Introduction of Software Engineering SOF107

CHAPTER 2: SOFTWARE PROCESSES

Dr. Hejab M. Al-Fawareh

alfawarehhejab.Khaled@xmu.edu.my

0199184370

XIAMEN UNIVERSITY MALAYSIA
厦門大學馬來西亞分校

University mission and vision

- **Vision**
- Xiamen University Malaysia aspires to become a university with a distinct global outlook, featuring first-class teaching and research, and embracing cultural diversity.
- **Mission**
- To nurture young talents with dignity and wisdom, turning them into fine citizens of the region who will contribute to the prosperity of the people and social progress of Malaysia, China and Southeast Asia.

Learning Outcome

- Describe appropriate knowledge of the fundamentals of software engineering.

At the end of these lectures students are able to:

1.Understand the concepts software processes and software process models.

2. Explain the general software process models.

3. Describe the fundamental process activities.

4. Understand why processes should be organized to cope with changes in the software requirements and design

5. Understand the notion of software process improvement and the Factors that affect software process quality.

6. Understand agile development concept and method.

Content

- Software Process
- Software Process Model.
- Agile Software Development

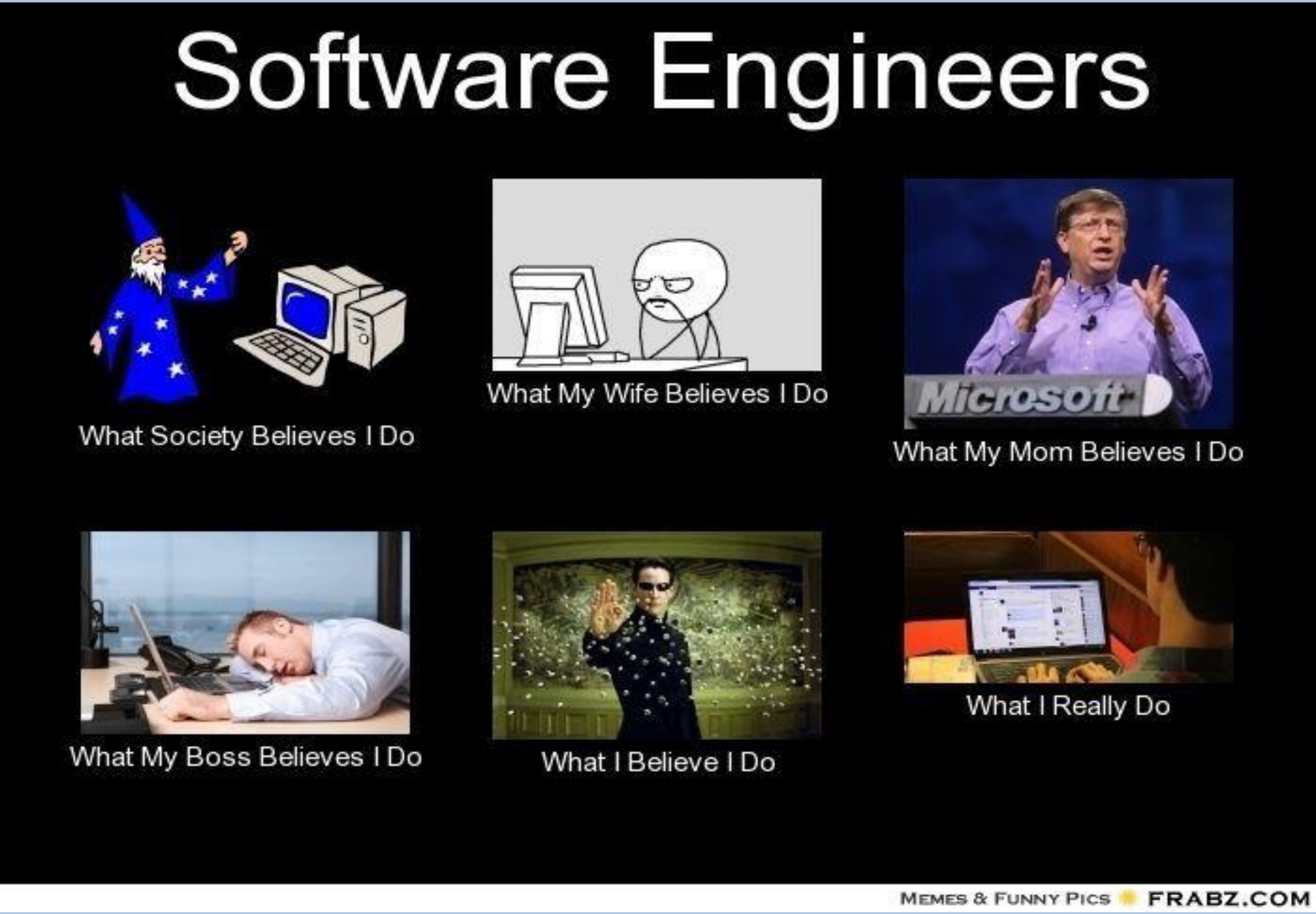
Introduction

- A software process is a set of related activities that leads to the production of software product.
- These activities may involve the development of software from scratch in a standard programming language like Java or C
- Software processes are complex and rely on people making decisions and judgments.
- There is no ideal process and most organizations have developed their own software development processes.
- Processes have evolved to take advantage of the capabilities of the people in an organization and the specific characteristics of the systems that are being developed.

types of software processes

There are 2 types of software processes: 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.
- agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
- In practice, most practical processes include elements of both plan-driven and agile approaches.


Software Engineers
What Society Believes I Do
What My Wife Believes I Do
Microsoft
What My Mom Believes I Do
What My Boss Believes I Do
What I Believe I Do
What I Really Do
MEMES & FUNNY PICS FRABZ.COM

Software process models

A software process model is a simplified representation of a software process.

Each process model represents a process from a particular perspective, and thus provides only partial information about that process.

The waterfall model

- The waterfall model is an example of a plan-driven process—in principle, you must plan and schedule all of the process activities before starting work on them.

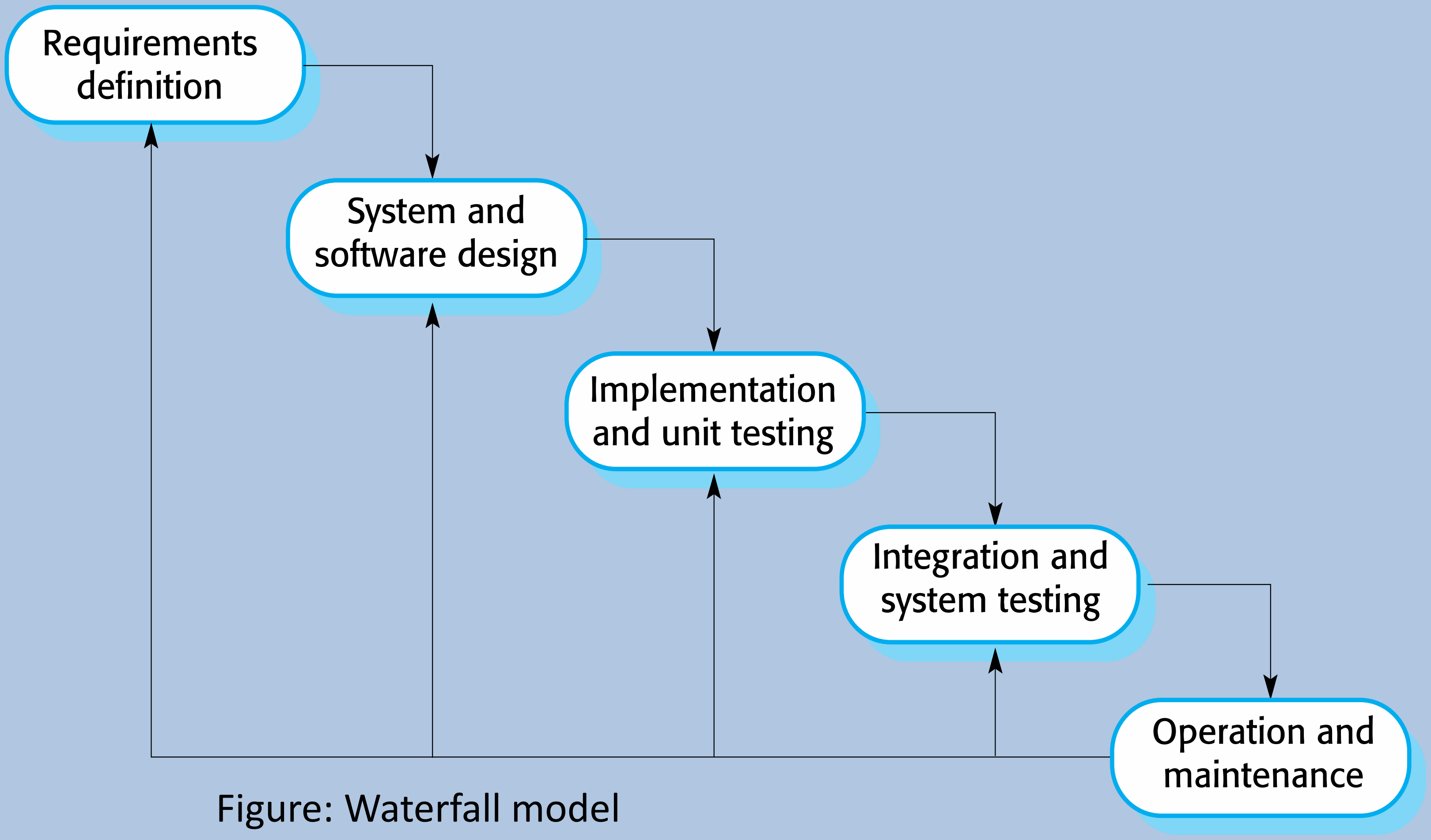

Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Figure: Waterfall model

There are separate identified phases in the waterfall model:

- Requirements analysis and definition

The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification

- System and software design

The systems design process allocates the requirements to either hardware or software systems by establishing an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.

- Implementation and unit testing

During this stage, the software design is realized as a set of programs or 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

The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle.

Problems of the Waterfall model

- It is difficult to make any change when the development process is underway. In principle, a phase has to be complete before moving onto the next phase.

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.

- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.

Incremental Model

- 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.

  It is an agile process.

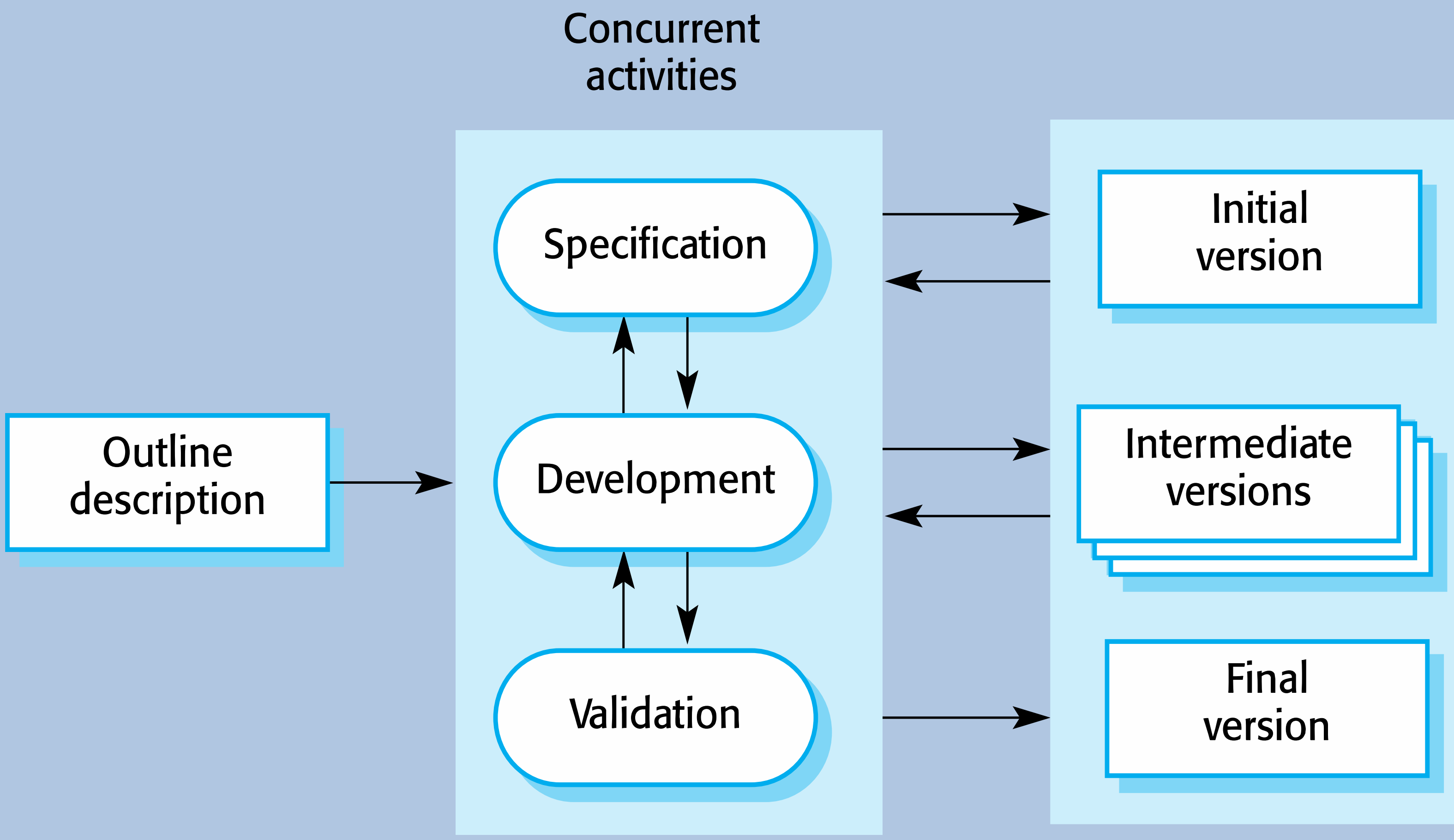


Figure: Incremental Model

Benefits of Incremental development

1. The cost of accommodating changing customer requirements is reduced. The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.

2. It is easier to get customer feedback on the development work that has been done. Customers can comment on demonstrations of the software and see how much has been implemented. Customers find it difficult to judge progress from software design documents.

3.More rapid delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included. Customers are able to use and gain value from the software earlier than is possible with a waterfall process

Problems of Incremental Approach

From **a management perspective**, the incremental approach has two problems:

1. The process is not visible.
2. 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.

Prototyping

- A prototype is **an initial version of a software system** that is used to demonstrate concepts, try out design options, and find out more about the problem and its possible solutions.
- **Rapid, iterative development** of the prototype is essential so that costs are controlled and system stakeholders can experiment with the prototype early in the software process.

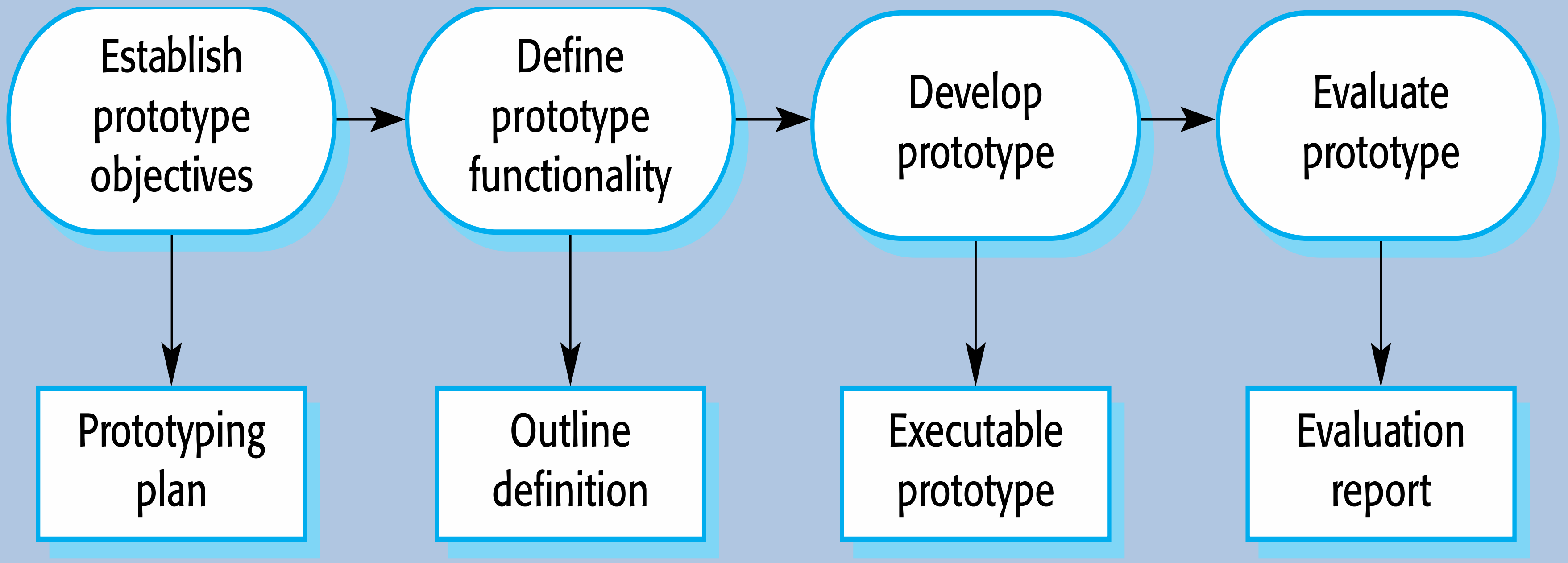


Figure: The process of software prototype development

- A software prototype can be used in a software development process to help anticipate changes that may be required:

- In the requirements engineering process, a prototype can help with the elicitation and validation of system requirements.
- In the system design process, a prototype can be used to explore particular software solutions and to support user interface design.
- A system prototype may be used while the system is being designed to carry out design experiments to check the feasibility of a proposed design. For example, a database design may be prototyped and tested to check that it supports efficient data access for the most common user queries. Prototyping is also an essential part of the user interface design process

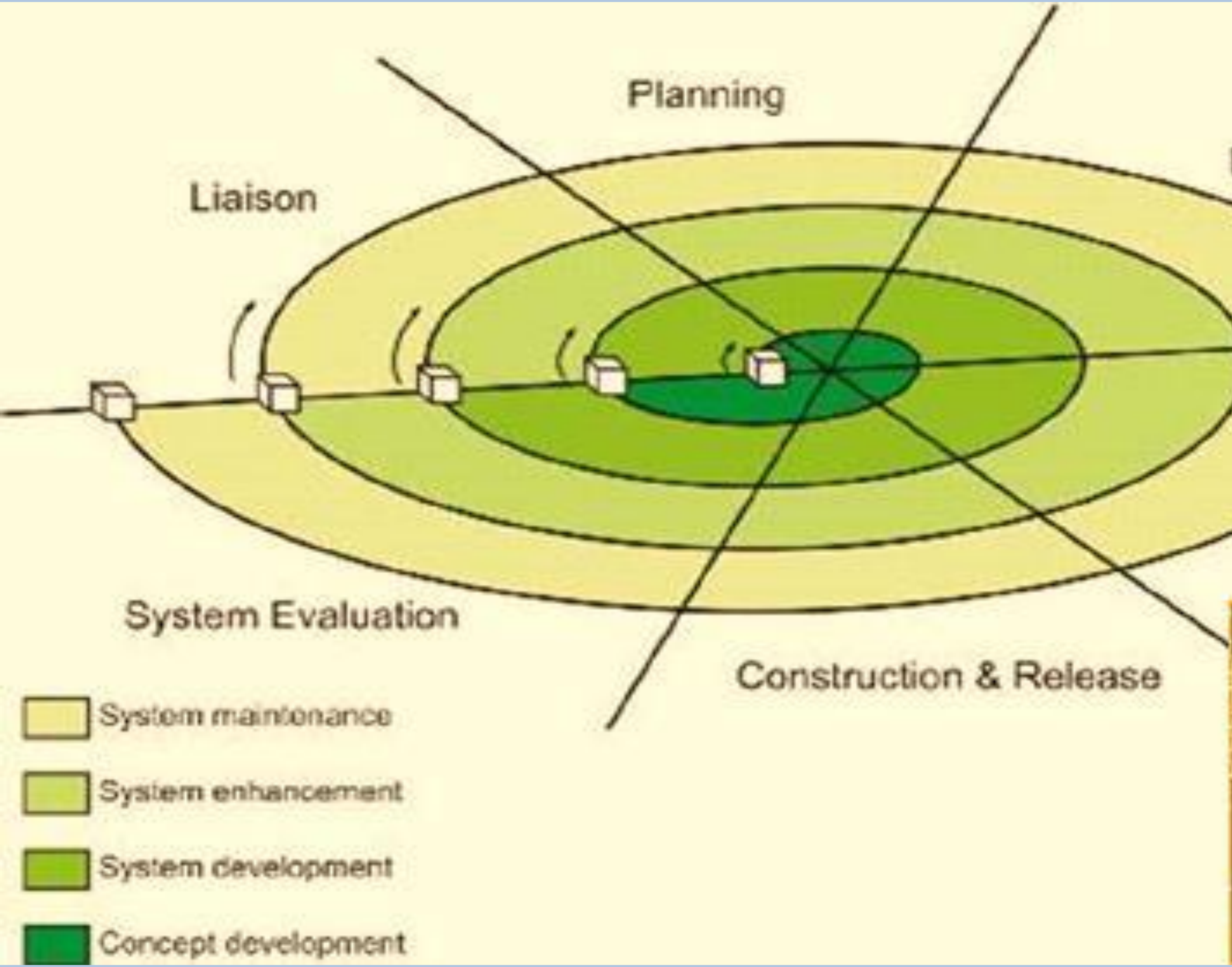
Benefits of Prototyping

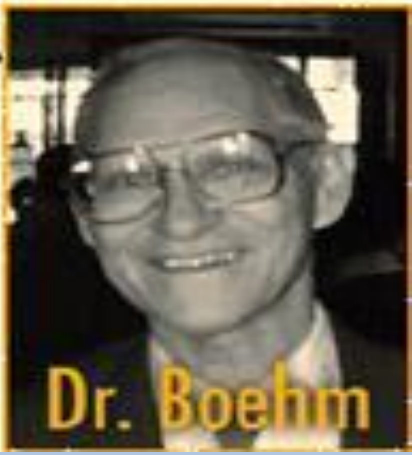
- Improved system usability.
- A closer match to users' real needs.
- Improved design quality.
- Improved maintainability.
- Reduced development effort.

A general problem

- with prototyping is that the prototype may not necessarily be used in the same way as the final system.
- The tester of the prototype may not be typical of system users.
- The training time during prototype evaluation may be insufficient.
- If the prototype is slow, the evaluators may adjust their way of working and avoid those system features that have slow response times.

Spiral Model



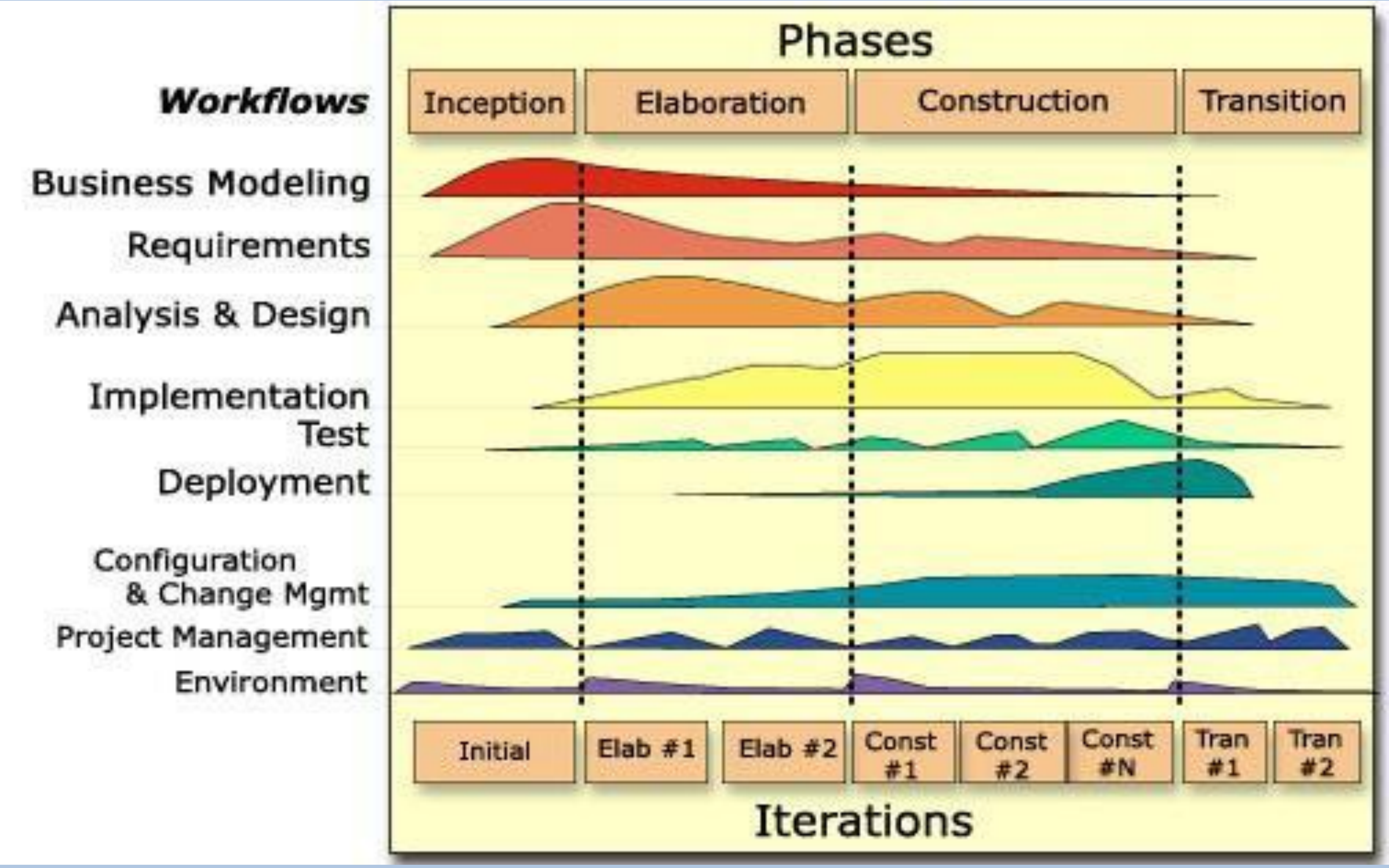


Spiral Model Phases

- Planning :It includes estimating the cost, schedule and resources for the iteration. It also involves understanding the system requirements for continuous communication between the system analyst and the customer.
- Risk Analysis : Identification of potential risk is done while risk mitigation strategy is planned and finalized.
- Engineering : It includes testing, coding and deploying software at the customer site.

Evaluation: evaluation of software by the customer. Also, includes identifying and monitoring risks such as schedule slippage and cost overrun.

Rational Unified Process



Rational Unified Process

- The Rational Unified Process (RUP) (Krutchen, 2003)
- an adaptable process model that has been derived from work on the UML (Unified Modelling Language) and the associated Unified Software Development Process (Rumbaugh, et al., 1999) (Arlow and Neustadt, 2005).
- It brings together elements from all of the generic process models and supports prototyping and incremental delivery

Rational Unified Process

RUP is normally described from three perspectives:

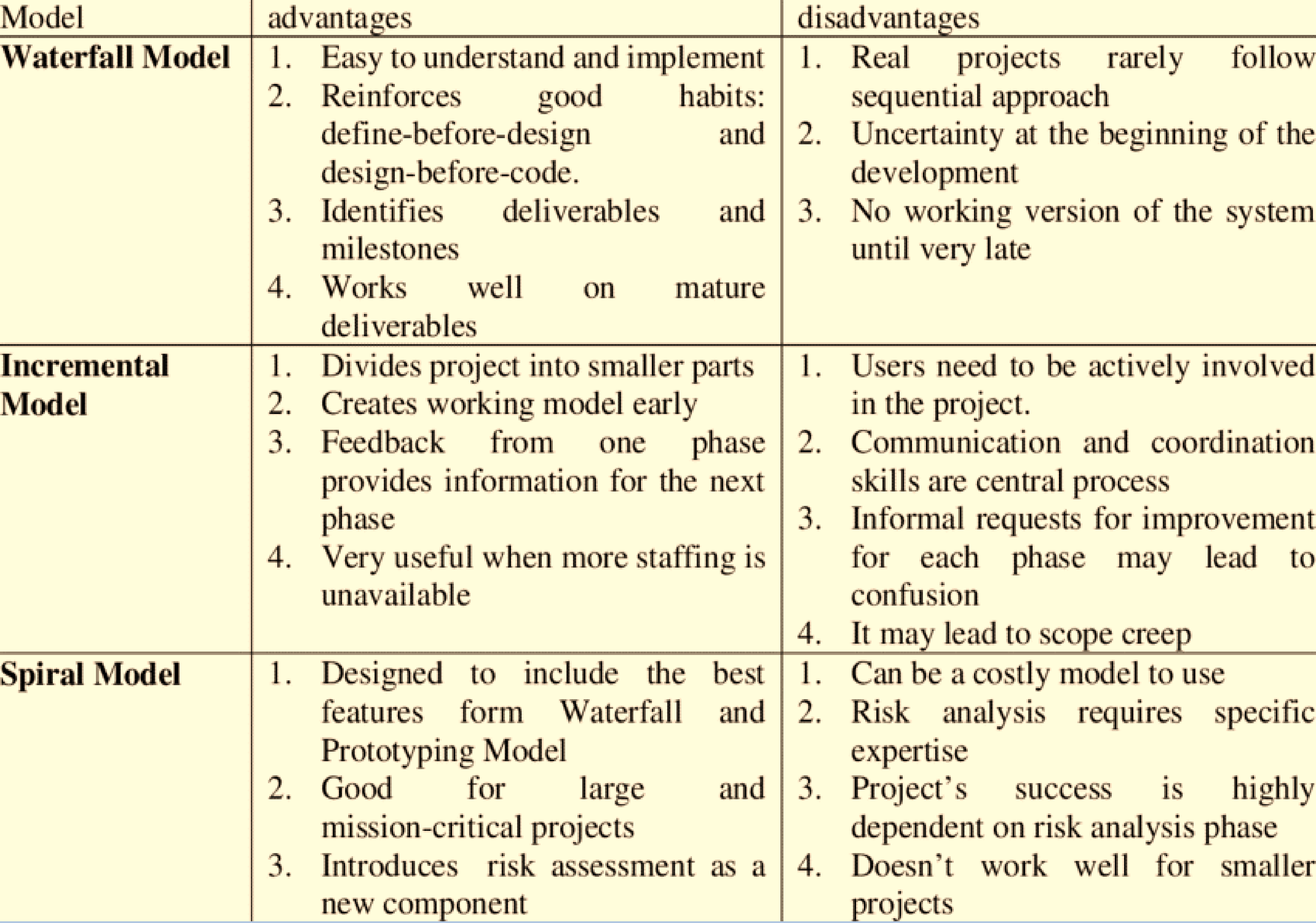
1. A dynamic perspective shows the phases of the model over time.
2. A static perspective shows the process activities that are enacted.
3. A practice perspective suggests good practices to be used during the process

RUP Workflow

- 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

RUP Workflow

- 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.
- 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.



Model	advantages	disadvantages
Waterfall Model	1. Easy to understand and implement 2. Reinforces good habits: define-before-design and design-before-code. 3. Identifies deliverables and milestones 4. Works well on mature deliverables	1. Real projects rarely follow sequential approach 2. Uncertainty at the beginning of the development 3. No working version of the system until very late
Incremental Model	1. Divides project into smaller parts 2. Creates working model early 3. Feedback from one phase provides information for the next phase 4. Very useful when more staffing is unavailable	1. Users need to be actively involved in the project. 2. Communication and coordination skills are central process 3. Informal requests for improvement for each phase may lead to confusion 4. It may lead to scope creep
Spiral Model	1. Designed to include the best features form Waterfall and Prototyping Model 2. Good for large and mission-critical projects 3. Introduces risk assessment as a new component	1. Can be a costly model to use 2. Risk analysis requires specific expertise 3. Project's success is highly dependent on risk analysis phase 4. Doesn't work well for smaller projects

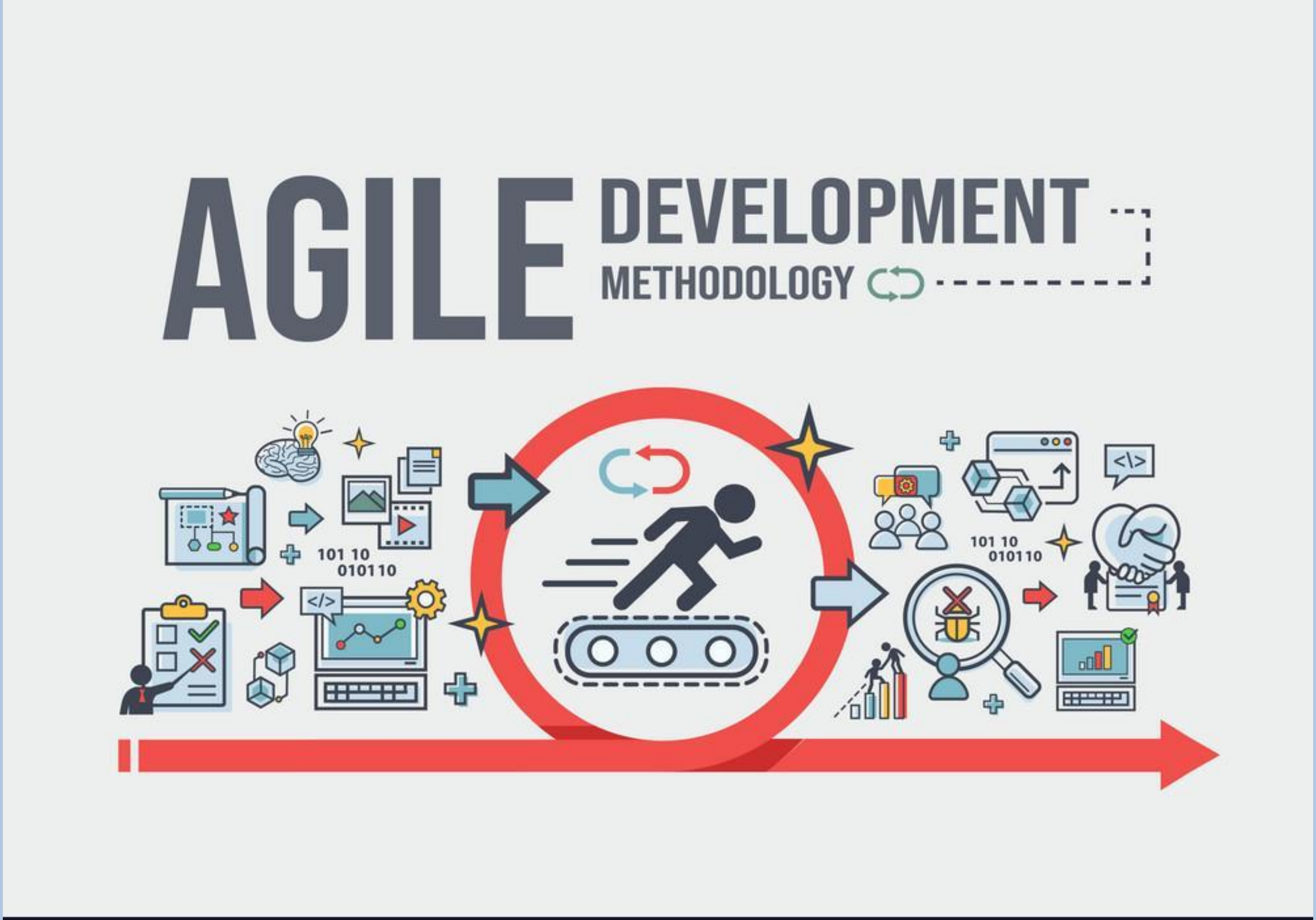
Exercises

1. **Compare the differences between the waterfall model and the incremental model**

2. You are a team manager at ABC company. You are responsible to advise to your customers that the incremental software development could be an effective solution for them. Prepare a short report to present to your customer about the effectiveness of the incremental software development.

3. Explain why there is no universal process model that is right for all kinds of software development.

Agile Software Development


AGILE DEVELOPMENT
METHODOLOGY
101 10
010110
</>

Rapid software development

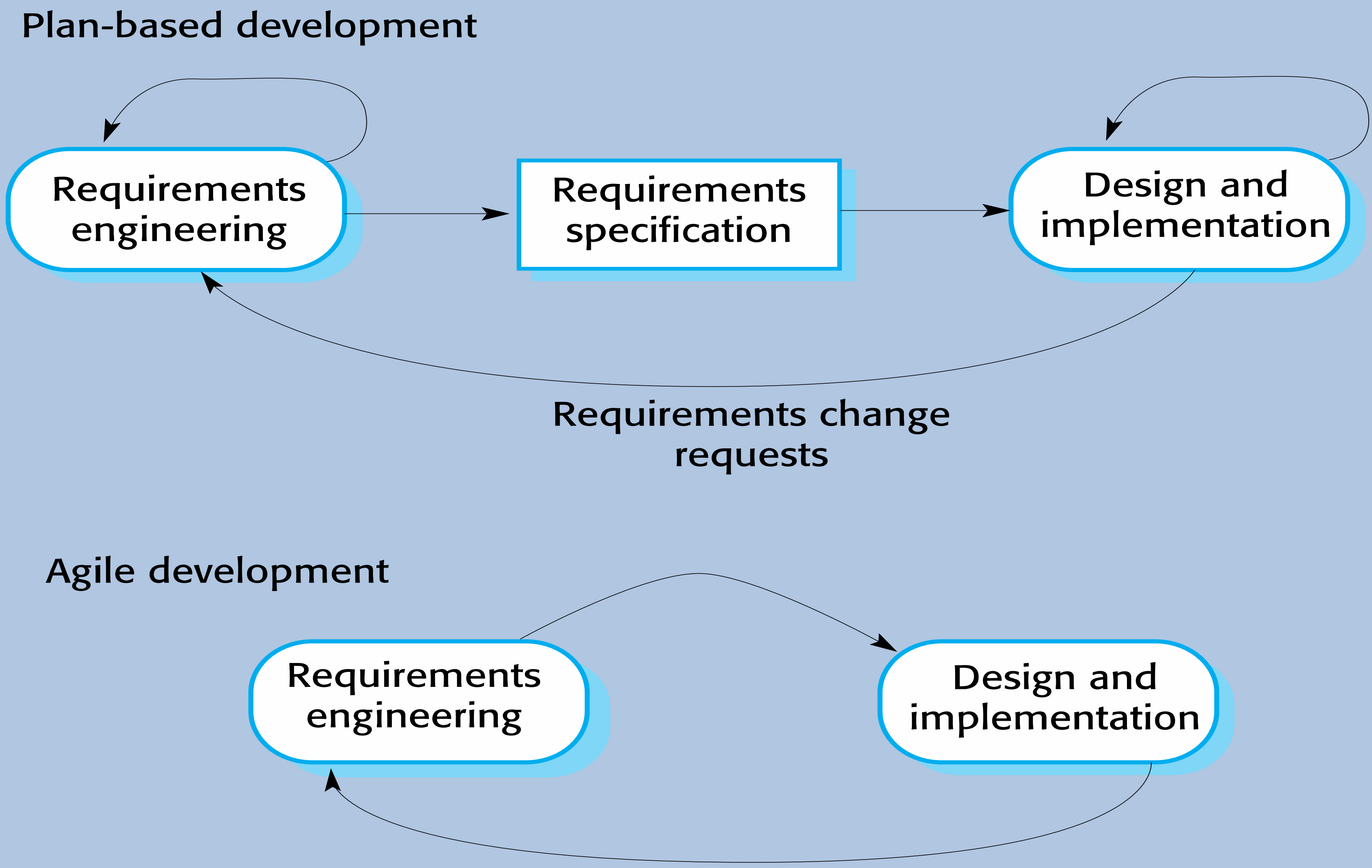
- Rapid development and delivery is now often the most important requirement for software systems
  - Businesses operate in a fast –changing requirement and it is practically impossible to produce a set of stable software requirements
  - Software has to evolve quickly to reflect changing business needs.

- Plan-driven development is essential for some types of system but does not meet these business needs.
- Agile development methods emerged in the late 1990s whose aim was to radically reduce the delivery time for working software systems

Agile development

- Program specification, design and implementation are inter-leaved
- The system is developed as a series of versions or increments with stakeholders involved in version specification and evaluation
- Frequent delivery of new versions for evaluation
- Extensive tool support (e.g. automated testing tools) used to support development.
- Minimal documentation – focus on working code

Plan-driven and agile development


Plan-based development
Requirements engineering
Requirements specification
Design and implementation
Requirements change requests
Agile development
Requirements engineering
Design and implementation

Plan-driven and agile development

- **Plan-driven development**
  - A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance.
  - Not necessarily waterfall model – plan-driven, incremental development is possible
  - Iteration occurs within activities.
- **Agile development**
  - Specification, design, implementation and testing are inter-leaved and the outputs from the development process are decided through a process of negotiation during the software development process.

Agile development

Specification, design, implementation and testing are inter-leaved and the outputs from the development process are decided through a process of negotiation during the software development process.

Agile methods

- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods. These methods:
  - Focus on the code rather than the design
  - Are based on an iterative approach to software development
  - Are intended to deliver working software quickly and evolve this quickly to meet changing requirements.

- methods is to reduce overheads in the software process (e.g. by limiting documentation) and to be able to respond quickly to changing .

- The aim of agile requirements without excessive rework.

Agile manifesto

- *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
- *That is, while there is value in the items on the right, we value the items on the left more.*

The principles of agile methods

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

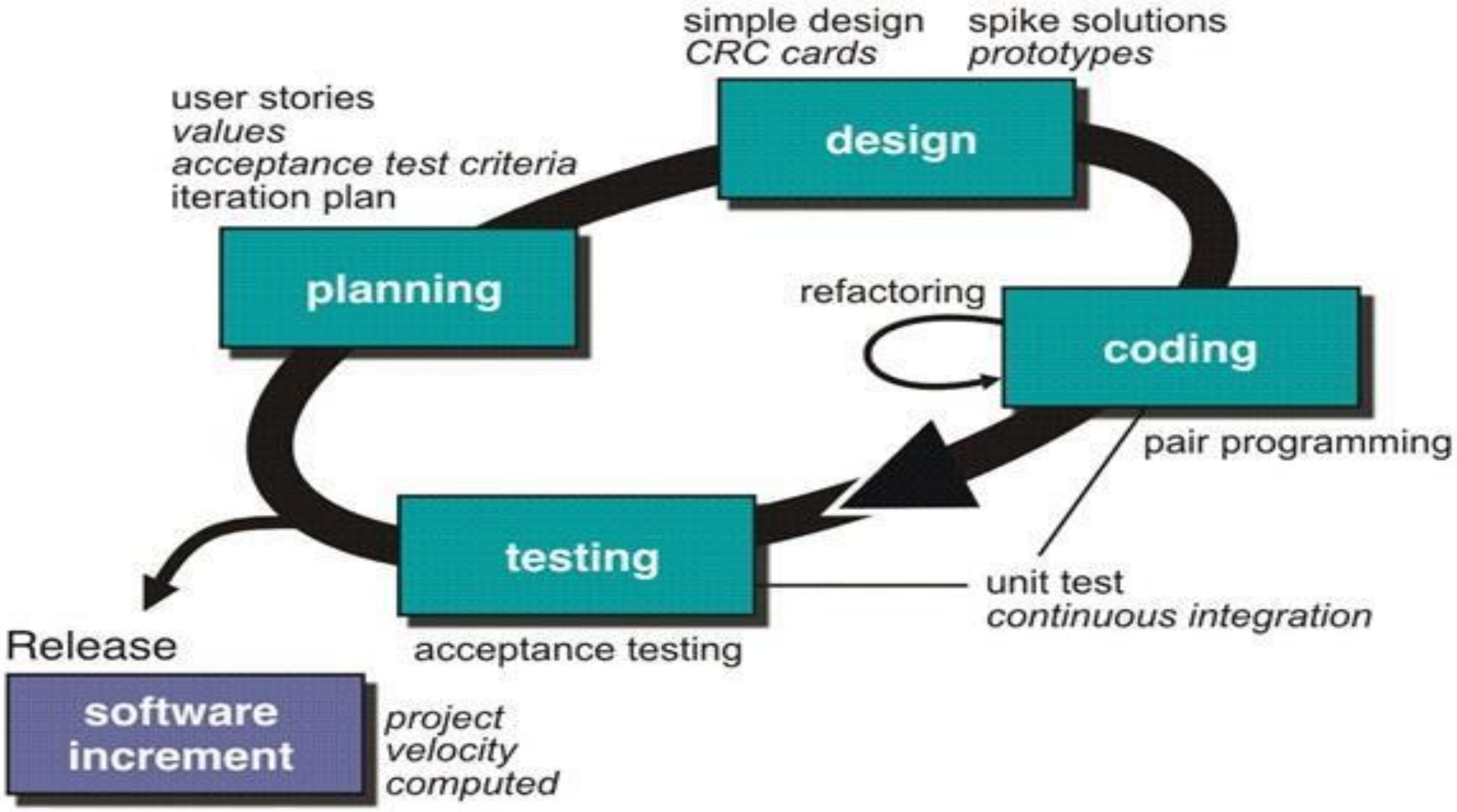
Agile method applicability

- Product development where a software company is developing a small or medium-sized product for sale.
  - Virtually all software products and apps are now developed using an agile approach
- Custom system development within an organization, where there is a clear commitment from the customer to become involved in the development process and where there are few external rules and regulations that affect the software.

Exercises

- Compare the differences between the **Plan-driven development** and **Agile development**

Extreme Programming (XP)



Extreme Programming (XP)

- The most widely used agile process, originally proposed by Kent Beck in 2004. It uses an object-oriented approach.

XP Planning

- Begins with the listening, leads to creation of “user stories” that describes required output, features, and functionality. Customer assigns a value(i.e., a priority) to each story.
- Agile team assesses each story and assigns a cost (development weeks. If more than 3 weeks, customer asked to split into smaller stories)
- Working together, stories are grouped for a deliverable increment next release.

Extreme Programming (XP)

- A commitment (stories to be included, delivery date and other project matters) is made. Three ways:

  1. Either all stories will be implemented in a few weeks,
  2. high priority stories first.
  3. the riskiest stories will be implemented first.

- After the first increment "project velocity", namely number of stories implemented during the first release is used to help define subsequent delivery dates for other increments. Customers can add stories, delete existing stories, change values of an existing story, split stories as development work proceeds.

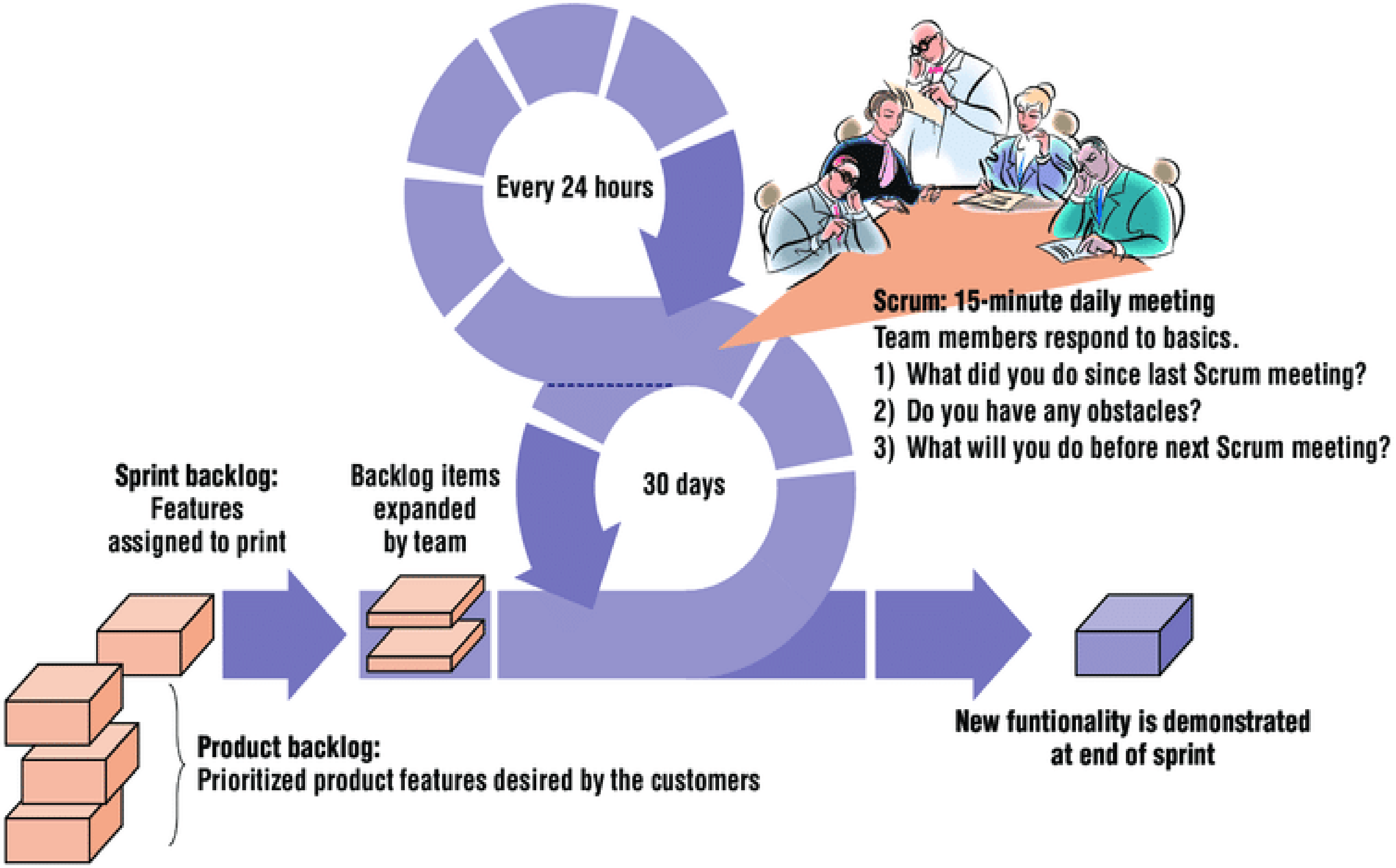
Extreme Programming (XP)

- XP Design (occurs both before and after coding as refactoring is encouraged)
- Follows the KIS principle (keep it simple) Nothing more nothing less than the story.
- Encourage the use of CRC (class-responsibility-collaborator) cards in an object-oriented context. The only design work product of XP. They identify and organize the classes that are relevant to the current software increment.
- For difficult design problems, suggests the creation of "spike solutions"—a design prototype for that portion is implemented and evaluated.
- Encourages "refactoring"—an iterative refinement of the internal program design. Does not alter the external behavior yet improve the internal structure. Minimize chances of bugs. More efficient, easy to read

Extreme Programming (XP)

- XP Coding
- Recommends the construction of a unit test for a story before coding commences. So implementer can focus on what must be implemented to pass the test.
- Encourages "pair programming". Two people work together at one workstation. Real time problem solving, real time review for quality assurance. Take slightly different roles.
- XP Testing
- All unit tests are executed daily and ideally should be automated. Regression tests are conducted to test current and previous components.
- "Acceptance tests" are defined by the customer and executed to assess customer visible functionality

SCRUM Process Flow


Every 24 hours
Scrum: 15-minute daily meeting
Team members respond to basics.
1) What did you do since last Scrum meeting?
2) Do you have any obstacles?
3) What will you do before next Scrum meeting?
Sprint backlog:
Features
assigned to print
Backlog items
expanded
by team
30 days
Product backlog:
Prioritized product features desired by the customers
New funtionality is demonstrated
at end of sprint

SCRUM Process Flow

- A software development method Originally proposed by Schwaber and Beedle (an activity occurs during a rugby match) in early 1990.
- Development work is partitioned into "packets"
- Testing and documentation are on-going as the product is constructed
- Work units occurs in "sprints" and is derived from a "backlog" of existing changing prioritized requirements
- Changes are not introduced in sprints (short term but stable) but in backlog.
- Meetings are very short (15 minutes daily) and sometimes conducted without chairs (what did you do since last meeting? What obstacles are you encountering? What do you plan to accomplish by next meeting?)
- "demos" are delivered to the customer with the time-box allocated. May not contain all functionalities. So customers can evaluate and give feedbacks