Software Engineering
CS-303

Software Process & Process Models

Lecture # 9, 10
11, 12 Feb

Rubab Jaffar
rubab.jaffar@nu.edu.pk

Today's Outline

- Integration and Configuration
- Software prototyping
- Incremental development process model
- Software process Activities

DFD Example level 0 and level 1

- https://www.visual-paradigm.com/tutorials/data-flow-diagram-example-food-ordering-system.jsp

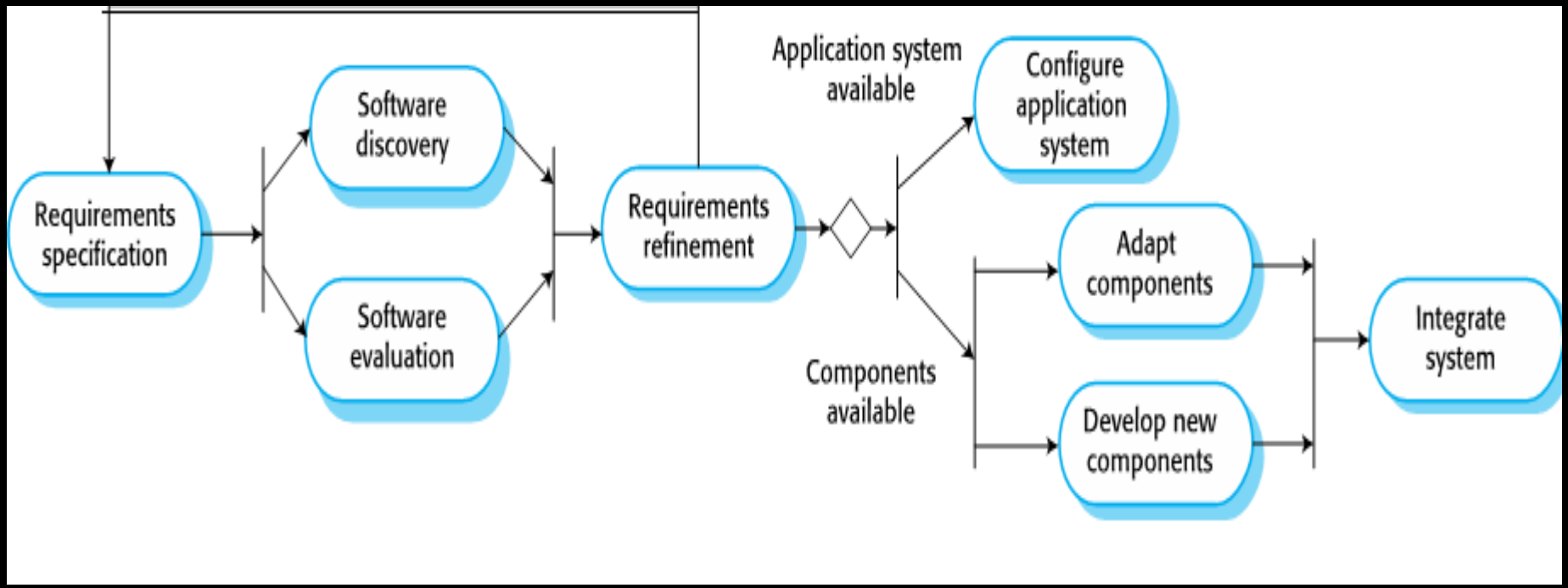
Integration and Configuration

- 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
  - Reuse covered in more depth in Chapter 15.

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



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

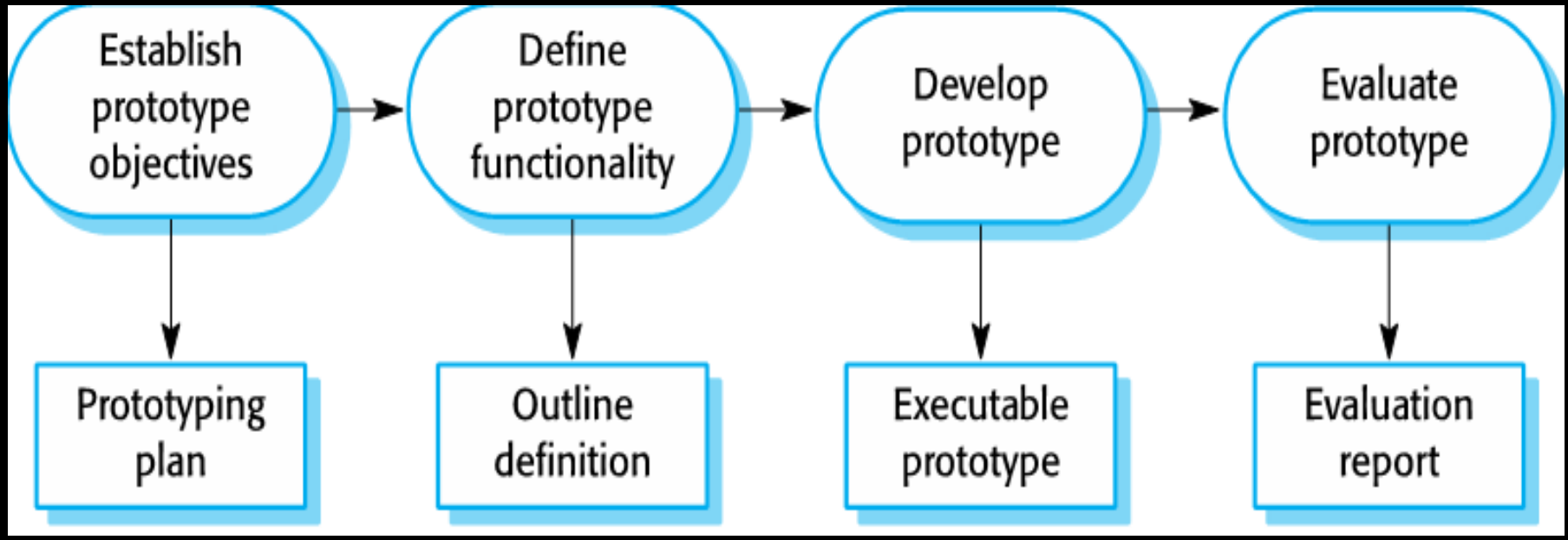
Software Prototyping

- 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.

Benefits of Prototyping

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

The Process of Prototype Development



Prototype Development

- 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

Throw-away Prototypes

- 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 organisational quality standards.

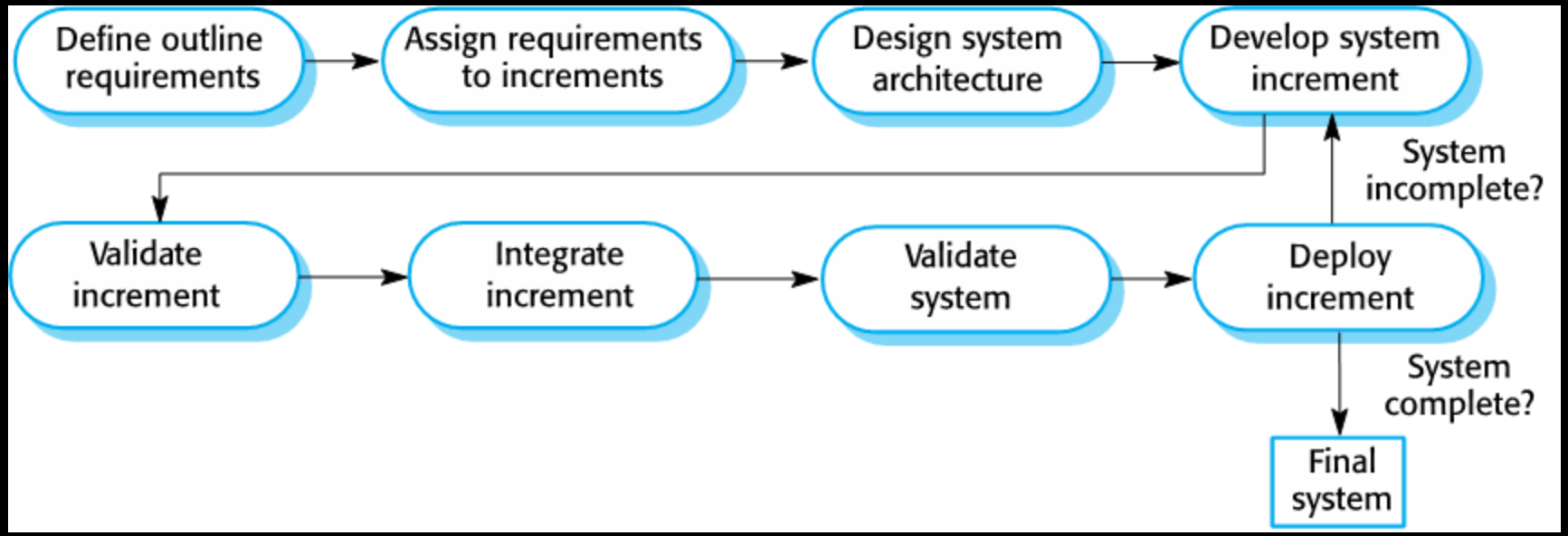
Incremental Delivery

- 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 and Delivery

- Incremental development
  - Develop the system in increments and evaluate each increment before proceeding to the development of the next increment;
  - Normal approach used in agile methods;
  - Evaluation done by user/customer proxy.
- Incremental delivery
  - Deploy an increment for use by end-users;
  - More realistic evaluation about practical use of software;
  - Difficult to implement for replacement systems as increments have less functionality than the system being replaced.

Incremental Delivery



Incremental Delivery 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.

Process Activities

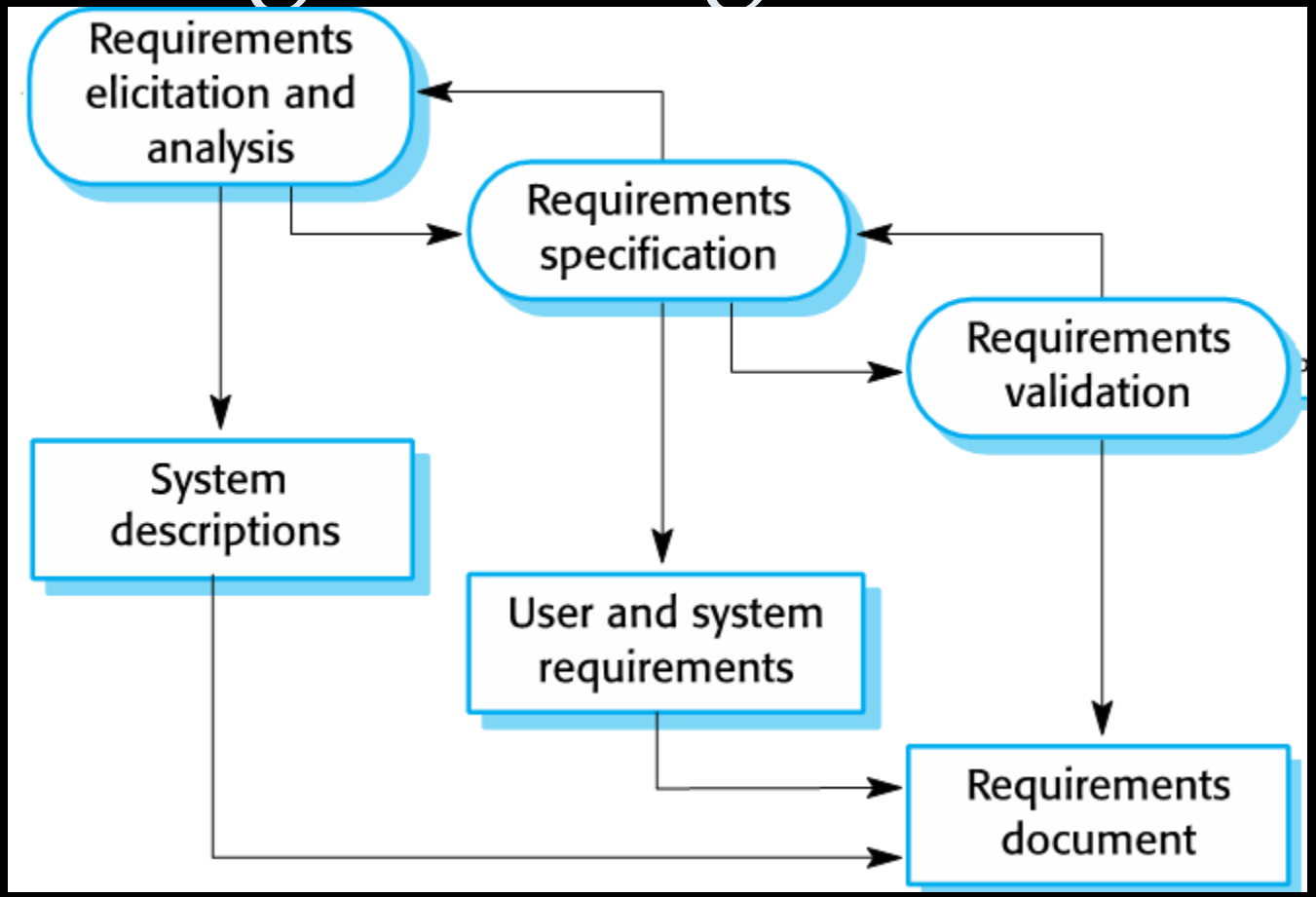
Process Activities

- Real software processes are inter-leaved 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.
- For example, in the waterfall model, they are organized in sequence, whereas in iterative development they are interleaved.

Software Specification

- The process of establishing what services are required and the constraints on the system's operation and development.
- Requirements engineering process
  - Requirements elicitation and analysis
    - What do the system stakeholders require or expect from the system?
  - Requirements specification
    - Defining the requirements in detail
  - Requirements validation
    - Checking the validity of the requirements

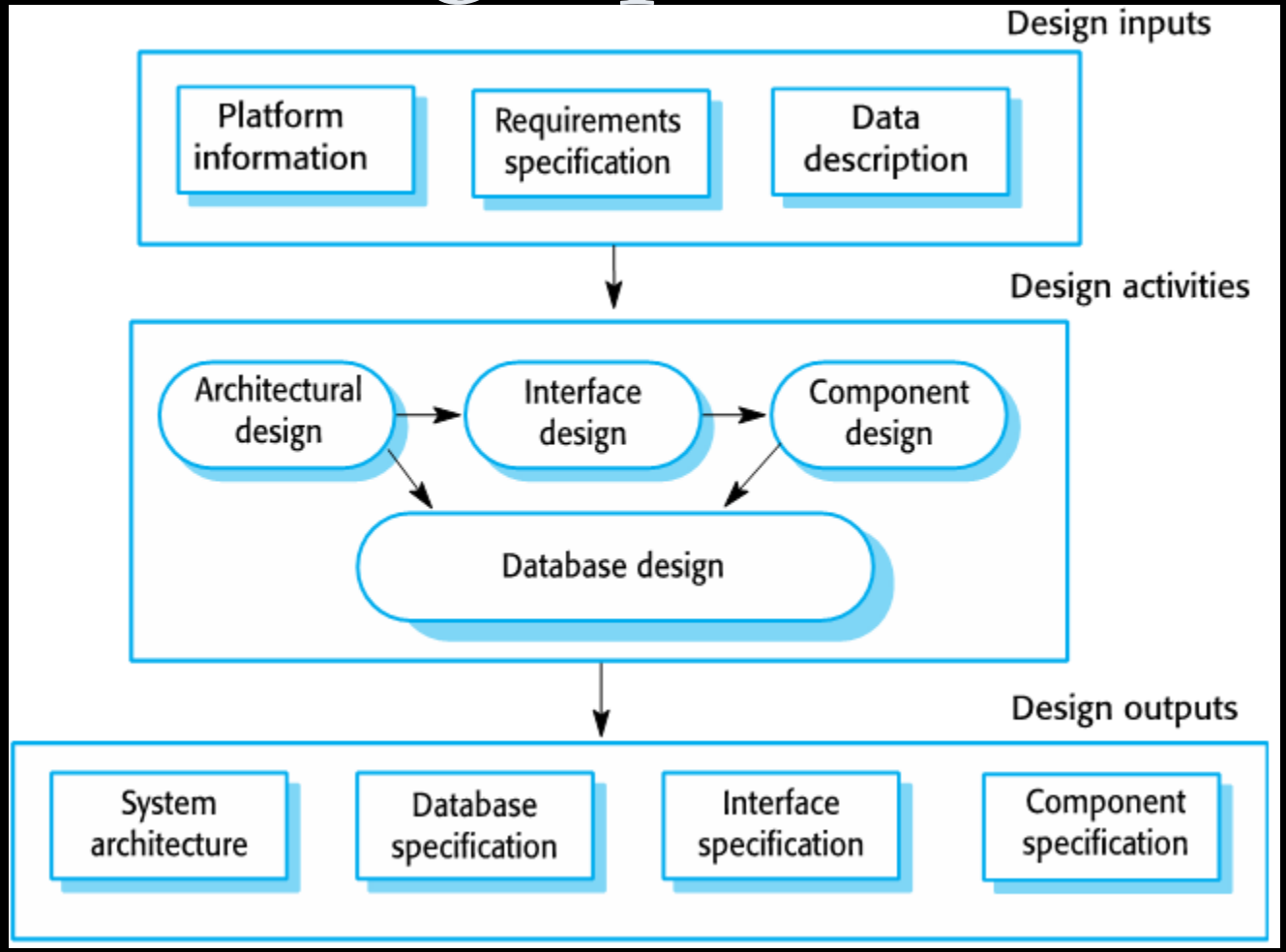
The Requirements Engineering Process



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.

A General Model of the Design process



Design Activities

- *Architectural design,* where you identify the overall structure of the system, the principal components (subsystems or modules), their relationships and how they are distributed.
- *Database design,* where you design the system data structures and how these are to be represented in a database.
- *Interface design,* where you define the interfaces between system components.
- *Component selection and design,* where you search for reusable components. If unavailable, you design how it will operate.

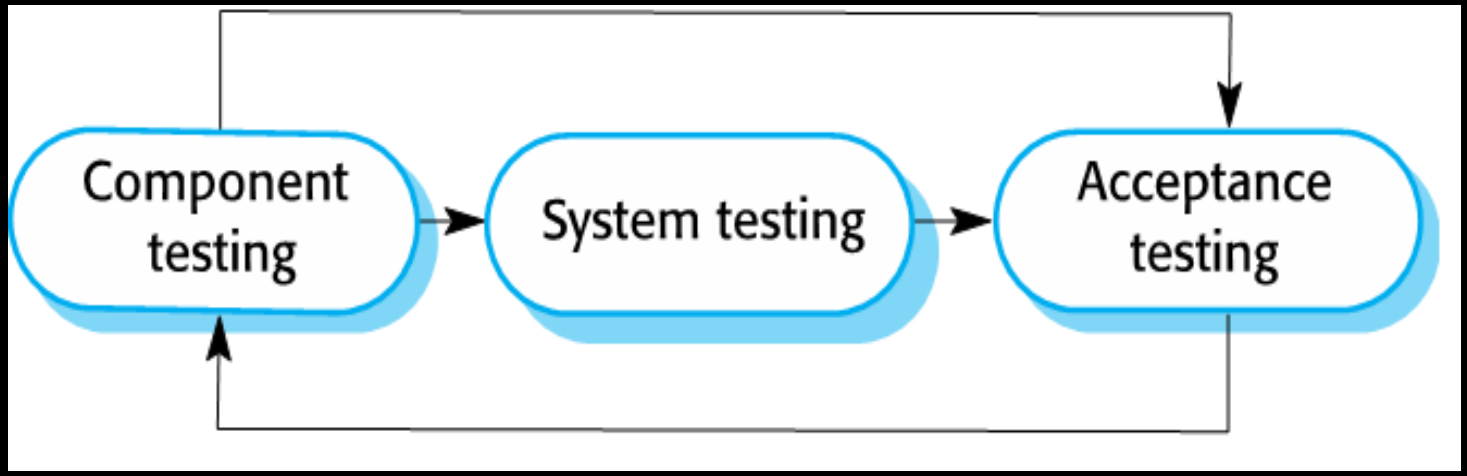
System Implementation

- The software is implemented either by developing a program or programs or by configuring an application system.
- Design and implementation are interleaved activities for most types of software system.
- Programming is an individual activity with no standard process.
- Debugging is the activity of finding program faults and correcting these faults.

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.
- Testing is the most commonly used V & V activity.

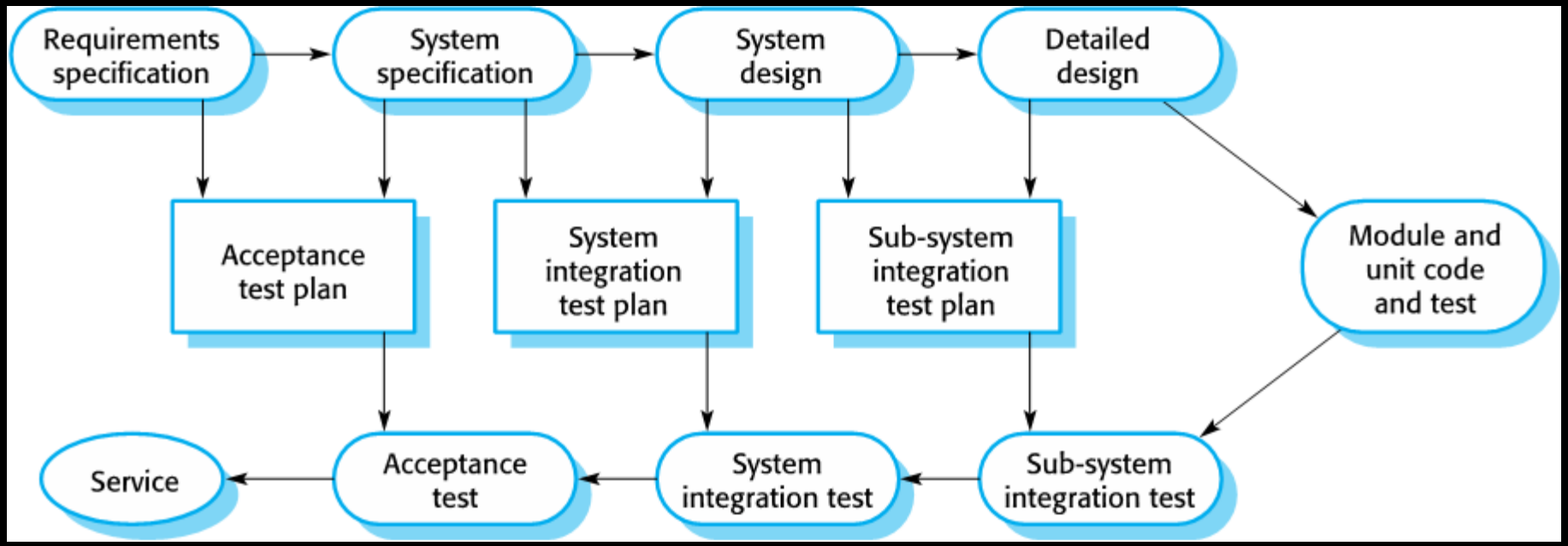
Stages of Testing



Testing Stages

- Component 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.
- Customer testing
  - Testing with customer data to check that the system meets the customer's needs.

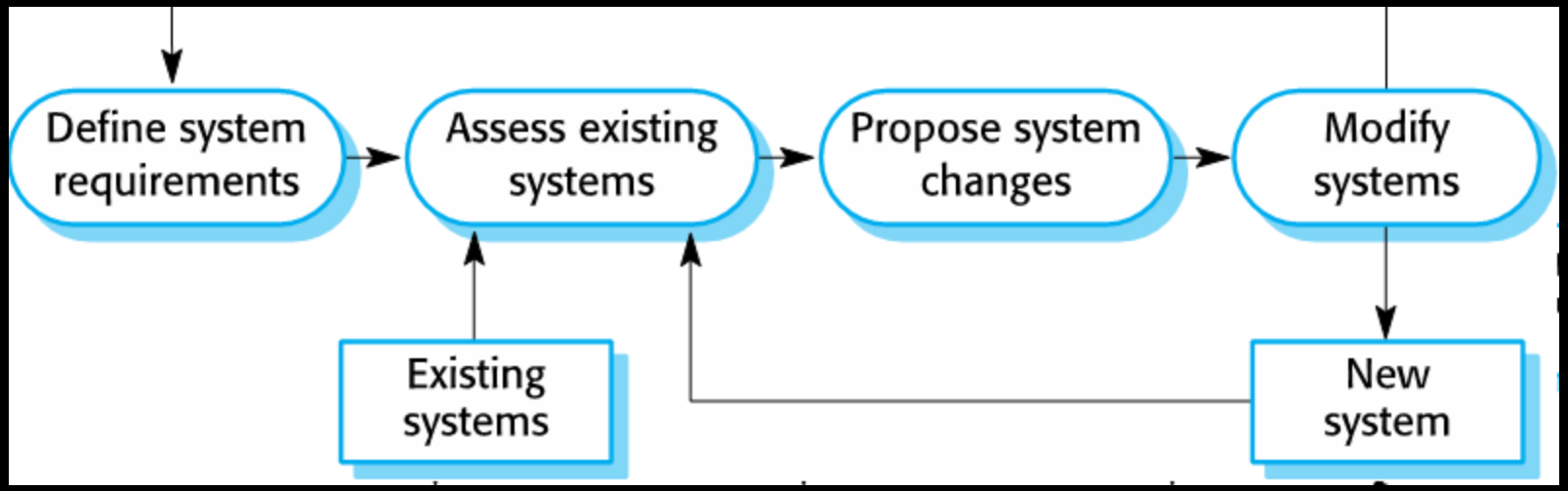
Testing phases in a plan-driven software process (V-model)



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



That is all

Software Engineering
CS-303

Agile Software Development

Lecture # 12, 13
13, Feb

Rubab Jaffar
rubab.jaffar@nu.edu.pk

Topics covered

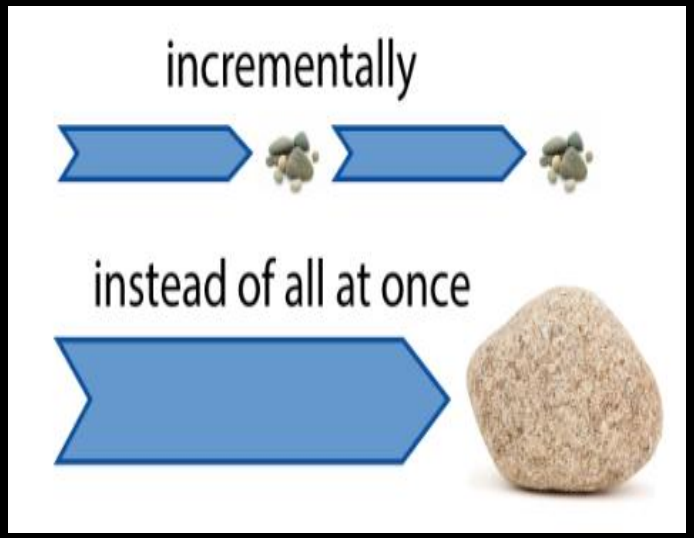
- Agile methods
- Agile development techniques
- Quiz

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

About Agile

- Agile is a Software Development Methodology
- It means '<u>ability to move quickly and easily' and responding swiftly to change</u>
- It's a time boxed, iterative approach
- Develop and deliver software incrementally from the start of the project, instead of trying to deliver it all at once near the end.



Example - Incremental

In the diagram above when we work **incrementally** we are adding piece by piece but expect that each piece is fully finished. Thus keep on adding the pieces until it's complete. As in the image above a person has thought of the application. Then he started building it and in the first iteration the first module of the application or product is totally ready and can be demoed to the customers. Likewise in the second iteration the other module is ready and integrated with the first module. Similarly, in the third iteration the whole product is ready and integrated. Hence, the product got ready step by step.

Example-- Iterative

In the diagram above when we work **iteratively** we create rough product or product piece in one iteration, then review it and improve it in next iteration and so on until it's finished. As shown in the image above, in the first iteration the whole painting is sketched roughly, then in the second iteration colors are filled and in the third iteration finishing is done. Hence, in iterative model the whole product is developed step by step.

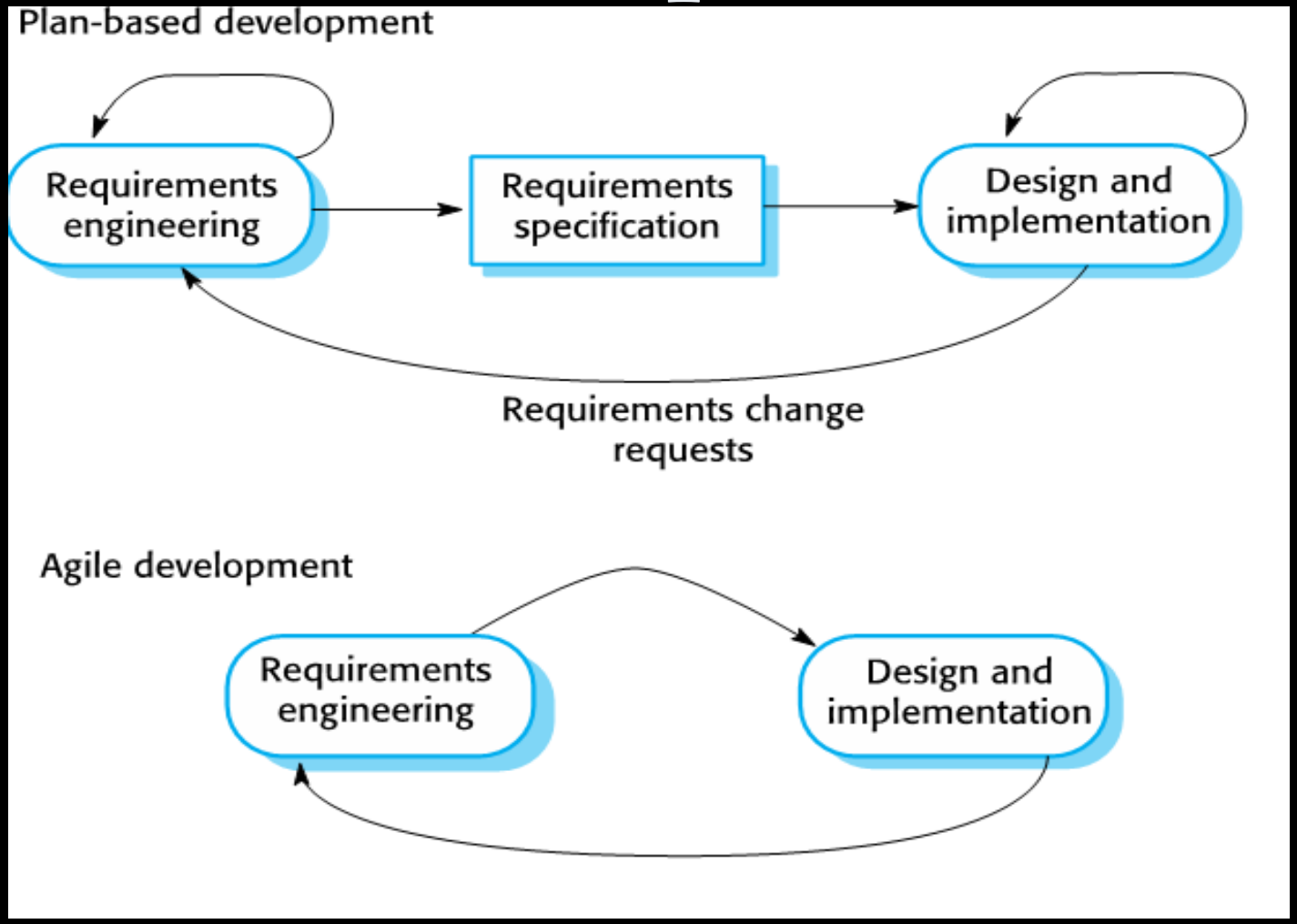
Agile Development Characteristics

- 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-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.

Plan-driven and Agile Development



Agile Methods

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

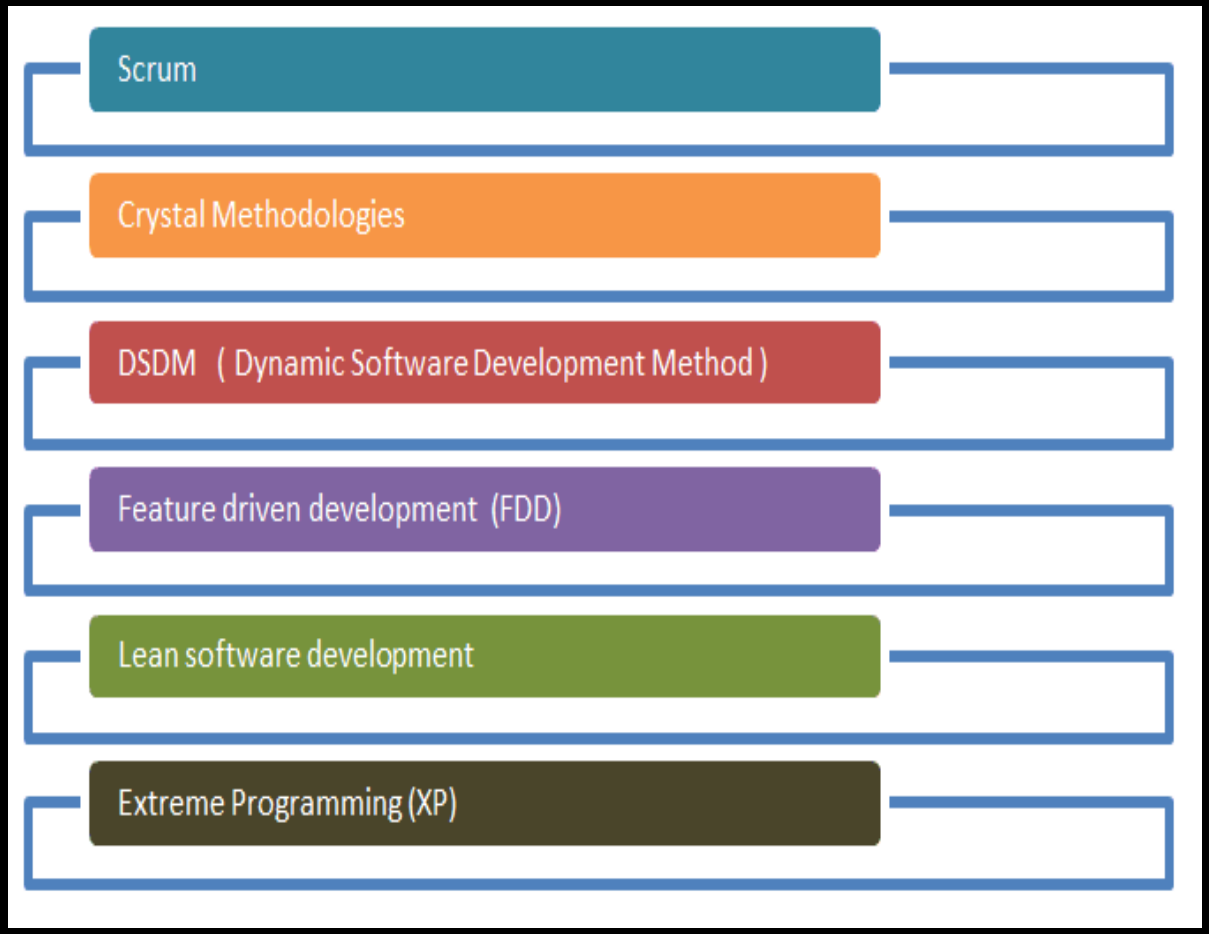
  Individuals and interactions over processes and tools
  Working software over comprehensive documentation
  Customer collaboration over contract negotiation
  Responding to change over following a plan

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

Agile Core Values

- **Individuals and interactions** over processes and tools---in agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
- **Working software** over comprehensive documentation--- working software will be more useful and welcome than just presenting documents to clients in meetings.
- **Customer collaboration** over contract negotiation--- requirements cannot be fully collected at the beginning of the software development cycle, therefore continuous customer or stakeholder involvement is very important.
- **Responding to change** over following a plan--- agile development is focused on quick responses to change and continuous development.

Agile Methods



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.

Agile Development Techniques

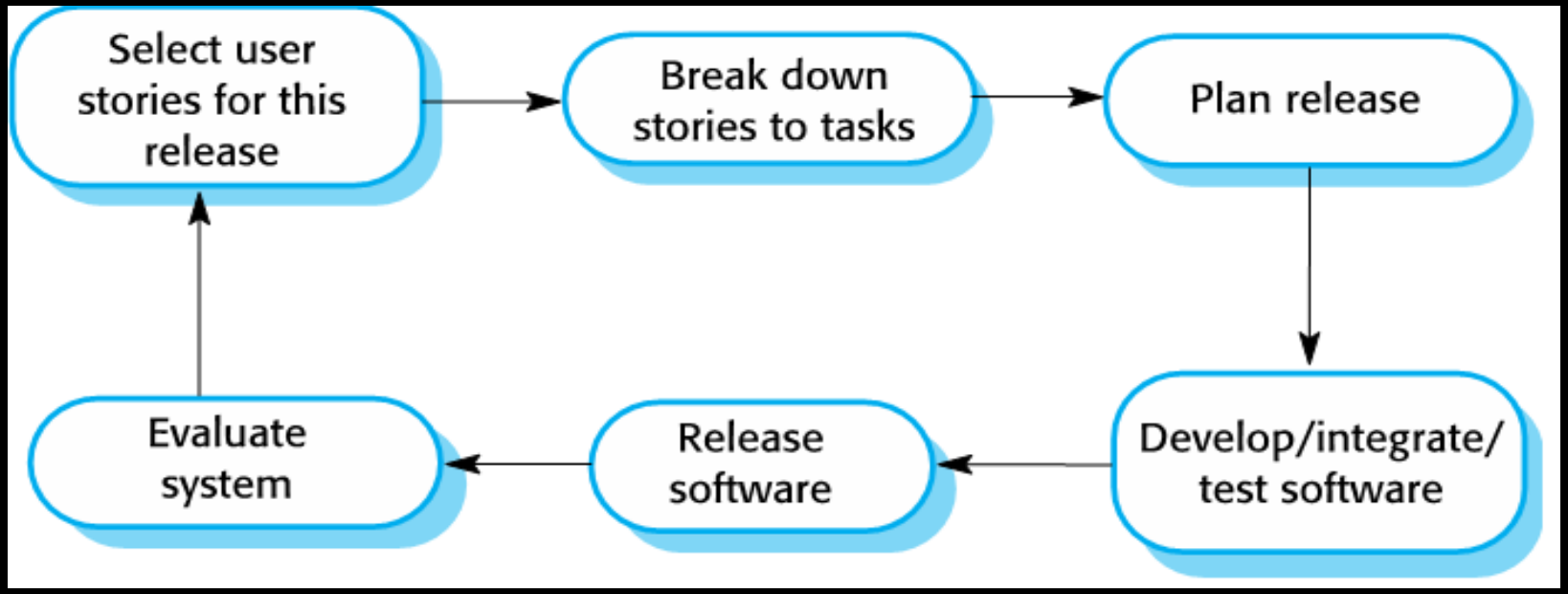
Extreme Programming(XP)

- The origin of extreme programming (XP) started in 1990s when Kent Black tried to find a better way of doing software development when he was handling a project. One significant difference in its approach is that it focuses on adaptability rather than on predictability. The reason behind this approach is that software development is a very fluid process where requirements cannot be fully predicted from the beginning but will always change as projects move on. Hence software development needs a methodology that is capable to adapt to changing requirements at any point during the project life.

Extreme Programming

- A very influential agile method, developed in the late 1990s, that introduced a range of agile development techniques.
- Extreme Programming (XP) takes an 'extreme' approach to iterative development.
  - New versions may be built several times per day;
  - Increments are delivered to customers every 2 weeks;
  - All tests must be run for every build and the build is only accepted if tests run successfully.

The Extreme Programming Release Cycle



Extreme Programming Practices (a)

Principle or practice	Description
Incremental planning	Requirements are recorded on story cards and the stories to be included in a release are determined by the time available and their relative priority. The developers break these stories into development 'Tasks'. See Figures 3.5 and 3.6.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

Extreme Programming Practices (b)

Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

XP and Agile Principles

- Incremental development is supported through small, frequent system releases.
- Customer involvement means full-time customer engagement with the team.
- People not process through pair programming, collective ownership and a process that avoids long working hours.
- Change supported through regular system releases.
- Maintaining simplicity through constant refactoring of code.

Influential XP Practices

- Extreme programming has a technical focus and is not easy to integrate with management practice in most organizations.
- Consequently, while agile development uses practices from XP, the method as originally defined is not widely used.
- Key practices
  - User stories for specification
  - Refactoring
  - Test-first development
  - Pair programming

That is all

Software Engineering
CS-303

Agile Software Development

Lecture # 14, 15
18, 19 Feb

Rubab Jaffar
rubab.jaffar@nu.edu.pk

Topics Covered

- Agile methods
- Agile development techniques
- Agile project management

Influential XP Practices

- Extreme programming has a technical focus and is not easy to integrate with management practice in most organizations.
- Consequently, while agile development uses practices from XP, the method as originally defined is not widely used.
- Key practices
  - User stories for specification
  - Refactoring
  - Test-first development
  - Pair programming

User Stories for Requirements

- Sitting down with the customers
- make a list of features they would like to see in their software.
- We call these things **user stories** and they become the **To Do list** for project. These tasks are the basis of schedule and cost estimates
- Like any other lists, there always seems to be more to do than time allows.
- Ask the customer to prioritize their list
- Get the most important stuff done first, and save the least important for last.

A ‘prescribing medication’ Story

Prescribing medication

The record of the patient must be open for input. Click on the medication field and select either ‘current medication’, ‘new medication’ or ‘formulary’.

If you select ‘current medication’, you will be asked to check the dose; If you wish to change the dose, enter the new dose then confirm the prescription.

If you choose, ‘new medication’, the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

If you choose ‘formulary’, you will be presented with a search box for the approved formulary. Search for the drug required then select it. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

In all cases, the system will check that the dose is within the approved range and will ask you to change it if it is outside the range of recommended doses.

After you have confirmed the prescription, it will be displayed for checking. Either click ‘OK’ or ‘Change’. If you click ‘OK’, your prescription will be recorded on the audit database. If you click ‘Change’, you reenter the ‘Prescribing medication’ process.

Examples of Task Cards for Prescribing Medication

Task 1: Change dose of prescribed drug

Task 2: Formulary selection

Task 3: Dose checking

Dose checking is a safety precaution to check that the doctor has not prescribed a dangerously small or large dose.
Using the formulary id for the generic drug name, lookup the formulary and retrieve the recommended maximum and minimum dose.
Check the prescribed dose against the minimum and maximum. If outside the range, issue an error message saying that the dose is too high or too low.
If within the range, enable the 'Confirm' button.

Refactoring

- Conventional wisdom in software engineering is to design for change. It is worth spending time and effort anticipating changes as this reduces costs later in the life cycle.
- XP, however, maintains that this is not worthwhile as changes cannot be reliably anticipated.
- Rather, it proposes constant code improvement (refactoring) to make changes easier when they have to be implemented.

Refactoring

- Programming team look for possible software improvements and make these improvements even where there is no immediate need for them.
- This improves the understandability of the software and so reduces the need for documentation.
- Changes are easier to make because the code is well-structured and clear.
- However, some changes requires architecture refactoring and this is much more expensive.

Examples of Refactoring

- Re-organization of a class hierarchy to remove duplicate code.
- Tidying up and renaming attributes and methods to make them easier to understand.
- The replacement of inline code with calls to methods that have been included in a program library.

Test-first Development

- Testing is central to XP and XP has developed an approach where the program is tested after every change has been made.
- XP testing features:
  - Test-first development.
  - Incremental test development from scenarios.
  - User involvement in test development and validation.
  - Automated test harnesses are used to run all component tests each time that a new release is built.

Test-driven Development

- Writing tests before code clarifies the requirements to be implemented.
- Tests are written as programs rather than data so that they can be executed automatically. The test includes a check that it has executed correctly.
  - Usually relies on a testing framework such as Junit.
- All previous and new tests are run automatically when new functionality is added, thus checking that the new functionality has not introduced errors.

Customer Involvement

- The role of the customer in the testing process is to help develop acceptance tests for the stories that are to be implemented in the next release of the system.
- The customer who is part of the team writes tests as development proceeds. All new code is therefore validated to ensure that it is what the customer needs.
- However, people adopting the customer role have limited time available and so cannot work full-time with the development team. They may feel that providing the requirements was enough of a contribution and so may be reluctant to get involved in the testing process.

Test-case Description for Dose Checking

Test 4: Dose checking

Input:
1. A number in mg representing a single dose of the drug.
2. A number representing the number of single doses per day.

Tests:
1. Test for inputs where the single dose is correct but the frequency is too high.
2. Test for inputs where the single dose is too high and too low.
3. Test for inputs where the single dose * frequency is too high and too low.
4. Test for inputs where single dose * frequency is in the permitted range.

Output:
OK or error message indicating that the dose is outside the safe range.

Test Automation

- Test automation means that tests are written as executable components before the task is implemented
  - These testing components should be stand-alone, should simulate the submission of input to be tested and should check that the result meets the output specification. An automated test framework (e.g. Junit) is a system that makes it easy to write executable tests and submit a set of tests for execution.
- As testing is automated, there is always a set of tests that can be quickly and easily executed
  - Whenever any functionality is added to the system, the tests can be run and problems that the new code has introduced can be caught immediately.

Pair Programming

- In pair programming, programmers sit together at the same computer to develop the software.
- Pairs are created dynamically so that all team members work with each other during the development process.
- Pair programming is not necessarily inefficient and there is some evidence that suggests that a pair working together is more efficient than 2 programmers working separately.

Pair Programming Advantages

- Pair programming involves programmers working in pairs, developing code together.
- The sharing of knowledge that happens during pair programming is very important as it reduces the overall risks to a project when team members leave.
- This helps develop common ownership of code and spreads knowledge across the team.
- It serves as an informal review process as each line of code is looked at by more than 1 person.
- It encourages refactoring as the whole team can benefit from improving the system code.

Agile Project Management

Agile Project Management

- The principal responsibility of software project managers is to manage the project so that the software is delivered on time and within the planned budget for the project.
- The standard approach to project management is plan-driven. Managers draw up a plan for the project showing what should be delivered, when it should be delivered and who will work on the development of the project deliverables.
- Agile project management requires a different approach, which is adapted to incremental development and the practices used in agile methods.

Scrum

- Scrum is an agile method that focuses on managing iterative development rather than specific agile practices.
- There are three phases in Scrum.
  - The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.
  - This is followed by a series of sprint cycles, where each cycle develops an increment of the system.
  - The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

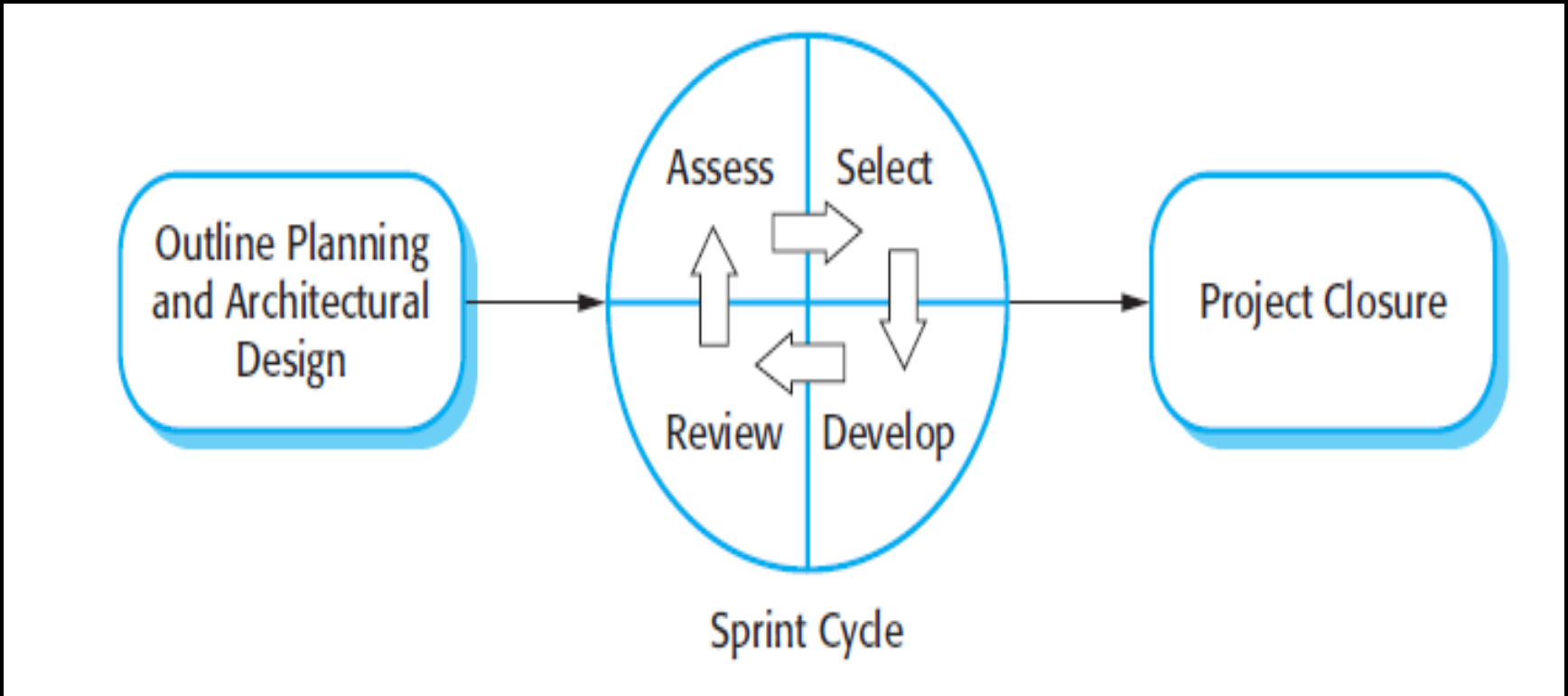
Scrum Terminology (a)

Scrum term	Definition
Development team	A self-organizing group of software developers, which should be no more than 7 people. They are responsible for developing the software and other essential project documents.
Potentially shippable product increment	The software increment that is delivered from a sprint. The idea is that this should be 'potentially shippable' which means that it is in a finished state and no further work, such as testing, is needed to incorporate it into the final product. In practice, this is not always achievable.
Product backlog	This is a list of 'to do' items which the Scrum team must tackle. They may be feature definitions for the software, software requirements, user stories or descriptions of supplementary tasks that are needed, such as architecture definition or user documentation.
Product owner	An individual (or possibly a small group) whose job is to identify product features or requirements, prioritize these for development and continuously review the product backlog to ensure that the project continues to meet critical business needs. The Product Owner can be a customer but might also be a product manager in a software company or other stakeholder representative.

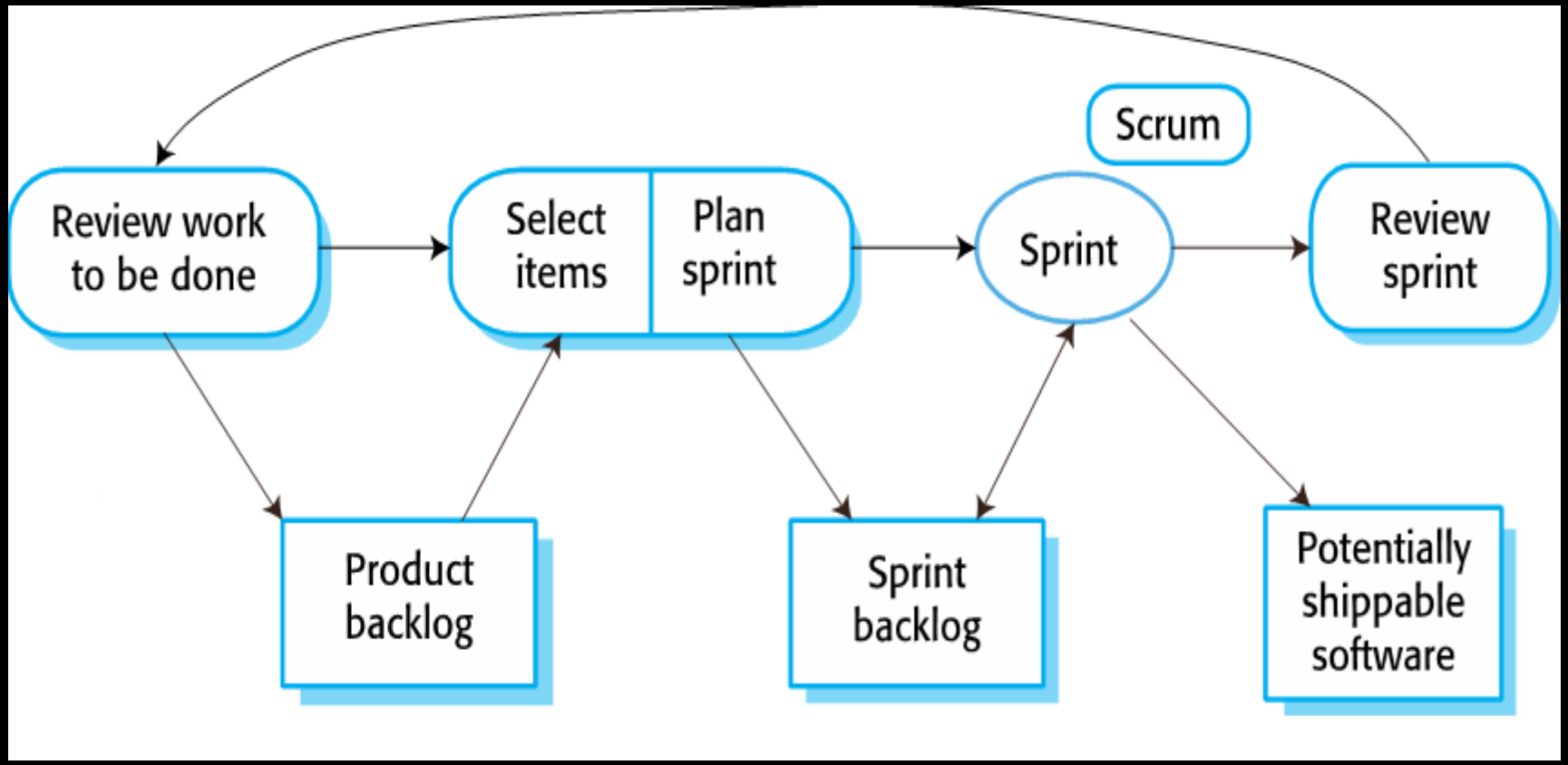
Scrum Terminology (b)

Scrum term	Definition
Scrum	A daily meeting of the Scrum team that reviews progress and prioritizes work to be done that day. Ideally, this should be a short face-to-face meeting that includes the whole team.
ScrumMaster	The ScrumMaster is responsible for ensuring that the Scrum process is followed and guides the team in the effective use of Scrum. He or she is responsible for interfacing with the rest of the company and for ensuring that the Scrum team is not diverted by outside interference. The Scrum developers are adamant that the ScrumMaster should not be thought of as a project manager. Others, however, may not always find it easy to see the difference.
Sprint	A development iteration. Sprints are usually 2-4 weeks long.
Velocity	An estimate of how much product backlog effort that a team can cover in a single sprint. Understanding a team's velocity helps them estimate what can be covered in a sprint and provides a basis for measuring improving performance.

The Scrum Process

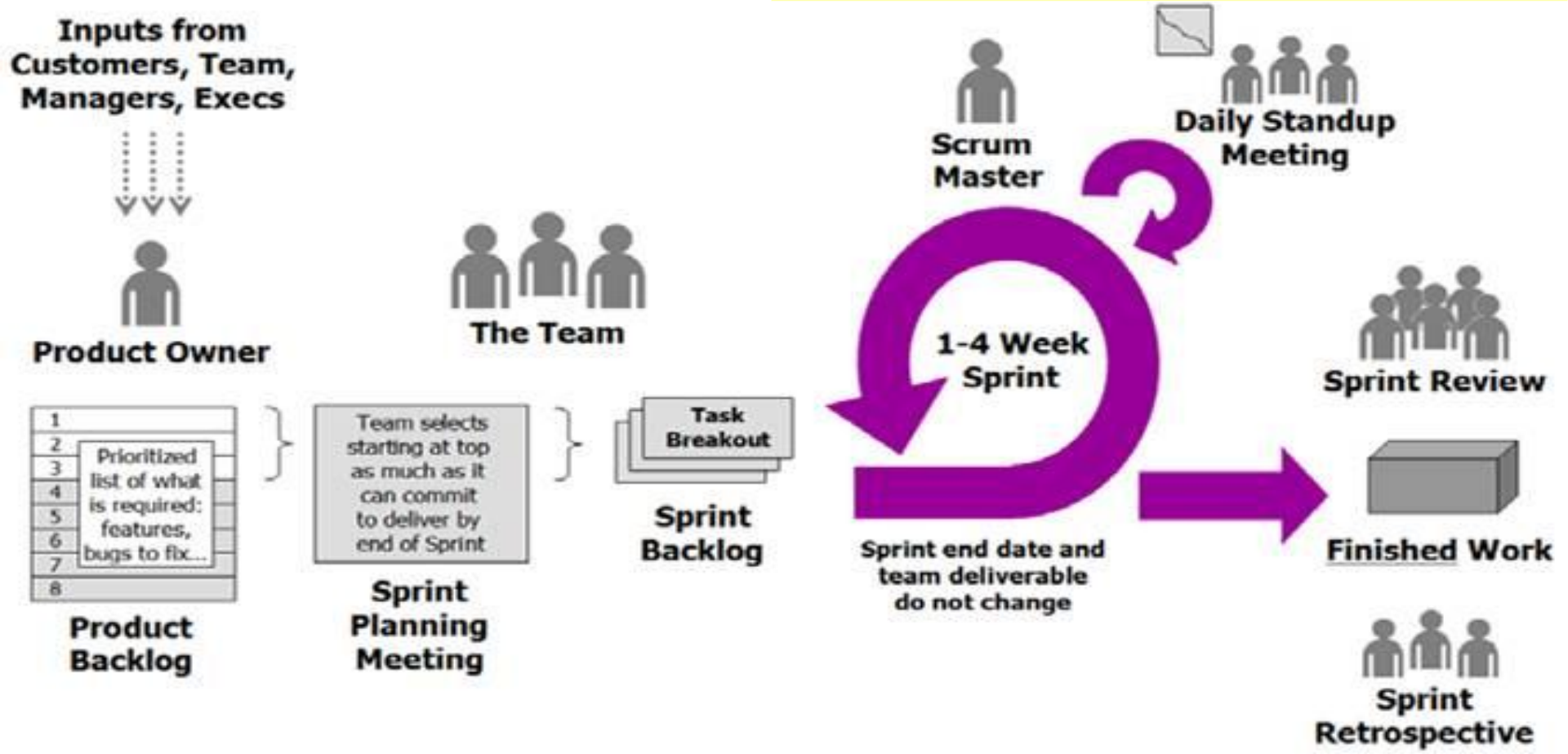


Scrum Sprint Cycle



The Scrum Sprint Cycle

1-What have you accomplished since yesterday?
2-Are your Sprint Backlog estimates accurate?
3-What are you working on today?
4-Is there anything blocking you?



The Scrum Sprint Cycle

- Sprints are fixed length, normally 2–4 weeks.
- The starting point for planning is the product backlog, which is the list of work to be done on the project.
- The selection phase involves all of the project team who work with the customer to select the features and functionality from the product backlog to be developed during the sprint.

The Sprint Cycle

- Once these are agreed, the team organize themselves to develop the software.
- During this stage the team is isolated from the customer and the organization, with all communications channelled through the so-called 'Scrum master'.
- The role of the Scrum master is to protect the development team from external distractions.
- At the end of the sprint, the work done is reviewed and presented to stakeholders. The next sprint cycle then begins.

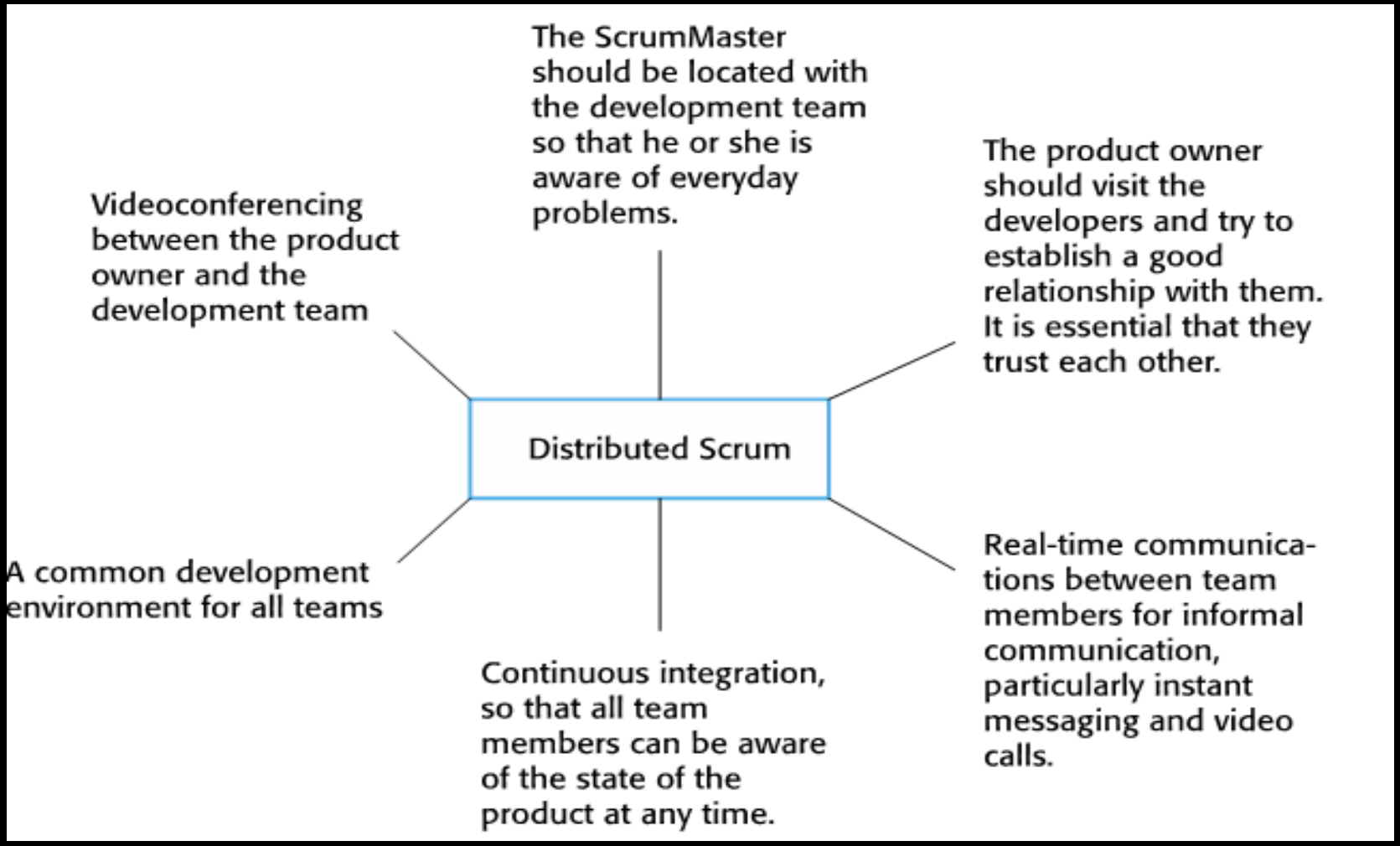
Teamwork in Scrum

- The 'Scrum master' is a facilitator who arranges daily meetings, tracks the backlog of work to be done, records decisions, measures progress against the backlog and communicates with customers and management outside of the team.
- The whole team attends short daily meetings (Scrums) where all team members share information, describe their progress since the last meeting, problems that have arisen and what is planned for the following day.
  - This means that everyone on the team knows what is going on and, if problems arise, can re-plan short-term work to cope with them.

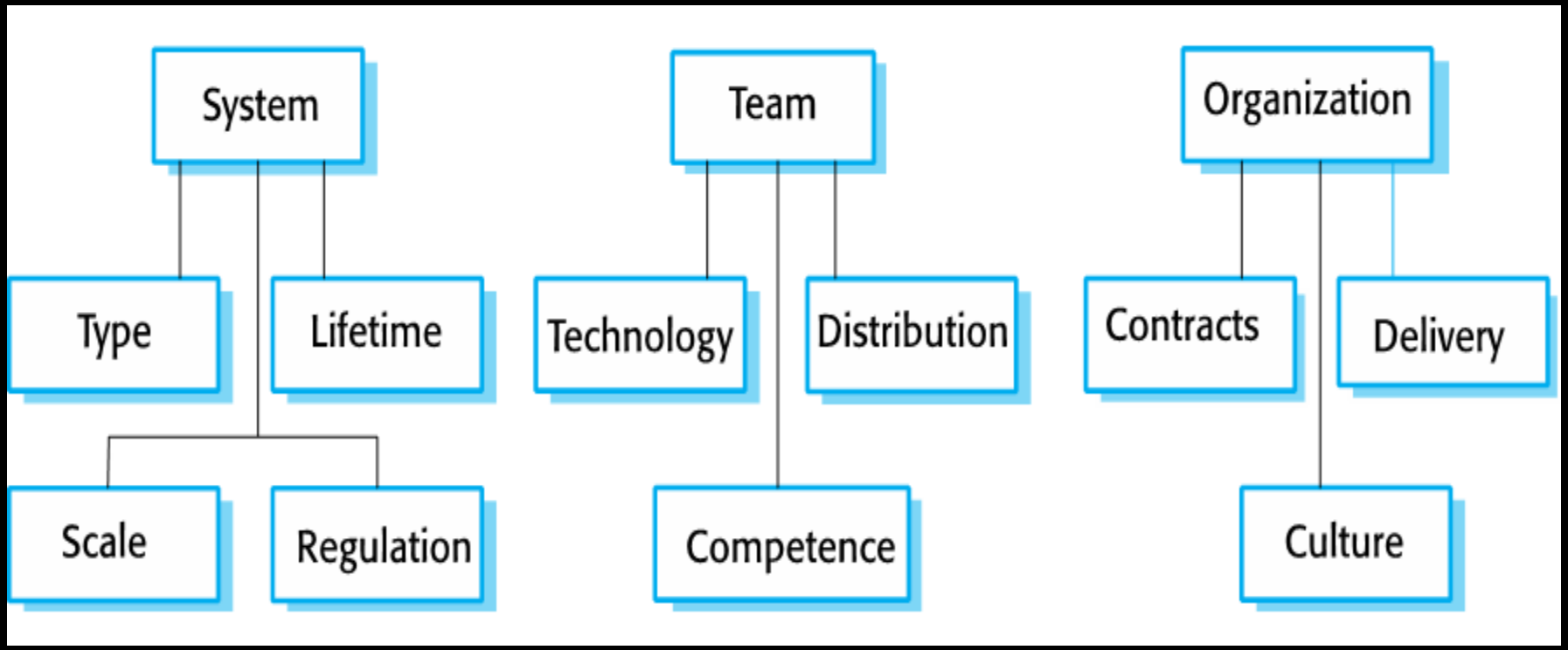
Scrum Benefits

- The product is broken down into a set of manageable and understandable chunks.
- Unstable requirements do not hold up progress.
- The whole team have visibility of everything and consequently team communication is improved.
- Customers see on-time delivery of increments and gain feedback on how the product works.
- Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.

Distributed Scrum



Agile and Plan-based Factors



That is all

Intro to Software Engineering

SE-110

Project Management

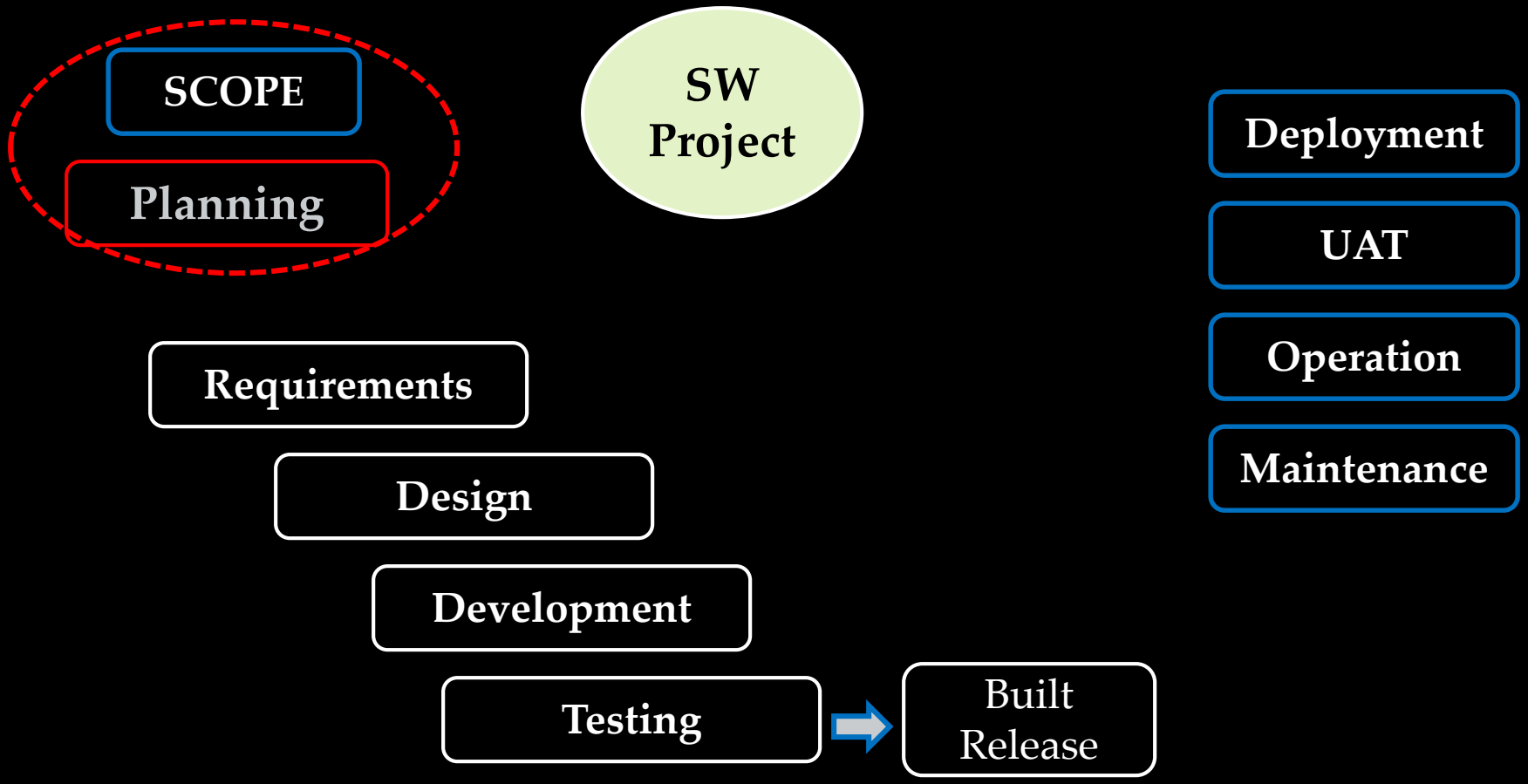
Lecture # 15, 16
19, 20 Feb

Rubab Jaffar
rubab.jaffar@nu.edu.pk

Today's Outline

- What is project?
- What is Management?
- Project Management
- Software Project Management(SPM)
- Project Management activities
  - Project planning
  - Project scheduling
  - Risk management

SDLC


SCOPE
Planning
SW Project
Deployment
UAT
Operation
Maintenance
Requirements
Design
Development
Testing
Built Release

What is not Project?

- It seems that everything we do is a project.
- A project is different from day-to-day business.
- Ongoing work effort is generally a repetitive process because it follows an organization's existing procedures.
- Any routine work is NOT a project.
  - Coming to university on regular basis in not a project.
  - Monthly/weekly grocery shopping is not a project.
  - Production of cars at automobile factory is not a project.
  - Conducting classes at University is not a project.

Project Concepts

It is a project when it meets the following criteria;

1. Any non-repetitive activity.

2. A temporary endeavor undertaken to create a unique product or service so require resources.

3. Any activity with a start and a finish. (Temporary)

4. A unique set of co-ordinated activities, undertaken by an individual or team to meet specific objectives within defined schedule, cost and performance parameters. *Project should have a primary customer/sponsor.*

What is Management?

- It is a set of activities undertaken by one or more persons for the purpose of planning & controlling the activities of others in order to achieve an objective or complete the project.

What is Project Management?

- Project management is "the application of knowledge, skills, tools and techniques to project activities to meet the project requirements".

- PM is the one who is responsible for establishing a communication in between the project team and the user.

- Project Manager and his/her team should collectively possess the necessary and requisite inter-personal and technical skills to facilitate control over the various activities within the project.

Project Manager Role

- **A Good Project Manager**
  - **Takes ownership of the whole project**
  - **Is proactive not reactive**
  - **Adequately plans the project**
  - **Is Authoritative (NOT Authoritarian)**
  - **Is Decisive**
  - **Is a Good Communicator**
  - **Manages by data and facts not uniformed optimism**
  - **Leads by example**
  - **Has sound Judgement**
  - **Is a Motivator**
  - **Is Diplomatic**

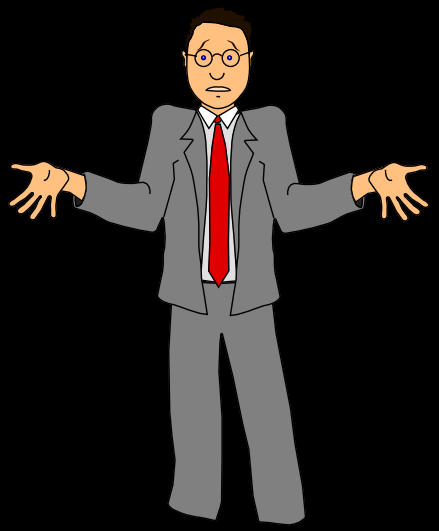
Software Project Issues

- Project management is needed because software development is always subject to budget and schedule constraints that are set by the organisation developing the software.

- Almost 70% of software organization have no defined engineering methods
- Software ends up
  - Late
  - Over budget
  - Fails to meet requirements
  - In a 1998 survey, 26% of software projects failed outright, 46% experienced cost and schedule overruns

- The major reason of theses software projects are not technical problems, but management problems"

Project Management….



Work Smart Not Hard !!!

Software Management Distinctions

- The product is intangible.
- The product is uniquely flexible.
- Software engineering is not recognized as an engineering discipline with the same status as mechanical, electrical engineering, etc. The software development process is not standardised.
- Many software projects are 'one-off' projects.

4 P'S OF PROJECT MANAGEMENT

People

Operation

Product

Process

Project

The Management Spectrum

- **Effective software project management focuses on these items (in this order)**
  - **The people**
    - **Deals with the cultivation of motivated, highly skilled people**
    - **Consists of the stakeholders, the team leaders, and the software team**
  - **The product**
    - **Product objectives and scope should be established before a project can be planned**
  - **The process**
    - **The software process provides the framework from which a comprehensive plan for software development can be established**
  - **The project**
    - **Planning and controlling a software project is done for one primary reason**

Project Staffing

- May not be possible to appoint the ideal people to work on a project
  - Project budget may not allow for the use of highly-paid staff;
  - Staff with the appropriate experience may not be available;
  - An organisation may wish to develop employee skills on a software project.
- Managers have to work within these constraints especially when there are shortages of trained staff.

What is planning?

Planning is the process of stating objectives and then determining the most effective activities or accomplishments necessary to reach the objectives

Who makes the plans?

Everybody must plan

Project manager initiate the planning process

Project manager coordinates planning activities into the overall project master plan

Characteristics of a project planner

Flexible

Creative

Responsive

Communicative

Analytic

Planning Process

Devising and maintaining a workable scheme to accomplish the business need that the project was undertaken to address

Project Planning

- Adequate planning leads to the correct completion of work

Planning

- Inadequate planning leads to frustration towards the end of the project & poor project performance

Project Start

Project End

Project Planning

- Probably the most time-consuming project management activity.
- Continuous activity from initial concept through to system delivery. Plans must be regularly revised as new information becomes available.
- Various different types of plan may be developed to support the main software project plan that is concerned with schedule and budget.

Project Planning Process

Establish the project constraints
Make initial assessments of the project parameters
Define project milestones and deliverables
while project has not been completed or cancelled loop
    Draw up project schedule
    Initiate activities according to schedule
    Wait (for a while)
    Review project progress
    Revise estimates of project parameters
    Update the project schedule
    Re-negotiate project constraints and deliverables
    if (problems arise) then
        Initiate technical review and possible revision
    end if
end loop

Types of Project Plan

Plan	Description
Quality plan	Describes the quality procedures and standards that will be used in a project. See Chapter 27.
Validation plan	Describes the approach, resources and schedule used for system validation. See Chapter 22.
Configuration management plan	Describes the configuration management procedures and structures to be used. See Chapter 29.
Maintenance plan	Predicts the maintenance requirements of the system, maintenance costs and effort required. See Chapter 21.
Staff development plan.	Describes how the skills and experience of the project team members will be developed. See Chapter 25.

The Software Development Plan /Project Plan

- This is usually what is meant by a project plan.
- Specifies the order of work to be carried out, resources, responsibilities, and so on.
- The project plan sets out:
  - The work breakdown;
  - The resources available to the project;
  - A schedule for the work.

Project Plan Structure

- Introduction.
- Project organisation.
- Risk analysis.
- Hardware and software resource requirements.
- Work breakdown.
- Project schedule.
- Monitoring and reporting mechanisms.

Work Breakdown Structure (WBS)

- **The Work Breakdown Structure is the foundation for effective project planning, costing and management.**
- **It is the most important aspect in setting-up a Project**
- **A Work Breakdown Structure (WBS) is a hierarchical (from general to specific) tree structure of deliverables and tasks that need to be performed to complete a project."**

- It is the foundation on which everything else builds

5 reasons to create a Work Breakdown Structure

- It helps with correct project organization.
- Assists in describing the project scope to stakeholders.
- Helps to distribute responsibilities.
- Shows the project's milestones and all the points to control.
- Allows estimating costs, risks and time correctly.

WBS

- There are many ways of breaking down the activities in a project, but the most usual is into:
- work packages;
- tasks;
- deliverables;
- milestones.

WBS

- A work package is a large, logically distinct section of work:
  - typically at least 12 months duration;
  - may include multiple concurrent activities;
  - independent of other activities;
  - but may depend on, or feed into other activities;
  - typically allocated to a single team.
- A task is typically a much smaller piece of work: A part of a work package.
- typically 3–6 person months effort;
- may be dependent on other concurrent activities;
- typically allocated to a single person.

WBS

- A deliverable is an output of the project that can meaningfully be assessed.
- Examples:
  - a report (e.g., requirements spec);
  - code (e.g., alpha tested product).
- Deliverables are indicators (but only indicators) of progress.
- A milestone is a point at which progress on the project may be assessed.
- Typically a major turning point in the project.
- EXAMPLES:
  - delivery of requirements spec;
  - delivery of alpha tested code.

WBS

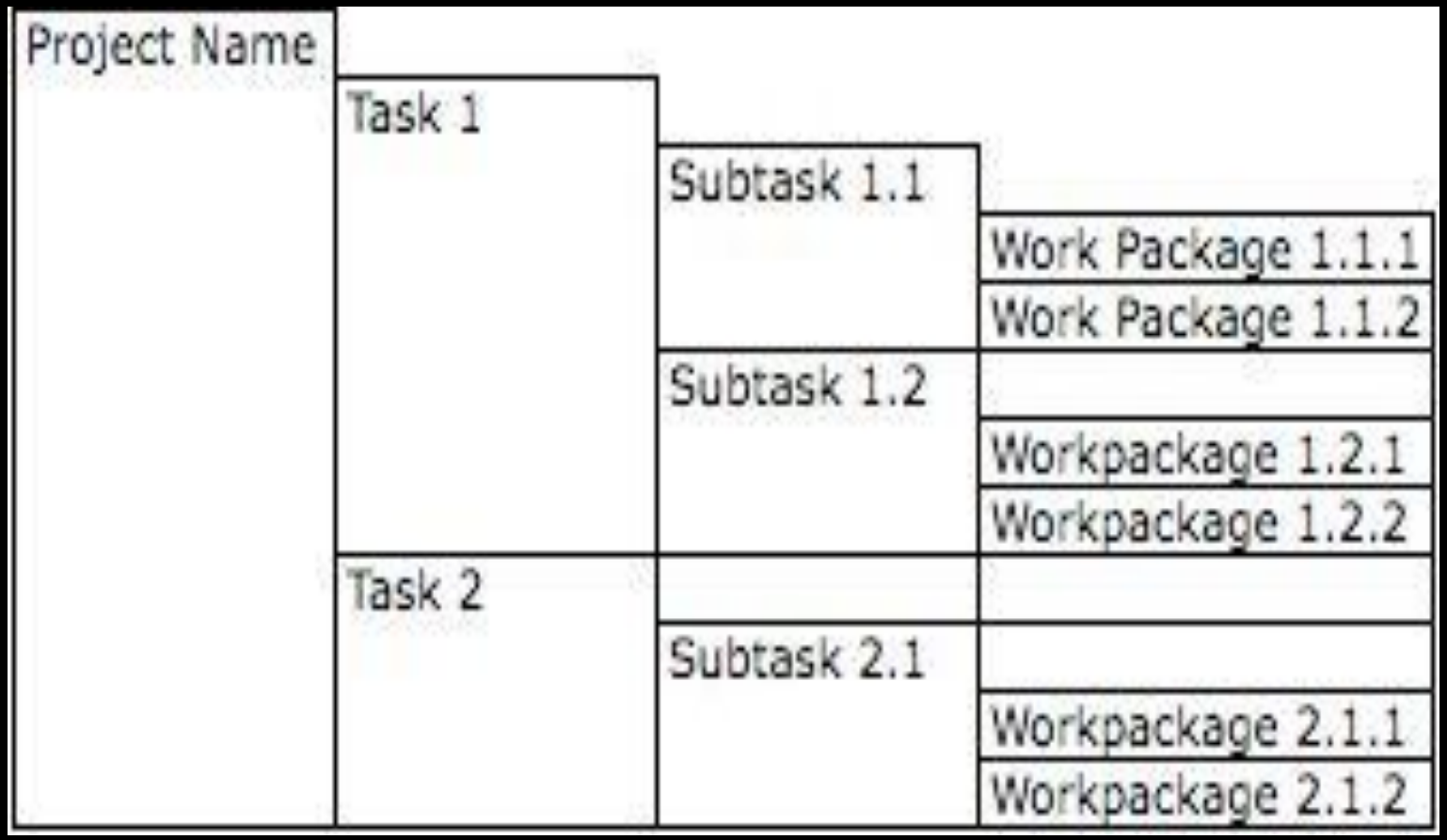
- Usually. . .
- work packages are numbered WP1, WP2, . . . ;
- tasks are numbered T1.1, T1.2, etc,
  - the first number is the number of the workpackage;
  - the second is a sequence number.
- deliverables are numbered D1.1, D1.2, etc
- milestones are numbered M1, M2 etc.

Steps to build a WBS

- Begin with the Charter, focusing on Objectives and Deliverables
- Break the main product(s) down into sub-products
- Set the structure to match how you'll manage the project
- Lowest level not too detailed, not too large
- Is there a need for Integration?
- Identify support activities
- Check for completeness - is all the effort included?
- Develop a coding structure if needed
- Assign work package managers

Displaying the WBS

Example of outlined WBS.



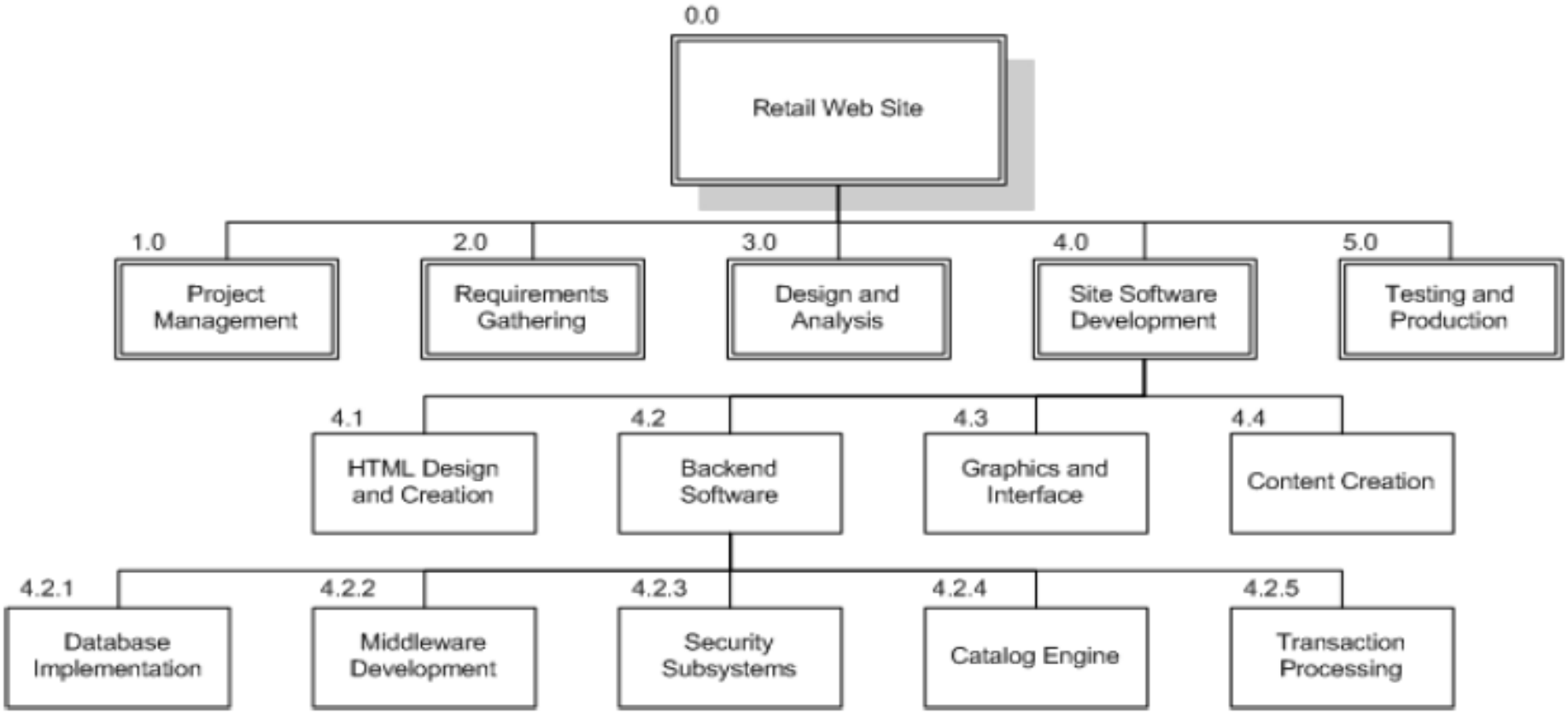
Displaying the WBS

Example of outlined WBS.

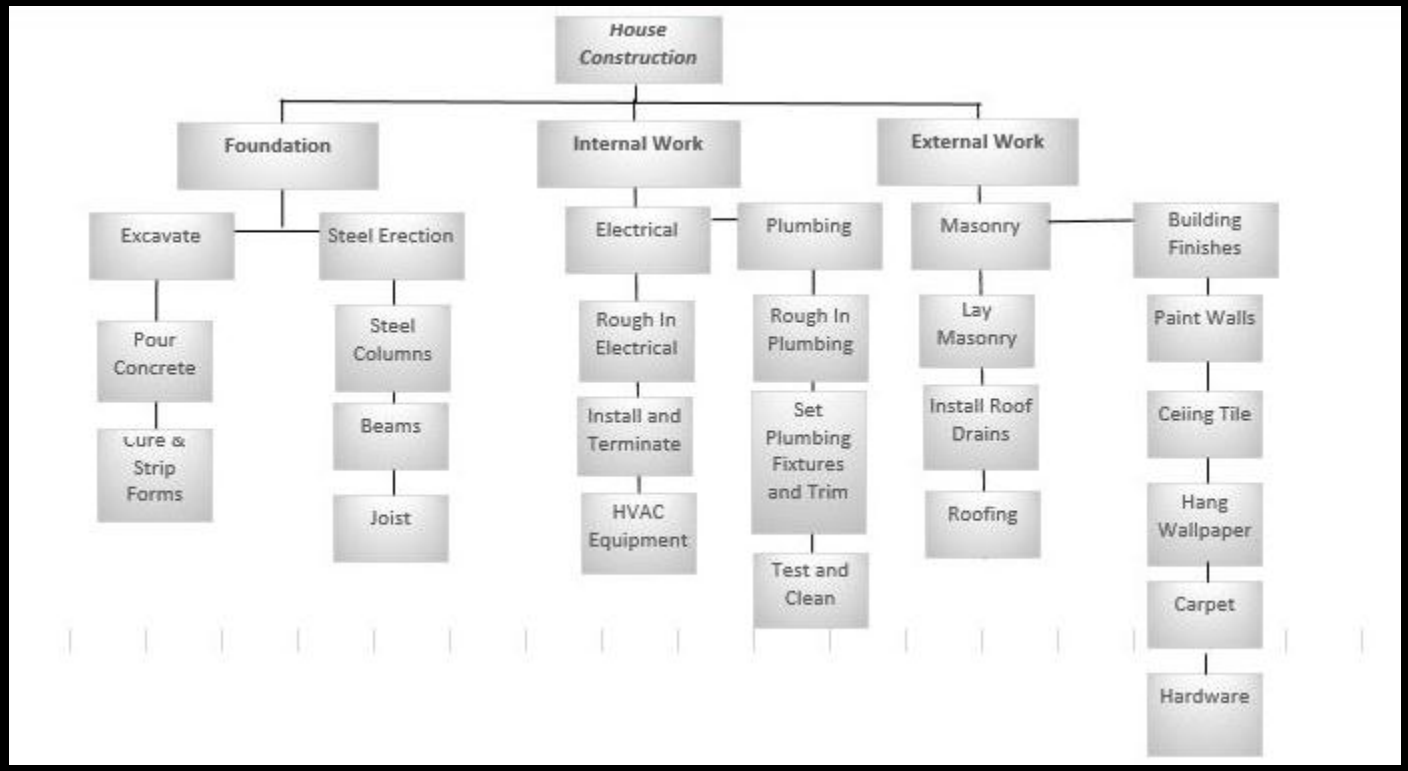
0.0 Retail Web Site
1.0 Project Management
2.0 Requirements Gathering
3.0 Analysis & Design
4.0 Site Software Development
    4.1 HTML Design and Creation
    4.2 Backend Software
        4.2.1 Database Implementation
        4.2.2 Middleware Development
        4.2.3 Security Subsystems
        4.2.4 Catalog Engine
        4.2.5 Transaction Processing
    4.3 Graphics and Interface
    4.4 Content Creation
5.0 Testing and Production

Displaying the WBS

Example of Chart WBS.



WBS– House Construction



Example WBS

- **Redecorate Room**
  - **Prepare materials**
    - **Buy paint**
    - **Buy a ladder**
    - **Buy brushes/rollers**
    - **Buy wallpaper remover**
  - **Prepare room**
    - **Remove old wallpaper**
    - **Remove detachable decorations**
    - **Cover floor with old newspapers**
    - **Cover electrical outlets/switches with tape**
    - **Cover furniture with sheets**
  - **Paint the room**
  - **Clean up the room**
    - **Dispose or store left over paint**
    - **Clean brushes/rollers**
    - **Dispose of old newspapers**
    - **Remove covers**

That is all