جامعة الجزيرة الخاصة
ALJAZEERA PRIVATE UNIVERSITY

Introduction

to

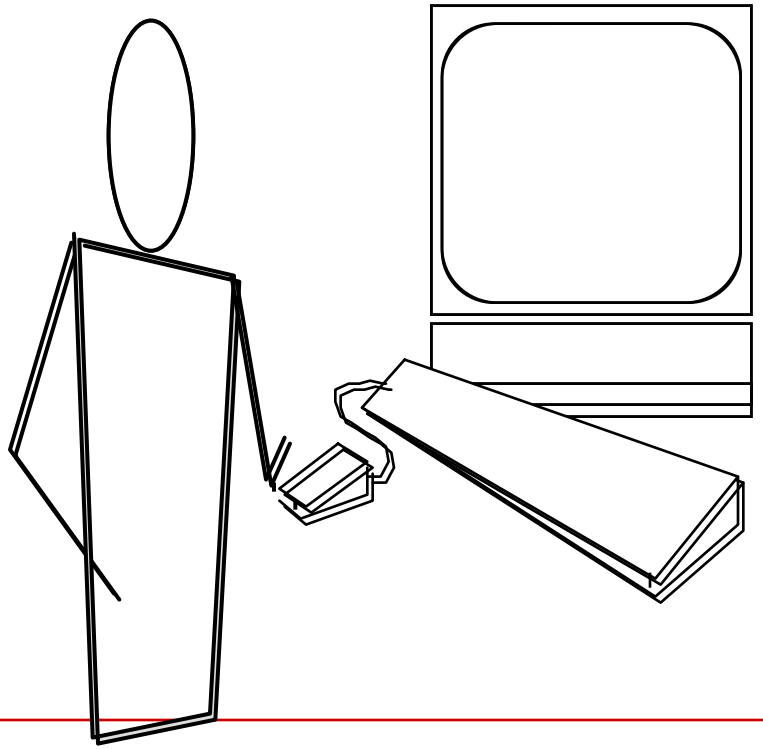
Software Engineering

What is Software?

Software is a set of items or objects that form a “configuration” that includes

- programs
- documents
- data ...

SW is a logical rather than a physical system element



What is software?

- Computer programs + associated documentation as : requirements, design models and user manuals.
- Software products may be
  - Generic = developed for a range of different customers (Excel or Word).
  - Custom = developed for a single customer according to their specification.
- New software can be created by :
  - developing new programs,
  - configuring generic software systems
  - reusing existing software.

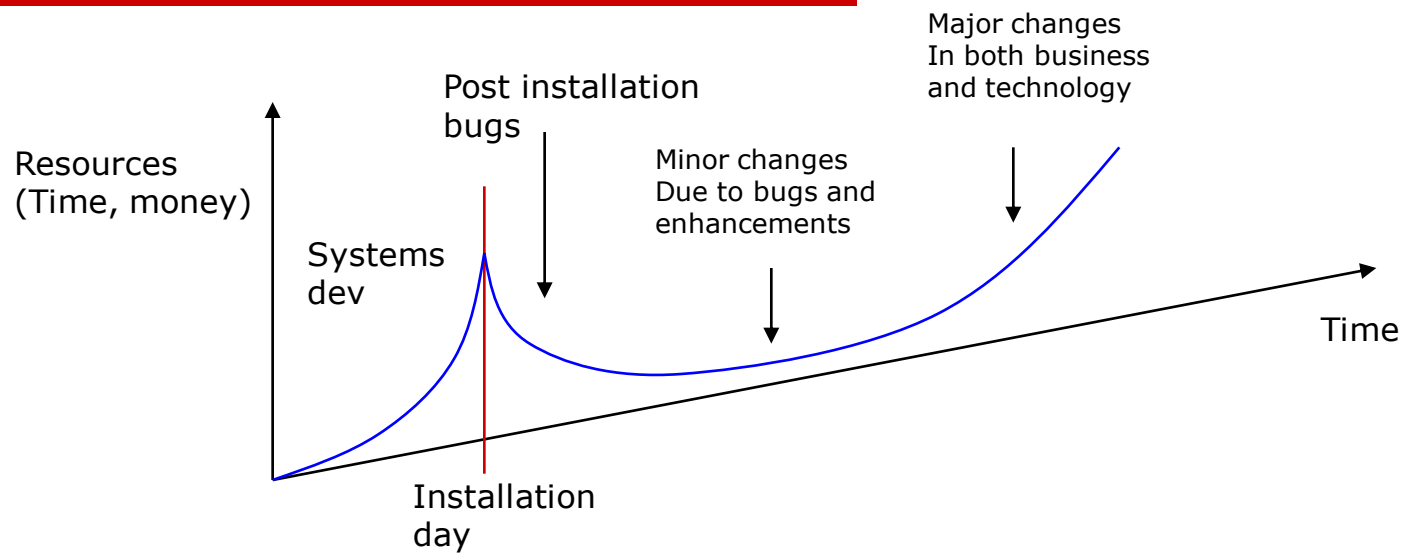
Software Applications

- System software
- Application software
- Engineering/scientific software
- Embedded software
- Web applications
- AI software
- Ubiquitous computing—wireless networks
- Cloud Computing
- Data mining
- Grid computing

Legacy Software

- Were developed decades ago and costly to maintain and risky to evolve.
- remain supportive to core business functions and are indispensable to the business
- Characterized by longevity and business criticality
- *Why must it change?*
  - software must be adapted to meet the needs of new computing environments or technology.
  - software must be enhanced to implement new business requirements.
  - software must be extended to make it interoperable with other more modern systems or databases.
  - software must be re-architected to make it viable within a network environment.

The impact of maintenance 1

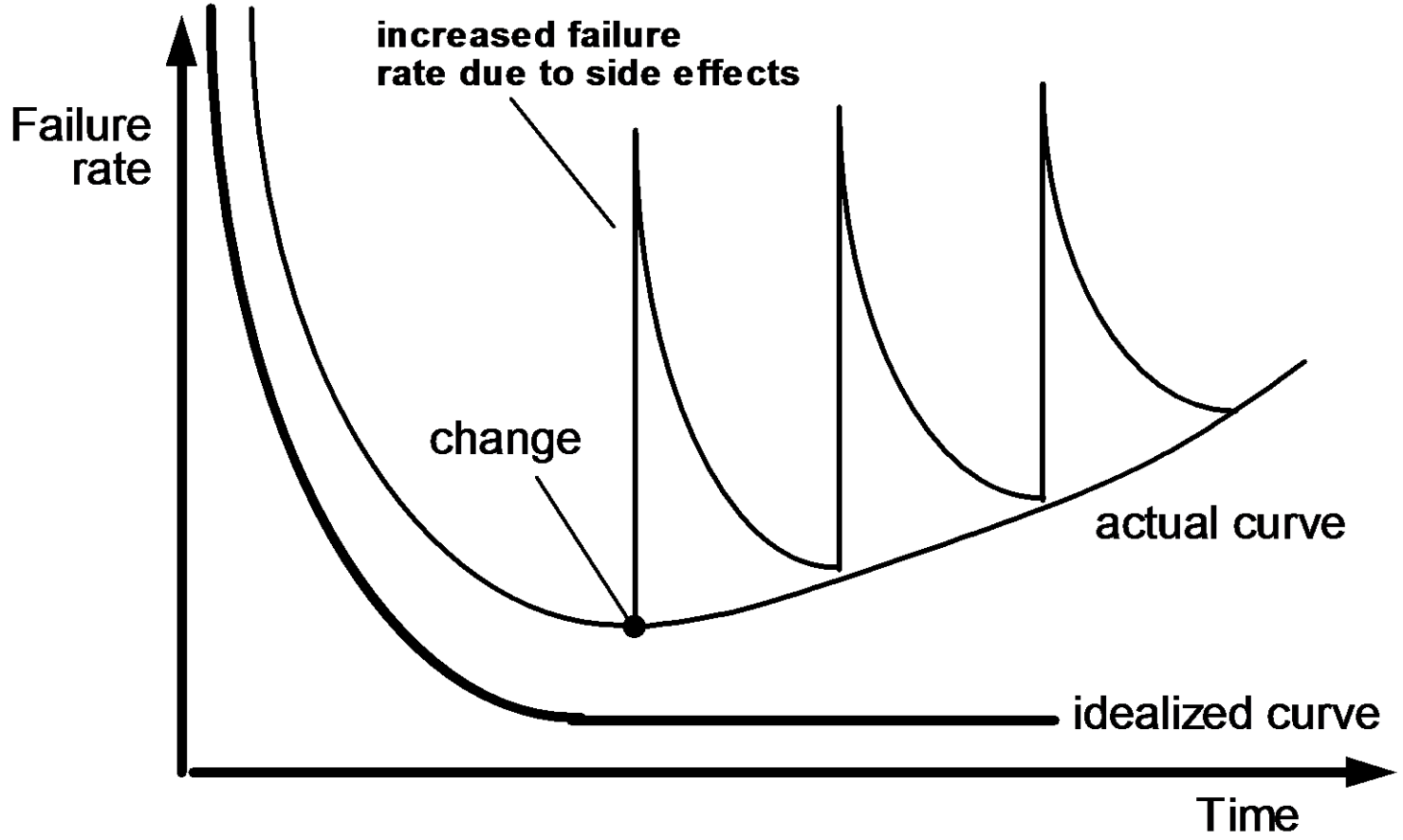


- Maintenance time represent 48% to 60% of total time spent developing systems

The impact of maintenance 2

- Maintenance is performed in 2 reasons:
  - Correct software errors
    - PC Software
    - Customized software
  - Enhance the soft capabilities in response to changing needs:
    - Users request additional features after they becom familiar with computer sys and its capabilities
    - The business changes over time
    - Hardware and software are changing at accelerated pace

Wear vs. Deterioration



Information Systems

- System types =
  - Business system
    - Realize well-defined goals (e.g plant)
    - May be devided in several sub-systems (e.g building maintenance, product assembly, agents billing, …)
  - Information system
    - Manage information that business system need (e.g store records about agents bills, ….)
    - Business re-engineering
- Information system provide a competitive advantage for business systems
  - Info helps the organization. So, Info is an organization resource, not an organization cost

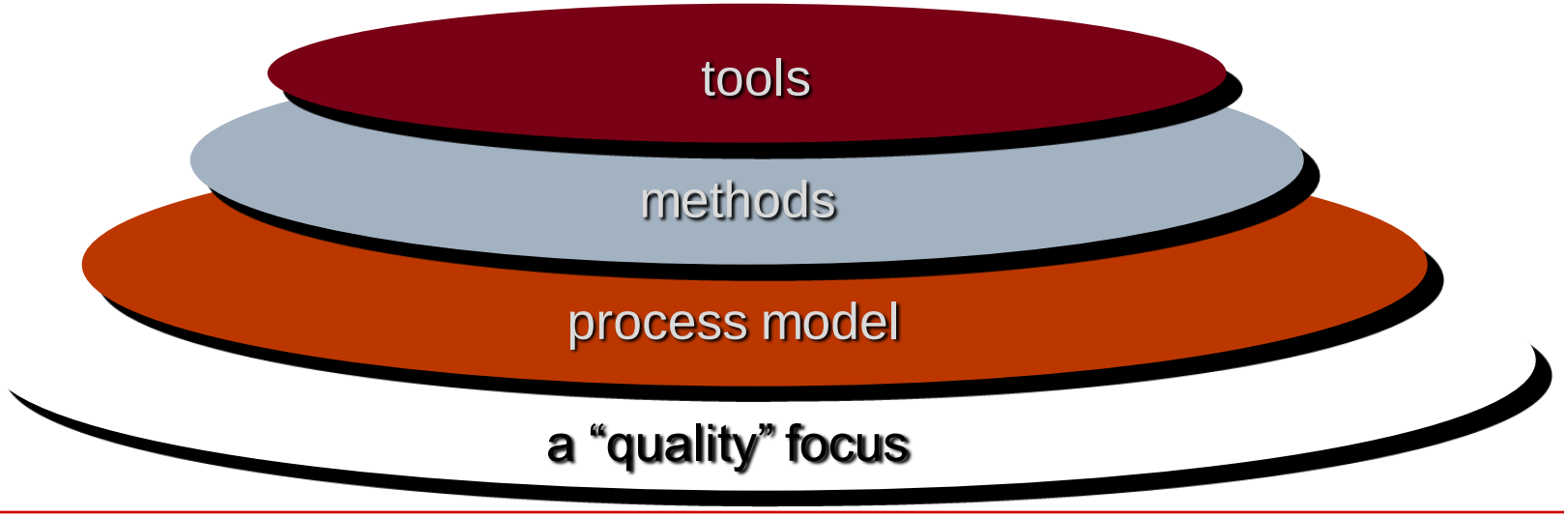
Software Crisis

- Application backlogs
- Maintenance phase takes 70% of programmer effort
  - Improvement takes 40% of maintenance phase
  - 45% to 65% from errors have been made before the programming phase
  - 50% of error removing cost come from errors made in the analysis

- **Solutions have to come in reasonable cost, time and quality**

Software Engineering

- **The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, in order to obtain economically software that is reliable and works efficiently on real machines**



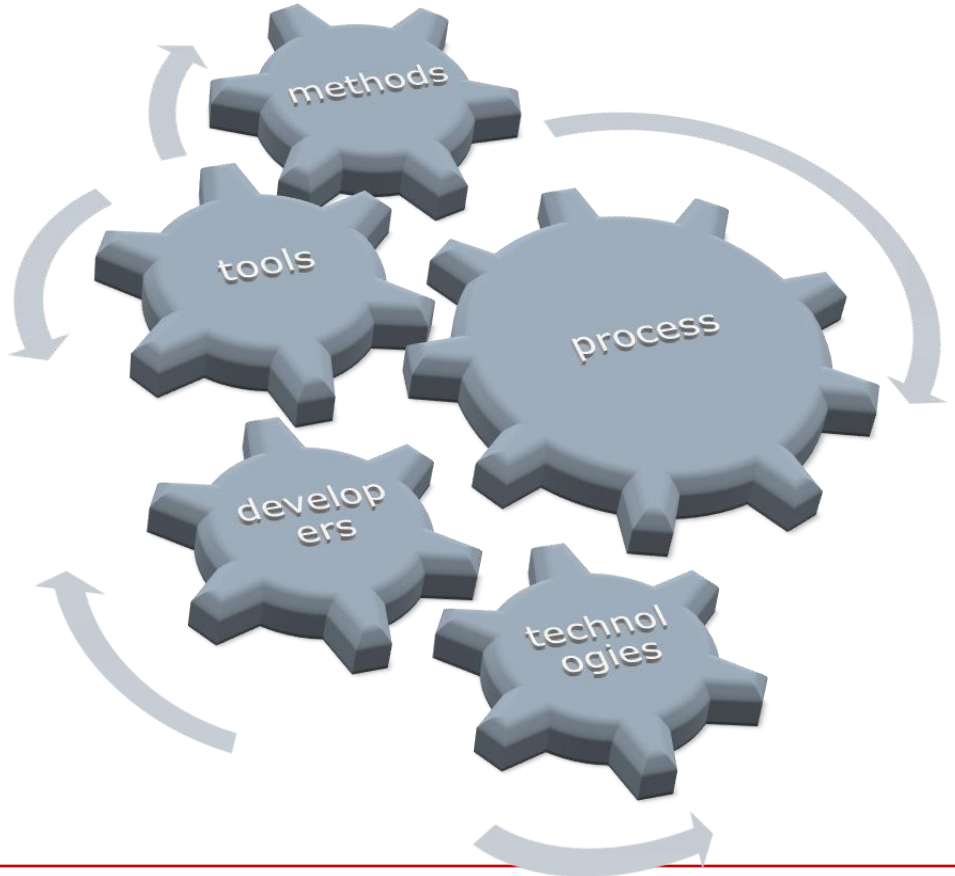
Goal of SoftWare Engineering

- Devise methodologies that are founded on the notion of evolution, that is, SW systems continually change, new SW systems are built from old ones, and all must interoperate and cooperate with each other
- Change (SW maintenance) drives SW evolution and occurs when:
  - Errors are corrected
  - SW adapted to new environments
  - Customer requests new features, and
  - Application reengineered to provide benefit in a modern context

SW Engineering Layers

- **SoftWare process** = basis for management control of SW projects and establishes the context in which technical methods are applied, work products are produced, milestones are established, quality is ensured, and change is properly managed
- **Methods** provide the technical "how to's" for building software. They encompass a broad array of tasks that include communication, requirements analysis, design modeling, construction, testing, and support
- **Tools** provide automated or semi-automated support for process and methods

Software Process



A Process Framework

- Establishes the foundation for a complete process by identifying a small number of framework activities that are applicable to all SW projects, regardless of their size or complexity

- Encompasses a set of umbrella activities that are applicable across the entire SW process

A Process Framework

- Process framework
  - Framework activities
    - work tasks
    - work products
    - milestones & deliverables
    - QA checkpoints
  - Umbrella Activities

Framework Activities

- Communication
- Planning
- Modeling
  - Analysis of requirements
  - Design
- Construction
  - Code generation
  - Testing
- Deployment

Umbrella Activities

- Software project management
- Formal technical reviews
- Software quality assurance
- Software configuration management
- Work product preparation and production
- Reusability management
- Measurement
- Risk management

Summary (1)

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

Summary (2)

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software must be reliable, secure, usable and maintainable.
- Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available.

What is a software process?

- **A set of activities whose goal is the development or evolution of software.**
- Generic activities in all software processes are:
  - **Specification** = what the system should do and its development constraints
  - **Development** = production of the software system
  - **Validation** = checking that the software is what the customer wants
  - **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.
- Generic process models
  - Waterfall;
  - Iterative development;
  - Component-based software engineering.
  - Using Formal Methods
  - ...

What are software engineering methods?

- Structured approaches to software development which include system models, notations, rules, design advice and process guidance.

- **Model descriptions** = Descriptions of graphical models which should be produced;
- **Rules** = Constraints applied to system models;
- **Recommendations** = Advice on good design practice;
- **Process guidance** = What activities to follow.

SDLC
(Systems Development Life Cycle)

SDLC (Systems Development Life Cycle)

- **Specification**
  - Identifying problems and objectives
  - Determining information requirements
- **Development**
  - Analyzing System needs
  - Designing recommended system
  - Implementation and documentation of software
- **Validation** = checking that the software is what the customer wants
- **Evolution** = Operating, evaluating & maintaining the System

Identifying problems and objectives

- **Goal** = Determining if the system users or managers need really a new information system
- **Main functions** =
  - Determining the problem
  - Identifyng the business objectives
  - Feasibility study = problem size, places of changes, tech solutions (components, …), cost-benefit analysis.

Determining information requirements

- **Goal** =understanding what information the users need to performe their job
- **Main functions** = knowing the details of current system = gathering next informations:
  - Involved people
  - Business activities
  - Environment
  - Timing
  - Current procedures
  - …

Analyzing System needs

- **Goal** = preparing system proposals + cost-benefit analysis + recommendations
- **Main functions** =
  - Input, output definition of the business's functions using
    - Data flow diagrams
    - Data dictionary
  - Decisions
    - Decision trees

Designing recommended system

- **Goal** = logical design, effective input, output.
- **Main functions** =
  - Data base design
  - User interfaces, screens, ..
  - Procedures using flowcharts, ...

Implementation and documentation of software

- **Goal** = program, documentation writing
- **Main functions** =
  - Programmers write or modify programs.
  - Analyst with users develop effective documentations :
    - Procedure manuals
    - Faq
    - Web sites
    - ……

Testing the system

- **Goal** = System correction, program review, new documentations, program updates
- **Main functions** = testing the system with sample data and current system data
  - Testing done by programmers
  - Testing done by programmers and system analysts
  - ..

Operating, evaluating & maintaining the System

- **Goal** = conversion from old to new system + discovery of problems or modification needs.
- **Main functions** =
  - Training users
  - Building data bases
  - Conversion data from old to new system
  - Installing equipment
  - Bringing new system into production

SDLC Summary

- We can begin a step of SDLC before finishing the previous step
  - Example = the design can be started before the analysis has finished, in condition that we have finished the analysis of most important functions.
- We can review a SDLC step if we find a need during the next step
- In any SDLC step except the maintenance step, It is possible to abort the project if the cost-benefit analysis was negative.

Problems encountered by the system Engineer

- No one correct solution for the system design.
- Choice between several available tools and approachs.
- He/she has to know the recent evolutions in component and software world and their tech details.
- New technologies or new updates can be come.
- He/she has to able to deal with people problems:
  - fear of new system / work loss
  - Learning new system / appearance as stupid
  - People groups
  - Department alliance
  - ……

Software Processes Models

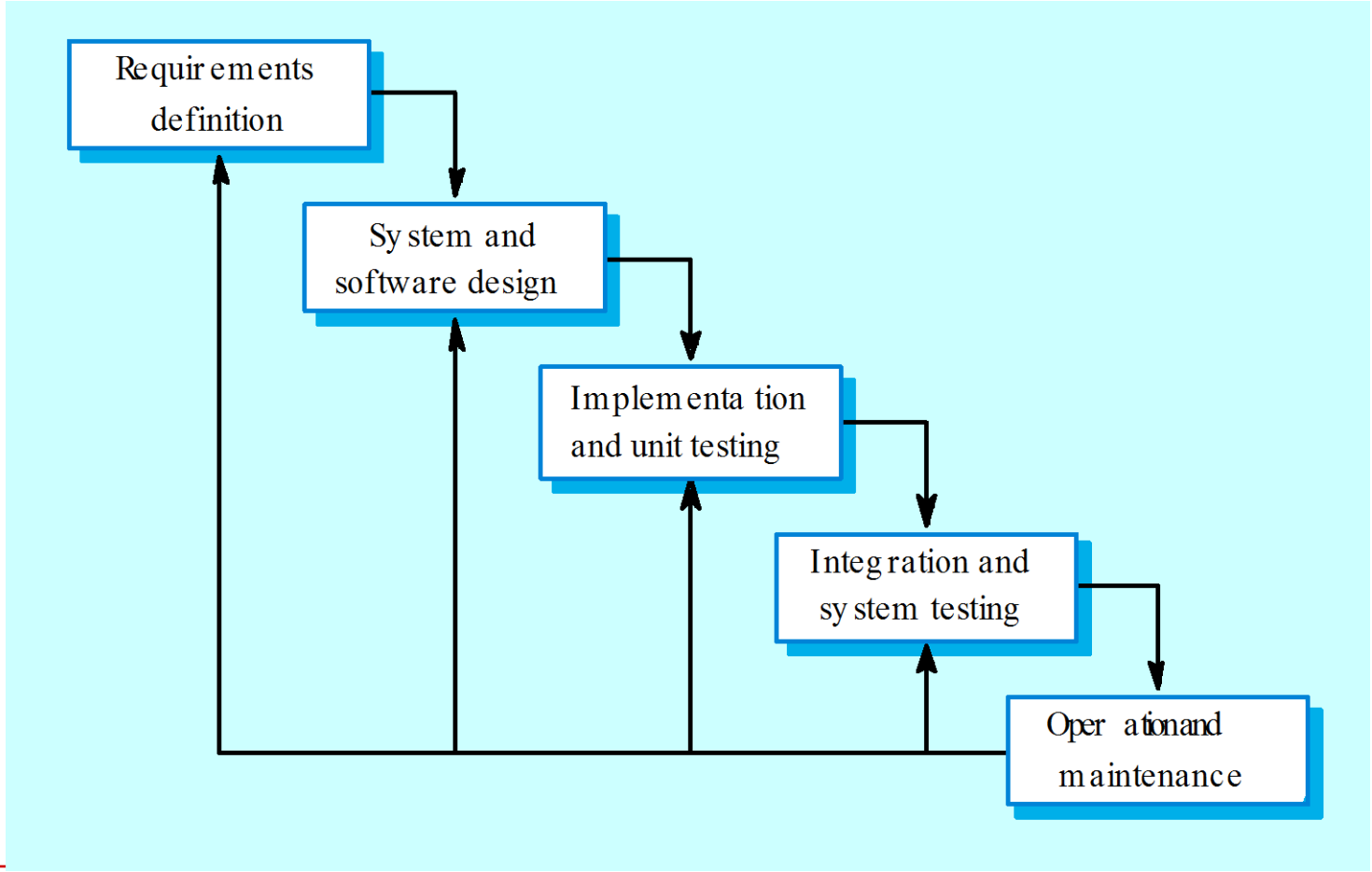
The software process

- A structured set of activities required to develop a software system
  - Specification;
  - Design;
  - Validation;
  - Evolution.
- 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

- 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.

Waterfall model



Waterfall model phases

- Requirements analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance

- The main drawback of the waterfall model = the difficulty of accommodating change after the process is underway. One phase has to be complete before moving onto the next phase.

Waterfall model problems

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
- Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
- Few business systems have stable requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.

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 and add new features as proposed by the customer.
- Throw-away prototyping
  - Objective is to understand the system requirements. Should start with poorly understood requirements to clarify what is really needed.

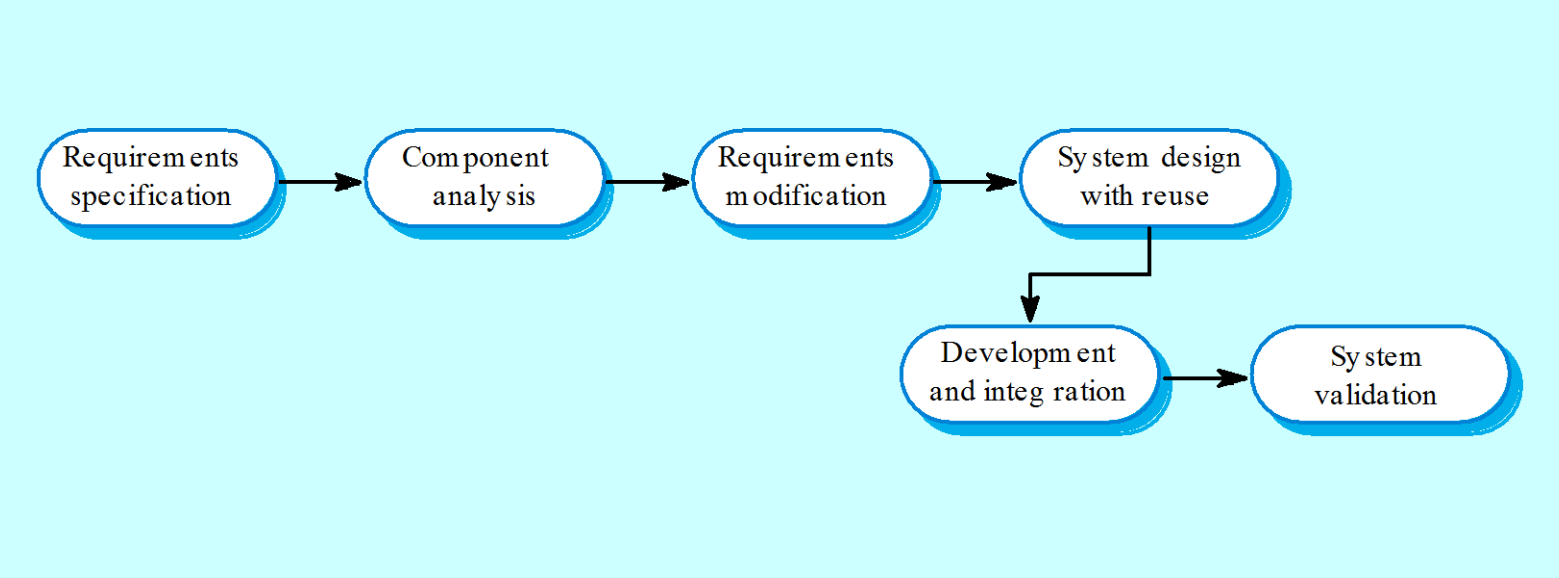
Evolutionary development

- Problems
  - Lack of process visibility;
  - Systems are often poorly structured;
  - Special skills (e.g. in languages for rapid prototyping) may be required.
- Applicability
  - For small or medium-size interactive systems;
  - For parts of large systems (e.g. the user interface);
  - For short-lifetime systems.

Component-based software engineering

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- Process stages
  - Component analysis;
  - Requirements modification;
  - System design with reuse;
  - Development and integration.
- This approach is becoming increasingly used as component standards have emerged.

Reuse-oriented development



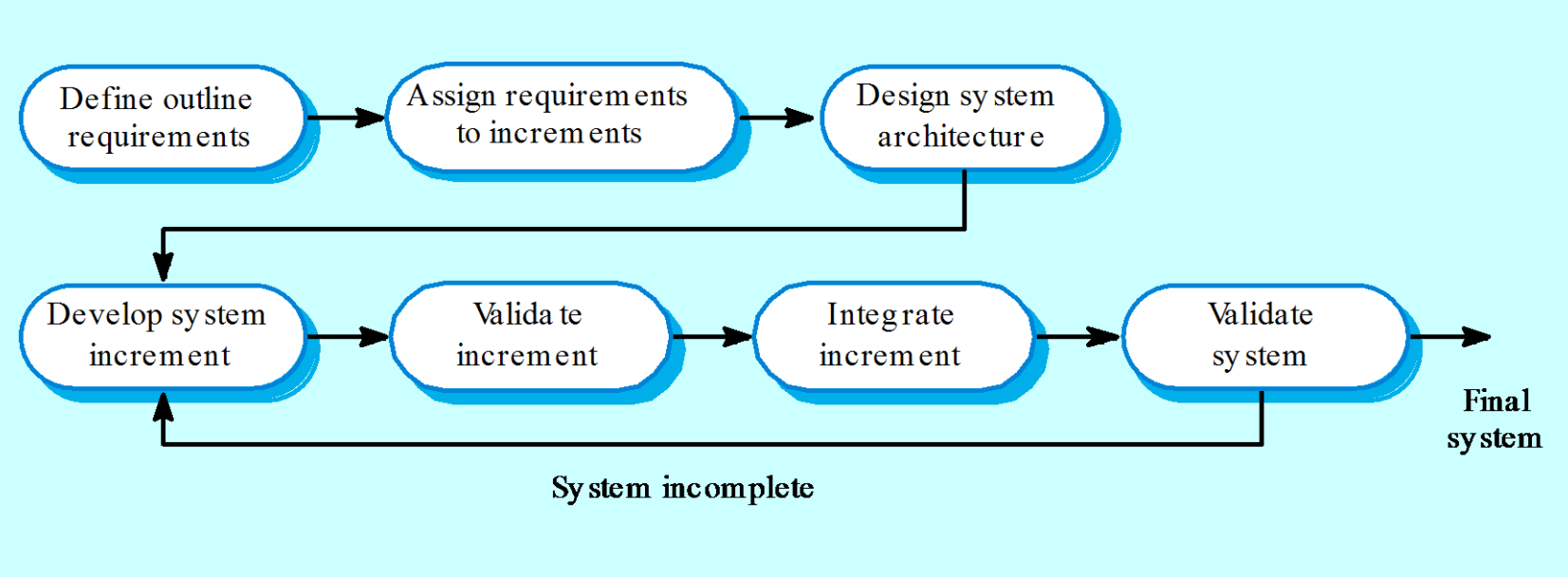
Process iteration

- System 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 models.
- Two (related) approaches
  - Incremental delivery;
  - Spiral development.

Incremental delivery

- 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.

Incremental development



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.