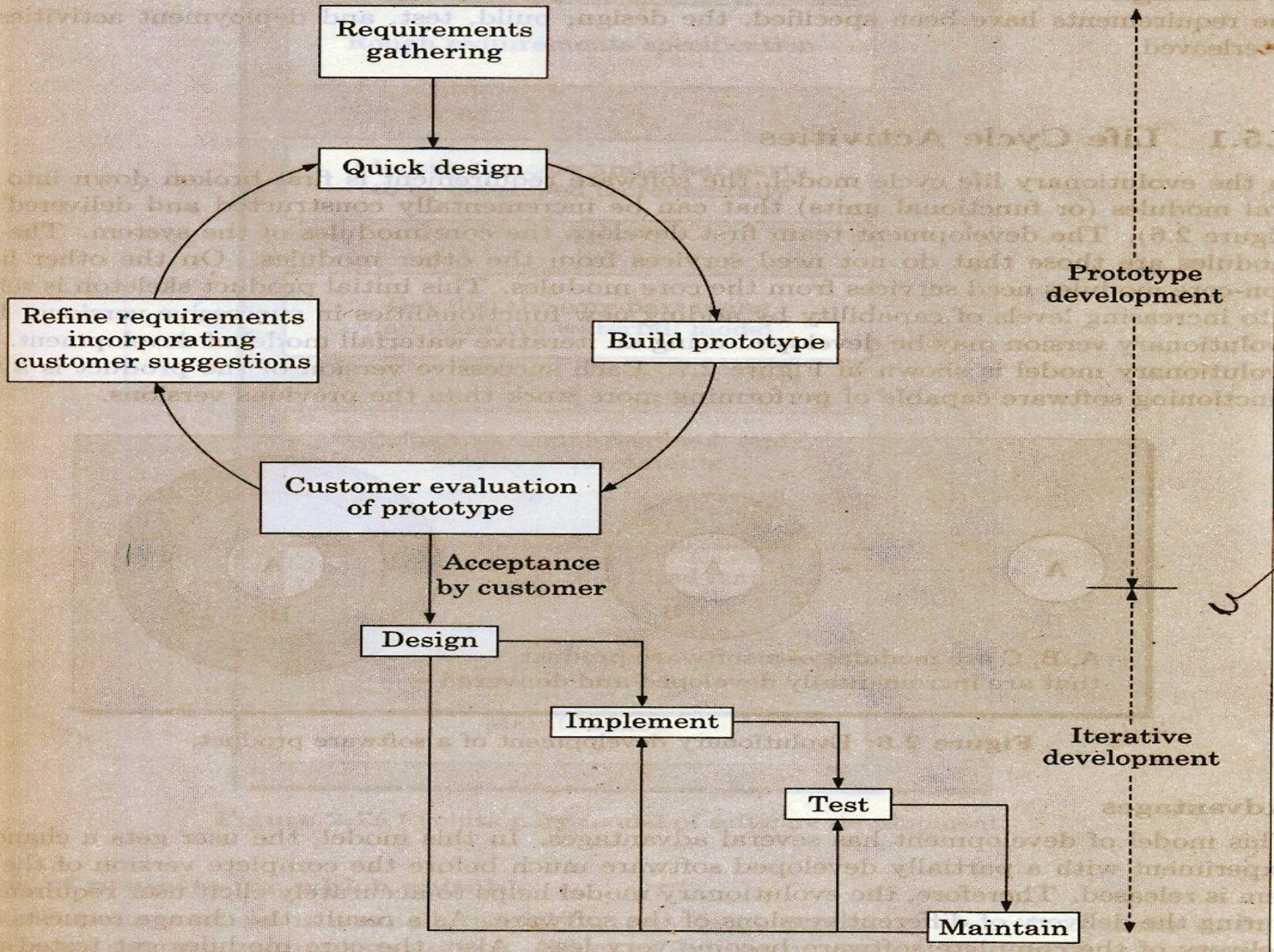
- Errors should be detected
  - in the same phase in which they are introduced.
- For example:
  - if a design problem is detected in the design phase itself,
    - the problem can be taken care of much more easily
    - than say if it is identified at the end of the integration and system testing phase.

Phase containment of errors

- Reason: rework must be carried out not only to the design but also to code and test phases.
- The principle of detecting errors as close to its point of introduction as possible:
  - is known as **phase containment of errors.**
- Iterative waterfall model is by far the most widely used model.
  - Almost every other model is derived from the waterfall model.

Prototyping Model

- Before starting actual development,
  - a working prototype of the system should first be built.
- A prototype is a toy implementation of a system:
  - limited functional capabilities,
  - low reliability,
  - inefficient performance.


Requirements gathering
Quick design
Refine requirements incorporating customer suggestions
Build prototype
Customer evaluation of prototype
Acceptance by customer
Design
Implement
Test
Maintain
Prototype development
Iterative development

Reasons for developing a prototype

- Illustrate to the customer:
  - input data formats, messages, reports, or interactive dialogs.
- All the requirements are not clear.

Prototyping Model (CONT.)

- The third reason for developing a prototype is:
  - it is impossible to "get it right" the first time,
  - we must plan to throw away the first product
    - if we want to develop a good product.

Prototyping Model (CONT.)

- Start with approximate requirements.
- Carry out a quick design.
- Prototype model is built using several short-cuts:
  - Short-cuts might involve using inefficient, inaccurate, or dummy functions.
    - A function may use a table look-up rather than performing the actual computations.

Prototyping Model (CONT.)

- The developed prototype is submitted to the customer for his evaluation:
  - Based on the user feedback, requirements are refined.
  - This cycle continues until the user approves the prototype.
- The actual system is developed using the classical waterfall approach.

Prototyping Model (CONT.)

- Requirements analysis and specification phase becomes redundant:
- Design and code for the prototype is usually thrown away:
  - However, the experience gathered from developing the prototype helps a great deal while developing the actual product.

Prototyping Model (CONT.)

- Even though construction of a working prototype model involves additional cost ---overall development cost might be lower for:
  - systems with unclear user requirements,
  - systems with unresolved technical issues.
- Many user requirements get properly defined and technical issues get resolved:
  - these would have appeared later as change requests and resulted in incurring massive redesign costs.

Evolutionary Model

- Evolutionary model (aka incremental model):
  - The system is broken down into several modules which can be incrementally implemented and delivered.
- First develop the core modules of the system.
- The initial product skeleton is refined into increasing levels of capability:
  - by adding new functionalities in successive versions.

Evolutionary Model (CONT.)

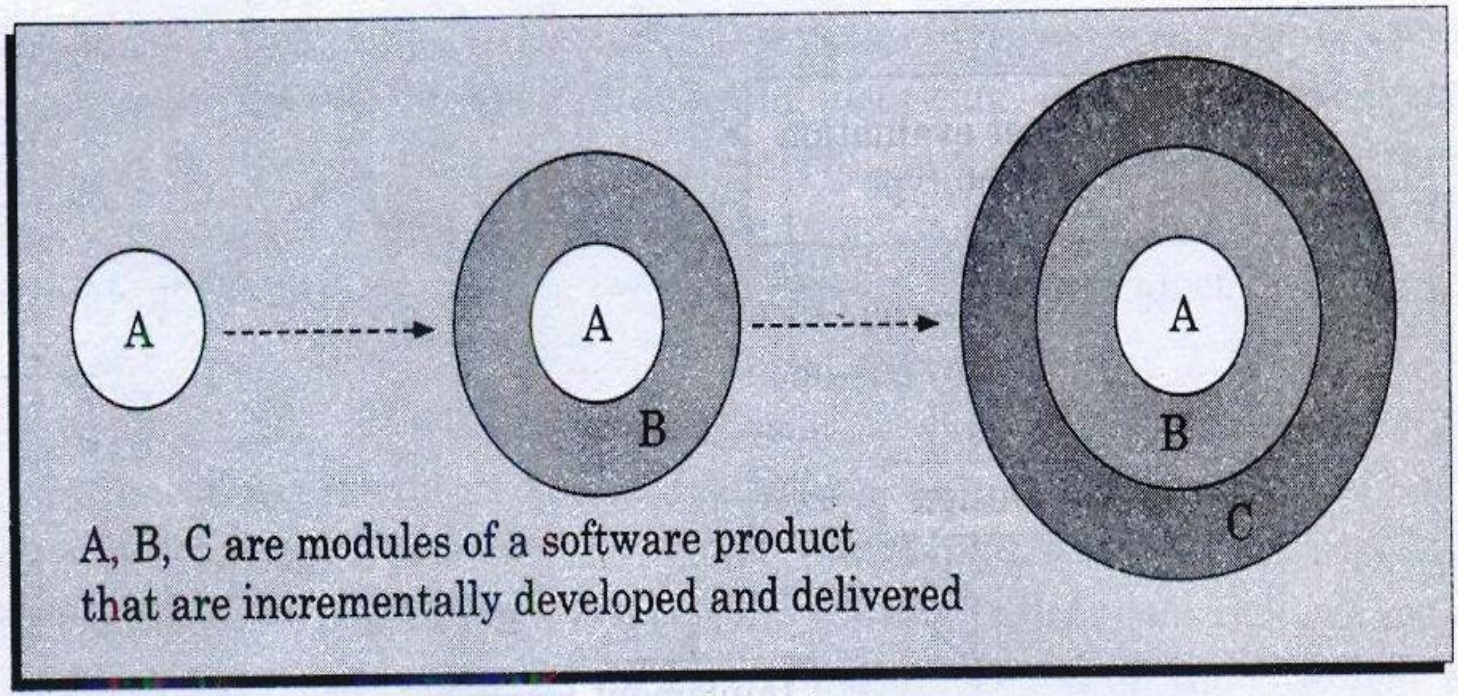


Figure 2.6: Evolutionary development of a software product.

Evolutionary Model

- This model reduce risk
- Incremental process may be incorporated rapidly.
- Software requirements changes rapidly.Then this model is effective, when the client request for new system
- Lack of understanding the project means, more no. of refinement, fill the system is accepted by the client.
- Initially it seems that there is no technical solution,but refinement is required.

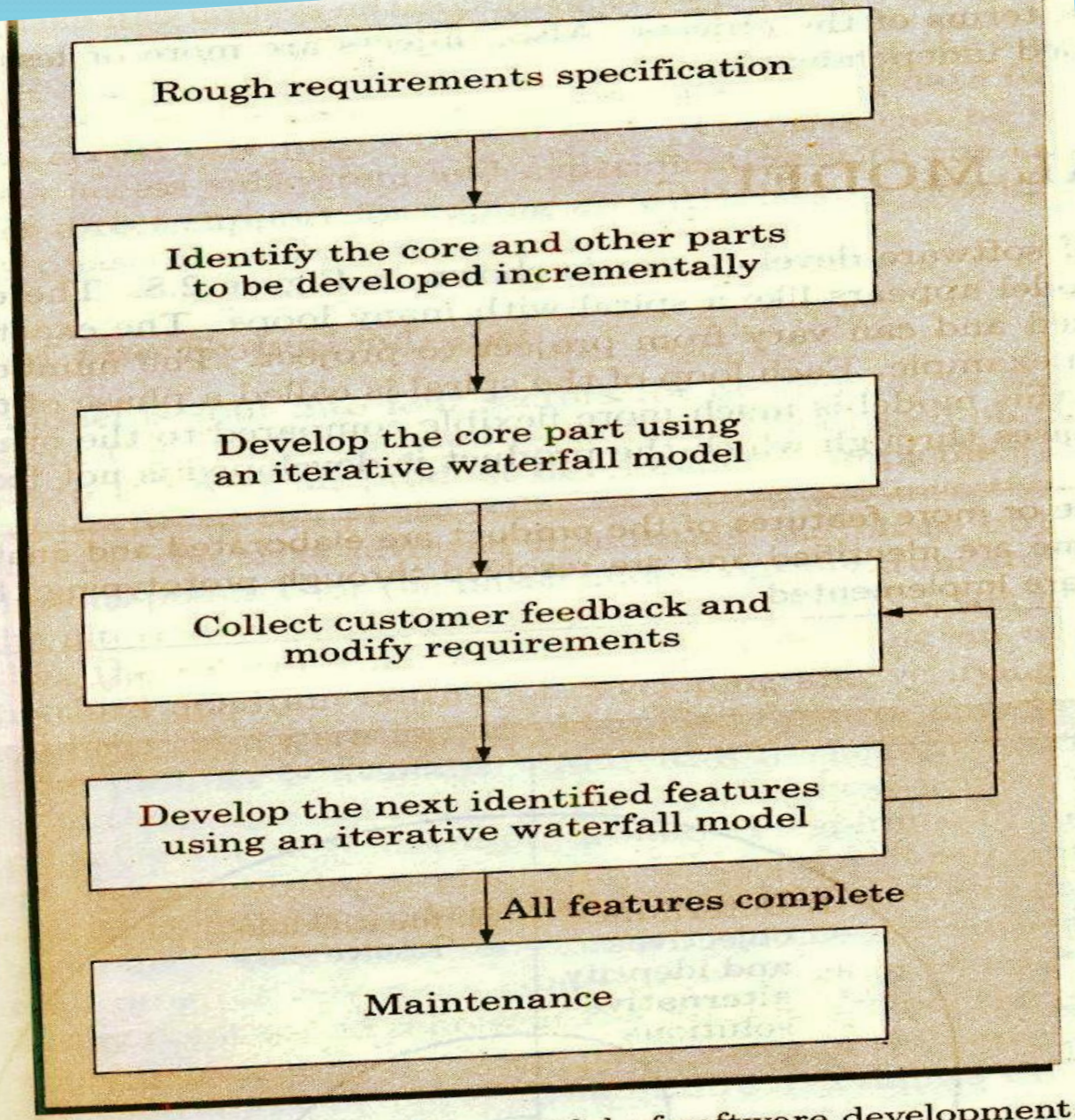


Figure 2.7: Evolutionary model of software development.

Advantages of Evolutionary Model

1. Users get a chance to experiment with a partially developed system:
   - much before the full working version is released,
2. Helps finding exact user requirements:
   - much before fully working system is developed.
3. Core modules get tested thoroughly:
   - reduces chances of errors in final product.
4. Effort for prototype is not wanted.
5. Faster than water fall model
6. High level interaction from the beginning.
7. Discovery of new methods may be involved

Disadvantages of Evolutionary Model

1. Often, difficult to subdivide problems into functional units:
   - which can be incrementally implemented and delivered.
   - evolutionary model is useful for very large problems,
     - where it is easier to find modules for incremental implementation.
2. Difficult to plan as amount of effort is uncertain
3. Documentation may be neglected
4. Poor structure coding.

Spiral Model

- In spiral model activities are organized in the form of spiral.The spiral has many cycle.
- Exact no of loops are not fixed
- Each loop of the spiral represents a phase S/W process.
  - "Risk":It is essentially any adverse circumstances that might hamper the successful completion of software project:Ex: Accessing data form a remote database can be that data access rate might be too slow.The risk can be resolved by building a prototype of the data access subsystem

Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Spiral Model

- "Radius" of the spiral at any represents the cost incurred in the project till them.
- "Angular Dimension" represents the progress made in the current phase.
- It is called Meta Model
  - A SINGLE-LOOP spiral actally represents the waterfall model
  - It uses prototyping as a risk reduction mechanism.
  - The iterations along the spiral model can be considered as evolutionary level through which the complete system built.

Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Spiral Model

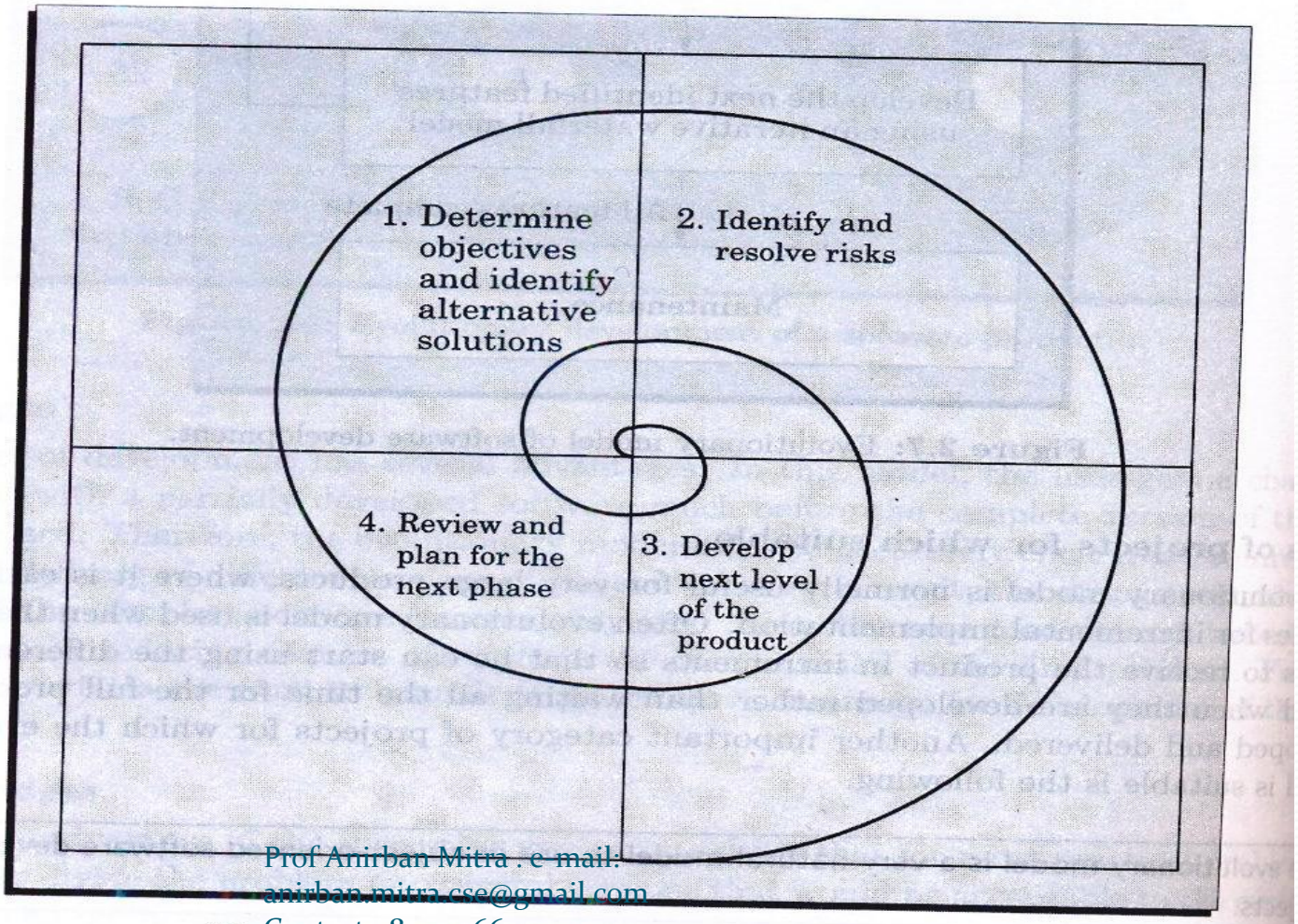
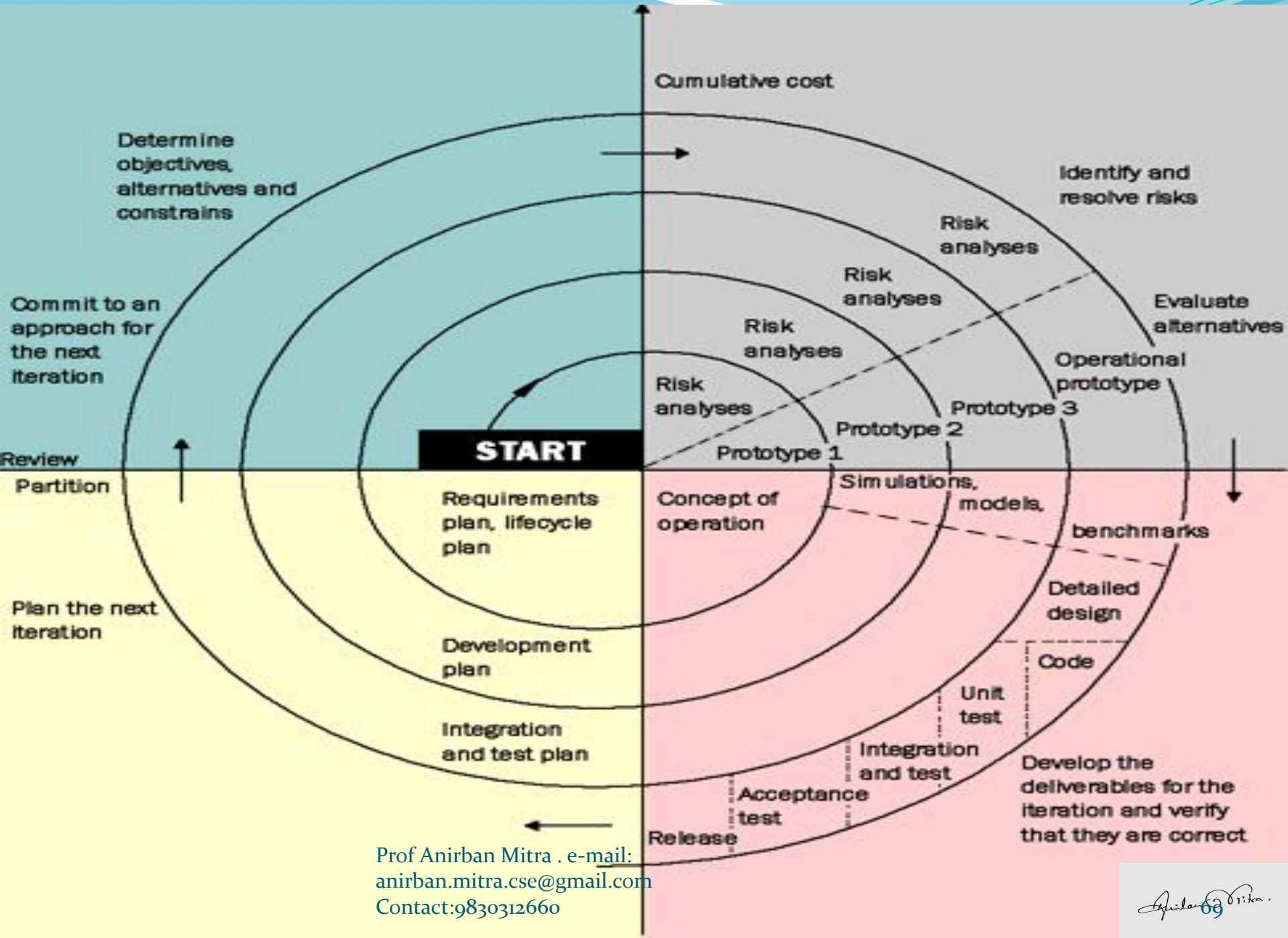


Figure 2.8: Spiral model of software development.

Prof Anirban Mitra . e-mail: anirban.mitra.cse@gmail.com
Contact:9830312660



Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Spiral Model begins with…

- Objective setting
- Identifying alternatives
- Risk assessment & reduction
- Development & validation
- Planning

Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Templates for a spiral round

- Objective: Goal of the analysis
- Constraints: Factors that limits possibilities.
- Alternatives:Different ways to active objective
- Risk: Analyzing the dangers associated with every alternatives.
- Risk Reduction: Strategies incorporated to reduce the risk
- Result:Outcome of risk reduction
- Plan:How to approach to the next phase
- Commitment:Management decision regarding continuation

Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Spiral Model

- Advantages:
  - For high risk project this model may be better than any other model
  - Lead time means decision of top level management
  - First level may be cancled and second cycle begins with new task

Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Spiral Model

- Disadvantages:
  - At some extent spiral model is theoritical and hence not possible to follow it at all the phases.The program developers don't like to adopt this model as the theoritical approach is not allways suitable with their practical work.

Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Spiral Model as a Meta Model

- The spiral model is called a meta model since it encompasses all other life cycle models. Risk handling is inherently built into this model. The spiral model is suitable for development of technically challenging software products that are prone to several kinds of risks. However, this model is much more complex than the other models – this is probably a factor deterring its use in ordinary projects.

Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Comparison of different life-cycle models

- The classical waterfall model can be considered as the basic model and all other life cycle models as embellishments of this model. However, the classical waterfall model can not be used in practical development projects, since this model supports no mechanism to handle the errors committed during any of the phases.

Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

Comparison of different life-cycle models…

- This problem is overcome in the iterative waterfall model. The iterative waterfall model is probably the most widely used software development model evolved so far. This model is simple to understand and use. However, this model is suitable only for well-understood problems; it is not suitable for very large projects and for projects that are subject to many risks.

- The prototyping model is suitable for projects for which either the user requirements or the underlying technical aspects are not well understood. This model is especially popular for development of the user-interface part of the projects.

- The evolutionary approach is suitable for large problems which can be decomposed into a set of modules for incremental development and delivery. This model is also widely used for object-oriented development projects. Of course, this model can only be used if the incremental delivery of the system is acceptable to the customer.

Prof Anirban Mitra . e-mail:
anirban.mitra.cse@gmail.com
Contact:9830312660

- The spiral model is called a meta model since it encompasses all other life cycle models. Risk handling is inherently built into this model. The spiral model is suitable for development of technically challenging software products that are prone to several kinds of risks. However, this model is much more complex than the other models – this is probably a factor deterring its use in ordinary projects.