Software Engineering

Lesson #02 - Lecture

Lesson #02 - Lecture

Your KBTU 202309 Software Engineering class information is updating …

Lesson #02 update is in progress

This will take around 2 hours to complete

Please, don’t turn off your head

Agenda: Lesson #02 - Software Engineering - Lecture

Introduction to Software Engineering

Part #01

Agenda: Lesson #02 - Software Engineering - Lecture

Software Processes &
Agile Software Development

Agenda: Lesson #02 - Software Engineering - Lecture

```
#include<iostream>
Using namespace std;

int main()
{
	cout << “Software Processes” << endl;

	return 0;
}
```

Agenda: Lesson #02 - Software Engineering - Lecture

Chapter 02 - Software processes

Agenda: Lesson #02 - Software Engineering - Lecture

1. Software process models
2. Process activities
3. Coping with change
4. Process improvement

Agenda: Lesson #02 - Software Engineering - Lecture

1. **Software process models**
2. Process activities
3. Coping with change
4. Process improvement

Objective

The objective of this sub-lesson is to introduce you to the idea of a software process - a coherent set of activities for software production

Software Processes

A software process is a set of related activities that leads to the production of a software system

Four fundamental SE activity

Software Specification

The functionality of the software and constraints on its operation must be defined

Four fundamental SE activity

Software Development

The software to meet the specification must be produced

Four fundamental SE activity

Software Validation

The software must be validated to ensure that it does what the customer wants

Four fundamental SE activity

Software Evaluation

The software must evolve to meet changing customer needs

Software Processes

Fundamental activities of software engineering

https://www.youtube.com/watch?v=Z2no7DxDWRI

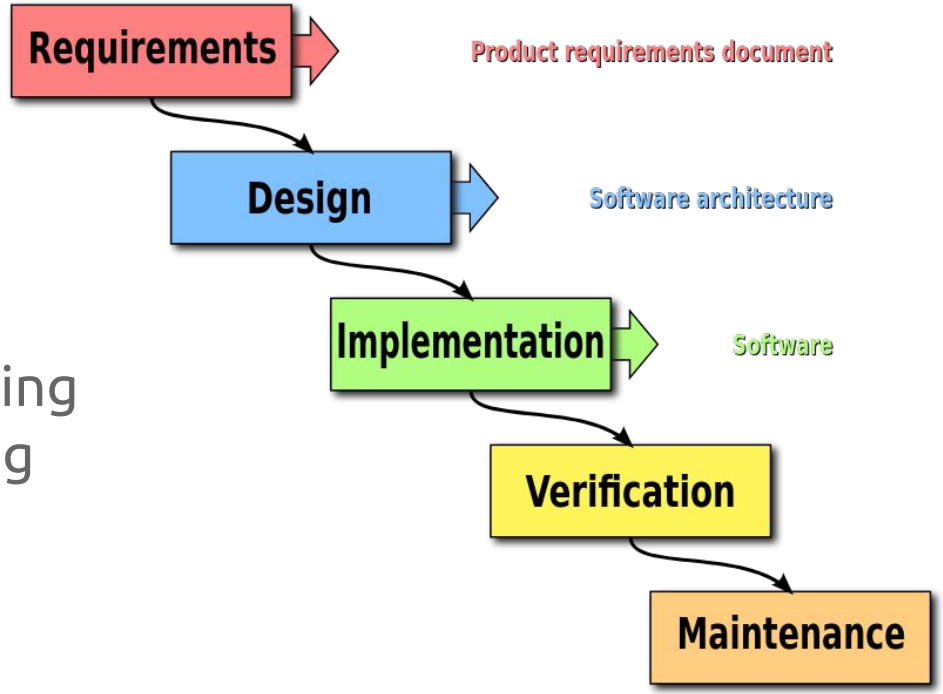
Software Process Models

Generic models are high-level, abstract descriptions of software processes that can be used to explain different approaches to software development

Software Process Models

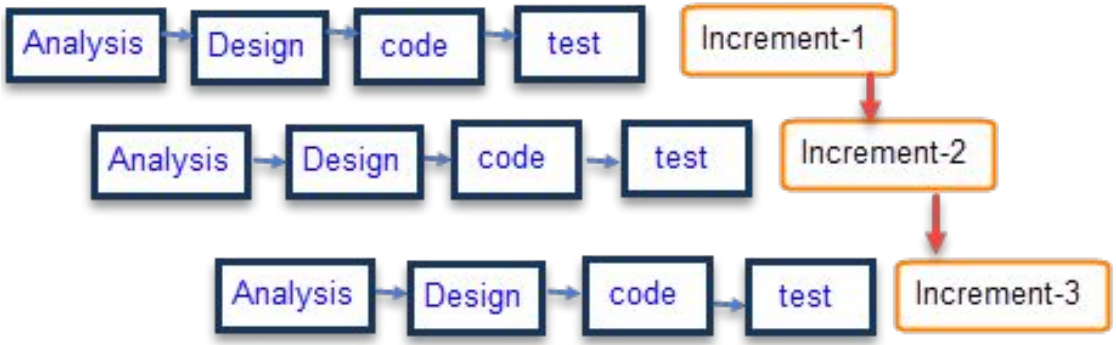
Waterfall Model

1. Requirements analysis and definition
2. System and software design
3. Implementation and unit testing
4. Integration and system testing
5. Operation and maintenance



Software Process Models

Incremental Development



Incremental Model

Software Process Models

Integration and configuration

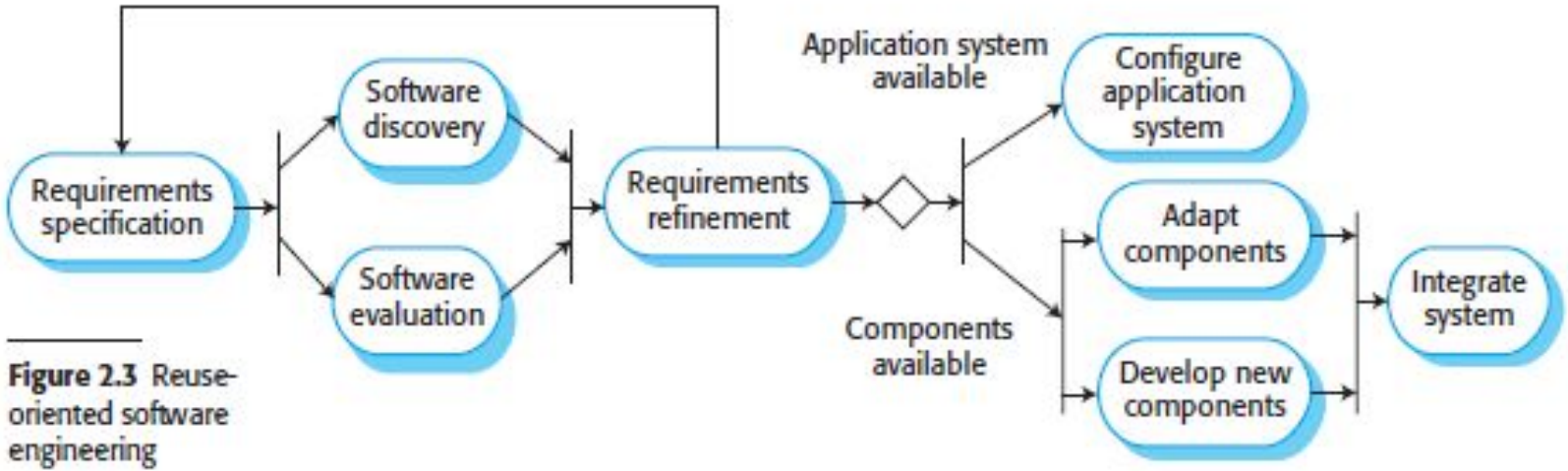


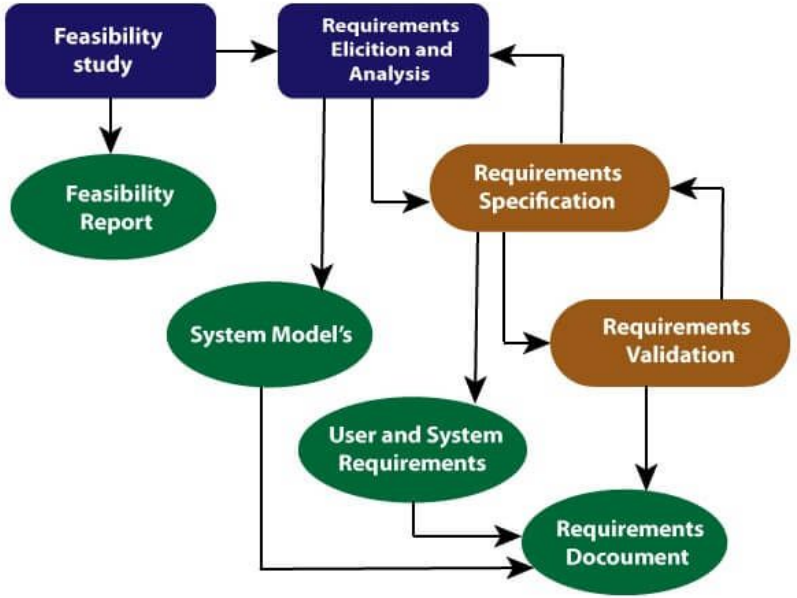
Figure 2.3 Reuse-oriented software engineering

Agenda: Lesson #02 - Software Engineering - Lecture

1. Software process models
2. **Process activities**
3. Coping with change
4. Process improvement

Software Process Activities

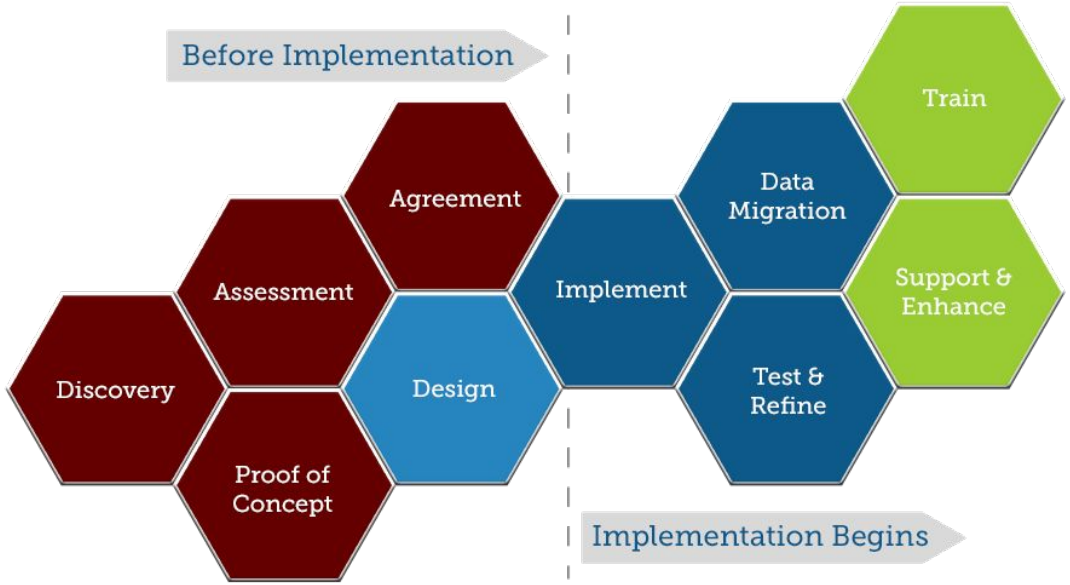
Software Specification



Requirement Engineering Process

Software Process Activities

Software Design & Implementation



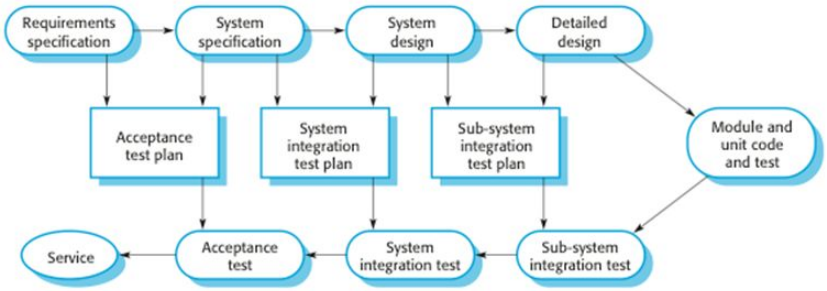
Software Process Activities

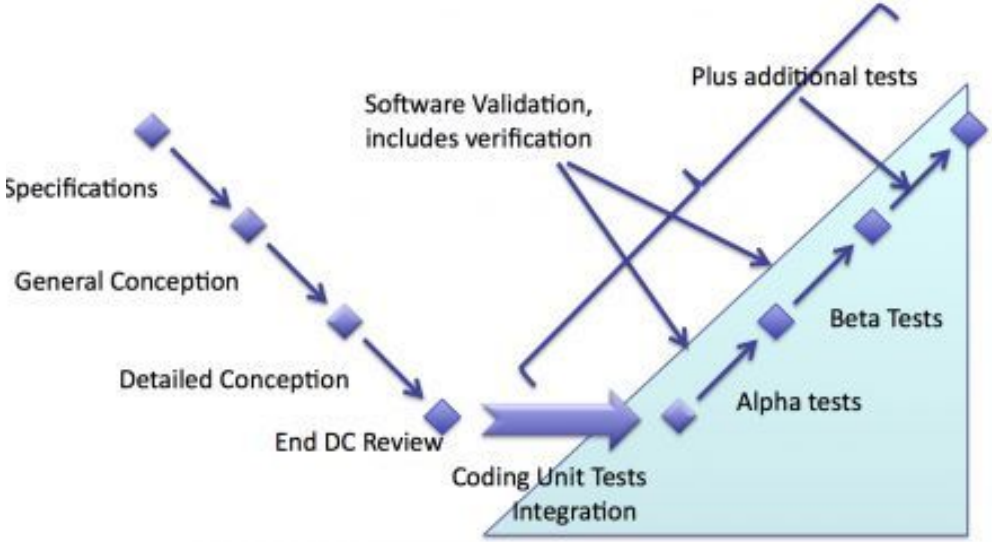
Software Design & Implementation

- Architectural Design
- Database design
- Interface design
- Component selection and design

Software Process Activities

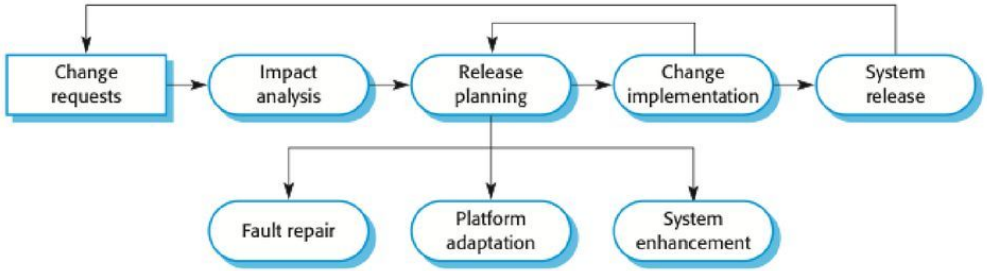
Software Validation





Software Process Activities

Software Evaluation

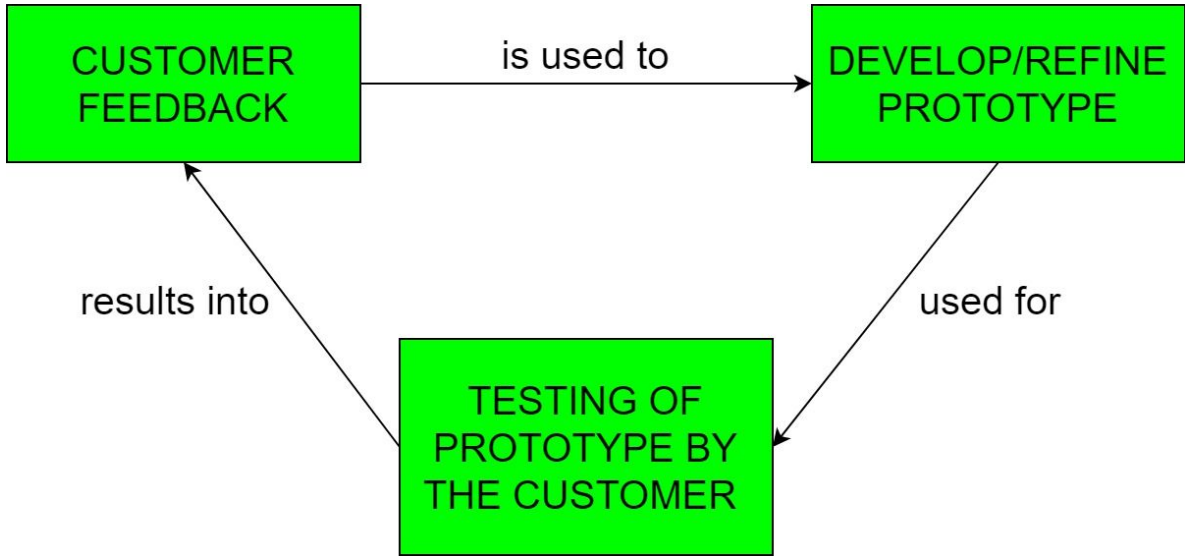


Agenda: Lesson #02 - Software Engineering - Lecture

1. Software process models
2. Process activities
3. **Coping with change**
4. Process improvement

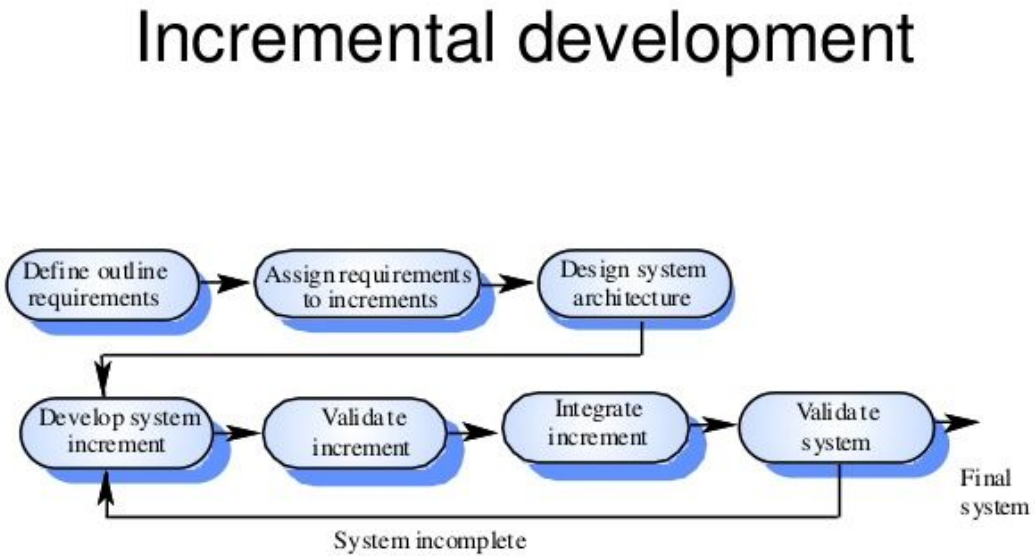
Coping with change

Prototyping



Coping with change

Incremental Delivery



Agenda: Lesson #02 - Software Engineering - Lecture

1. Software process models
2. Process activities
3. Coping with change
4. **Process improvement**

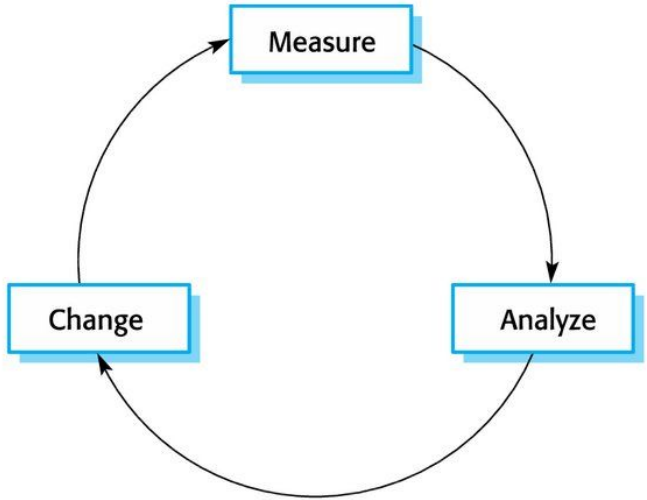
Process Improvement

Nowadays, there is a constant demand from industry for cheaper, better software, which has to be delivered to ever-tighter deadlines

Consequently, many software companies have turned to software process improvement as a way of enhancing the quality of their software, reducing costs, or accelerating their development processes

Process Improvement

Process improvement means understanding existing processes and changing these processes to increase product quality and/or reduce costs and development time



Agenda: Lesson #02 - Software Engineering - Lecture

Software Processes &
Agile Software Development

Agenda: Lesson #02 - Software Engineering - Lecture

```
#include<iostream>
Using namespace std;

int main()
{
	cout << “Agile software development” << endl;

	return 0;
}
```

Agenda: Lesson #02 - Software Engineering - Lecture

Chapter 03 - Agile software development

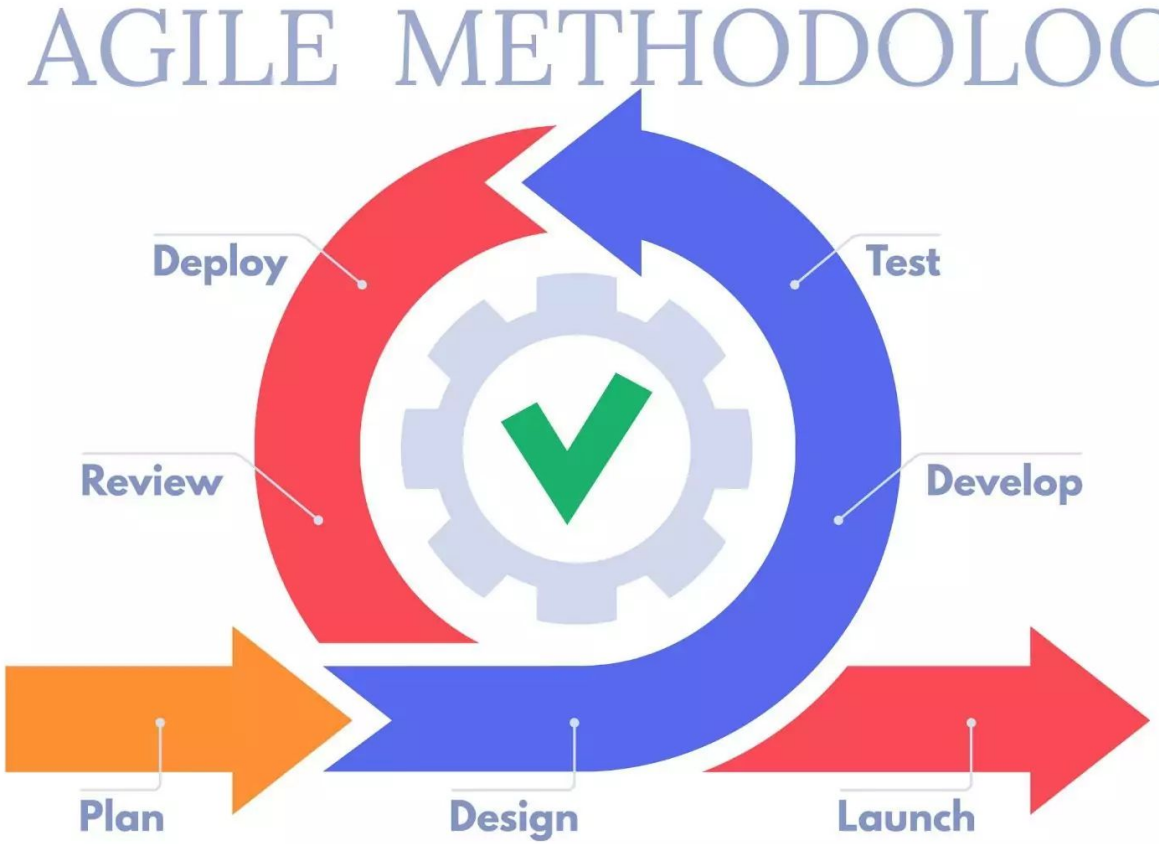
Agenda: Lesson #02 - Software Engineering - Lecture

1. Agile methods
2. Agile development techniques
3. Agile project management
4. Scaling agile methods

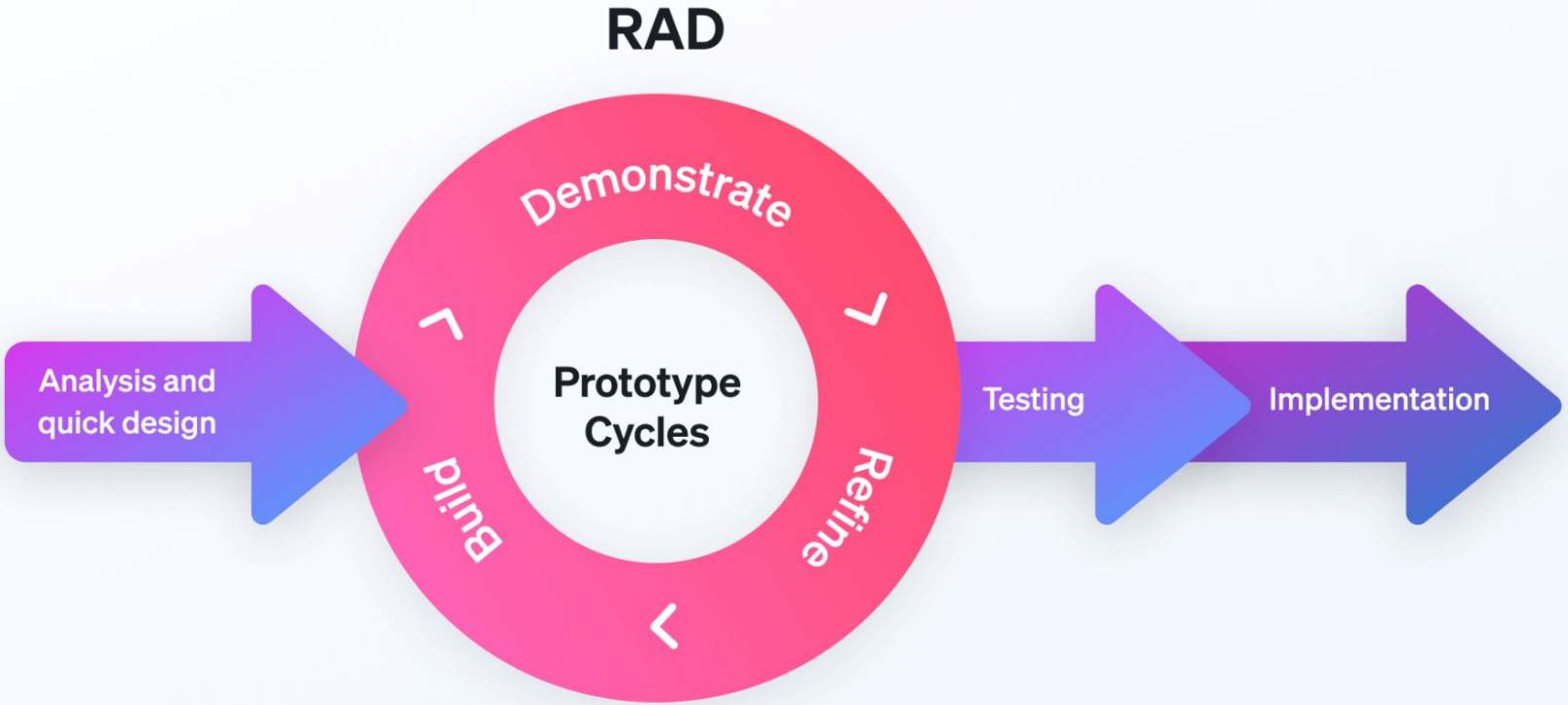
Agenda: Lesson #02 - Software Engineering - Lecture

1. **Agile methods**
2. Agile development techniques
3. Agile project management
4. Scaling agile methods

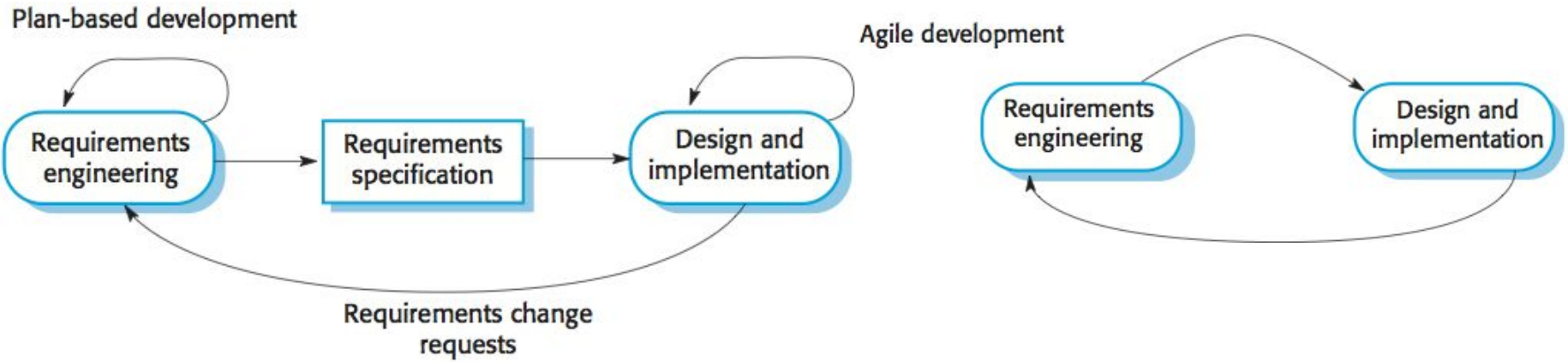
Agile methods



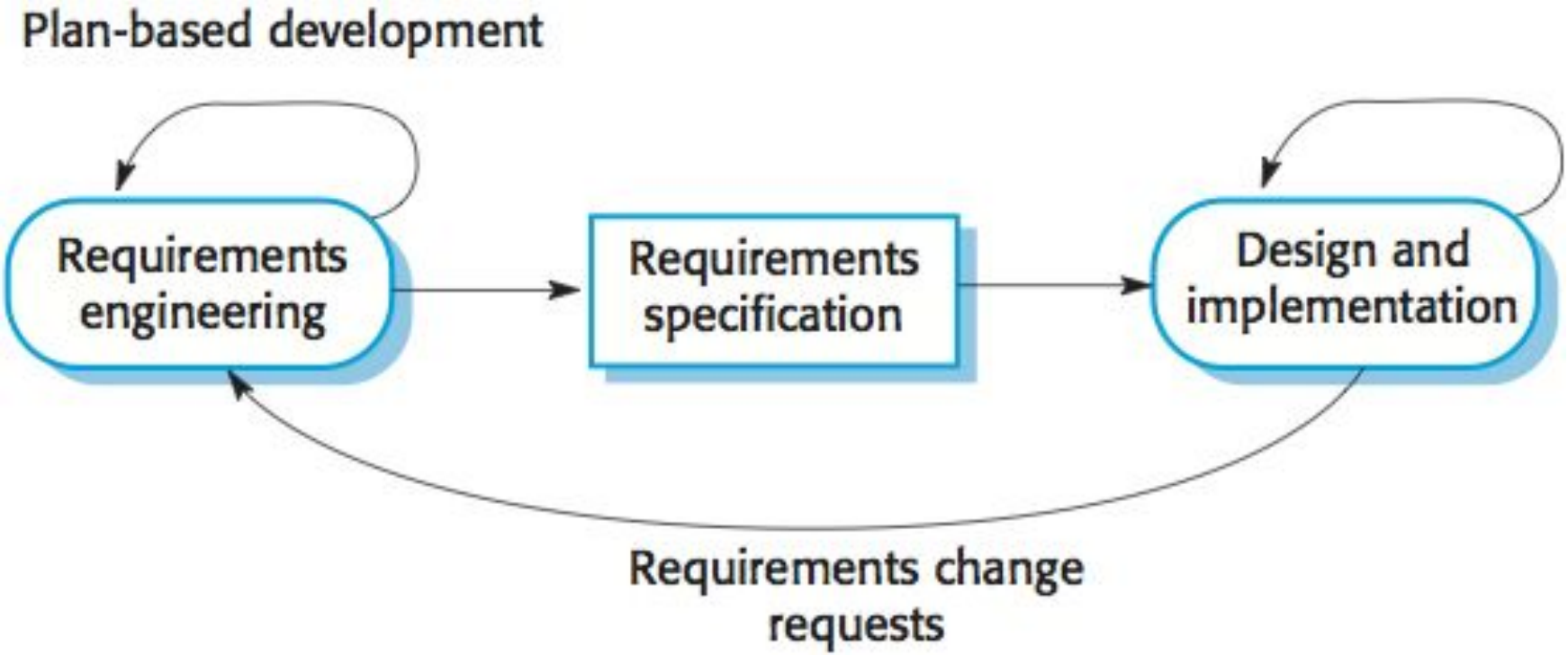
Rapid software development



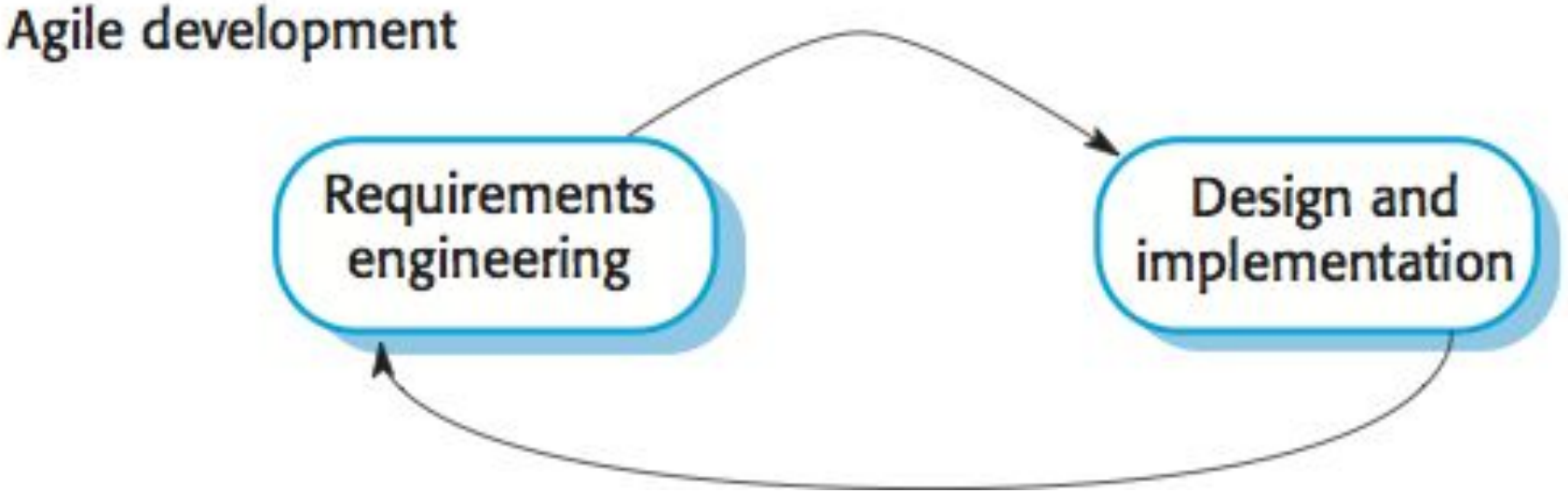
Plan-based vs Agile development



Agile methods



Agile methods



Plan-based vs Agile development

Plan-based and agile software processes

https://www.youtube.com/watch?v=q8X2Rk5sRFI&t=14s

Agile methods

The Agile Manifesto

Individuals and Interactions	over	Processes and Tools
Working Product	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 methods

Agile methods have been particularly successful for two kinds of system development

- 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

Agile methods

Agile methods have been particularly successful for two kinds of system development

- 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 stakeholders and regulations that affect the software

12 Principles of Agile Software Development

1. Satisfy the customer through early and continuous delivery.
2. Welcome changing requirements, even late in development.
3. Deliver working software frequently
4. Business people and developers work together daily
5. Build projects around motivated individuals.
6. Convey information via face-to-face conversation.
7. Working software is the primary measure of progress.
8. Maintain a constant pace indefinitely.
9. Give continuous attention to technical excellence
10. Simplify: maximizing the amount of work not done
11. Teams self-organize.
12. Teams retrospect and tune behavior

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 methods

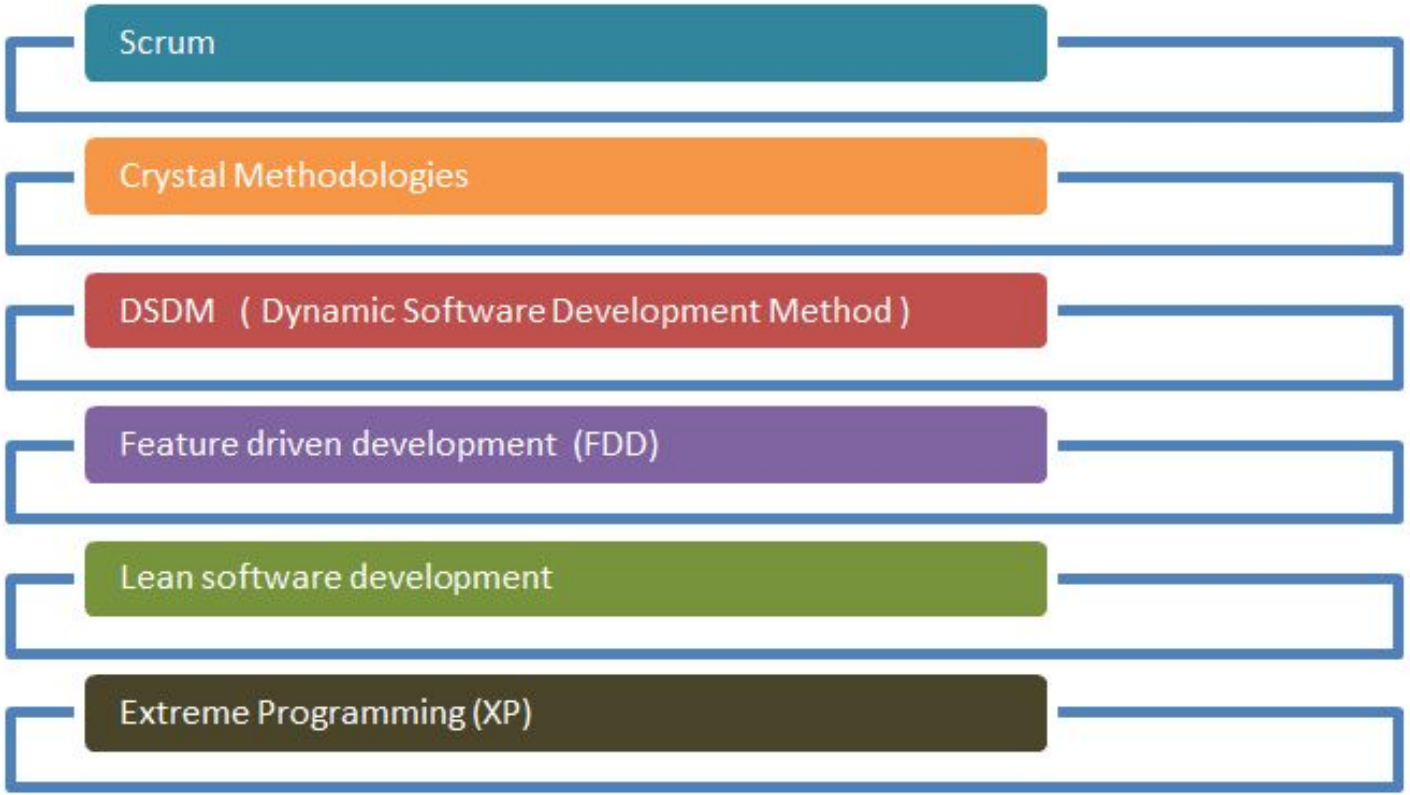
Agile methods for large systems

https://www.youtube.com/watch?v=L1JcQDHJzHA

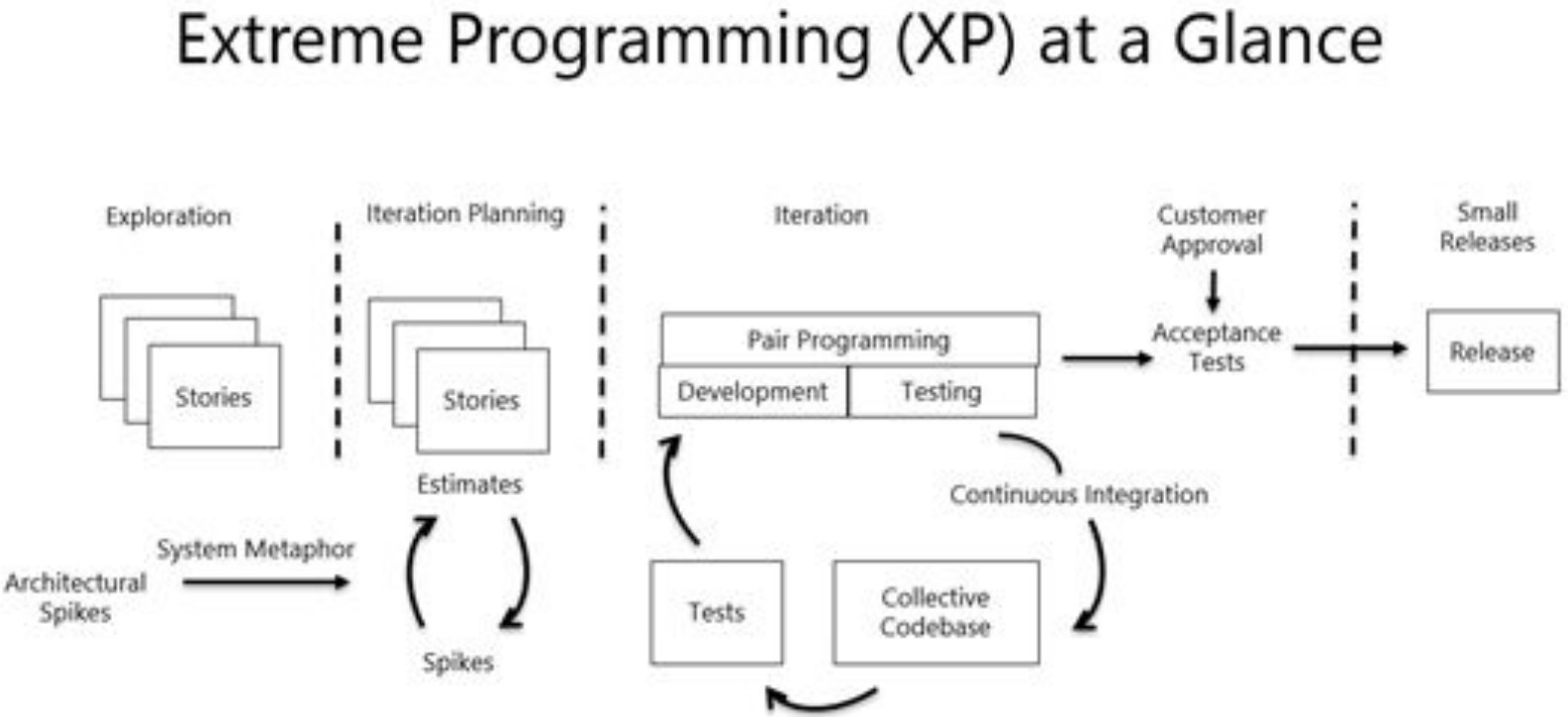
Agenda: Lesson #02 - Software Engineering - Lecture

1. Agile methods
2. **Agile development techniques**
3. Agile project management
4. Scaling agile methods

Agile development techniques

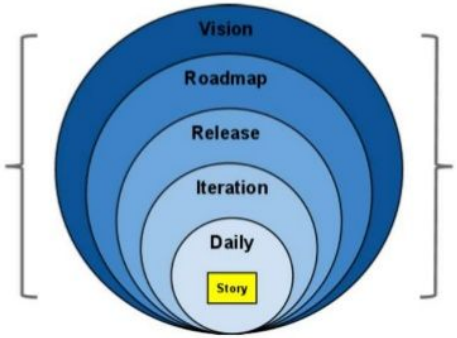


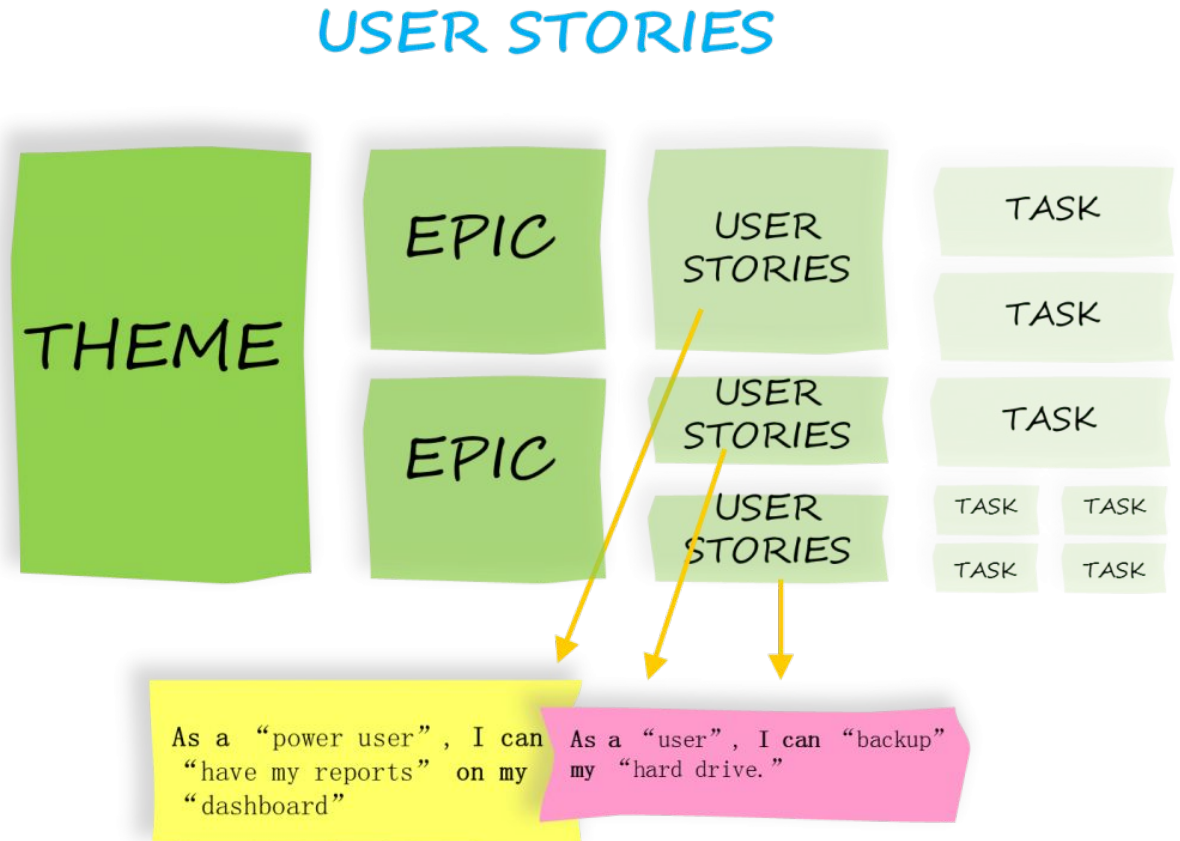
Agile development techniques



User stories

User stories in agile software development



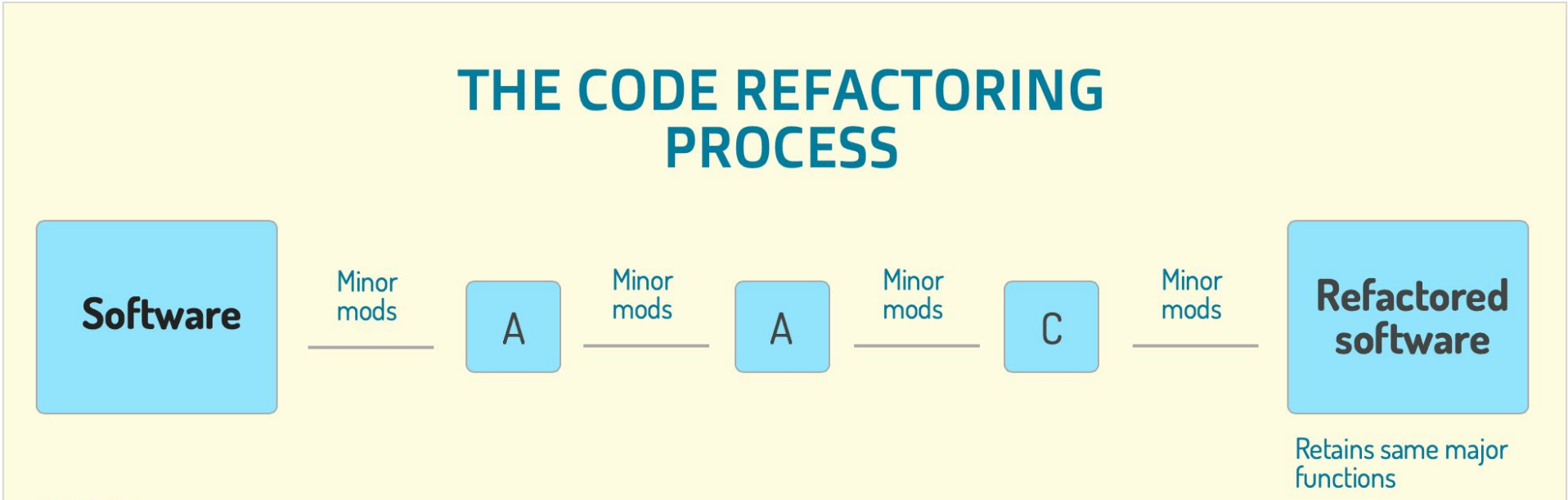


User stories

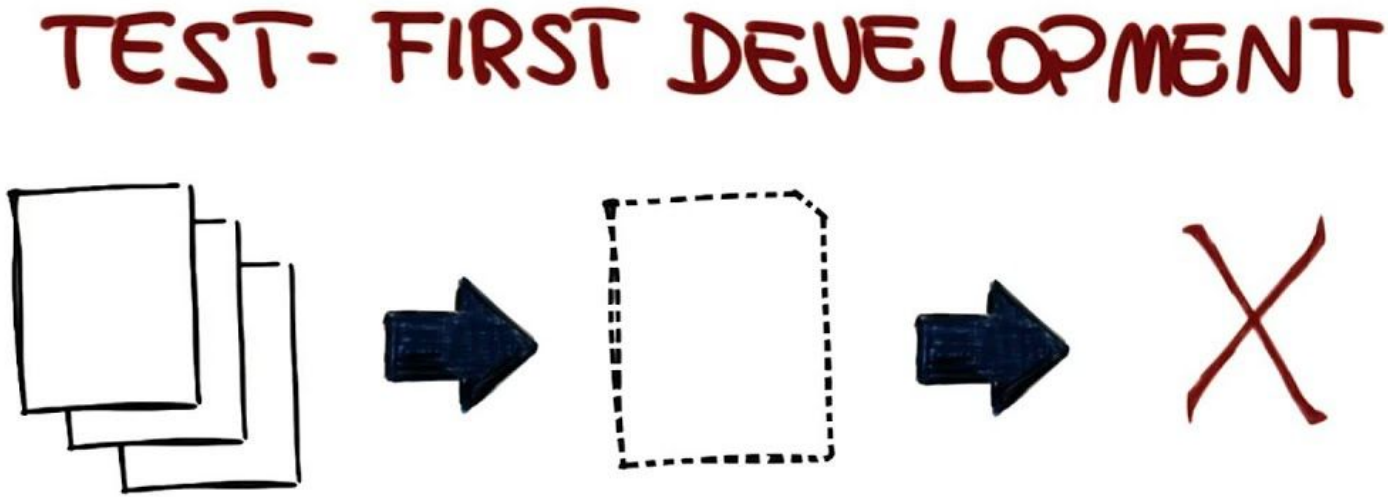
User stories

https://www.youtube.com/watch?v=UpYdVSV3dG8&t=42s

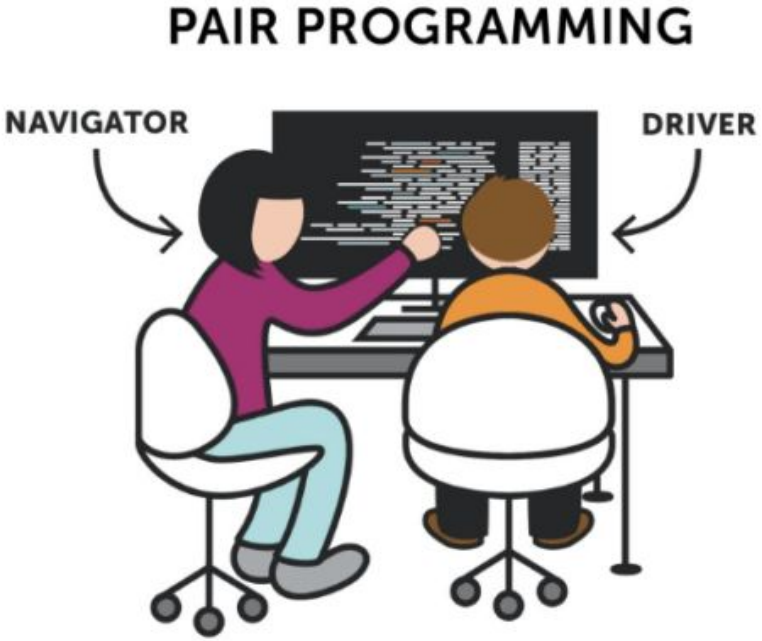
Refactoring



Test-first development



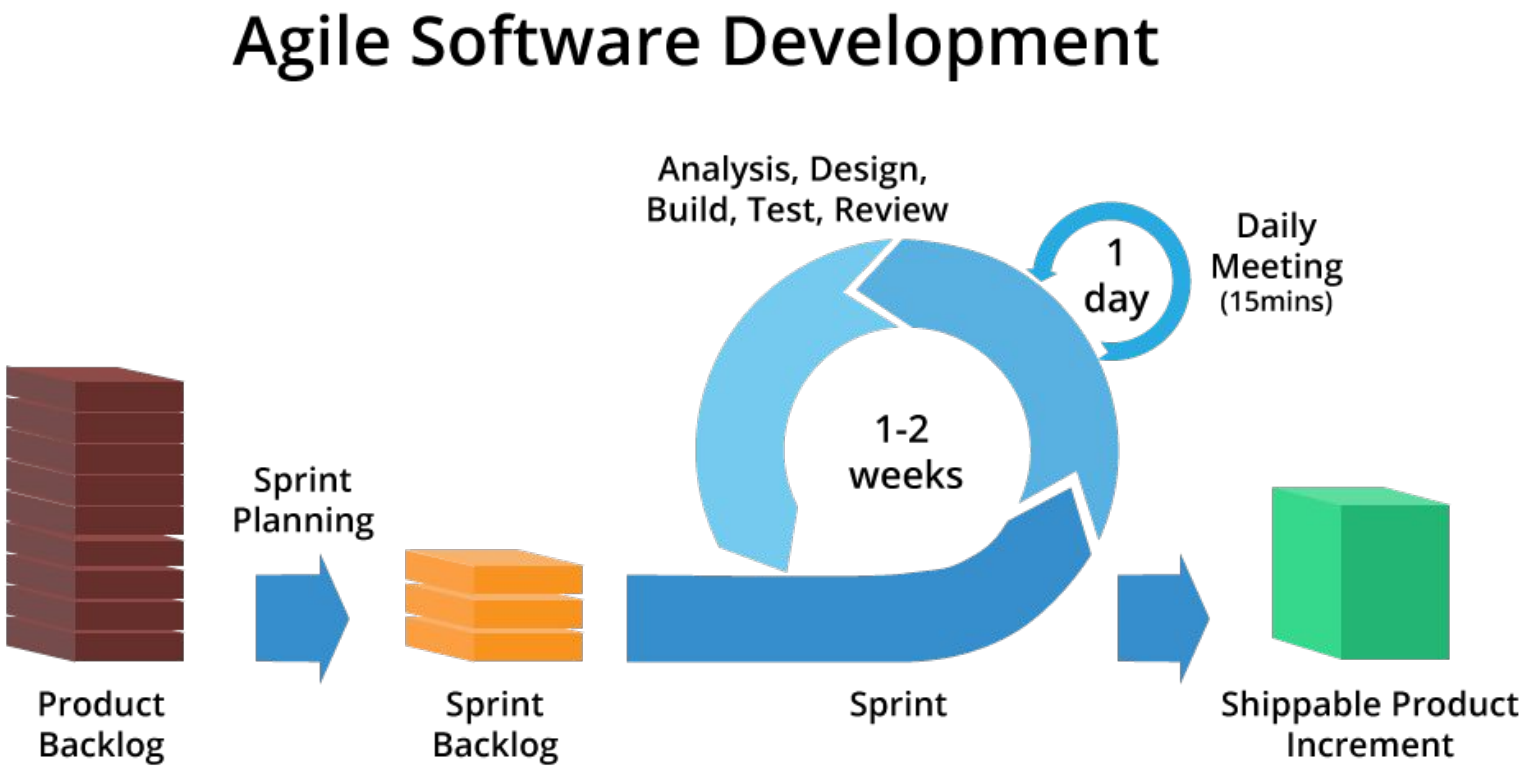
Pair programming



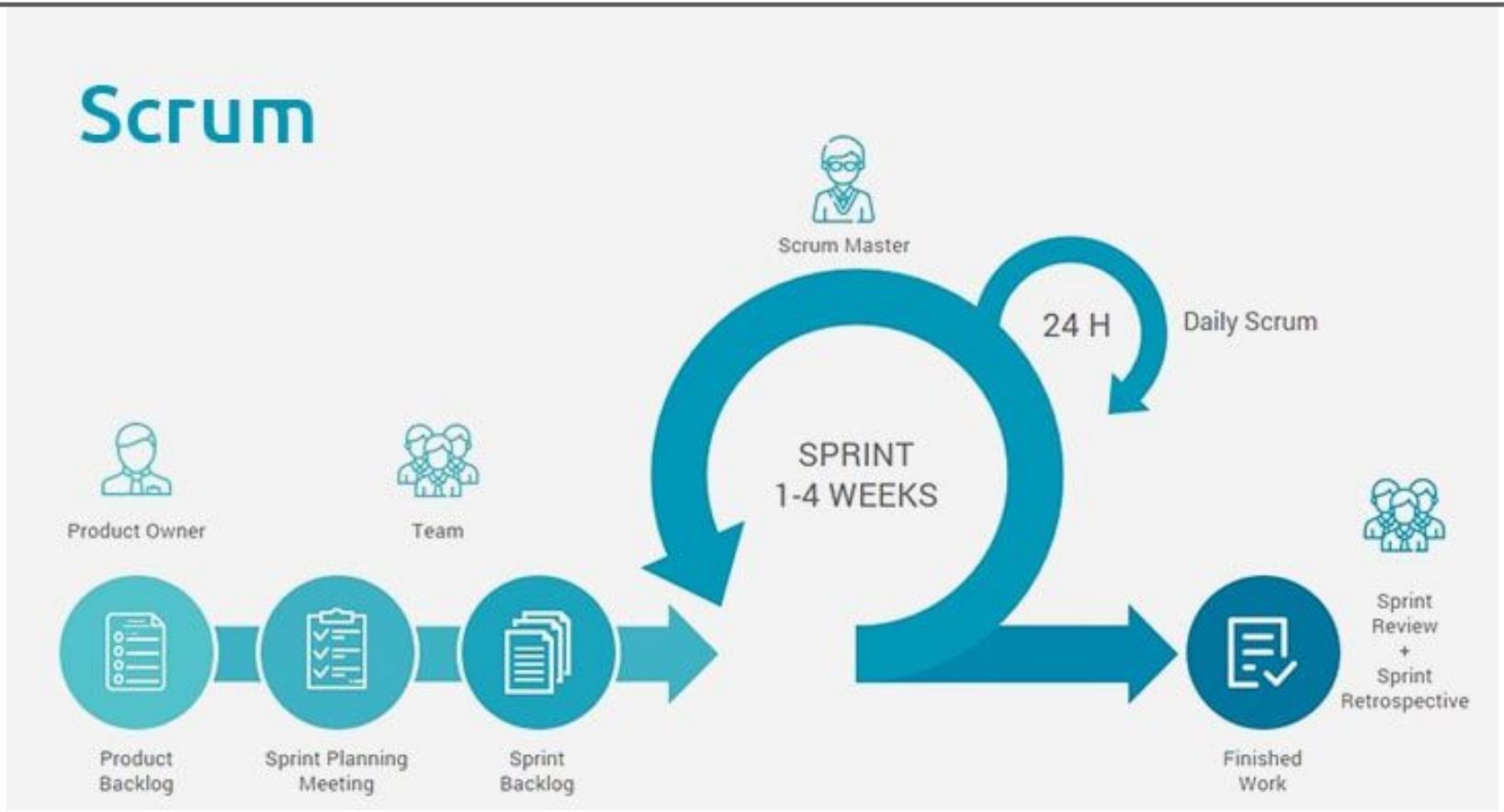
Agenda: Lesson #02 - Software Engineering - Lecture

1. Agile methods
2. Agile development techniques
3. **Agile project management**
4. Scaling agile methods

Agile project management



Agile project management



Agenda: Lesson #02 - Software Engineering - Lecture

1. Agile methods
2. Agile development techniques
3. Agile project management
4. **Scaling agile methods**

Scaling Agile Methods

Practical problems with agile methods

Agile and plan-driven methods

Agile methods for large systems

Agile methods across organisations

Scaling Agile

Scaling Agile

https://www.youtube.com/watch?v=GuK46hw3CyI

Agenda: Lesson #02 - Software Engineering - Lecture

Q & A