Software Processes

The software process

- The software process is a structured set of activities whose goal is development or evolution of software. These activities may involve the development of software from a standard programming language like C, C++or Java. Increasingly, however, new software is developed by extending and modifying existing systems and by configuring and integrating off-the-shelf software or system components.
- Software processes are complex and, like all intellectual and creative processes, rely on people making decisions and judgements.
- Because of the need for judgement and creativity, attempts to automatic processes have met with limited success. CASE tools can support some process activities.

Some Fundamental activities which are common to all software process.

- **Software Specification**: The functionality of software and constraints on its operation must be defined
- **Software Design and Implementation**: The software to meet the specifications must be produced.
- **Software Validation**: The software must be validated to ensure that it does what the customer wants
- **Software Evolution**: The software must evolve to meet changing customer needs.

The software Process Model

- A software process model is an abstract representation of a process.
- It presents a description of a process from some particular perspective
- To solve actual problems in an industry setting, a software engineer or a team of engineers must incorporate a development strategy that encompasses the process, methods, and tools and the generic phases.
- This strategy is often referred to as a *process model* or a *software engineering paradigm.*
- A process model for software engineering is chosen based on the nature of the project and application, the methods and tools to be used, and the controls and deliverables that are required.

Generic software process models

The waterfall(Linear) model: Separate and distinct phases of specification and development
Evolutionary development: Specification and development are interleaved
Formal systems development: A mathematical system model is formally transformed to an implementation
Reuse-based development: The system is assembled from existing components

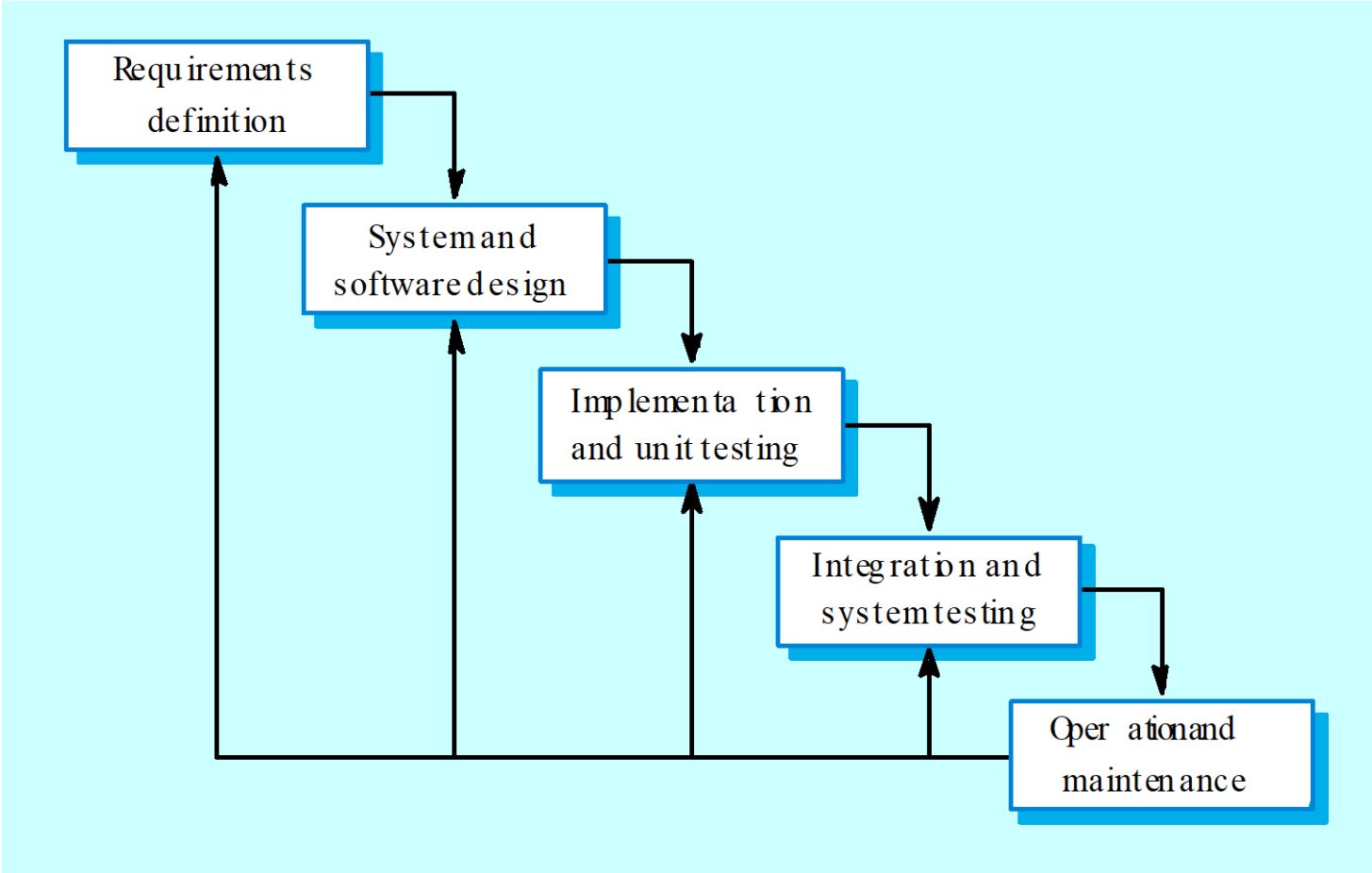
Generic software process models

- The waterfall model
  - Separate and distinct phases of specification and development.
- Evolutionary development
  - Specification, development and validation are interleaved.
- Component-based software engineering
  - The system is assembled from existing components.
- There are many variants of these models e.g. formal development where a waterfall-like process is used but the specification is a formal specification that is refined through several stages to an implementable design.

Sequential (Waterfall) model

- The first published model of the software development process was Waterfall model. It was proposed by Royce in 1970s.
- Because of the cascade from one phase to another, this model is known as waterfall/software life cycle.
- This model takes the fundamental process activities of specification, development, validation and evolution and represents them as separate phases such as requirements specification, software **design,** implementation, testing and maintenance**.**

Waterfall model



Waterfall model phases are

Requirements analysis and definition: The system's service, constraints and goals are established by consultation with system users.

They are often defined in detail and serve as a system specification.

System and software design: The system design process partitions the requirements to either hardware or software systems. It establishes overall system architecture.

Software design involves identifying and describing the fundamental software system abstractions and their relationship

Implementation and unit testing:

During this stage, the software design is realised as asset of programs and program units.

Unit testing involves verifying that each unit meets its specification

Integration and system testing:

The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met.

After testing, the software system is delivered to the customer.

Operation and maintenance:

- This is the longest life-cycle phase**.** The system is installed and put into practical use.
- Maintenance involves correcting errors which were not discovered earlier stages of the life-cycle, improving the implementation of system units and enhancing the system’s services as new requirements are discovered.

Advantages

- Easy to understand even by non-technical person, i.e customers.
- Each phase has well defined inputs and outputs.
- Easy to use as software development proceeds,
- Each stage has well defined deliverables.
- Helps the project manager in proper planning of the project.

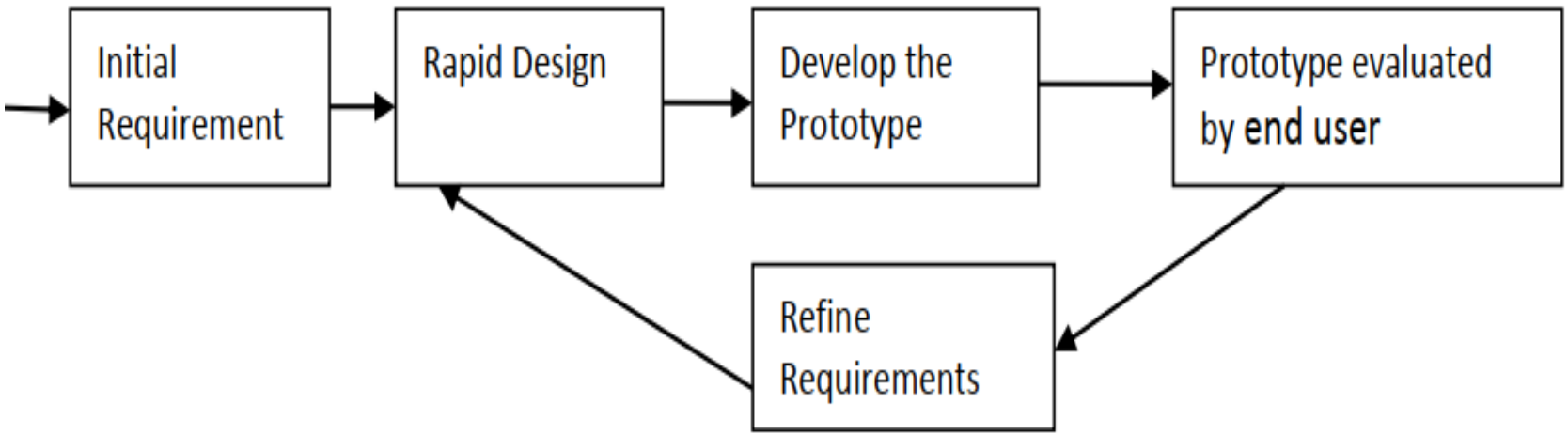
Waterfall model problems

- The drawback of the waterfall model is the difficulty of accommodating change after the process is underway because of sequential nature
- Inflexible partitioning of the project into distinct stages
- This makes it difficult to respond to changing customer requirements

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

Prototyping Model

Prototyping: An iterative process of software development in which requirements are converted to a working system that is continually revised through close work between developer and user.



Prototyping Model

- A customer defines a set of general objectives for software but does not identify
  - detailed input,
  - processing, or
  - output requirements.

  In other cases, the developer may be unsure of the efficiency of an algorithm,

  the adaptability of an operating system, or

  the form that human/machine interaction should take.
- In these, and many other situations, a *prototyping model may* offer the best approach.

- The prototyping model begins with requirements gathering.
- Developer and customer meet and define the overall objectives for

  the software,

  identify whatever requirements are known,

  and outline areas where further definition is mandatory.
- A "quick design" then occurs.

  The quick design focuses on a representation of those aspects of the software that will be visible to the customer/user

  (e.g., input approaches and output formats).

  The quick design leads to the construction of a prototype.
- The prototype is evaluated by the customer/user and used to refine requirements for the software to be developed.
- Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time enabling the developer to better understand what needs to be done.

- Prototyping model should be used
  when requirements of the system are
    not clearly understood or
    are unstable.
- *It can also be used if requirements are changing quickly.*
- This model can be successfully used for
  developing user interfaces,
  high technology software intensive systems, and
  systems with complex algorithms and interfaces.
- *It is also a very good choice to demonstrate technical feasibility of the product.*

Advantages

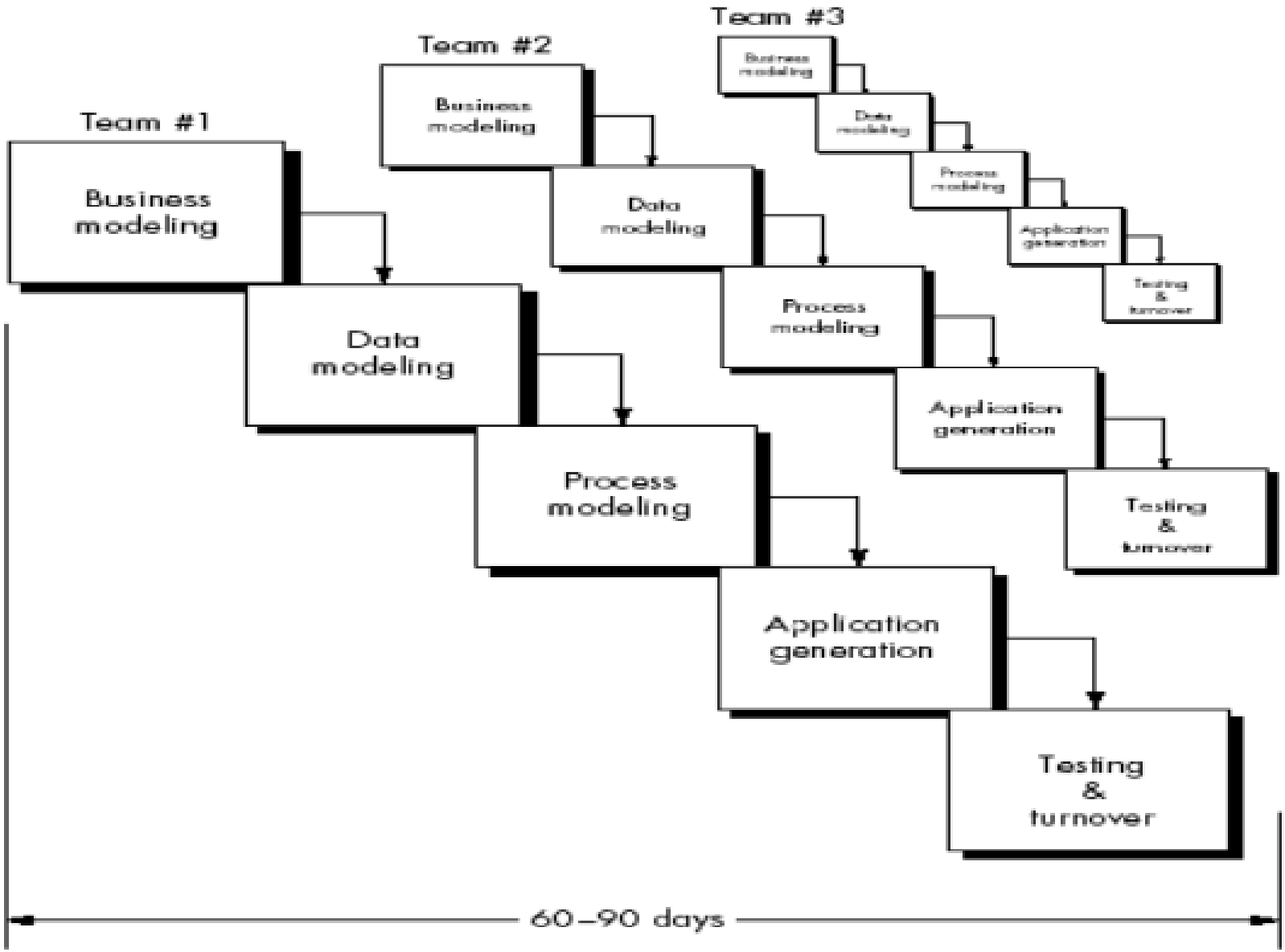
- A partial product is built in the initial stages. Therefore customers get a chance to see the product early in the life cycle and thus give necessary feedback.
- New requirements can be easily added
- Requirements become more clear resulting into an accurate product.
- As user is involved from the starting of the project, he/she tends to be more secure, comfortable and satisfied.
- Flexibility in design and development is also supported by the model.

Disadvantages

- After seeing an early prototype end users demand the actual system to be delivered.
- Developers in a hurry to build prototypes may end up with sub-optional solutions.
- If not managed properly, the iterative process of prototype demonstration and refinement can continue for long duration.
- If end user is not satisfied with initial prototype, he/she may loose interest in the project.
- Poor documentation

Rapid Application Development Model

- Rapid application development (RAD) is an incremental software development process model that *emphasizes an extremely short development cycle.*
- The RAD model is a “high-speed” adaptation of the linear sequential model in which rapid *development is achieved by using component-based construction.*
- If requirements are well understood and project scope is constrained,
- the RAD process enables a development team to create a “fully functional system” within very short time periods
- *(e.g.60 to 90 days)*


Team #1
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
Team #2
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
Team #3
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
60–90 days

It is used primarily for information systems applications, *the RAD approach* encompasses the following phases:
Business modeling: The information flow among business functions is modeled in a way that answers the following questions:

What information drives the business process?

What information is generated?
Who generates it?

Where does the information go?

Who processes it?
Data modeling: The information flow defined as part of the business modeling phase is refined into a set of data objects that are needed to support the business.

The characteristics (called *attributes*) of each object are identified and the relationships between these objects defined.
Process modeling: The data objects defined in the data modeling phase are transformed to achieve the information flow necessary to implement a business function.

Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

Application generation: RAD assumes the use of fourth generation techniques).

- Rather than creating software using conventional third generation programming languages the RAD process works to reuse existing program components (when possible) or create reusable components (when necessary).
- In all cases, automated tools are used to facilitate construction of the software.
  Testing and turnover: Since the RAD process emphasizes reuse, many of the program components have already been tested.
- This reduces overall testing time. However, new components must be tested and all interfaces must be fully exercised.
- Obviously, the time constraints imposed on a RAD project demand

Advantages

- As customer is involved at all stages of development. It leads to product achieving customer satisfaction**.**
- Usage of powerful development tools results into reduced software development cycle time.
- Makes use of reusable components, to decrease the cycle time

Drawbacks of RAD

- For large but scalable projects, RAD requires sufficient human resources to create the right number of RAD teams.
- RAD requires developers and customers who are committed to the rapid-fire activities necessary to get a system complete in a much abbreviated time frame.
- If commitment is lacking from either constituency, RAD projects will fail.
- Not all types of applications are appropriate for RAD.

- If a system cannot be properly *modularized, building the components necessary for RAD will be problematic.*
- If high performance is an issue and performance is to be achieved through tuning the interfaces to system components, the RAD approach may not work.
- *RAD is not appropriate when technical risks are high.*
- This occurs when a new application makes heavy use of new technology or
- when the new software requires a high degree of interoperability with existing computer programs.

Evolutionary Software Process Model

- The linear sequential model is designed for straight-line development.
- In essence, this waterfall approach assumes that a complete system will be delivered after the linear sequence is completed.
- The prototyping model is designed to assist the customer (or developer) in understanding requirements. In general, it is not designed to deliver a production system.
- **The evolutionary** nature of software is not considered in either of these classic software engineering paradigms.
- **Evolutionary model** is based on the idea of developing an initial implementation, exposing to this to user comment and refining it through many versions until an adequate system has been developed.
- *Specification, development and validation activities are interleaved rather than separate, with rapid feedback across activities.*
- *Evolutionary models are iterative*. They are characterized in a manner that enables software engineers to develop increasingly more complete versions of the software.

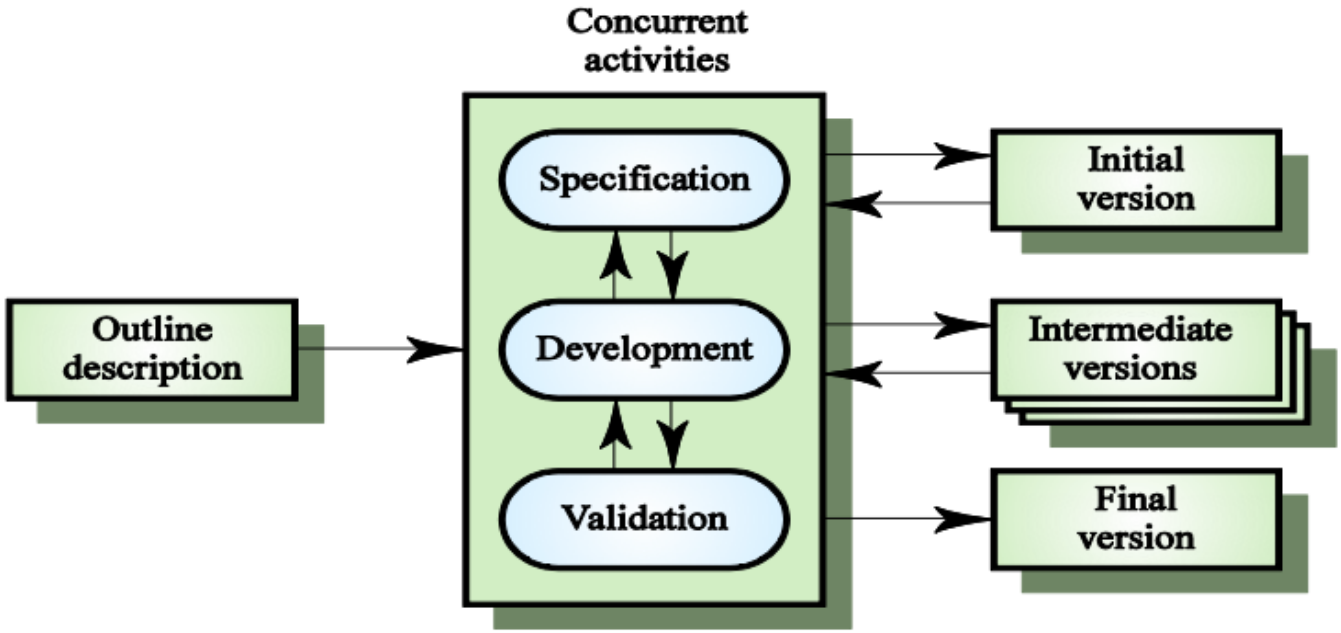


Fig: Evolutionary Development

There are two fundamental types of evolutionary development

Exploratory development

• Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements

Throw-away prototyping

• Objective is to understand the system requirements. Should start with poorly understood requirements

Advantages

- *An evolutionary approach is often more effective than waterfall approaches in producing systems that meet the immediate needs of customers.*
- *The advantage of software process that is based on an evolutionary approach is that the specification can be developed incrementally.*
- *As users develop a better understanding of their problem, this can be reflected in the software system.*

Problems towards engineering and management perspective

- ***The process is not visible:*** *Managers need regular deliverables to measure progress.*
- *If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.*
- ***Systems are often poorly structured:***
- *Continual change tends to corrupt the software structure.*
- *Incorporating software changes becomes increasingly difficult and costly.*

Incremental Model

- In an incremental model, customers identify, in outline, the services to be provided by the system.
- They identify which of the services are most important and which are least important to them.
- A number of delivery increments are then defined, with each increment providing a subset of the system functionality.
- The allocation of services to increments depends on the service priority with the highest priority services delivered first.
- Once the system increments have been identified, the requirements for the services to be delivered in the first increment are defined in detail, and that increment is developed.
- During development, further requirements analysis for later increments can take place, but requirements changes for the current increment are not accepted.

- Once increment is completed and delivered, customers can put it into services>they can experiment with the system that helps to clarify their requirements for later increments and for later versions of the current increment.
- As new increments are completed, they are integrated with existing increments so that the system functionality improves each delivered increment.
- The common services may be implemented early in the process or may be implemented incrementally as functionality is required by an increment.
- (In incremental model, rather than deliver the system as a single delivery, the development and 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
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve)

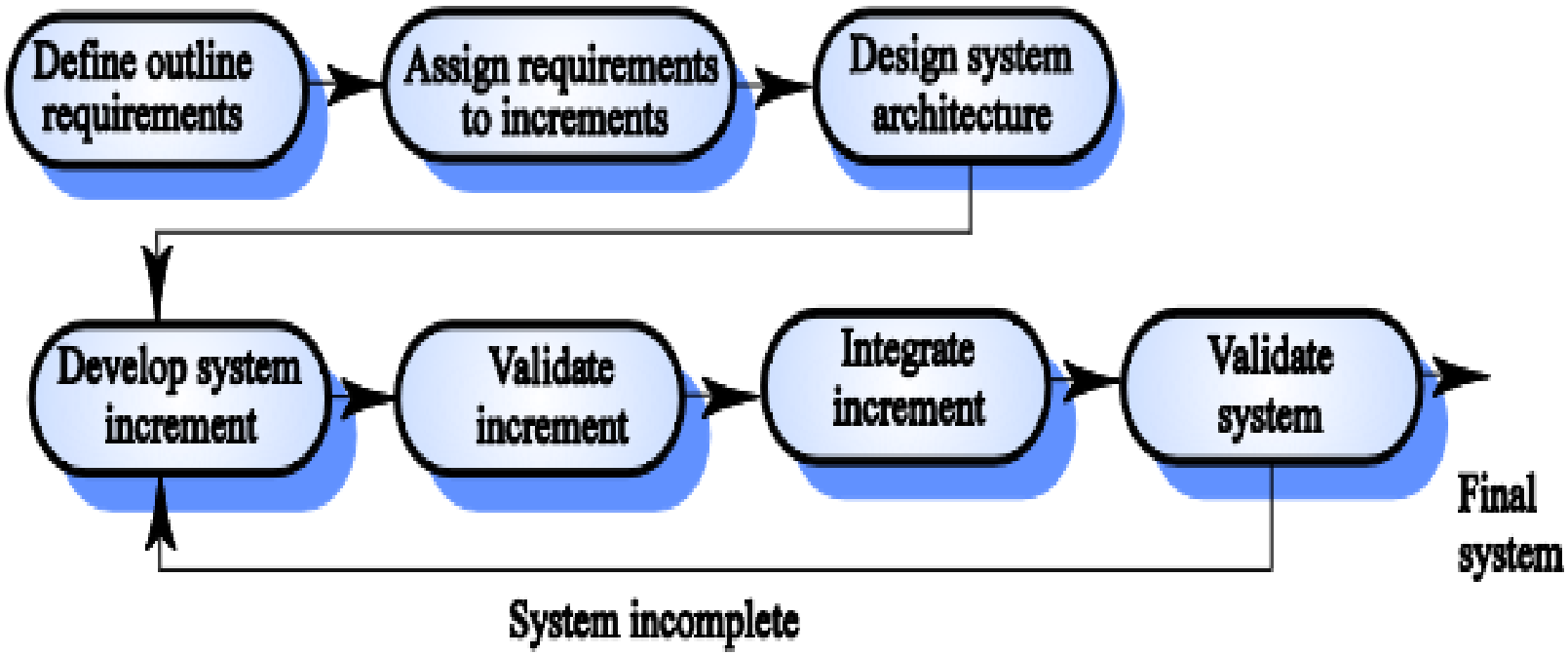


Fig: Incremental delivery

Incremental development advantages

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

Variant in this approach called Extreme programming

- New approach to development based on the development and delivery of very small increments of functionality
- Relies on constant code improvement, user involvement in the development team and pairwise programming

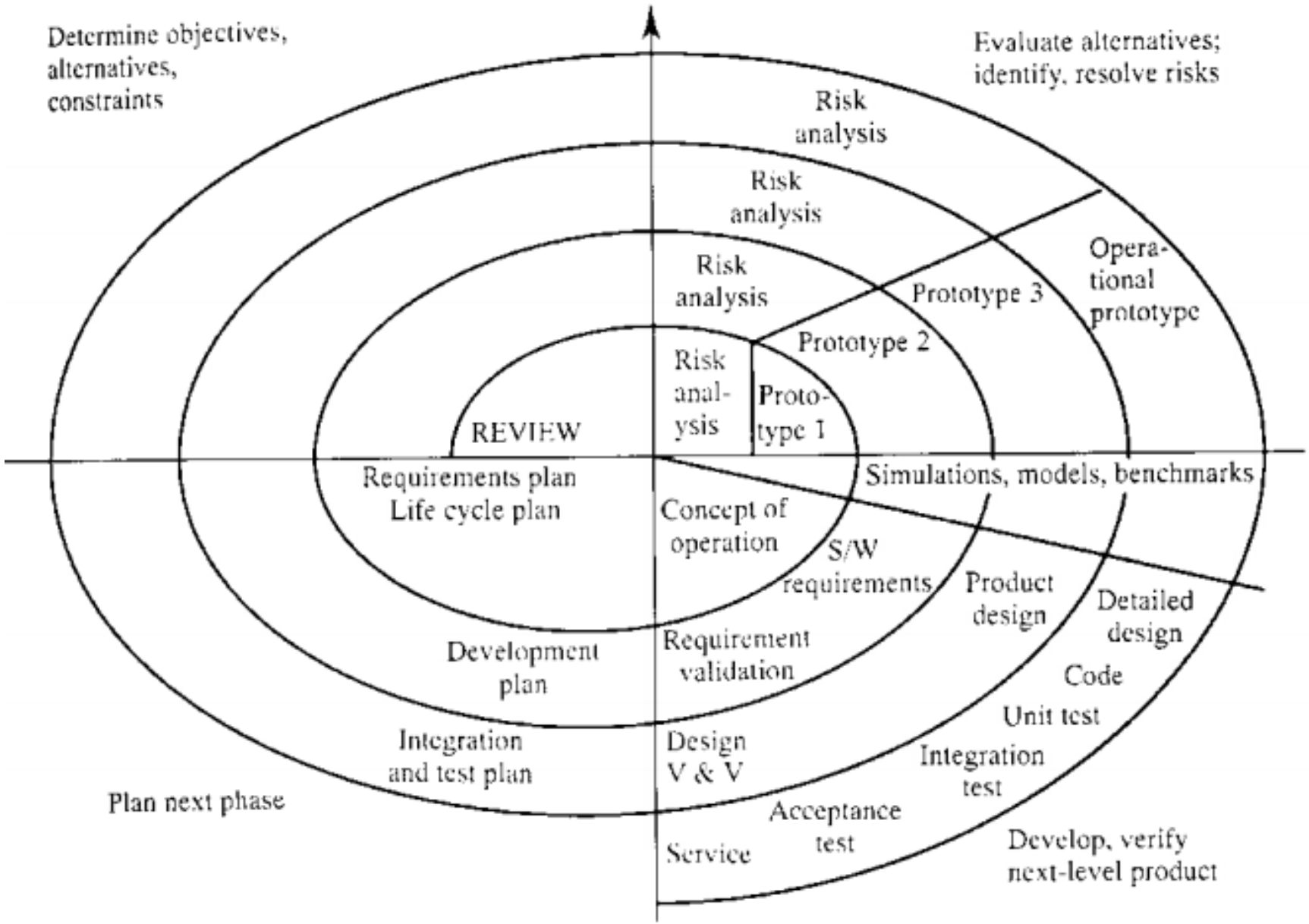
Disadvantages

- As product is delivered in parts, total development cost is higher.
- Well defined interfaces are required to connect modules developed with each phase.
- Testing of modules also results into overhead and increased cost.

Spiral development

- The spiral model was originally proposed by Boehm in 1988.
- In this model, process is represented as a spiral rather than as a sequence of activities with backtracking from one activity to another.
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required
- Risks are explicitly assessed and resolved throughout the process

Spiral Model



Spiral model sectors are

Objective setting: (Specific objectives for the phase are identified)

- In this sector, specific objectives are defined.
- Constraints on the process and product are identified and a detailed management plan is drawn up.
- Project risks are identified.
- Alternative strategies, depending on these risks, may be planned.
- **Risk assessment and reduction:** Risks are assessed and activities put in place to reduce the key risks)
- For each of the identified project risks, a detailed analysis is carried out. Steps are taken to reduce the risk.
- For example, if there is risk that requirement are inappropriate, a prototype system may be developed.

Development and validation

- A development model for the system is chosen which can be any of the generic models
- After risk evaluation, a development model for the system is chosen which can be any generic model.
- For example. the waterfall model may be the most appropriate development model if the main identified risk is sub-system integration
- **Planning: (**The project is reviewed and the next phase of the spiral is planned)

The project is reviewed and a decision made whether to continue with a further loop of the spiral.
If it is decided to continue, plans are drawn up for the next phase of the project.

Advantages

- The model tries to resolve all possible risks involved in the project starting with the highest risk.
- End users get a chance to see the product early in life cycle.
- With each phase as product is refined after customer feedback, the model ensures a good quality product.
- The model makes use of techniques like reuse, prototyping and component based design

- **Disadvantages:**
  The model requires expertise in risk management and excellent management skills.
- This model is not suitable for small projects as cost of risk analysis may exceed the actual cost of the project.
- Different persons involved in the project may find it complex to use.

Computer-Aided Software Engineering (CASE):

- **Computer-aided software engineering** (CASE) tools are software programs that automate or support the drawing and analysis of system models and provide for the translation of system models into application programs.
- Some CASE tools also provide prototyping and code generation capabilities.
  A **CASE repository** is a system developers' database.
- It is a place where developers can store system models, detailed descriptions and specifications, and other products of system development.
- Synonyms include **dictionary** and **encyclopedia**.

CASE Facilities: To use the repository, the CASE tools provide some combination of the following facilities.

Diagramming tools: are use to draw the system models required or recommended in most system development methodologies. Usually, the shapes on one system model can be linked to other models and to detailed descriptions.

Dictionary tools: are used to record, delete, edit, and output detailed documentation and specifications. The descriptions can be associated with shapes appearing on system models that were drawn with diagramming tools.

Design tools: can be used to develop system components such as inputs and outputs.

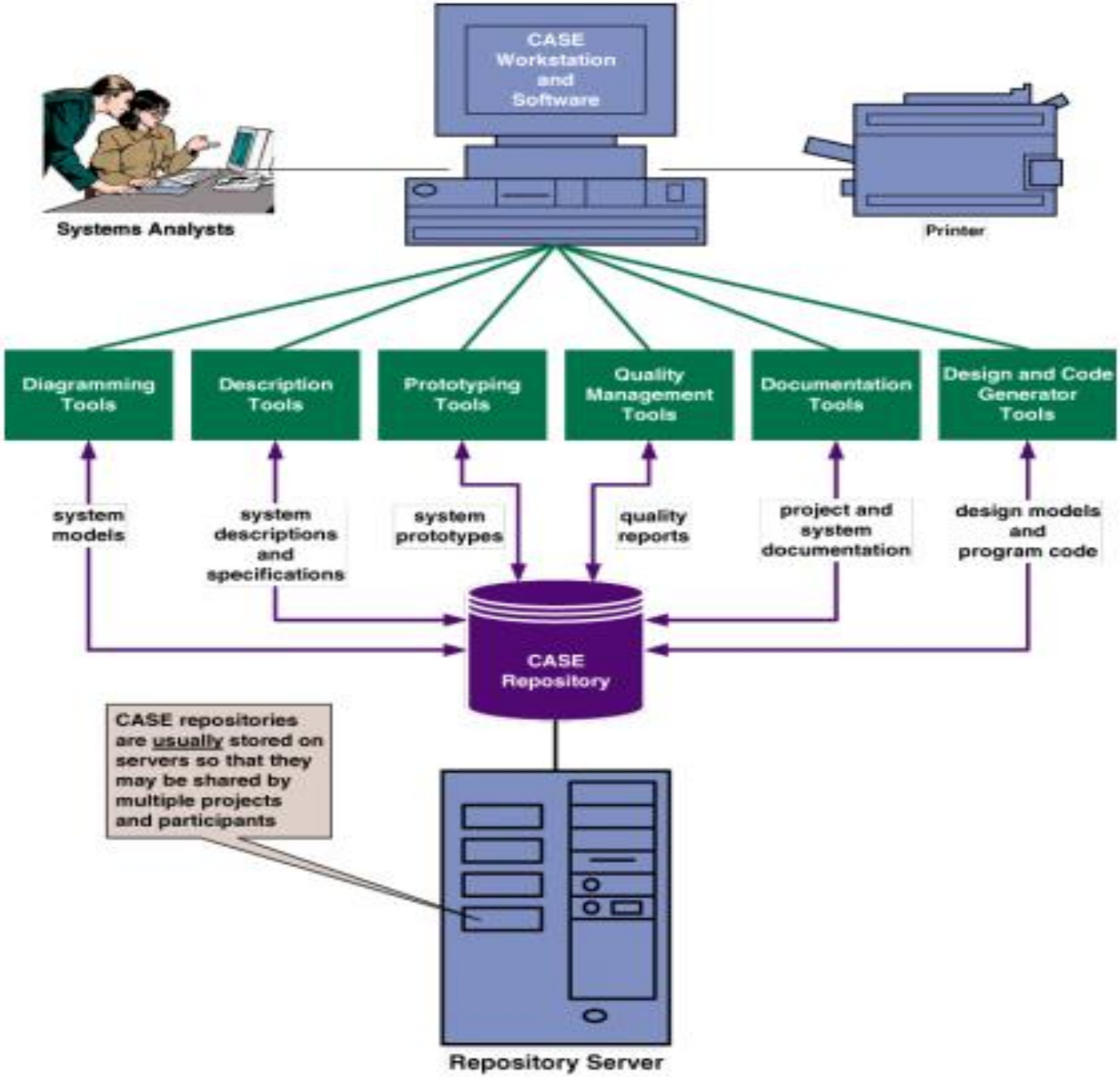
Quality management tools: analyze system models, descriptions and specifications, and designs for completeness, consistency, and conformance to accepted rules of the methodologies.

Documentation tools: are used to assemble, organize, and report on system models, descriptions and specifications, and prototype that can be reviewed by system owners, users, designers, and builders.

Design and code generator tools: automatically generate database designs and application programs or significant portions of those programs.

Forward engineering requires the systems analyst to draw system models, either from scratch or from templates. The resulting models are subsequently transformed into program code

Reverse engineering allows a CASE tool to read existing program code and transform that code into a representative system model that can be edited and refined by the systems analyst.


CASE
Workstation
and
Software
Systems Analysts
Printer
Diagramming Tools
Description Tools
Prototyping Tools
Quality Management Tools
Documentation Tools
Design and Code Generator Tools
system models
system descriptions and specifications
system prototypes
quality reports
project and system documentation
design models and program code
CASE
Repository
CASE repositories are usually stored on servers so that they may be shared by multiple projects and participants
Repository Server

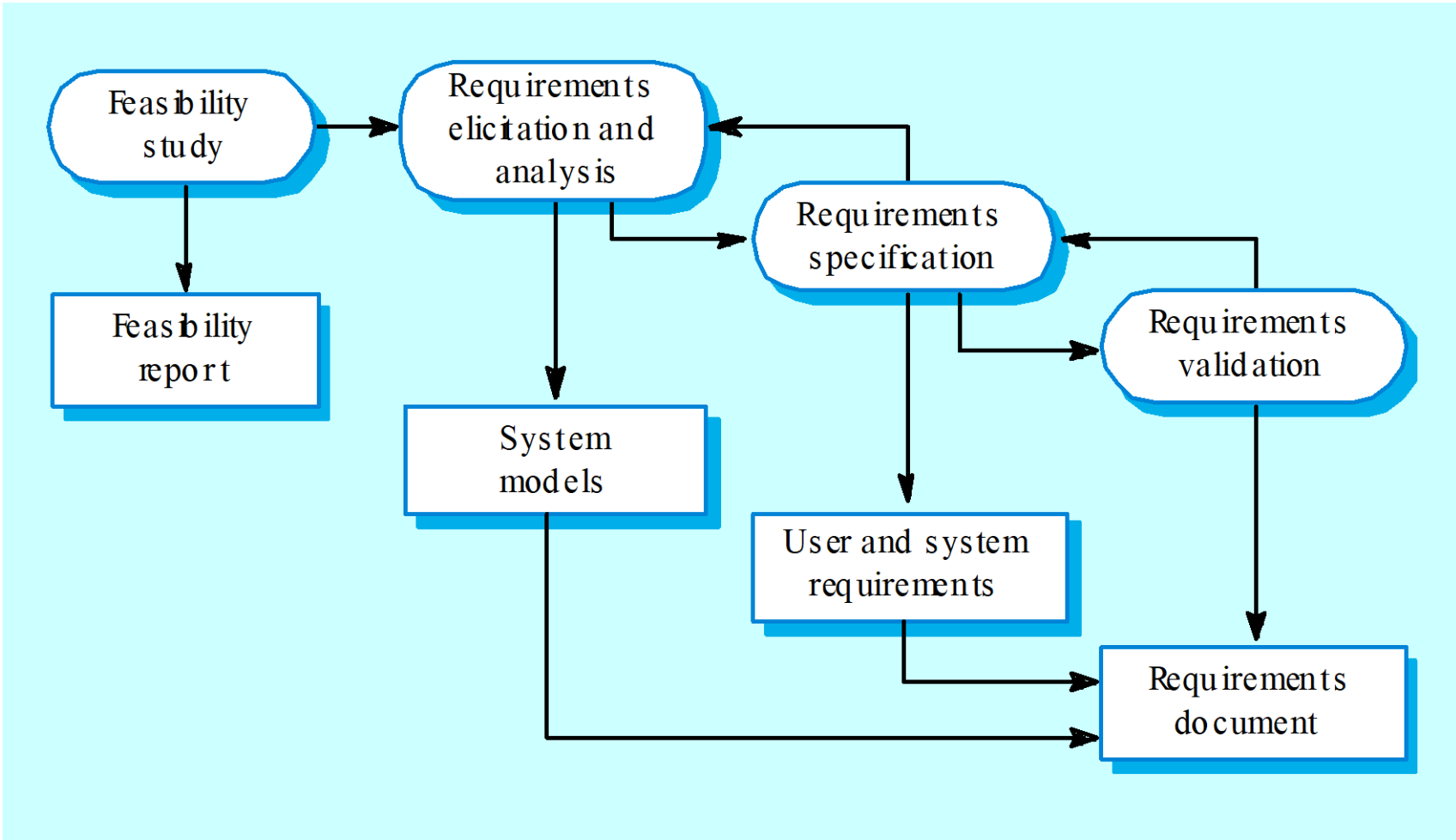
Process activities

- Software specification
- Software design and implementation
- Software validation
- Software evolution

Software specification

- The process of establishing what services are required and the constraints on the system's operation and development.
- Requirements engineering process
  - Feasibility study;
  - Requirements elicitation and analysis;
  - Requirements specification;
  - Requirements validation.

The requirements engineering process



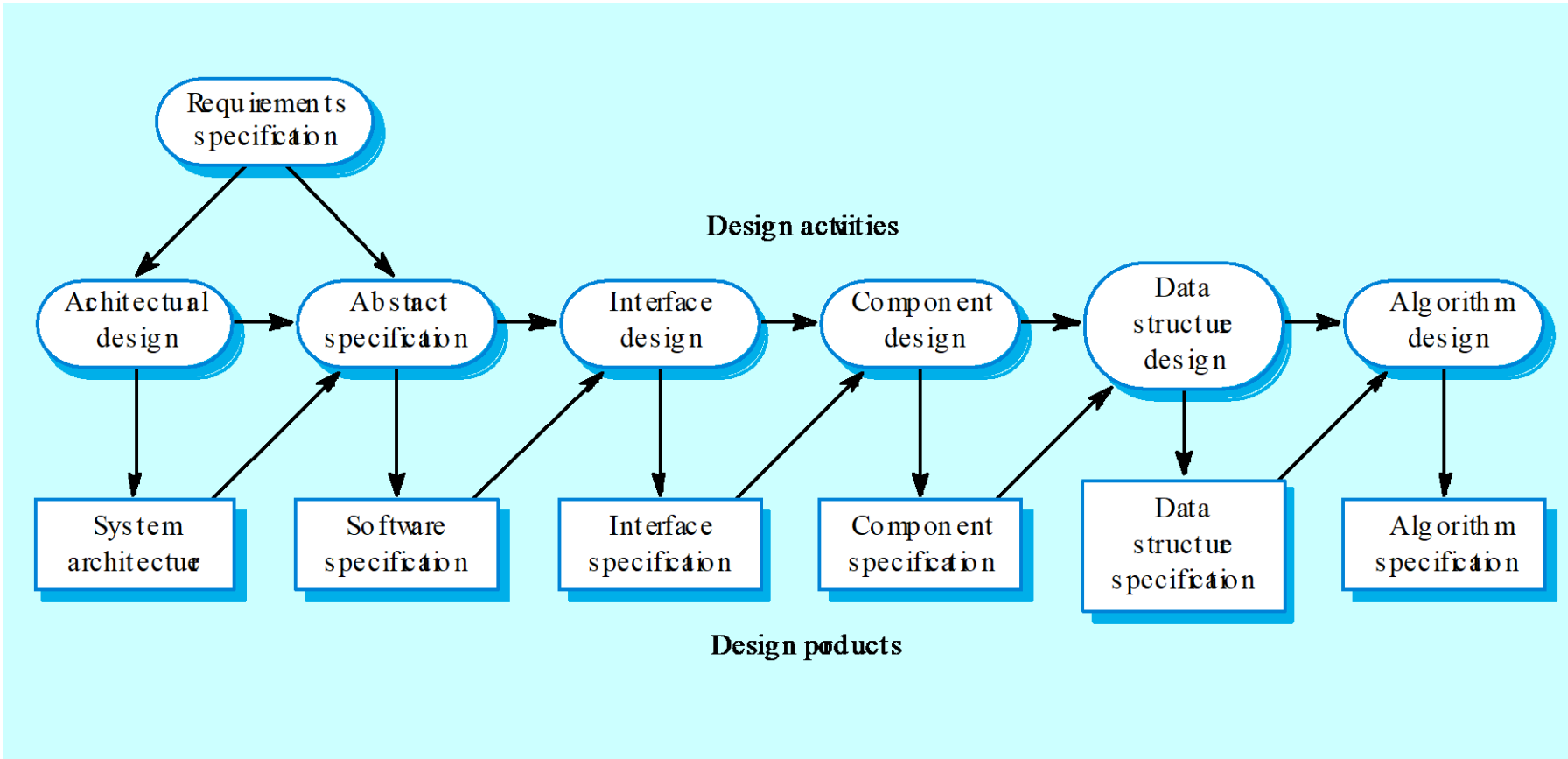
Software design and implementation

- The process of converting the system specification into an executable system.
- Software design
  - Design a software structure that realises the specification;
- Implementation
  - Translate this structure into an executable program;
- The activities of design and implementation are closely related and may be inter-leaved.

Design process activities

- Architectural design
- Abstract specification
- Interface design
- Component design
- Data structure design
- Algorithm design

The software design process



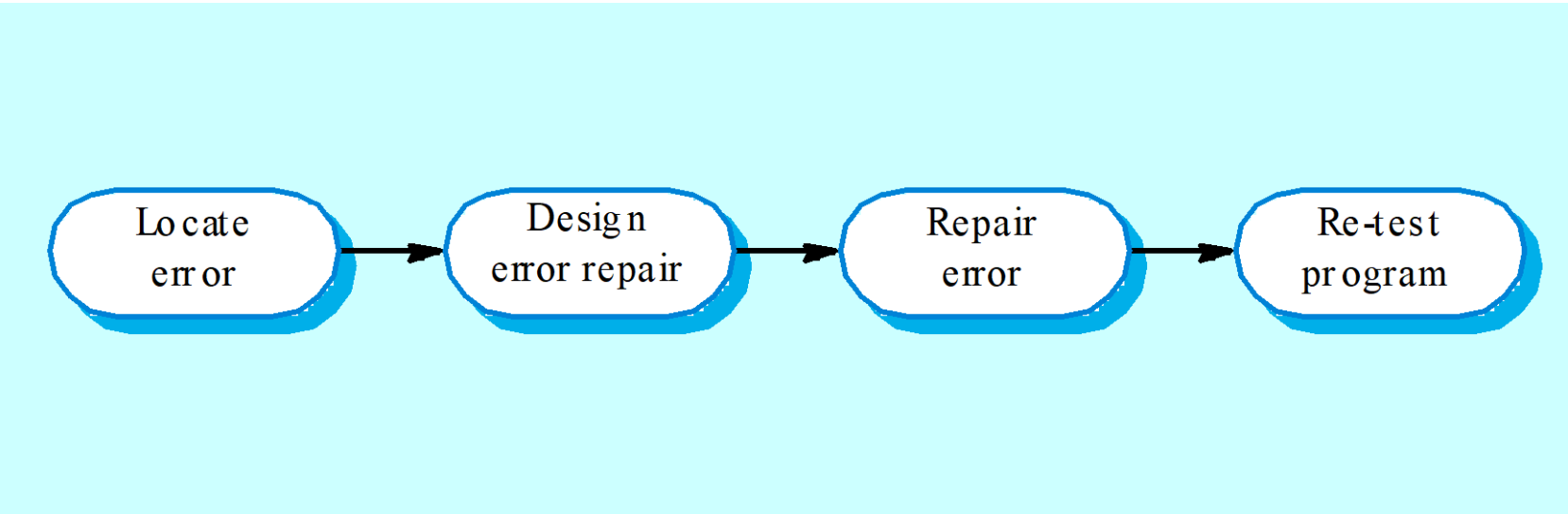
Structured methods

- Systematic approaches to developing a software design.
- The design is usually documented as a set of graphical models.
- Possible models
  - Object model;
  - Sequence model;
  - State transition model;
  - Structural model;
  - Data-flow model.

Programming and debugging

- Translating a design into a program and 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 and remove these faults in the debugging process.

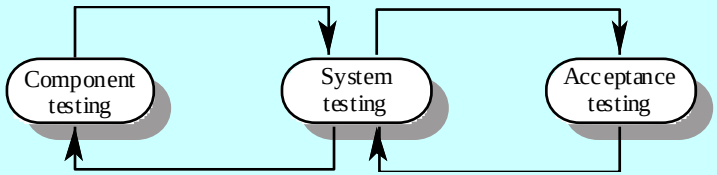
The debugging process



Software validation

- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- Involves checking and review processes and 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.

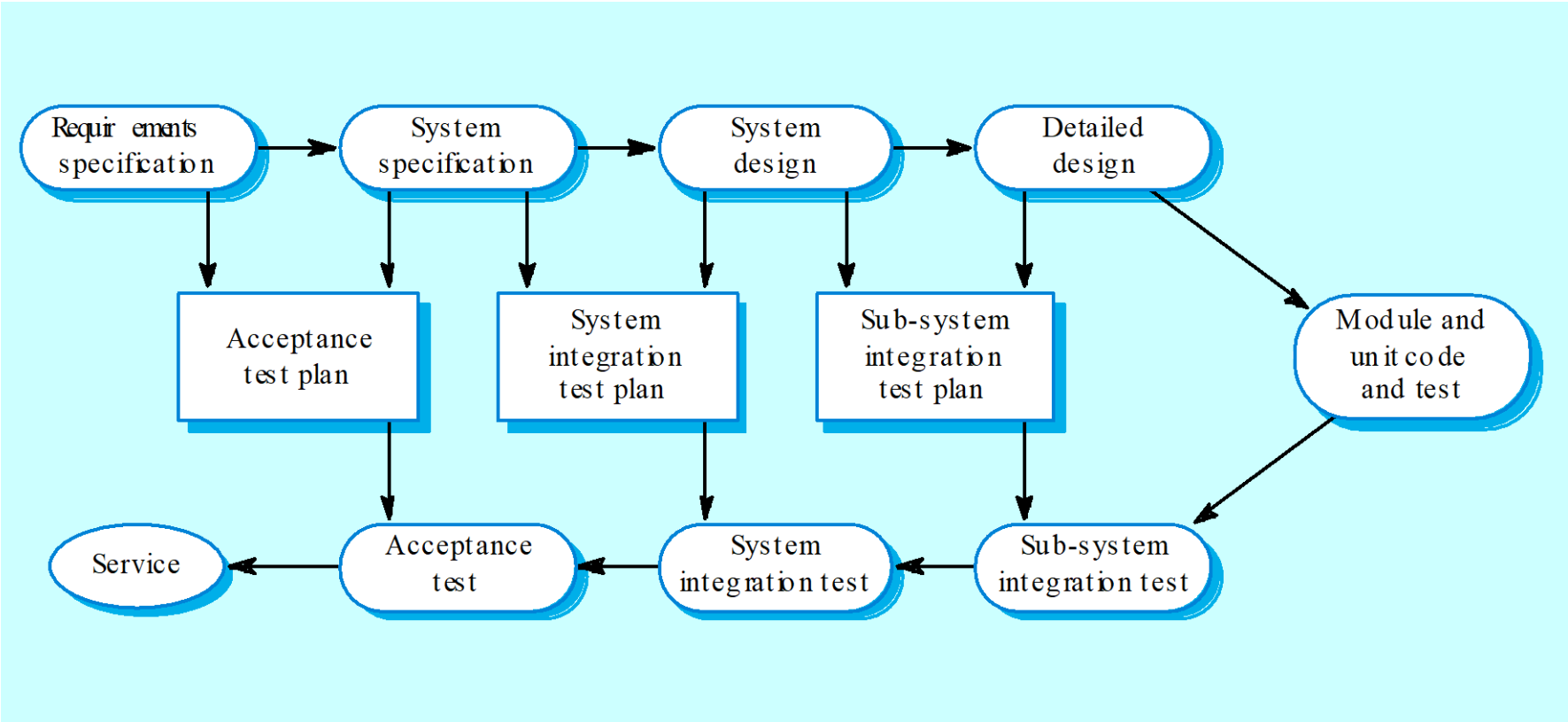
The testing process



Testing stages

- Component or unit testing
  - Individual components are tested independently;
  - Components may be functions or objects or coherent groupings of these entities.
- System testing
  - Testing of the system as a whole. Testing of emergent properties is particularly important.
- Acceptance testing
  - Testing with customer data to check that the system meets the customer’s needs.

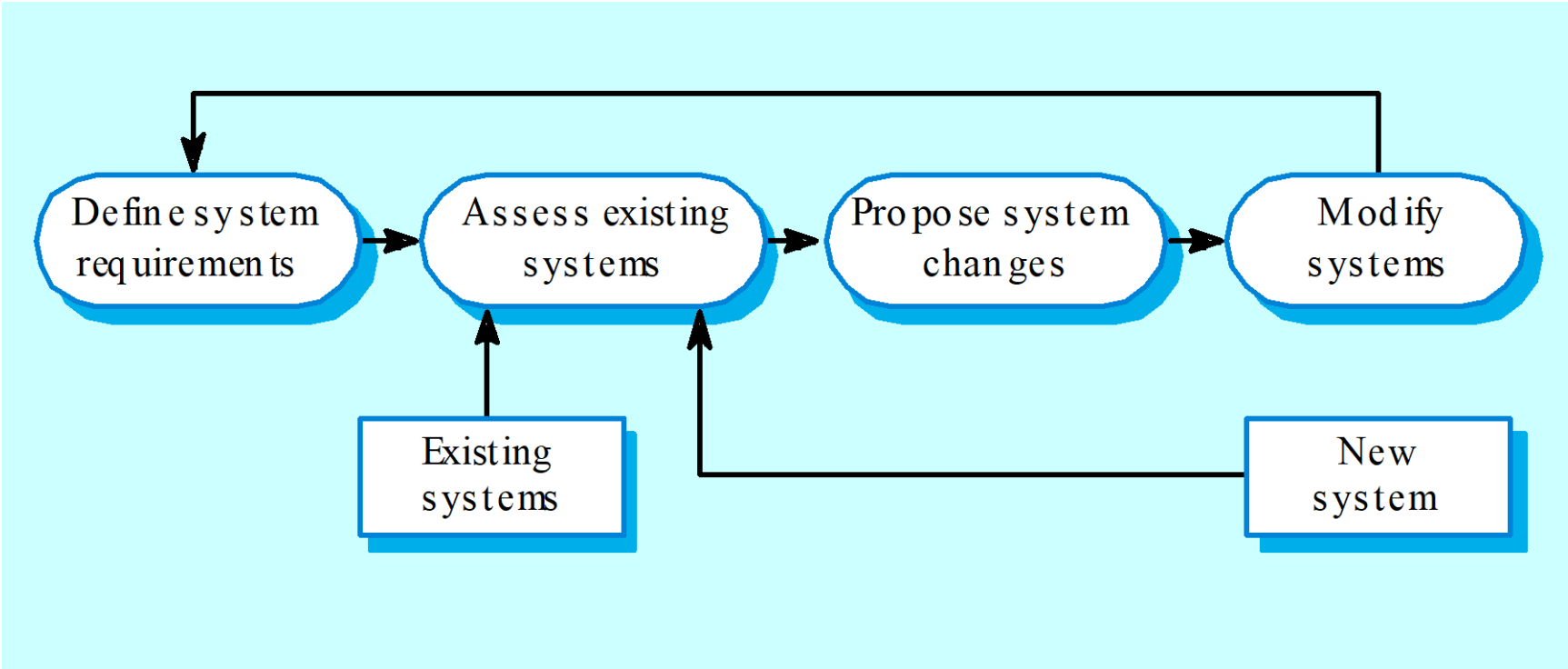
Testing phases



Software evolution

- Software is inherently flexible and can change.
- As requirements change through changing business circumstances, the software that supports the business must also evolve and change.
- Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new.

System evolution



Key points

- Requirements engineering is the process of developing a software specification.
- Design and implementation processes transform the specification to an executable program.
- Validation involves checking that the system meets to its specification and user needs.
- Evolution is concerned with modifying the system after it is in use.
- The Rational Unified Process is a generic process model that separates activities from phases.
- CASE technology supports software process activities.