SOFTWARE PROCESS

SOFTWARE ENGINEERING 1

SOON PHEI TIN

OBJECTIVES

- understand the concepts of software processes and software process models;
- introduced to three generic software process models and when they might be used;
- know about the fundamental process activities of software requirements engineering, software development, testing, and evolution;

INTRODUCTION

- A software process is a set of related activities that leads to the production of a software product. These activities may involve the development
  - of software from scratch
  - by extending and modifying existing systems
  - by configuring and integrating off-the-shelf software or system components

INTRODUCTION

- There are many different software processes, but all must include four activities that are fundamental to software engineering:
  - Software specification
  - Software design and implementation
  - Software validation
  - Software evolution
- In practice, they are complex activities in themselves and include sub-activities
- There are also supporting process activities such as documentation and software configuration management

INTRODUCTION

- Software processes are complex and, there is no ideal process
- Most organizations have developed their own software development processes that take advantage of the capabilities of the people in an organization and the specific characteristics of the systems that are being developed
  - For a critical systems, a very structured development process is required.
  - For business systems, with rapidly changing requirements, a less formal, flexible process is likely to be more effective

INTRODUCTION

- software processes are categorized as either plan-driven or agile processes.
- Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.

SOFTWARE PROCESS MODELS

SOFTWARE ENGINEERING 1

SOON PHEI TIN

SOFTWARE PROCESS MODELS

- A software process model is a simplified representation of a software process.
- Discuss very general process models and present these from an architectural perspective, framework of the process rather than the details of specific activities.
- These generic models are not definitive descriptions of software processes, they are abstractions of the process that can be used to explain different approaches to software development
- You can think of them as process frameworks that may be extended and adapted to create more specific software engineering processes.

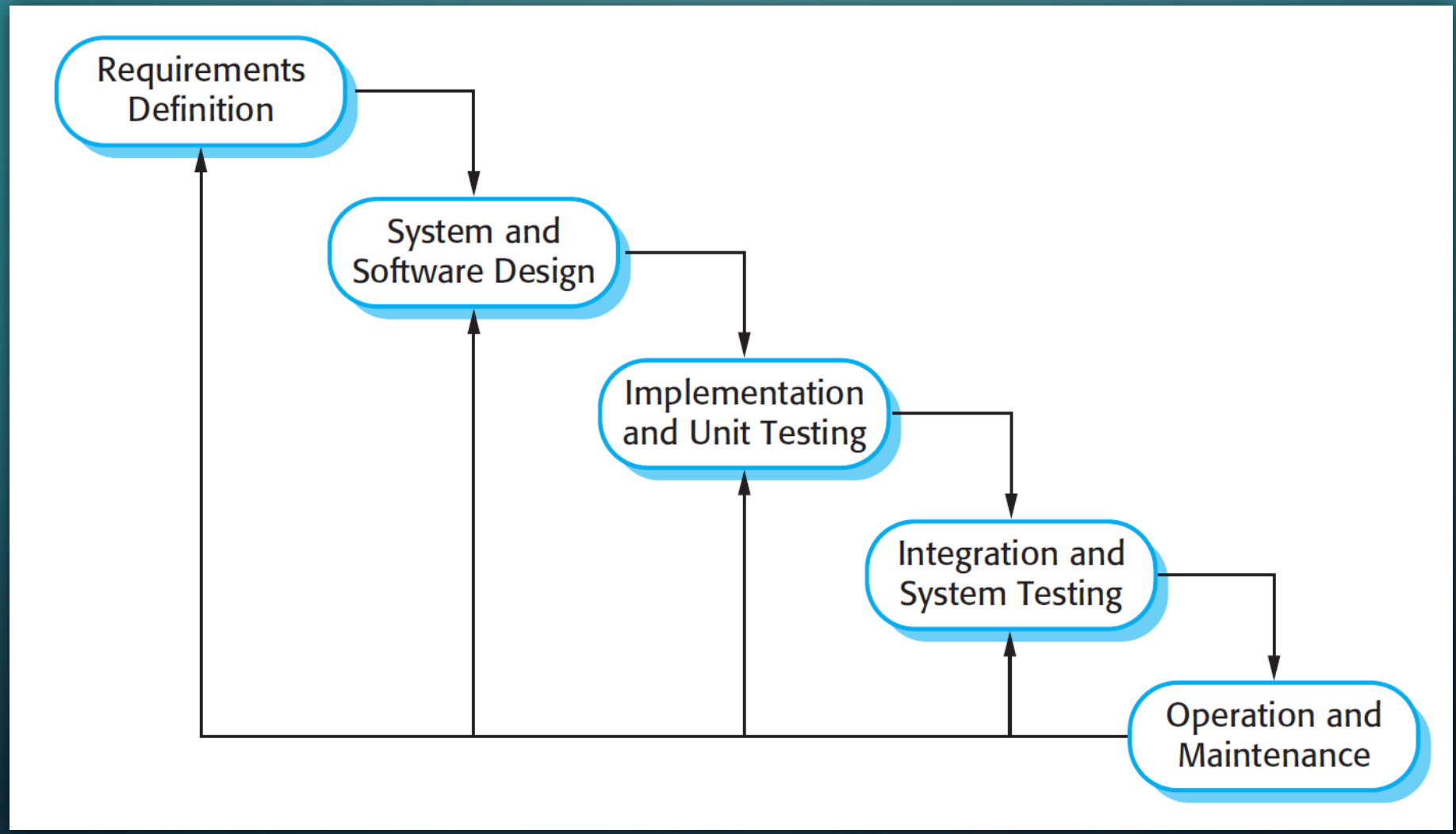
SOFTWARE PROCESS MODELS

- The models in our discussion: -
  - The waterfall model - This takes the fundamental process activities of specification, development, validation, and evolution and represents them as separate process phases such as requirements specification, software design, implementation, testing, and so on.
  - Incremental development - This approach interleaves the activities of specification, development, and validation. The system is developed as a series of versions (increments), with each version adding functionality to the previous version.
  - Reuse-oriented software engineering - This approach is based on the existence of a significant number of reusable components. The system development process focuses on integrating these components into a system rather than developing them from scratch.

SOFTWARE PROCESS MODELS

- These models are not mutually exclusive and are often used together, especially for large systems development.

THE WATERFALL MODEL



THE WATERFALL MODEL

- Because of the cascade from one phase to another, this model is known as the ‘waterfall model’ or software life cycle. The waterfall model is an example of a plan-driven process
- The principal stages of the waterfall model directly reflect the fundamental development activities:
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance

THE WATERFALL MODEL

- In principle, the result of each phase is one or more documents that are approved (‘signed off’). The following phase should not start until the previous phase has finished.
- In practice, these stages overlap and feed information to each other
- The software process is not a simple linear model but involves feedback from one phase to another.
- Documents produced in each phase may then have to be modified to reflect the changes made

THE WATERFALL MODEL

- Because of the costs of producing and approving documents, iterations can be costly and involve significant rework.
- Therefore, after a small number of iterations, it is normal to freeze parts of the development, such as the specification, and to continue with the later development stages.
- Problems are left for later resolution, ignored, or programmed around.
- This premature freezing of development tasks may mean that the system won’t deliver what the users are expecting.

THE WATERFALL MODEL

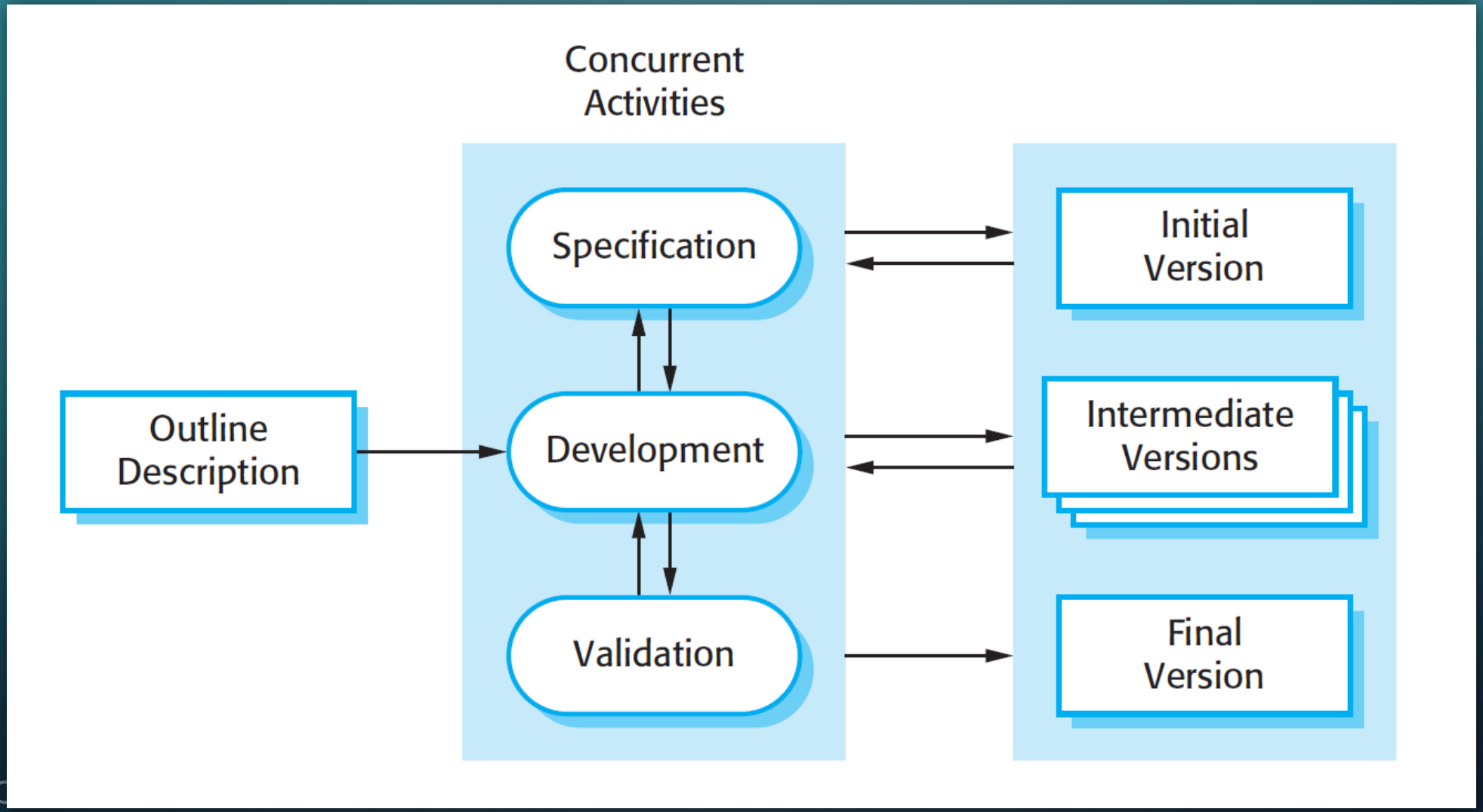
- During the final life cycle phase (operation and maintenance) the software is put into use.
- Errors and omissions in the original software requirements are discovered.
- Program and design errors emerge and the need for new functionality is identified.
- The system must therefore evolve to remain useful. Making these changes (software maintenance) may involve repeating previous process stages.

THE WATERFALL MODEL

- In the Waterfall Model, documentation is produced at each phase. This makes the process visible so managers can monitor progress against the development plan.
- Its major problem is the inflexible partitioning of the project into distinct stages.
- Commitments must be made at an early stage in the process, which makes it difficult to respond to changing customer requirements.
- In principle, the waterfall model should only be used when the requirements are well understood and unlikely to change radically during system development.

Is Waterfall Model a suitable model to develop a web-based e-commerce system? Why?

INCREMENTAL DEVELOPMENT



INCREMENTAL DEVELOPMENT

- Incremental development is based on the idea of developing an initial implementation, exposing this to user comment and evolving it through several versions until an adequate system has been developed
- Specification, development, and validation activities are interleaved rather than separate, with rapid feedback across activities.
- Incremental software development, which is a fundamental part of agile approaches, is better than a waterfall approach for most business, e-commerce, and personal systems

INCREMENTAL DEVELOPMENT

- By developing the software incrementally, it is cheaper and easier to make changes in the software as it is being developed.
- Each increment or version of the system incorporates some of the functionality that is needed by the customer.
- Generally, the early increments of the system include the most important or most urgently required functionality. This means that the customer can evaluate the system at a relatively early stage in the development to see if it delivers what is required.
- If not, then only the current increment has to be changed and, possibly, new functionality defined for later increments

INCREMENTAL DEVELOPMENT

- Incremental development has three important benefits, compared to the waterfall model:
  1. The cost of accommodating changing customer requirements is reduced.
  2. It is easier to get customer feedback on the development work that has been done.
  3. More rapid delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included.

INCREMENTAL DEVELOPMENT

- From a management perspective, the incremental approach has two problems:
  - The process is not visible.
  - System structure tends to degrade as new increments are added.

- Other problems with incremental development includes: -
  - large organizations have bureaucratic procedures that have evolved over time and there may be a mismatch between these procedures and a more informal iterative or agile process
  - Formal procedures are required by external regulations (e.g., accounting regulations)

INCREMENTAL DEVELOPMENT

- The problems of incremental development become particularly acute for large, complex, long-lifetime systems, where different teams develop different parts of the system.
- Large systems need a stable framework or architecture and the responsibilities of the different teams working on parts of the system need to be clearly defined with respect to that architecture.
- This has to be planned in advance rather than developed incrementally.

REUSE-ORIENTED SOFTWARE ENGINEERING

- In the majority of software projects, there are some forms of informal software reuse.
- This informal reuse takes place irrespective of the development process that is used.
- In the 21st century, software development processes that focus on the reuse of existing software have become widely used.
- Reuse-oriented approaches rely on a large base of reusable software components and an integrating framework for the composition of these components.

REUSE-ORIENTED SOFTWARE ENGINEERING

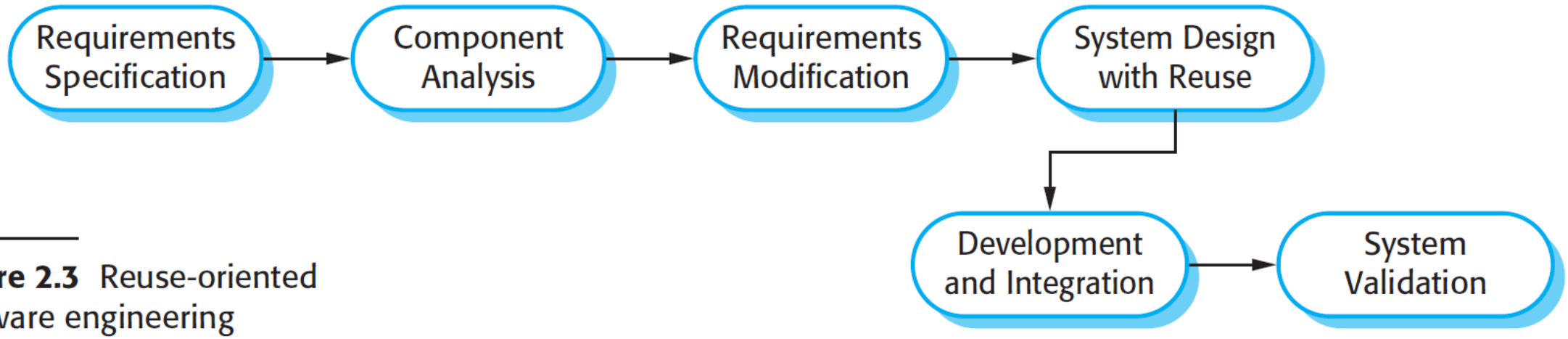


Figure 2.3 Reuse-oriented software engineering

REUSE-ORIENTED SOFTWARE ENGINEERING

- There are three types of software component that may be used in a reuse-oriented process:
  - Web services that are developed according to service standards and which are available for remote invocation.
  - Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE.
  - Stand-alone software systems that are configured for use in a particular environment.

REUSE-ORIENTED SOFTWARE ENGINEERING

- Advantages
  - reducing the amount of software to be developed and so reducing cost and risks.
  - usually also leads to faster delivery of the software.
- However, requirements compromises are inevitable, and this may lead to a system that does not meet the real needs of users.
- Furthermore, some control over the system evolution is lost as new versions of the reusable components are not under the control of the organization using them.

SOFTWARE PROCESS ACTIVITIES

SOFTWARE ENGINEERING 1

SOON PHEI TIN

SOFTWARE PROCESS ACTIVITIES

- Real software processes are interleaved sequences of technical, collaborative, and managerial activities with the overall goal of specifying, designing, implementing, and testing a software system.
- The four basic process activities of specification, development, validation, and evolution are organized differently in different development processes.
- In the waterfall model, they are organized in sequence
- In incremental development they are interleaved
- How these activities are carried out depends on the type of software, people, and organizational structures involved

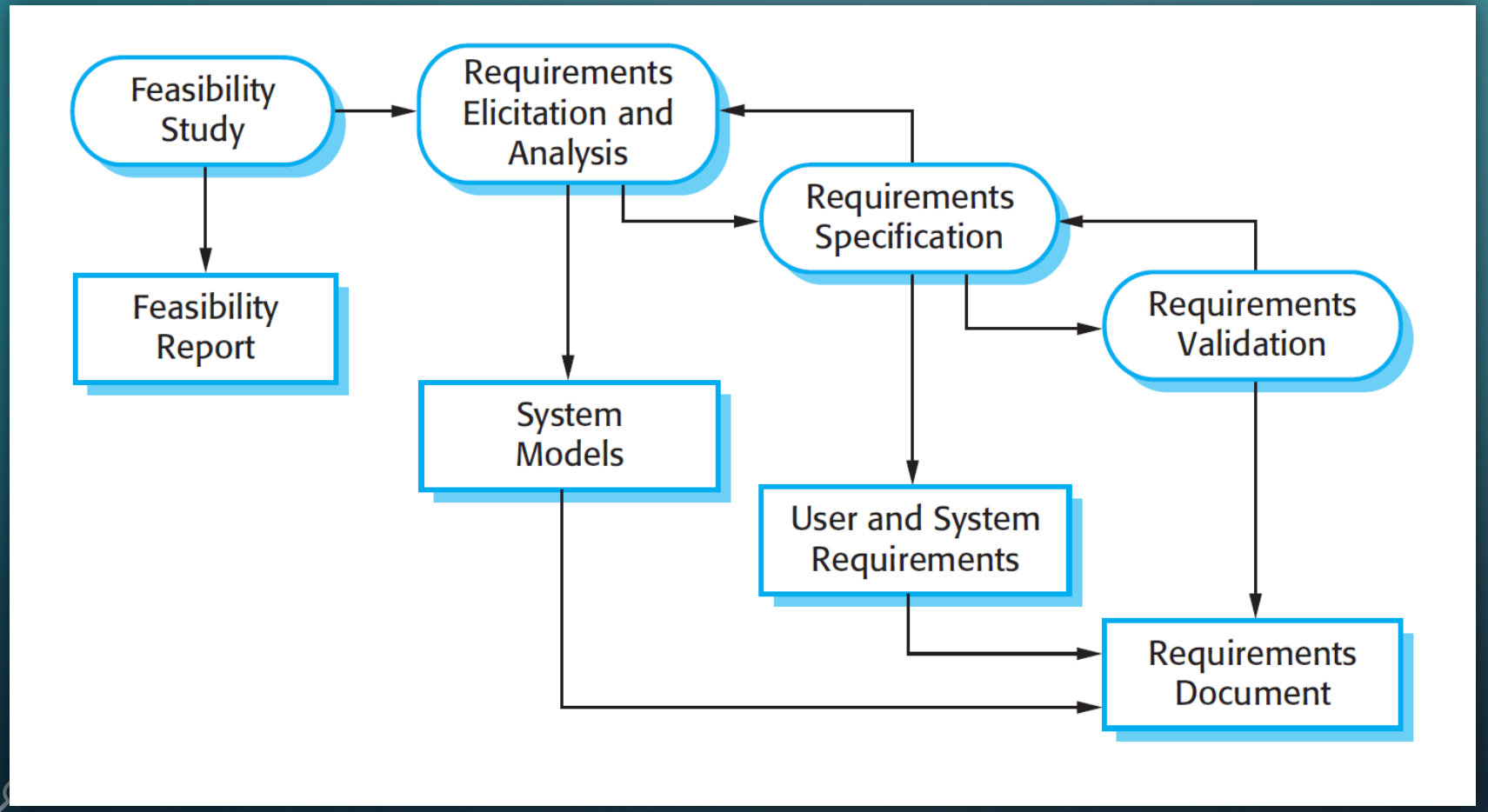
SOFTWARE PROCESS ACTIVITIES

- Software Specification
- Software Design and Implementation
- Software Validation
- Software Evolution

SOFTWARE SPECIFICATION

- Software specification or requirements engineering is the process of understanding and defining
  - what services are required from the system
  - identifying the constraints on the system's operation and development.
- Requirements engineering is a particularly critical stage of the software process as errors at this stage inevitably lead to later problems in the system design and implementation.

SOFTWARE SPECIFICATION



The activities of analysis, definition, and specification are interleaved

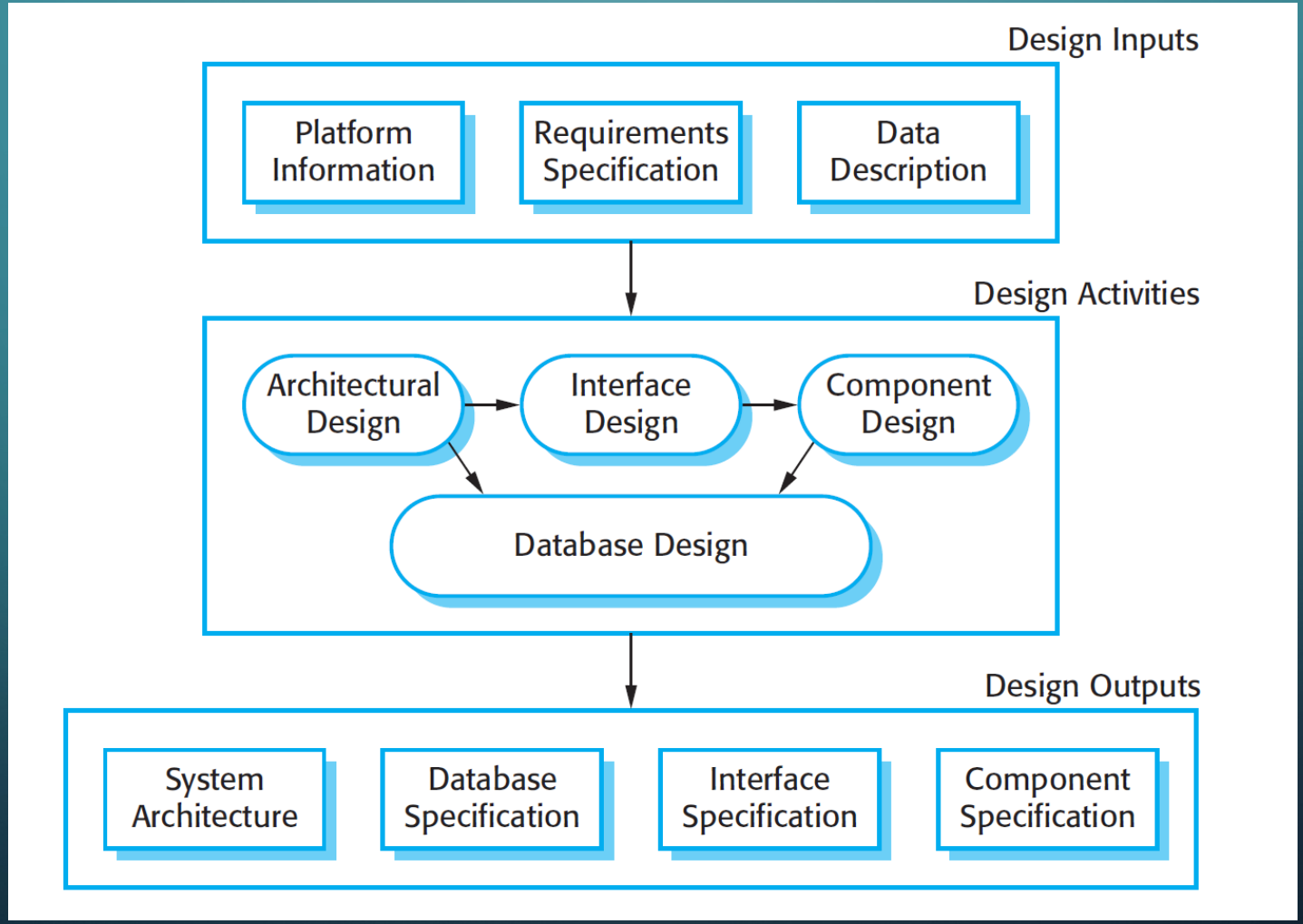
SOFTWARE SPECIFICATION

- The requirements engineering process aims to produce an agreed requirements document that specifies a system satisfying stakeholder requirements.
- Requirements are usually presented at two levels of detail.
  - End-users and customers need a high-level statement of the requirements;
  - system developers need a more detailed system specification.
- There are four main activities in the requirements engineering process:
  - Feasibility study
  - Requirements elicitation and analysis
  - Requirements specification
  - Requirements validation

SOFTWARE DESIGN AND IMPLEMENTATION

- The implementation stage of software development is the process of converting a system specification into an executable system.
- It always involves processes of software design and programming but, if an incremental approach to development is used, may also involve refinement of the software specification.
- A software design is a description of the structure of the software to be implemented, the data models and structures used by the system, the interfaces between system components and, sometimes, the algorithms used.

SOFTWARE DESIGN AND IMPLEMENTATION



SOFTWARE DESIGN AND IMPLEMENTATION

- Designers do not arrive at a finished design immediately but develop the design iteratively. They add formality and detail as they develop their design with constant backtracking to correct earlier designs.

SOFTWARE DESIGN AND IMPLEMENTATION

- Design activities
  - Architectural design, where you identify the overall structure of the system, the principal components (sometimes called sub-systems or modules), their relationships, and how they are distributed.
  - Interface design, where you define the interfaces between system components. This interface specification must be unambiguous.

SOFTWARE DESIGN AND IMPLEMENTATION

- Design activities
  - Component design, where you take each system component and design how it will operate.
    - a simple statement of the expected functionalities
    - a list of changes to be made to a reusable component
    - a detailed design model (model-driven approach).
  - Database design, where you design the system data structures and how these are to be represented in a database.

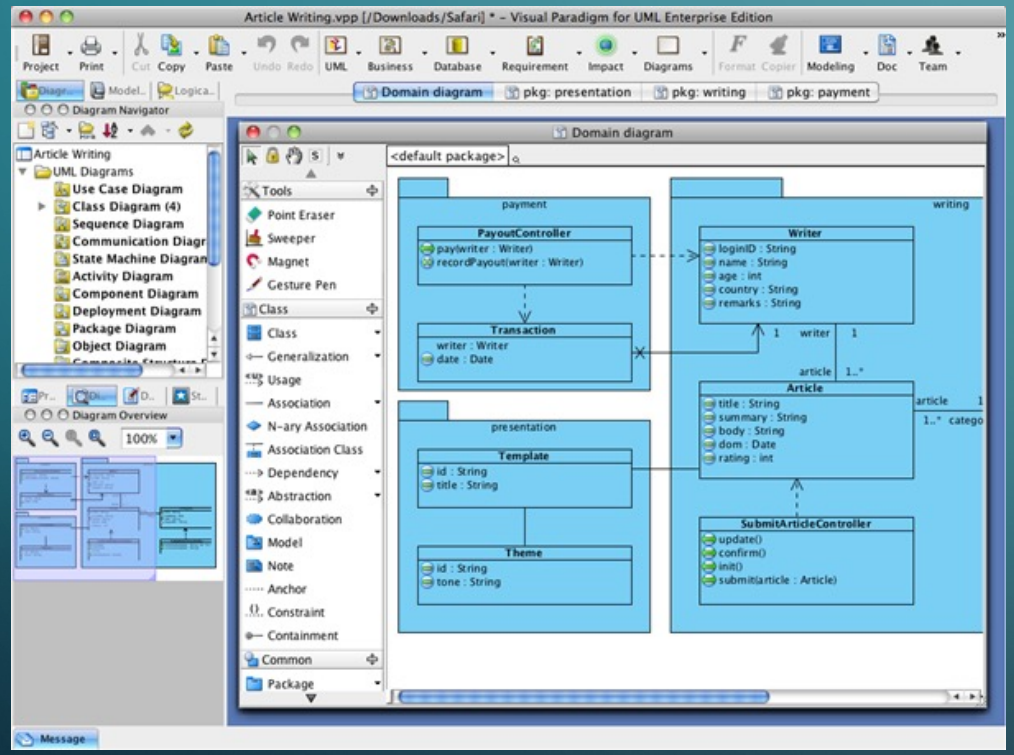
SOFTWARE DESIGN AND IMPLEMENTATION

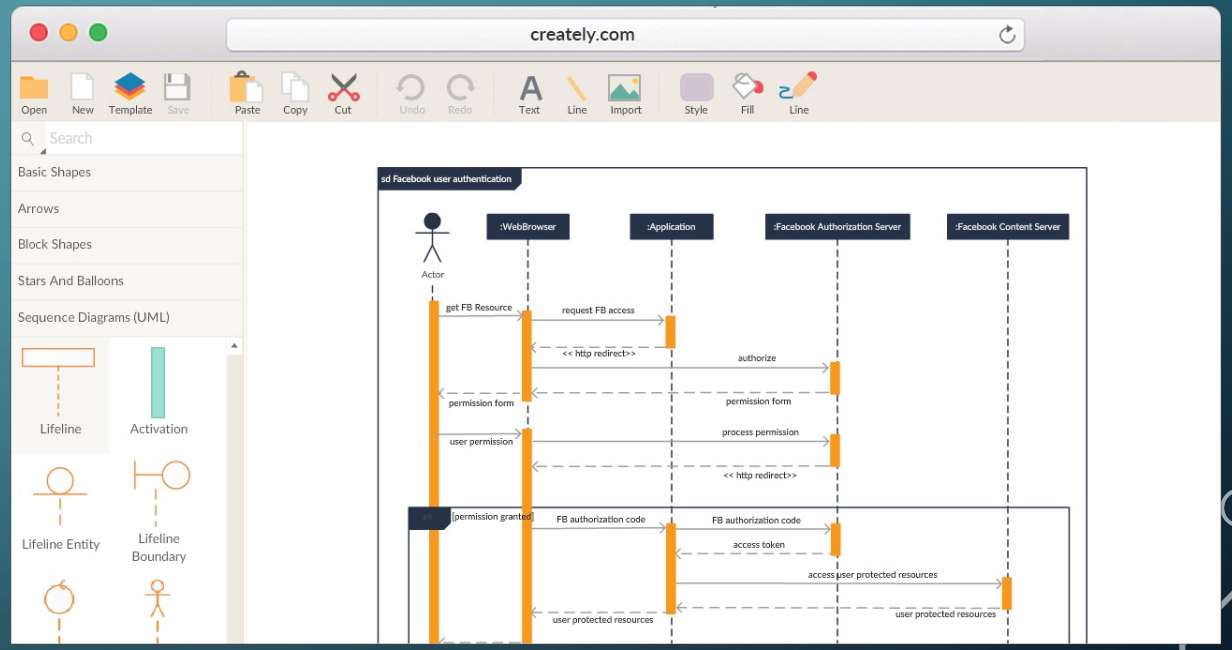
- The detail and representation of the design output vary considerably.
  - For critical systems, detailed design documents setting out precise and accurate descriptions of the system must be produced.
  - If a model-driven approach is used, these outputs may mostly be diagrams.
  - Where agile methods of development are used, the outputs of the design process may not be separate specification documents but may be represented in the code of the program.

SOFTWARE DESIGN AND IMPLEMENTATION

- Model-driven development (MDD) or model-driven engineering (Schmidt, 2006), where models of the software are created at different levels of abstraction
- The models are developed in sufficient detail so that the executable system can be generated from them
- Software development tools may be used to generate a skeleton program from MDD. This includes code to define and implement interfaces, and, in many cases, the developer need only add details of the operation of each program component

SOFTWARE DESIGN AND IMPLEMENTATION

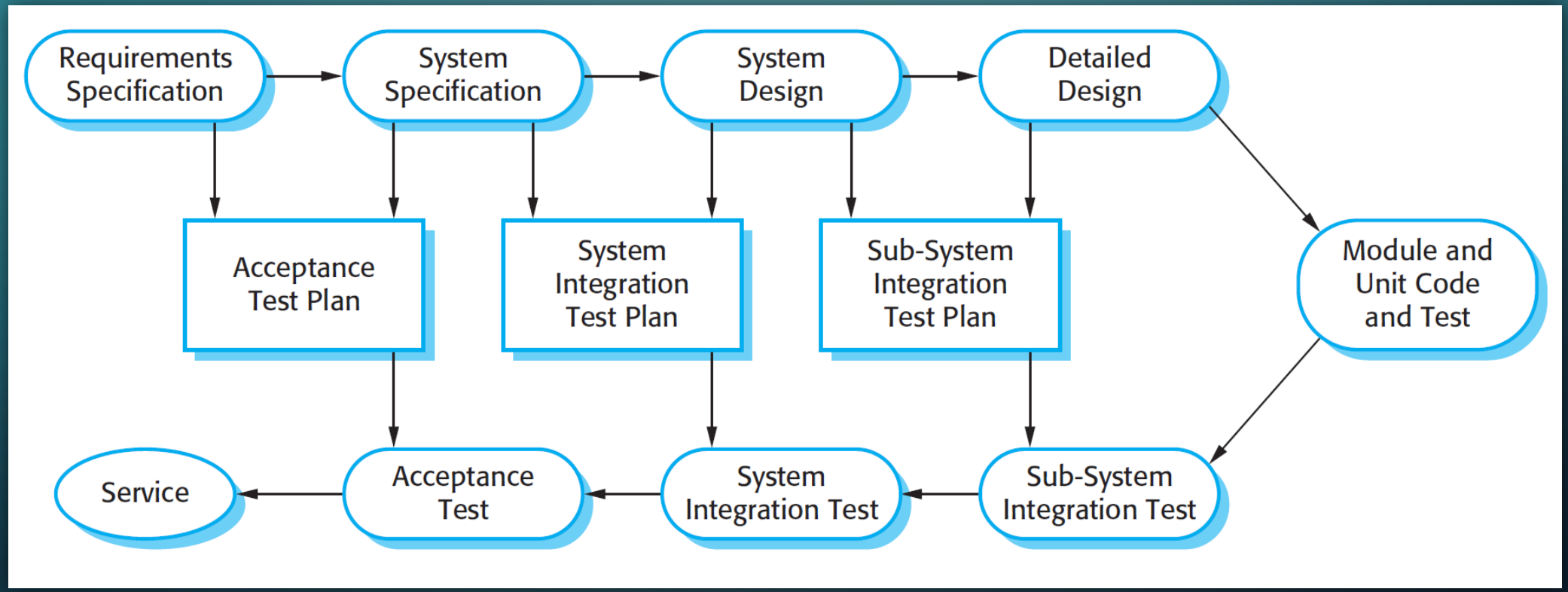




SOFTWARE VALIDATION

- Software validation or, more generally, verification and validation (V&V) is intended to show that
  - a system both conforms to its specification
  - it meets the expectations of the system customer
- Program testing, where the system is executed using simulated test data, is the principal validation technique
- Validation may also involve checking processes, such as inspections and reviews, at each stage of the software process from user requirements definition to program development

SOFTWARE VALIDATION



SOFTWARE VALIDATION

- A three-stage testing process involved testing for system components then the testing for an integrated system and, finally, the testing of the system with the customer’s data

SOFTWARE VALIDATION

- The stages in the testing process are:
  - Development testing - The components making up the system are tested by the people developing the system. Each component is tested independently, without other system components.
  - System testing - System components are integrated to create a complete system. This process is concerned with finding errors that result from unanticipated interactions between components and component interface problems.
  - Acceptance testing - This is the final stage in the testing process before the system is accepted for operational use. The system is tested with data supplied by the system customer rather than with simulated test data.

SOFTWARE VALIDATION

- Normally, component development and testing processes are interleaved. Programmers make up their own test data and incrementally test the code as it is developed.
- If an incremental approach to development is used, each increment should be tested as it is developed, with these tests based on the requirements for that increment.
- When a plan-driven software process is used (e.g., for critical systems development), testing is driven by a set of test plans. An independent team of testers works from these pre-formulated test plans

SOFTWARE EVOLUTION

- Once a decision has been made to manufacture hardware, it is very expensive to make changes to the hardware design. However, changes can be made to software at any time during or after the system development. Even extensive changes are still much cheaper than corresponding changes to system hardware
- Historically, there has always been a split between the process of software development and the process of software evolution (software maintenance).
- People think of software development as a creative activity in which a software system is developed from an initial concept through to a working system. However, they sometimes think of software maintenance as dull and uninteresting
- Rather than two separate processes, it is more realistic to think of software engineering as an evolutionary process