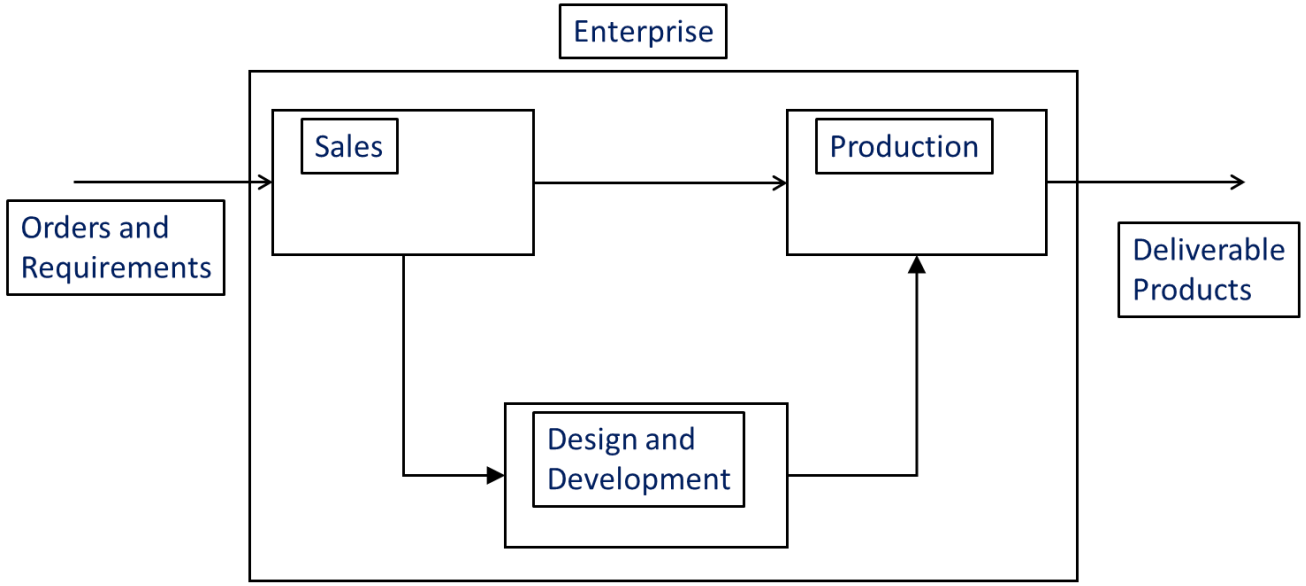
CSCI 4050/6050
Software Engineering

System Development

Dr. Eman Saleh eman.saleh@uga.edu

Part of a larger activity

- Software systems are usually developed to support a part (or parts) of the overall mission of the enterprise.



- Many different types of software system exist and there is no universal set of software techniques that is applicable to all situations.
- The software engineering methods and tools used depend on the type of application being developed, the requirements of the customer and the background of the development team

Types of Systems

- Stand-alone systems
  - run on a local computer, such as a PC; include all necessary functionality; do not need to be connected to a network
- Interactive transaction-based systems
  - run on a remote computer and are accessed by users from their own local systems; these include Web applications, e.g. E-Commerce applications
- Embedded control systems
  - software systems that control and manage hardware devices; likely, there are more embedded systems than any other type of system

Types of Systems

- Batch processing systems
  - business systems that are designed to process data in large quantities (batches); process large numbers of individual inputs to create corresponding outputs
- Entertainment systems
  - systems that are primarily for personal use, which are intended to entertain the user.
- Systems for modelling and simulation
  - systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects

Types of Systems

- Data collection systems
  - systems that collect data from their environment using a set of sensors and send that data to other systems for processing
- Systems of systems
  - systems that are composed of a number of other software systems; integration of systems

Web-Based Systems

- The Web has become a platform for running applications and organizations are increasingly developing Web-based systems rather than local, stand-alone systems
- Web Services (WS) allow application functionality to be accessed over the Web
- Cloud computing is an approach to the provision of computer services where applications run remotely on the "cloud"
  - Users do not buy software but pay according to use

Major effects on SE for Web-Based Systems

- Software reuse is the dominant approach for constructing Web-based systems
  - we think about how you can assemble them from pre-existing software components and systems
- Web-based systems should be developed and delivered incrementally
  - it is impractical to specify all the requirements for such systems in advance
- User interfaces are constrained by the capabilities of Web browsers
  - technologies such as AJAX, jQuery, ExtJS, allow rich interfaces to be created within a Web browser

Software Engineering Fundamentals

- Some fundamental principles apply to all types of software systems, irrespective of the development techniques used:
  - systems should be developed using a managed and understood development process; different processes are used for different types of software
  - dependability and performance are important for all types of system
  - understanding and managing the software specification and requirements (what the software should do) are important
  - if appropriate, we should reuse software that has already been developed rather than write new one

Parties in System Development

Parties involved in system development:

- Users (direct/indirect)
- Customer/procurer
- Producer/developer

All three may, and usually are, different

What do we mean stakeholders?

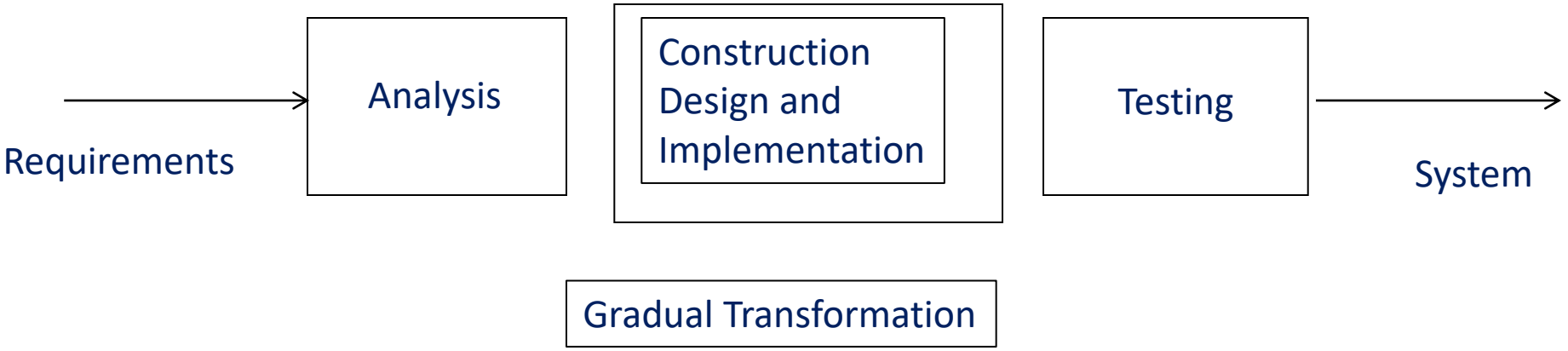
Parties in System Development

Users play a central role

Systems should be:

- specified based on user needs
- validated whether it really functions according to user needs
- documented by describing the system from the user's perspective

System Development



Notation, Method, Methodology

Notation

- representation of a model
- may be graphical (e.g. UML) or textual

Method

- Repeatable technique for solving a problem (e.g., a sorting algorithm)

Methodology

- Collection of methods for solving a class of problems, for example:
  - Booch Methodology
  - Object Modeling Technique (OMT)
  - Unified Software Development Process

Software Development Process

Methodology of developing software systems is frequently referred to as the Software Development Process, or simply the Software Process

It is a structured set of activities required to develop a software system

Software Development Process

Many different software processes exist, but all involve:

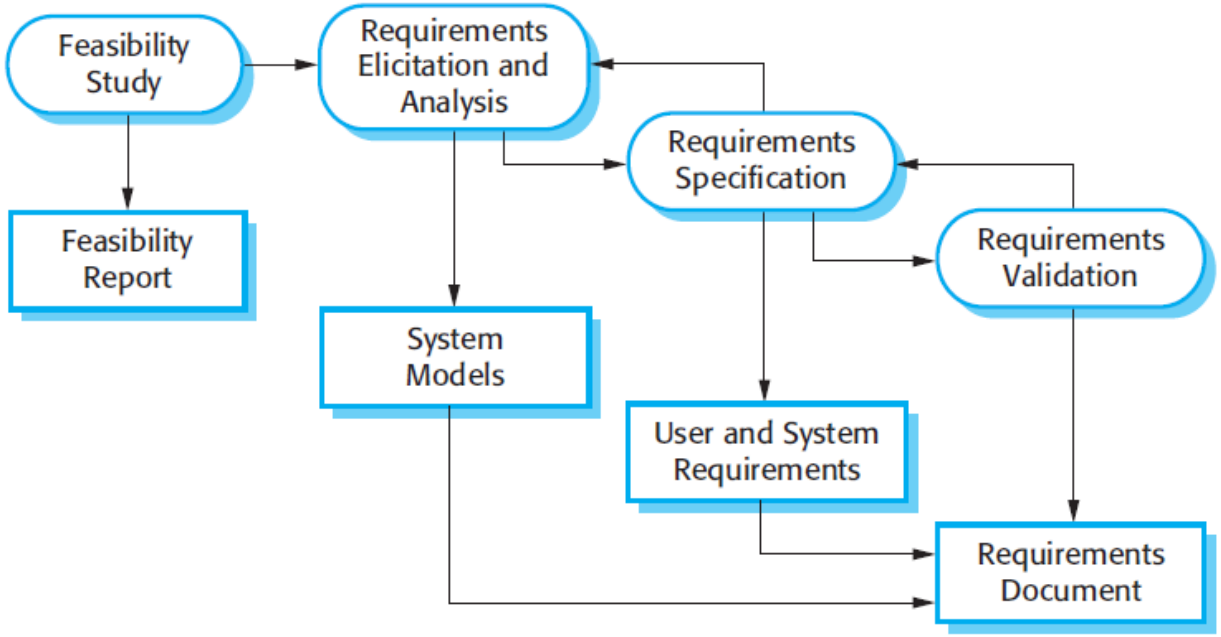
- Specification (Requirements Engineering) – defining what the system should do
- Design and implementation – defining the organization of the system and implementing the system
- Validation – checking that it does what the customer wants
- Evolution – changing the system in response to changing customer needs

Software Engineering Activities

1- Requirements Engineering

Software specification or requirements engineering is the process of understanding and defining what services are required and identifying the constraints on these services.
Requirements engineering processes ensures your software will meet the user expectations and ending up with a high-quality software.

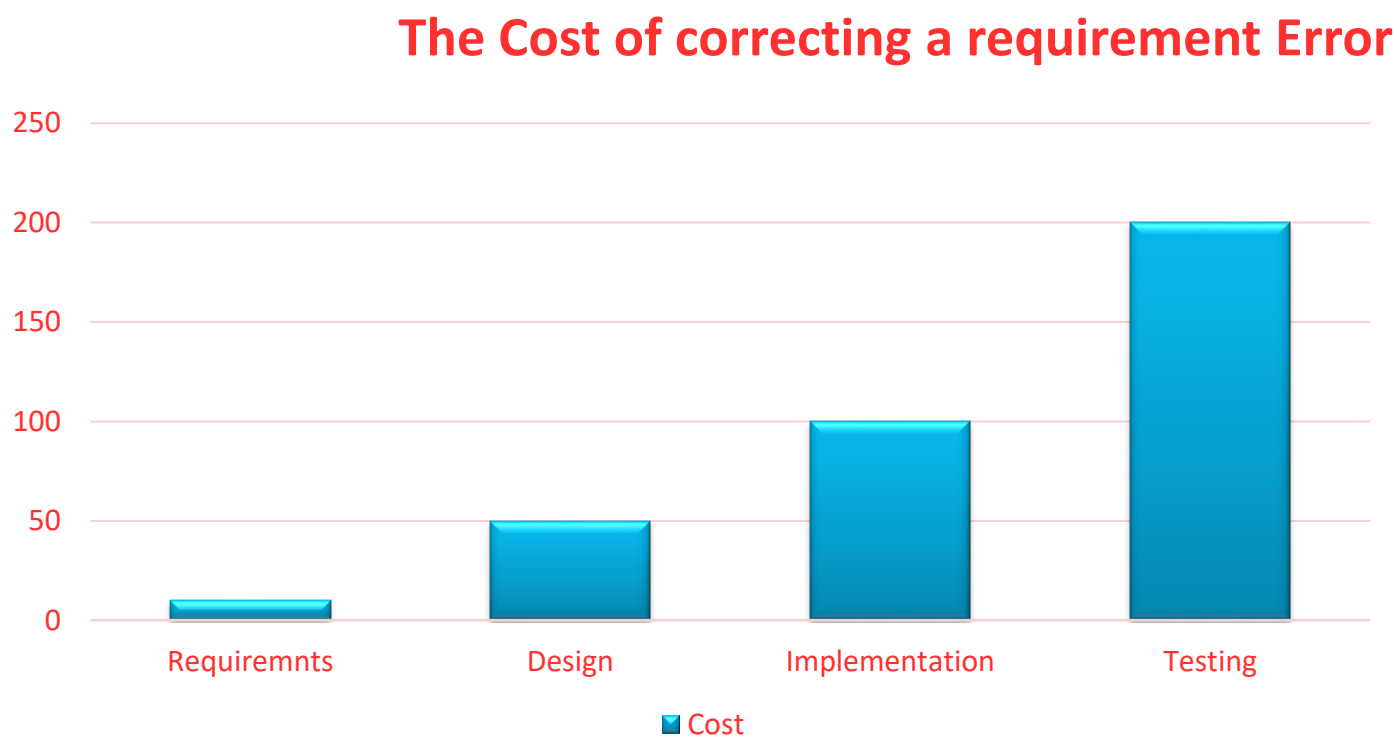
It's a critical stage of the software process as errors at this stage will reflect later on the next stages, which definitely will cause you a higher costs.
At the end of this stage, a requirements document that specifies the requirements will be produced and validated with the stockholders.
There are four main activities (or sub-activities) of requirements engineering:



The requirements engineering process.

- Of course, the activities in the requirements process are not simply executed in a strict sequence, but they are interleaved. For example, analysis activity continues during the specification as new requirements come to light.

- *In agile methods, requirements are developed incrementally according to user priorities and the elicitation of requirements comes from users who are part of the development team.*


The Cost of correcting a requirement Error
250
200
150
100
50
0
Requiremnts
Design
Implementation
Testing
Cost

Software Engineering Activities

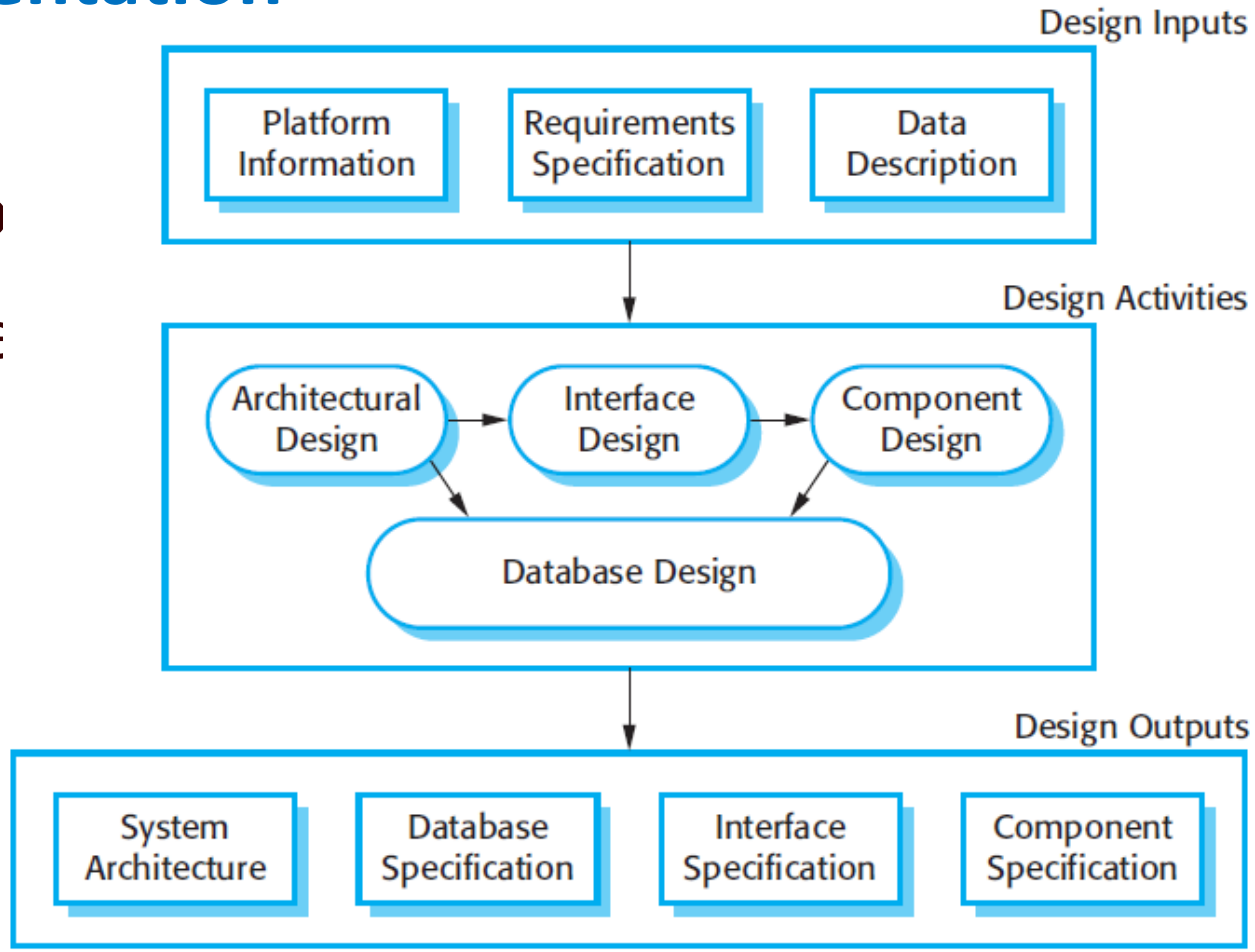
2- Software Design and Implementation

Software Design

A software design is a description o the structure of the software to be implemented, data models, the Use interface design, subsystem interfa design (i.e. interfaces between system components), and possibly the algorithms used.

Software Implementation

is the process of converting a system specification and design into an executable system. (coding)



The Software Design activities.

Software Engineering Activities

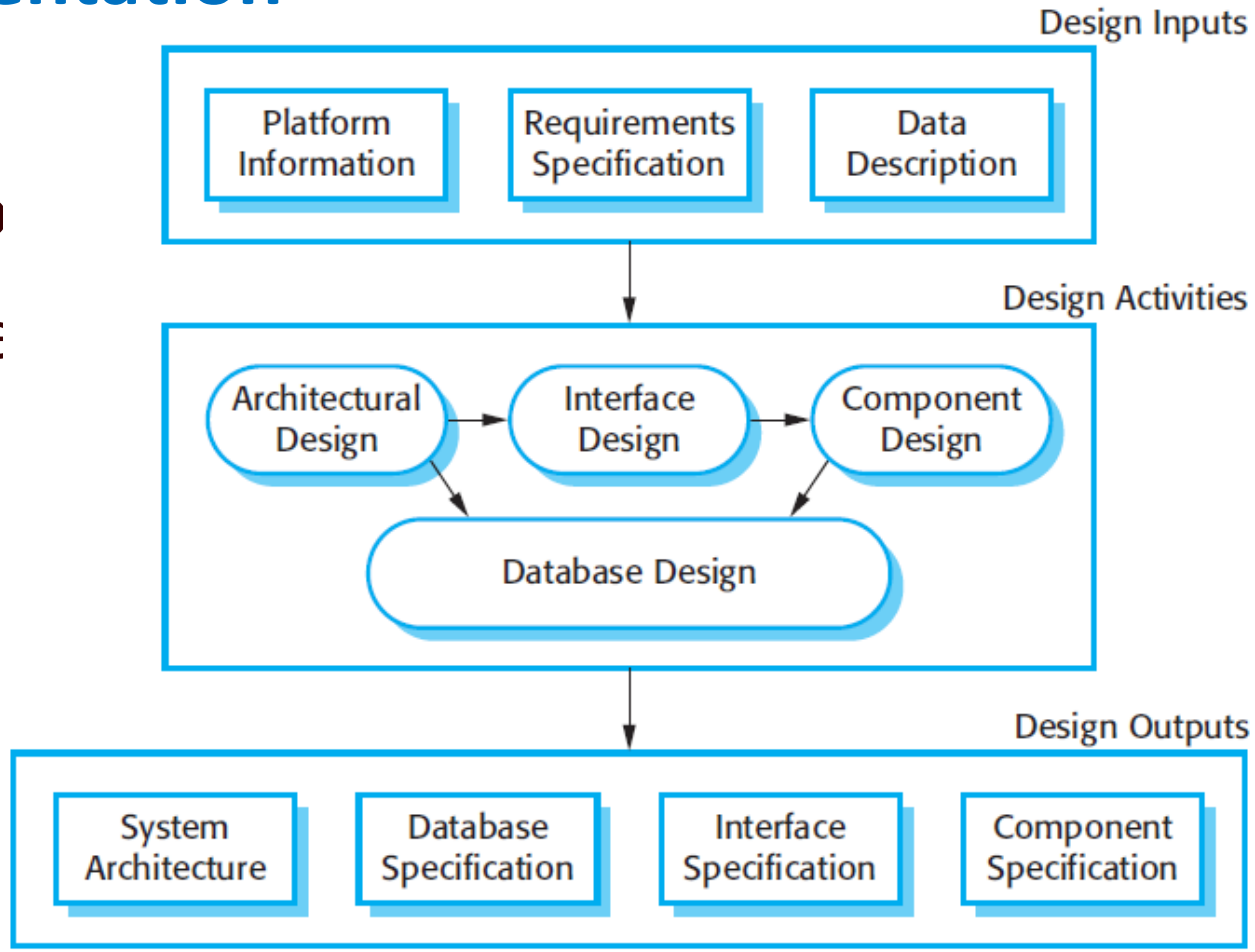
2- Software Design and Implementation

Software Design

A software design is a description o the structure of the software to be implemented, data models, the Use interface design, subsystem interfa design (i.e. interfaces between system components), and possibly the algorithms used.

Software Implementation

is the process of converting a system specification and design into an executable system. (coding)



The Software Design activities.

3- Software Verification and Validation (V&V)

Verification and Validation is a more general term compared to testing; V&V are independent procedures that are used together for checking that the software system meets its requirements and specifications and that it fulfills its intended purpose.

Testing: The process of executing the system using a set of test cases with the intent of finding and removing bugs. It is an important validation technique.

The main phases of software testing

4- Software Evolution

The process of modifying a software product after delivery.

There are four types of maintenance: -

1- Corrective maintenance is concerned with fixing errors that are observed when the software is in use.

2- Adaptive maintenance is concerned with the change in the software to make it adaptable to new environment such as to run the software on a new operating system.

3-Perfective maintenance is concerned with the change in the software that occurs while adding new functionalities in the software.

4- Preventive maintenance involves implementing changes to prevent the occurrence of errors. The distribution of types of maintenance by type and by percentage of time consumed.

Types of Software Processes

Plan-driven processes

all of the process activities are planned in advance and progress is measured against this plan

Agile processes

planning is incremental and iterative, and it is easier to change the process to reflect changing customer requirements

Types of Software Processes

- Most practical processes include elements of both plan-driven and agile approaches
- There are no right or wrong software processes

Software Process Models

- Waterfall model
- Incremental development
- Iterative development
- Agile development
- Reuse-oriented development
- Prototyping
- Spiral model
- Formal transformation

Waterfall Model

Also referred as the software lifecycle

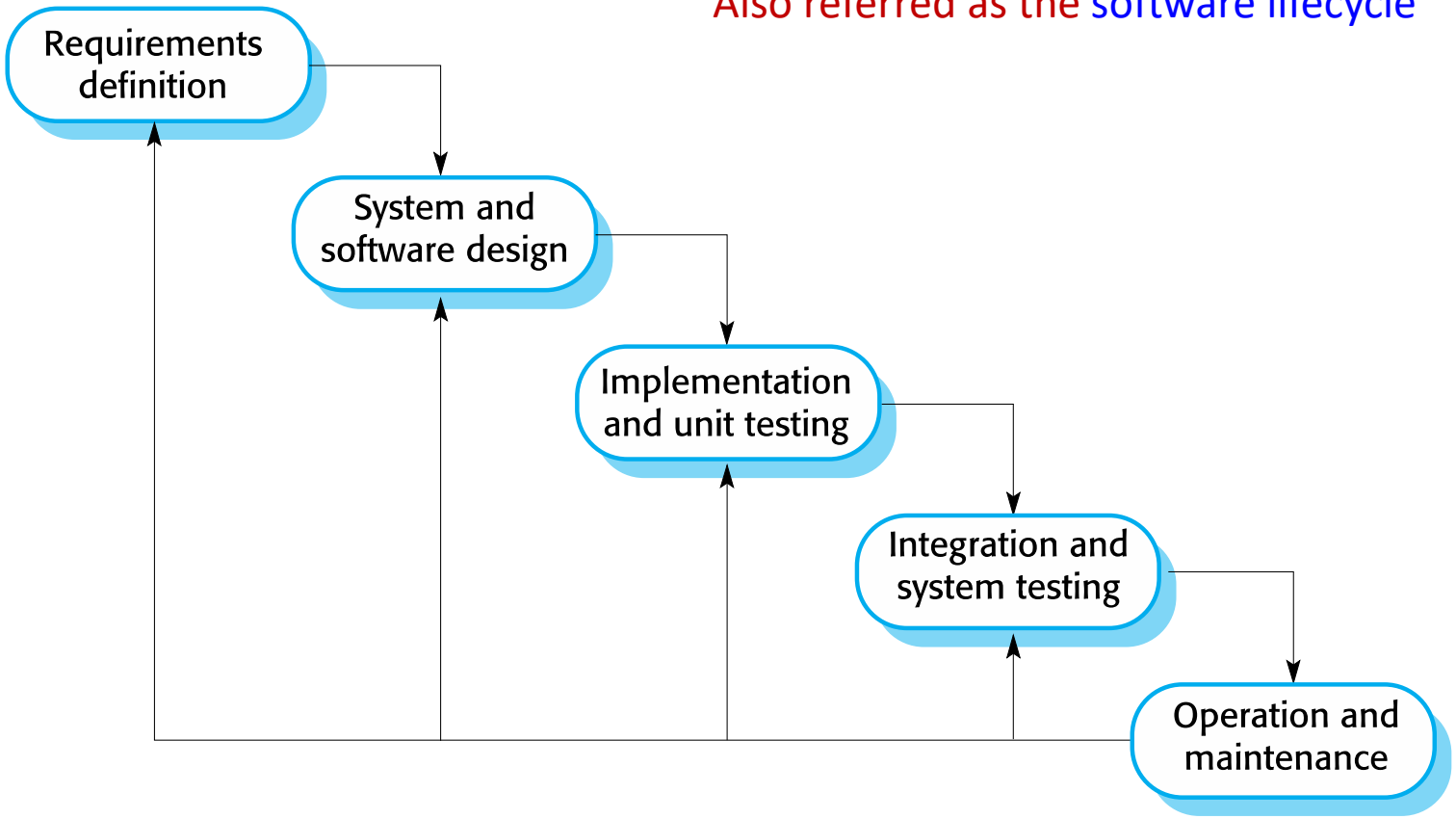


Figure from: Software Engineering (9th Edition), by I. Sommerville

Waterfall Model

- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway
- In principle, a phase has to be completed 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.
  - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Incremental Development

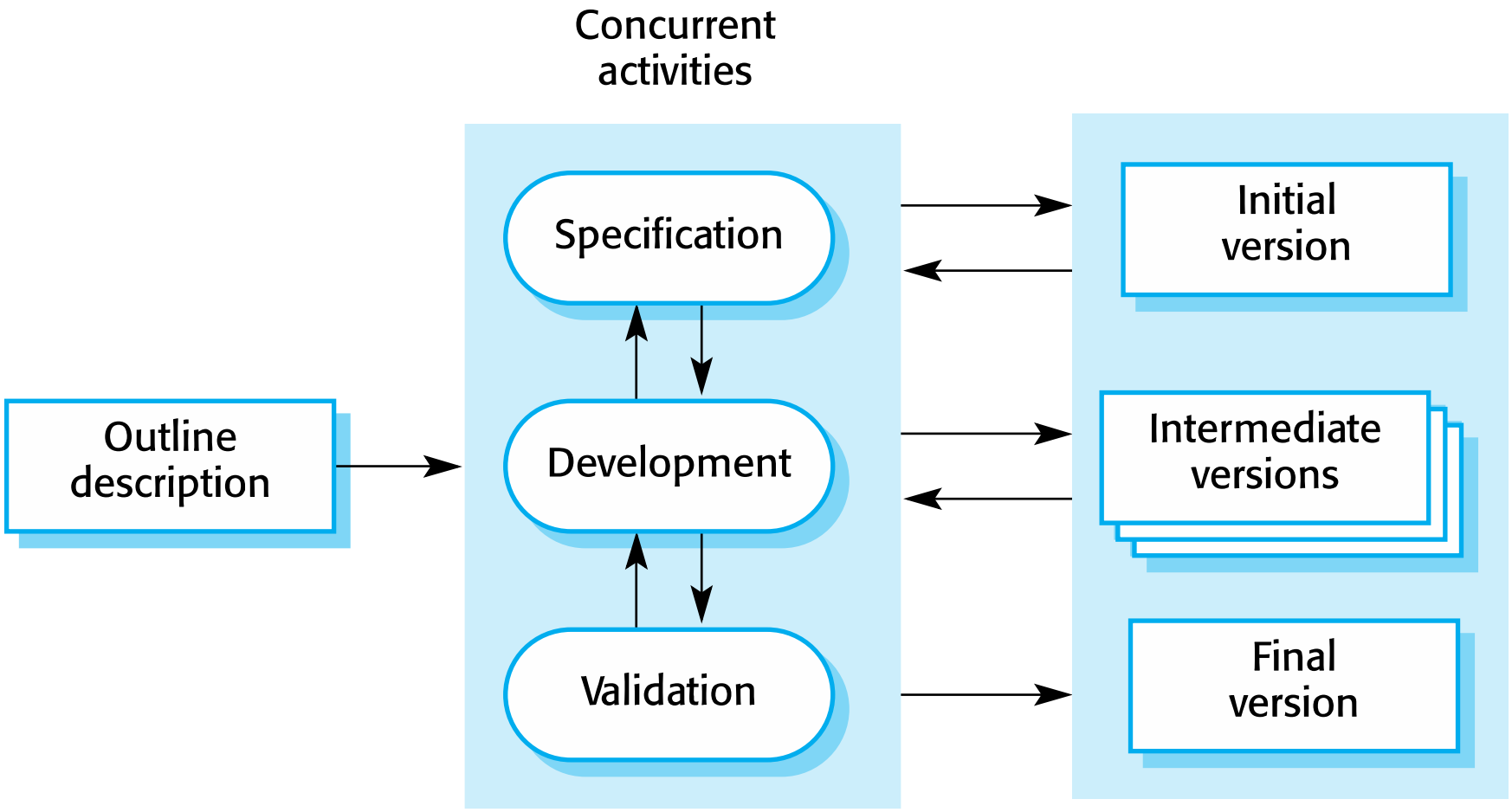
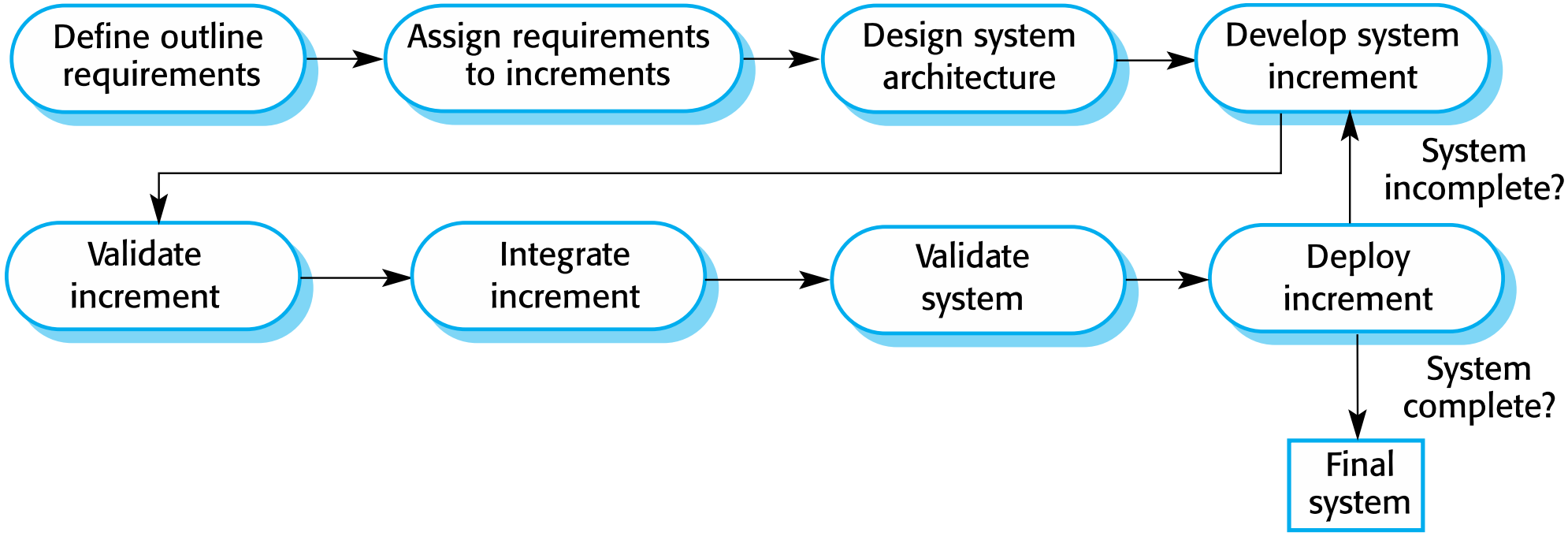


Figure from: Software Engineering (9th Edition), by I. Sommerville

Incremental delivery process



Incremental Development

- The cost of accommodating changing customer requirements is reduced
- It is easier to get customer feedback on the development work that has been done
- More rapid delivery and deployment of useful software to the customer is possible
- 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.

Incremental development problems

- 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.
- System structure tends to degrade as new increments are added.
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.
- Iterative development is problematic when the new system is intended to replace an existing system. Users need all the functionality and it is often impractical t use both the new(incomplete) system with the old system; as they are likely to have different UIs and different Databases.

Coping with change

Coping with change

- Change is inevitable in all large software projects.
  - Business changes lead to new and changed system requirements
  - New technologies open up new possibilities for improving implementations
  - Changing platforms require application changes
- Change leads to rework so the costs of change include both rework (e.g. re-analyzing requirements) as well as the costs of implementing new functionality

Reducing the costs of rework

- **Change anticipation** (change avoidance), where the software process includes activities that can anticipate possible changes before significant rework is required.
  - For example, a prototype system may be developed to show some key features of the system to customers. *Reduces the numbers of required changes.*

- **Change tolerance,** where the process is designed so that changes can be accommodated at relatively low cost.
  - This normally involves some form of incremental development. Proposed changes may be implemented in increments that have not yet been developed. If this is impossible, then only a single increment (a small part of the system) may have be altered to incorporate the change.

Coping with changing requirements

Two ways of coping with change and changing system requirements. These are:

- System prototyping, where a version of the system or part of the system is developed quickly to check the customer's requirements and the feasibility of design decisions. This approach supports change anticipation.
- Incremental delivery, where system increments are delivered to the customer for comment and experimentation. This supports both change avoidance and change tolerance.

Prototyping Model

- A prototype is an initial version of a system used to demonstrate concepts and try out design options
- A prototype can be used in:
  - The requirements engineering process to help with requirements elicitation and validation
  - In design processes to explore options and develop a UI design
  - In the testing process to run back-to-back tests

Prototyping Model

Checkout these prototypes

https://www.fullstacklabs.co/prototypes

Prototyping Model

Benefits of prototyping

- Improved system usability
- A closer match to users' real needs
- Improved design quality
- Improved maintainability
- Reduced development effort

Prototyping Model

- May be based on rapid prototyping languages or tools
- May involve leaving out functionality
  - Prototype should focus on areas of the product that are not well-understood
  - Error checking and recovery may not be included in the prototype
  - Focus on functional rather than non-functional requirements such as reliability and security

Prototyping Model

- Prototypes should be discarded after development as they are not a good basis for a production system:
  - It may be impossible to tune the system to meet non-functional requirements
  - Prototypes are normally undocumented
  - The prototype structure is usually degraded through rapid change
  - The prototype probably will not meet normal organizational quality standards

Iterative Development

Includes the elements of the Waterfall Model
but organized into an iterative, incremental process

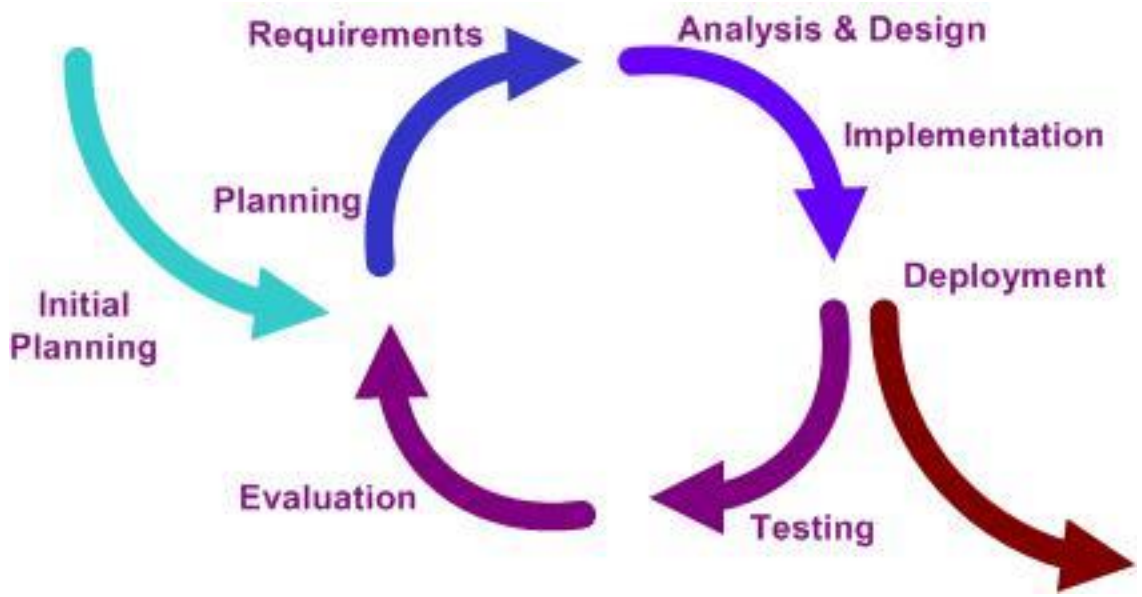


Figure from: wikipedia.org

Iterative Development

Rational Unified Process (RUP)

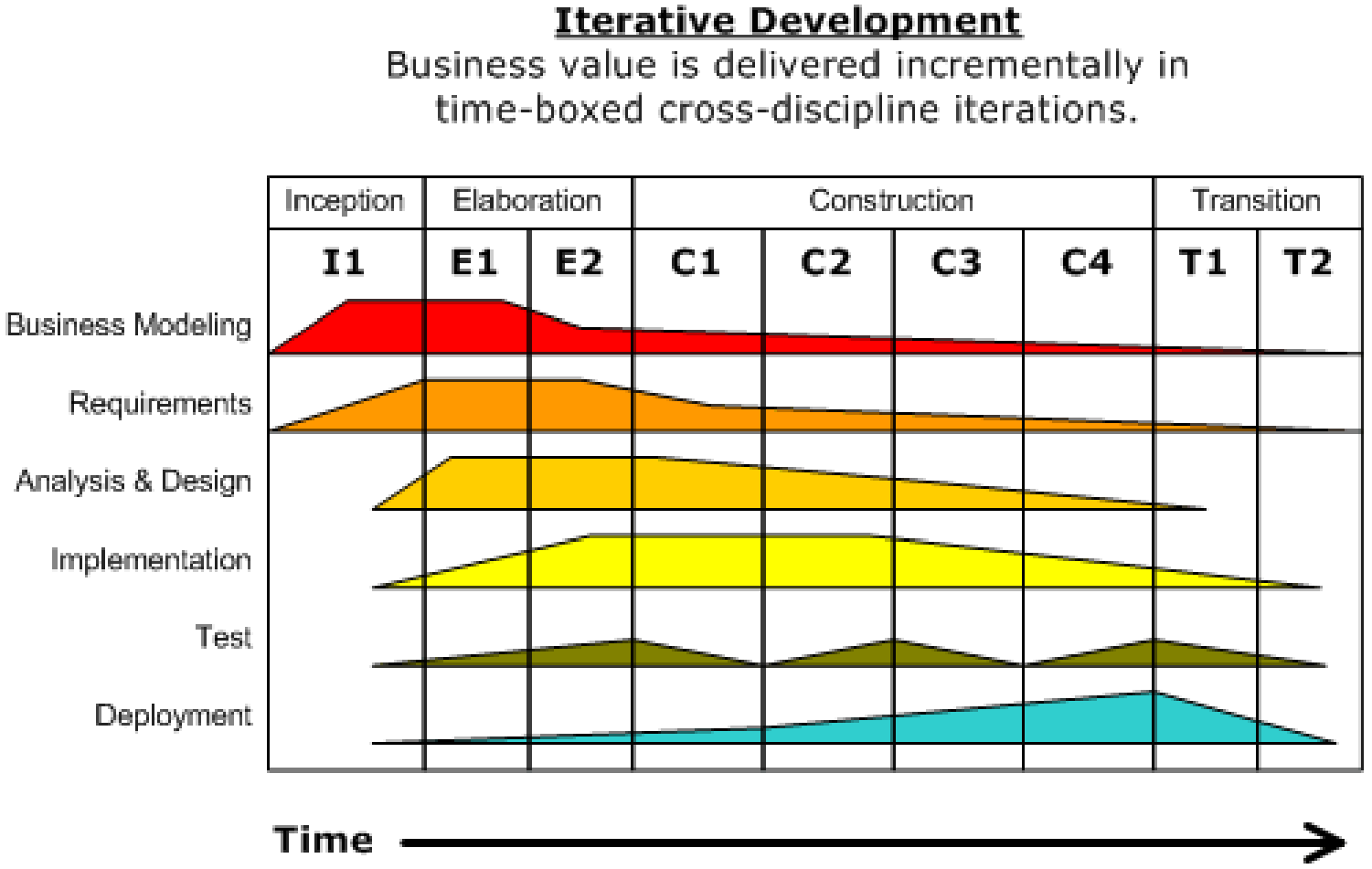
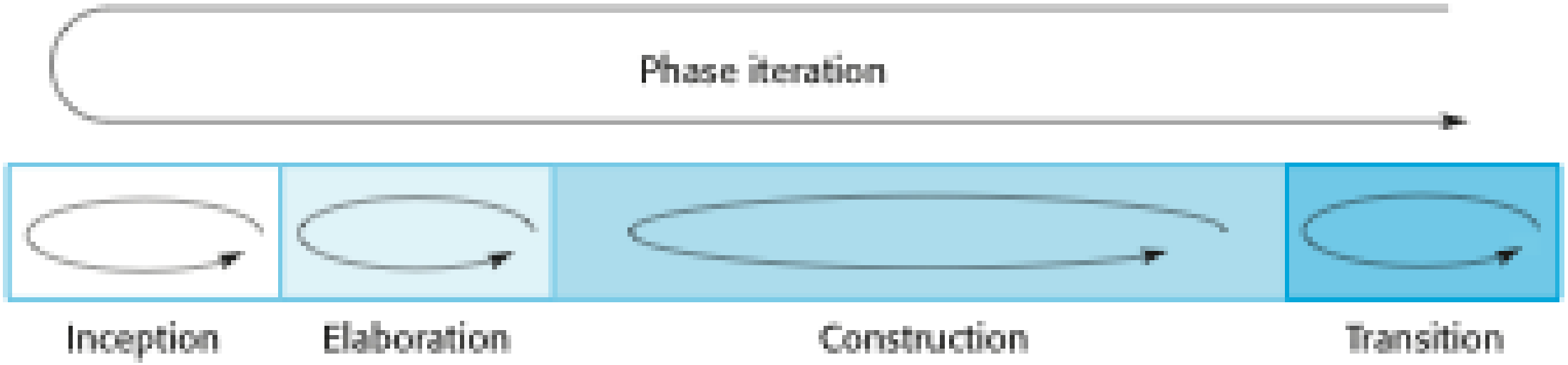


Figure from: wikipedia.org

Phases in the Rational Unified Process



RUP phases

- Inception
  - Establish the business case for the system.
- Elaboration
  - Develop an understanding of the problem domain and the system architecture.
- Construction
  - System design, programming and testing.
- Transition
  - Deploy the system in its operating environment.

RUP iterations

- In-phase iteration
  - Each phase is iterative with results developed incrementally.
- Cross-phase iteration
  - As shown by the loop in the RUP model, the whole set of phases may be enacted incrementally.

RUP

The following iterations are characteristic:

- a short Inception iteration to establish scope and vision, and to define the business case
- a single Elaboration iteration, during which requirements are defined, and the architecture established
- several Construction iterations during which the use cases are realized and the architecture fleshed-out
- several Transition iterations to migrate the product into the user community

Static workflows in the Rational Unified Process

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.

Static workflows in the Rational Unified Process

Workflow	Description
Testing	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow managed changes to the system (see Chapter 25, Sommerville book).
Project management	This supporting workflow manages the system development
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

Integration and configuration

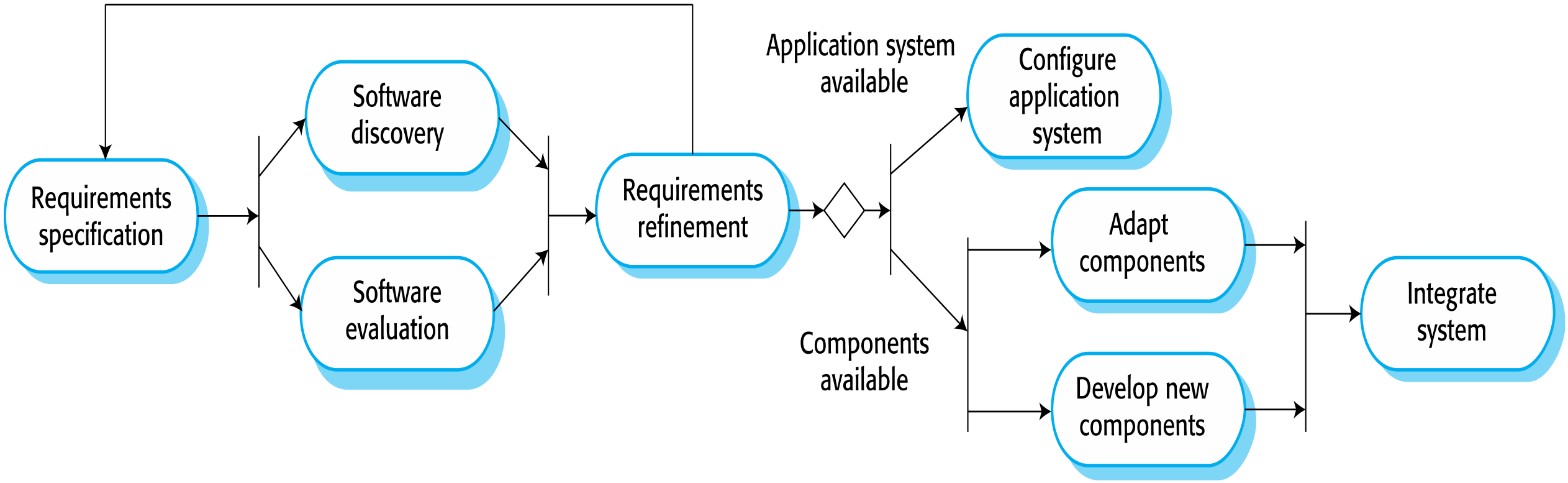
Reuse-oriented development

- Based on software reuse where systems are integrated from existing components or application systems (sometimes called COTS -Commercial-off-the-shelf systems).
- Reused elements may be configured to adapt their behaviour and functionality to a user's requirements
- Reuse is now the standard approach for building many types of business system

Types of reusable software

- Stand-alone application systems (sometimes called COTS) that are configured for use in a particular environment.
- Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE.
- Web services that are developed according to service standards and which are available for remote invocation.

Reuse-oriented software engineering



From Sommerville book, Chapter 2 Software Processes

Key process stages

- Requirements specification
- Software discovery and evaluation
- Requirements refinement
- Application system configuration
- Component adaptation and integration

Advantages and disadvantages

- Reduced costs and risks as less software is developed from scratch
- Faster delivery and deployment of system
- But requirements compromises are inevitable so system may not meet real needs of users
- Loss of control over evolution of reused system elements

Reuse-Oriented Model

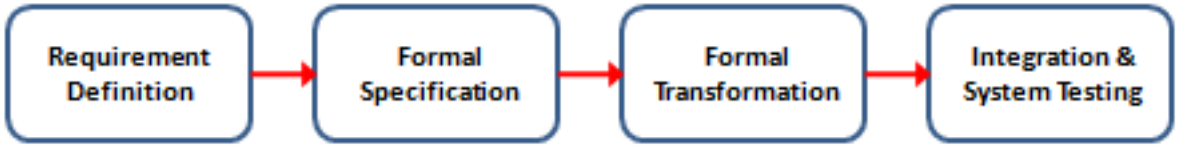
Examples of software component types:

- 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 (COTS) that are configured for use in a particular environment

Formal Transformation Model

- Based on the transformation of a mathematical specification through different representations to an executable program

- Transformations are ”correctness-preserving” so it is straightforward to show that the program conforms to its specification



Formal Transformation Model

- Problems
  - Need for specialized skills and training to apply the technique
  - Difficult to formally specify some aspects of the system such as the user interface
- Applicability
  - Critical systems especially those where a safety or security case
  - must be made before the system is put into operation

Our Software Process

It includes the following phases:

- Requirements elicitation
- Analysis
- System Design
- Object Design
- Implementation
- Testing

A note on the SE course

- In our SE course, we must learn many aspects of software engineering.
- We may use some aspects of Scrum when appropriate.
- Using Scrum requires a very good, up-front understanding of software development as well as the necessary tools and techniques.
- It is not the best approach to *learning* software engineering.

Software Cost

- Software development is expensive
- It is difficult to estimate the development cost
- All phases of a software process (software lifecycle) involve cost
- How is the cost distributed?

Software Cost Distribution

Estimated cost	Phase
2%	Concept & Definition
4%	Requirements Definition
7%	Software Architecture Design
6%	Detailed Software Design
7%	Code and Unit Test
12%	Integration and System Test
3%	Acceptance Testing
1%	Replication, Storage and Shipment
2%	Delivery, installation and Training
55%	Maintenance
1%	Retirement

Agile Development

- In the late 90s, Agile Development has been created as a response to fully planned, “heavy-weight” development.
- Created to address constantly changing requirements and need for rapid software development.
- Experience shows that when heavy-weight processes are applied to small/medium systems, the overhead of planning and design dominates the development process.

Agile Development

- System is developed through small, frequent, incremental releases.
- Requirements include relatively simple customer stories, easy to modify.
- Complete requirements document is usually not created.
- Customers are continuously engaged, and their representatives often take part in the development.

Agile Development

Agile Manifesto (Beck, Beedle, Schwaber, and 13 others, 2001)

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

principles underlying agile development

Agile Development

Agile methodologies include:

- Extreme Programming (XP)
- Dynamic systems development method (DSDM)
- Kanban (inspired by Toyota Production System and lean manufacturing)
- Scrum

Next Lecture

- Scrum Methodology and Scrum project management