

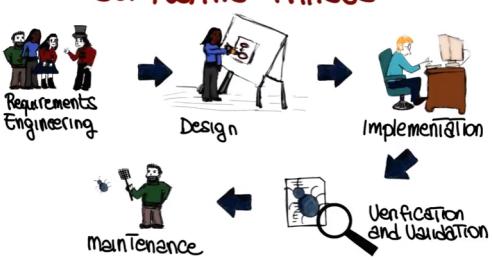
 DA-IICT

---

## IT 314: Software Engineering

*Software Process Models - AGILE*

**SOFTWARE PHASES**



```
graph LR; RE[Requirements Engineering] --> D[Design]; D --> I[Implementation]; I --> VV[Verification and Validation]; VV --> M[Maintenance]; M --> RE; I --> VV
```

The diagram illustrates the five phases of software development: Requirements Engineering, Design, Implementation, Verification and Validation, and Maintenance. Each phase is represented by a small illustration and a label. Arrows indicate a sequential flow from Requirements Engineering through Implementation, followed by a feedback loop from Maintenance back to Requirements Engineering, and another arrow from Implementation to Verification and Validation.

---

1



---

## Agile Software Development

Agile reduces the risk by delivering the value of the project very early

---

## Agile Software Development

Traditional Software Development - Opposed to Agile

1. PLAN



3. TEST

WHAT'S THE  
**PROBLEM**  
WITH A  
**TRADITIONAL**  
**PROJECT** ?

2. BUILD



## Agile Software Development

**CHANGE!**

2. BUILD



## Agile Software Development

USER STORY:  
AS AN ATTENDEE  
I WANT TO CROSS  
THE CANYON  
SO THAT I CAN  
ATTEND THE  
CONFERENCE

Simple plans

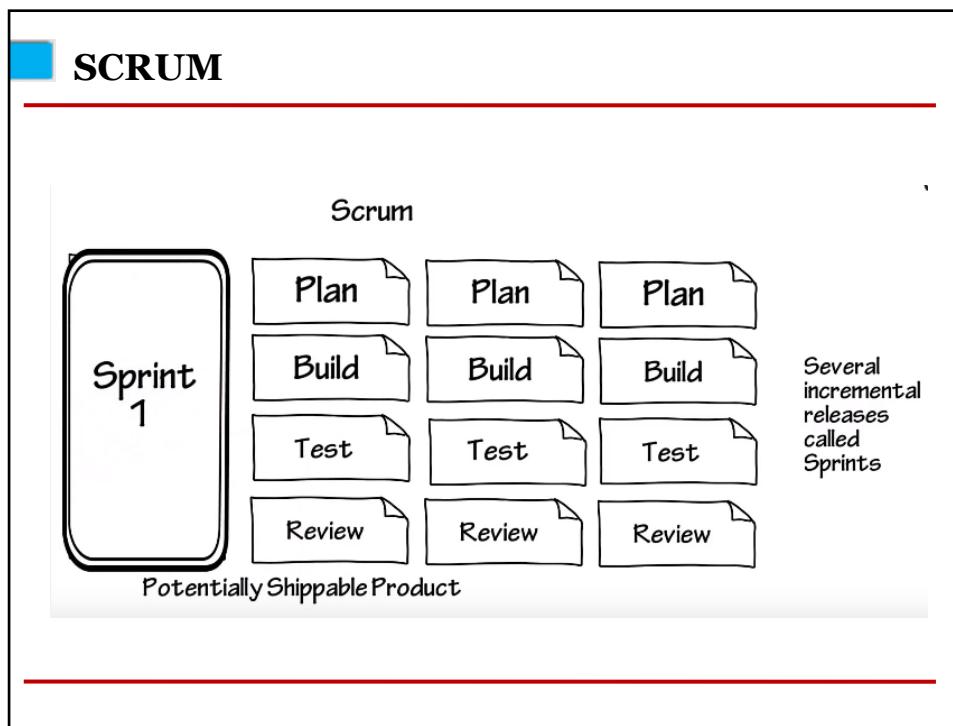
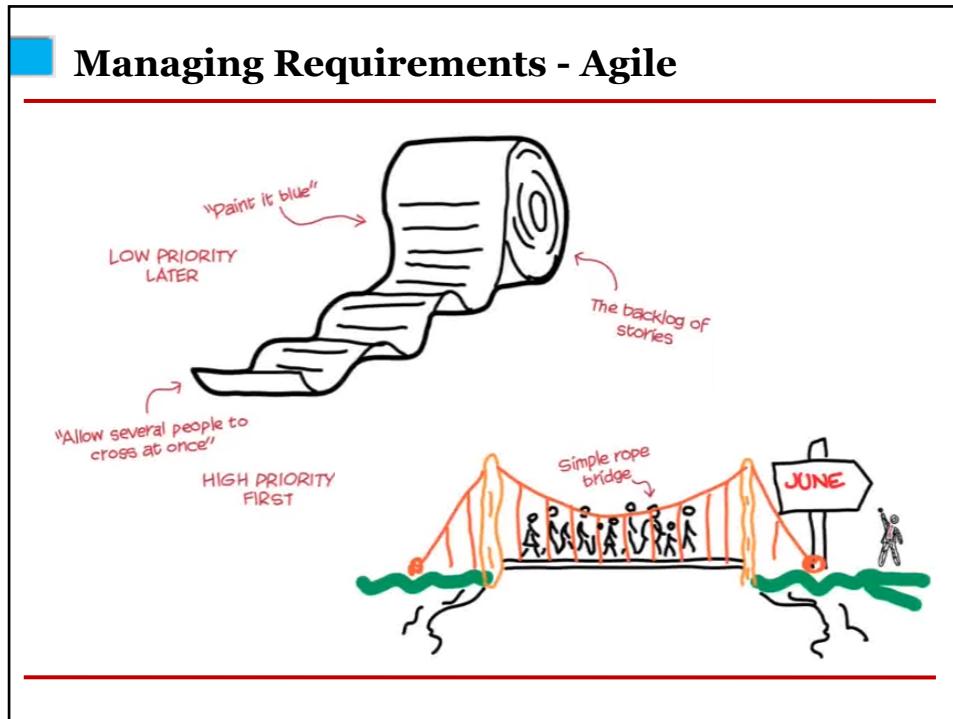
Deliver something early

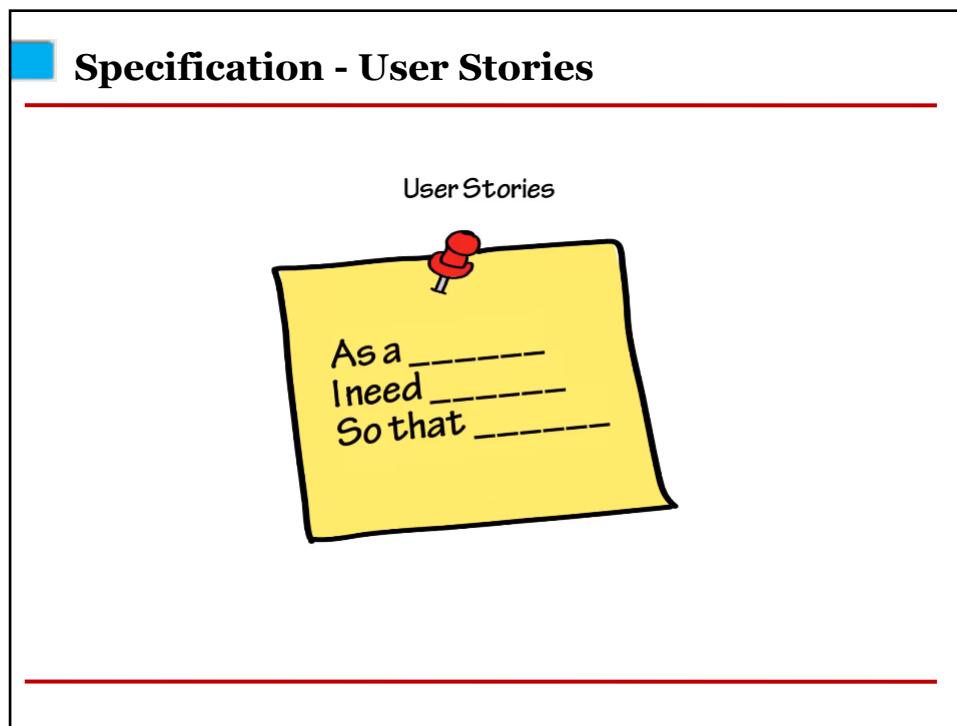
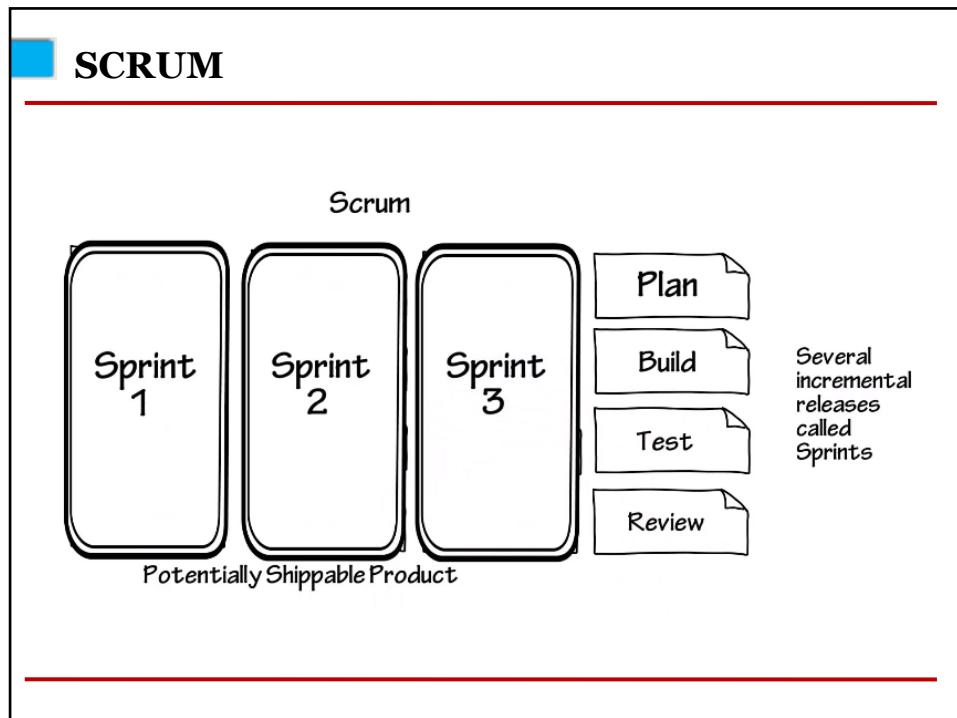
Test all the time

SEPTEMBER CONFERENCE

## Agile Software Development

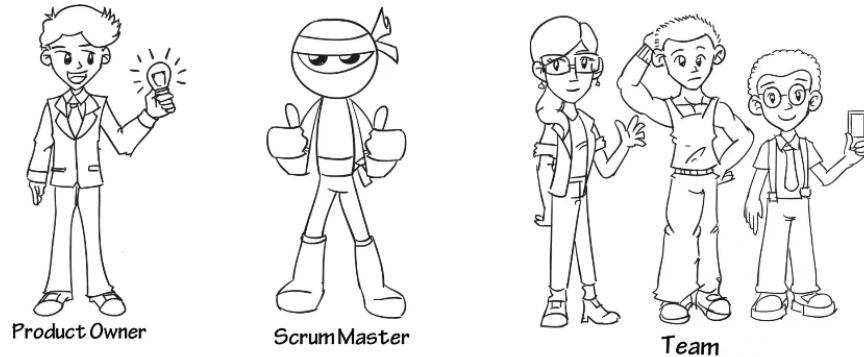
	Focus on the code		Customer involvement
	People over process		Expectation that requirements will change
	Iterative approach		Simplicity





## 3R - SCRUM

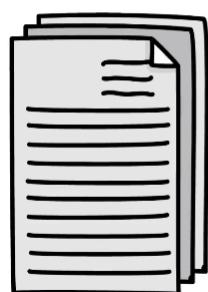
### 3 Roles



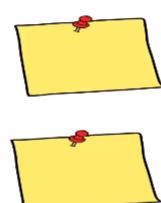
## 3A - SCRUM

### 3 Artifacts

Product Backlog



Sprint Backlog



Burndown Chart



## 3C - SCRUM

### 3 Ceremonies

Sprint Planning



Daily Scrum



Sprint Review



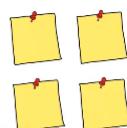
## SCRUM Process



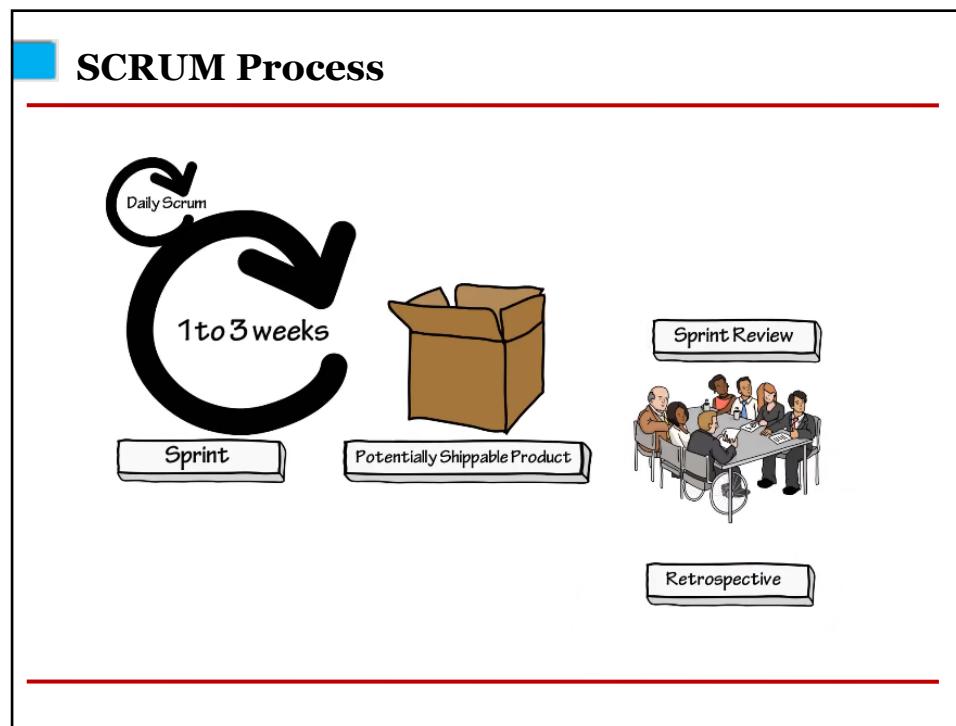
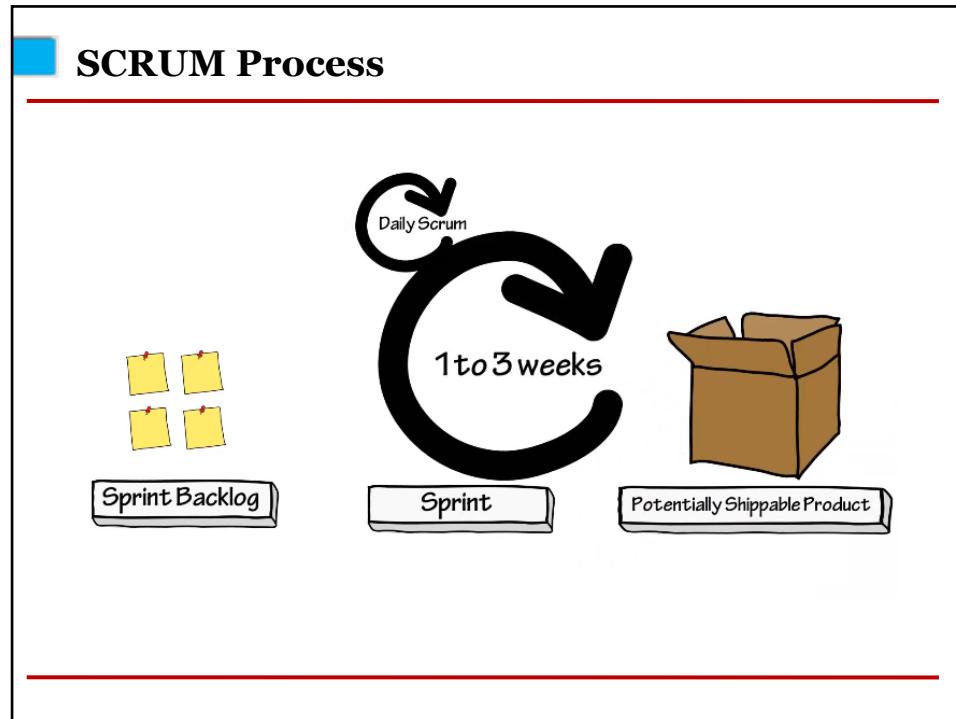
Product Backlog

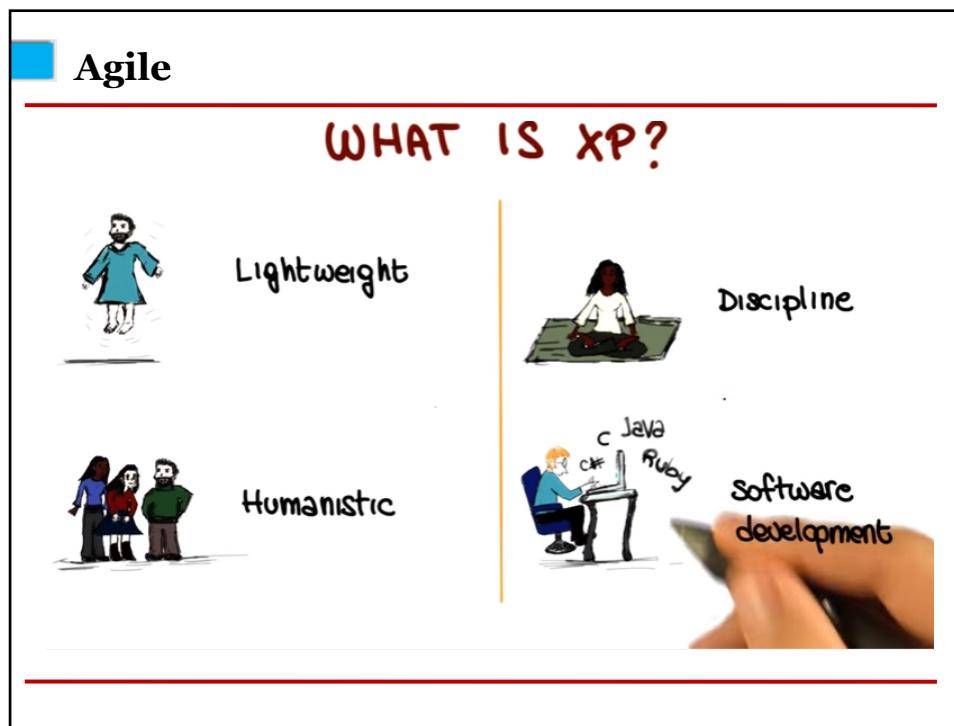
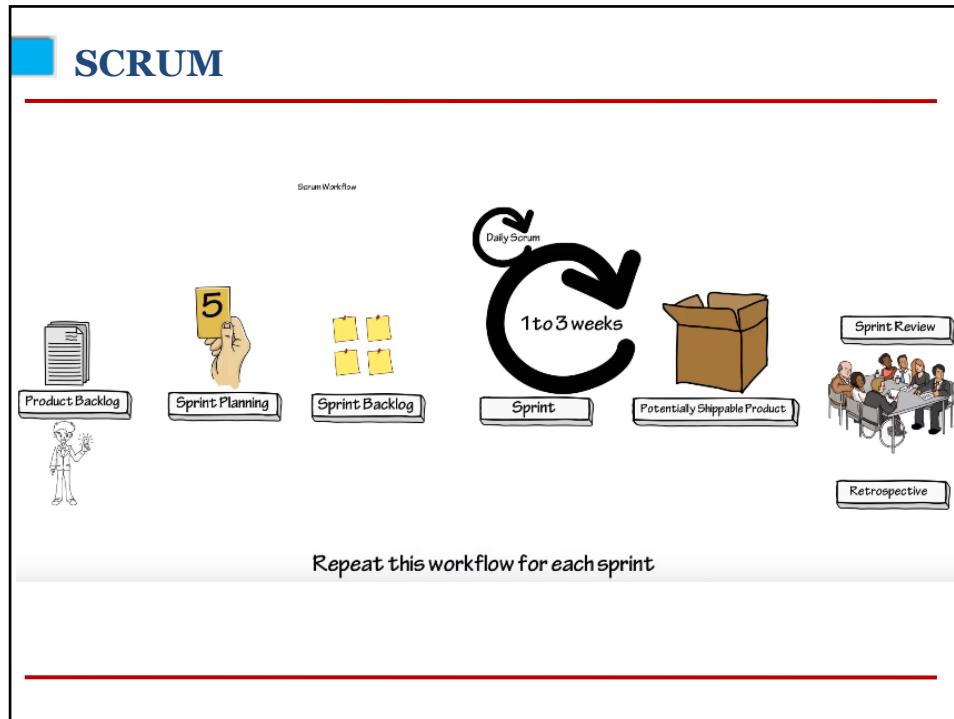


Sprint Planning

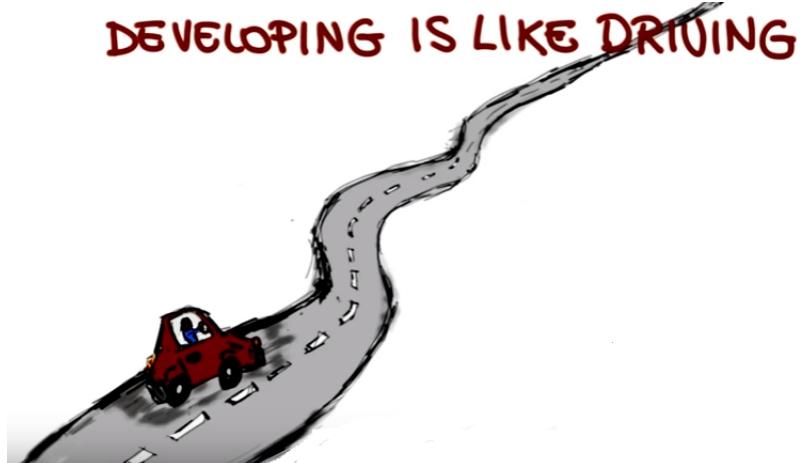


Sprint Backlog





## DEVELOPING IS LIKE DRIVING

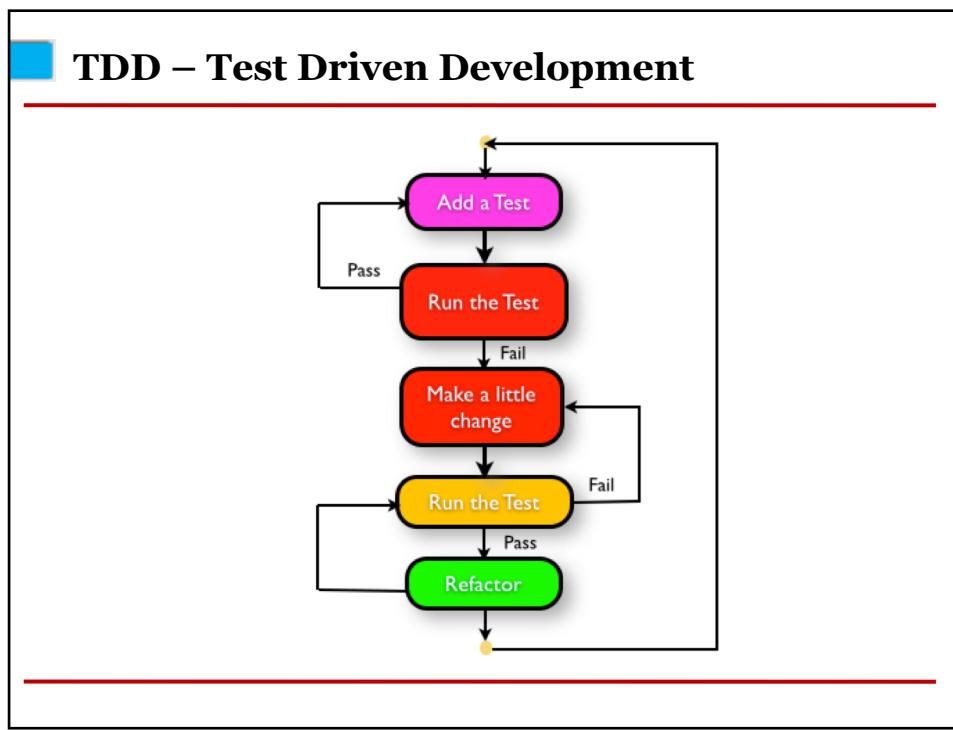
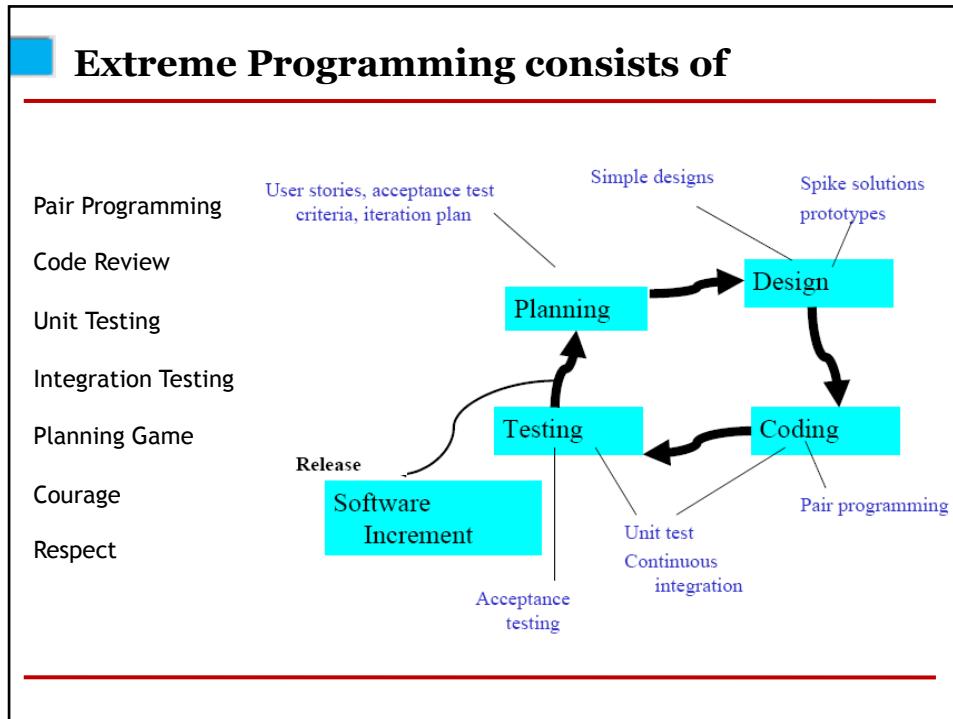


## MENTALITY OF SUFFICIENCY

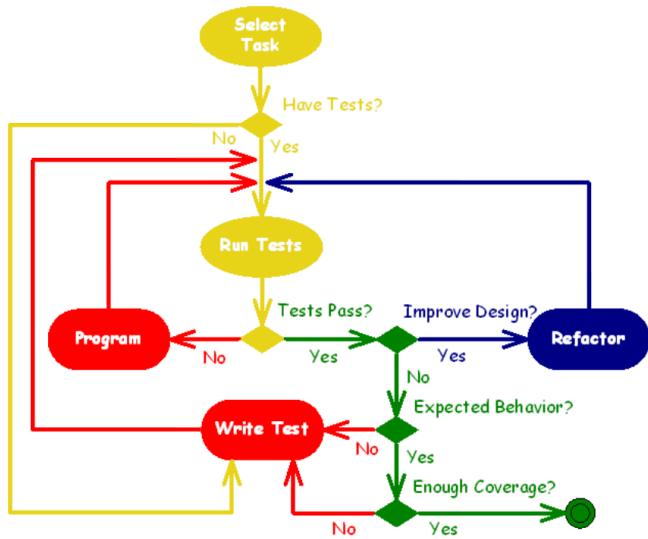


How would you program if you had all the time in the world?

- Write tests
- Restructure often
- Talk with fellow programmers and with the customer often



## TDD – Test Driven Development



## TDD – Test Driven Development

Test-driven development is not about testing.

Test-driven development is about development (and design), specifically improving the quality and design of code.

### Cycle:

- Write the test
- Run the test (there is no implementation code, test does not pass)
- Write just enough implementation code to make the test pass
- Run all tests (tests pass)
- Refactor
- Repeat

## TDD – Test Driven Development

---

Ensures quality

Keeps code clear, simple and testable

Provides documentation for different team members

Repeatable tests

Enable rapid change

**Questions??**

Next Lectures...

Other Process Models  
(RUP, Agile (XP) and so on...., CMM