**Revision summary**

**Lesson 1: Course Introduction**

* RUP: Rational Unified Process
* Agile Modelling and Agile development techniques including SCRUM and extreme programming are widely used
* Goal is to optimize and chose the best technique for the size and requirements of the project

Fundamental goals of software engineering are

1. A clear vision about a project that can be communicated to others
2. Minimize mistakes
3. Fulfil large scale intentions

Qualities for successful software development include

1. Clear thinking
2. Ability to adapt to change
3. Ability to make fewer mistakes
4. Ability to achieve goals

**Lesson 2: Software Engineering Best practices**

Software development process: A set of activities needed to transform a user requirement into a software system (On time and within budget)

Need a software engineering process to reduce project fail

Iterative && agile have better success rate as compared to ad-hoc and traditional projects which have greater project failures

Success – Delivered with expected functionality, on time and within budget

Challenge – Miss one or more of the above

Failure – No solution delivered

Processes used include

-iterative

-Agile

-Ad-hoc

-Waterfall

Smaller teams have higher success rates than larger teams for all types of software development processes

Importance of analysis and Design

1. Maintainability – ease to maintain (67% of cost used for maintaining a project, the rest for development)
2. Good start, fewer bugs

Best Practices for Software Development

1. Visually model software e.g UML
2. Develop software iteratively
3. Use Component Based architectures e.g Enterprise

UML is a language used for

* Specifying
* Visualizing
* Constructing
* Documenting

UML diagrams include

* Collaboration diagrams
* Sequence diagrams
* Object diagrams
* Class diagrams
* Deployment diagrams
* State diagrams
* Activity diagrams
* Component diagrams

An actor is someone or something outside the system that interacts with the system

A UC is a sequence of actions a system performs that yields an observable result of value to an output

Waterfall: requirements analysis, design, code & unit testing, subsystem testing, system testing

Iterative has less risk than waterfall

Life cycle phases: Inception, Elaboration, Construction, Transition

Inception: Define vision and scope of project

Elaboration: Plan project, specify features, baseline architecture

Construction: Build project

Transition: Transition product to users

**Lesson 3**

Problem Analysis: require enough knowledge of the problem domain

Stakeholders: Anyone who represents an interest group whose needs must be satisfied by the project.

Problems are mapped to needs which are in turn mapped to features

(Move in the direction of concrete specification)

Vision Document has

* Problem statement
* Key needs and features
* Scope
* Stakeholders
* Business case for the project

From the RUP, the vision document defines the stakeholders view of the products to be developed specified in terms of stakeholders’ key need and features.

Identifying and documenting the problem, client needs and features.

**Lesson 4: RUP Process Overview**

Requirements Specifications (System or Business Analyst) – Architectural Analysis (Architect) – Use Case analysis – Architectural Design – Use Case Design (Designer) – Class Design – Subsystem Design

Use case are used to capture RUP

Requirement analysis: Output is a use case diagram (It has an actor and a use case)

Architectural Analysis (by a Software architect): Preparation for UC analysis (high level view of the system)

Use Case Analysis: Process of understanding in detail the business domain (Systems analysis) and the user requirements (UC analysis)

Architectural Design: Identify design classes, packages, subsystems and their interfaces, system layering strategy, reusable parts at the system level, components tools for the project

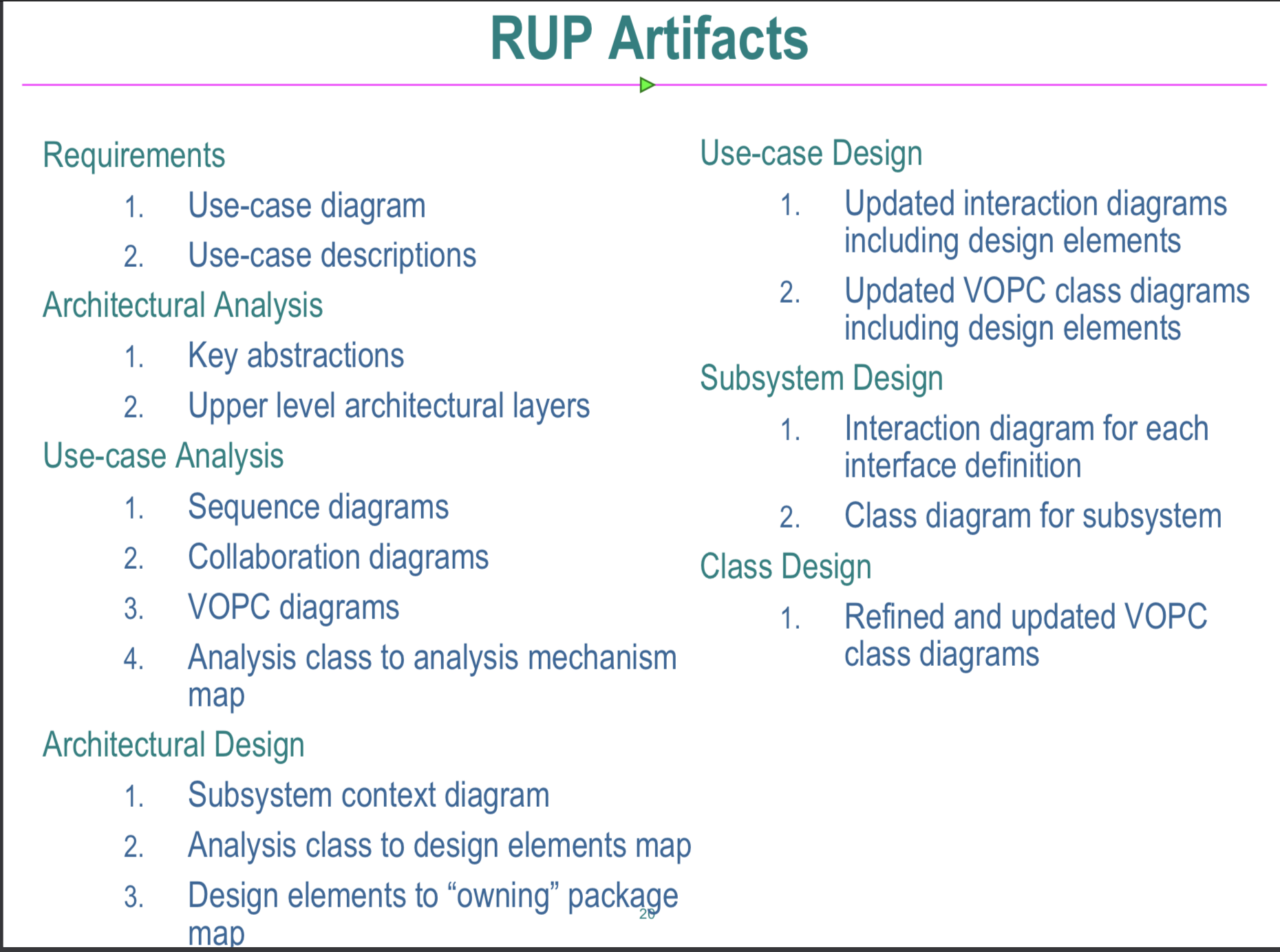
UC Design: Inspect and elaborate architectural design from the perspective of the UC

Subsystem Design: working out details of a subsystem design

Class Design: final preparation for implementation. The overall architecture as well as functional structure is already set. Refining operations, attributes and the relationships for all classes.

Construction and Transitions: Implementer and Tester.

See page 20:



**Lesson 5**

We use UC analysis to capture elements of the SRS document

Elements of SRS include

* Use case Models
* Supplementary Specifications
* Glossary

A UC is a sequence of actions performed by an actor interacting with system to achieve a goal showing how the goal might succeed or fail

UC is described in terms of scenarios and flows

SRS package: UC model + supplementary specifications + Glossary output is design model, test model, end user end docs

Problem description – vision doc

Needs to feature to software requirements to design/test/doc requirements

Use Case Description

1. Title
2. Actors
3. Pre-conditions
4. Post Conditions
5. Business rules
6. Basic flow
7. Alternate paths

Supplementary specifications for SRS – reliability, performance, security, Design constraints, usability

Glossary: terms and definitions

**Lesson 6: Architectural Analysis**

See page 20

A design pattern is a customizable solution to a common design problem

Initial system architecture, we determine if we are building

* Stand-alone desktop application
* Enterprise System
* Web Application
* Client-Server application
* Distributed System (RESTful or SOAP)
* Or a hybrid of 2 or more

Standalone desktop is layered

ORM allow us to gain all advantages of OOAD and let our persistence manager map Entity classes to Relational Database tables, columns and rows

Key abstractions

**Lesson 7**