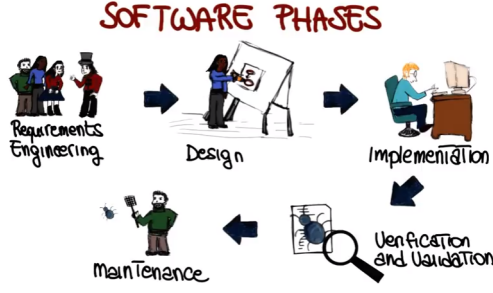

DA-IICT

---


## IT 314: Software Engineering

*Software Process Models – RUP|XP|TDD*

**SOFTWARE PHASES**



1



## RUP – Rational Unified Process

---

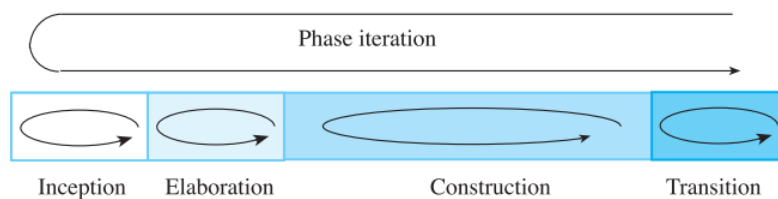
- Life Cycle model proposed by Booch, Jacobson, and Rumbaugh (“The three Amigos”) derived from the work on UML
- Rational Unified Process (RUP) uses Unified Modeling Language (UML) as core notation
- Described from 3 perspectives
  - □ A dynamic perspective that shows phases over time;
  - □ A static perspective that shows process activities;
  - □ A practice perspective that suggests good practice.
- Unified Process is distinguished by being
  - □ Use-case driven
  - □ Architecture-centric
  - □ Iterative and incremental

---

## RUP – Rational Unified Process

- RUP proposes a phase model that identifies four discrete phases in the software process
- **Inception**
  - □ Establish the business case for the system
  - □ Decide to cancel or continue the project
- **Elaboration**
  - □ Develop an understanding of the problem domain and the system architecture.
- **Construction**
  - □ System design, programming and testing.
- **Transition**
  - □ Deploy the system in its operating environment.

## Iterative Phase Model



- Each phase may be enacted in an iterative way with the results developed as increments
- The whole set of phases may also be enacted incrementally  
Whole set = cycle (later on..)
- An iteration represents a set of activities for which there is a milestone ("well-defined intermediate event")
- The scope and results of the iteration are captured via discrete work products called artifacts.

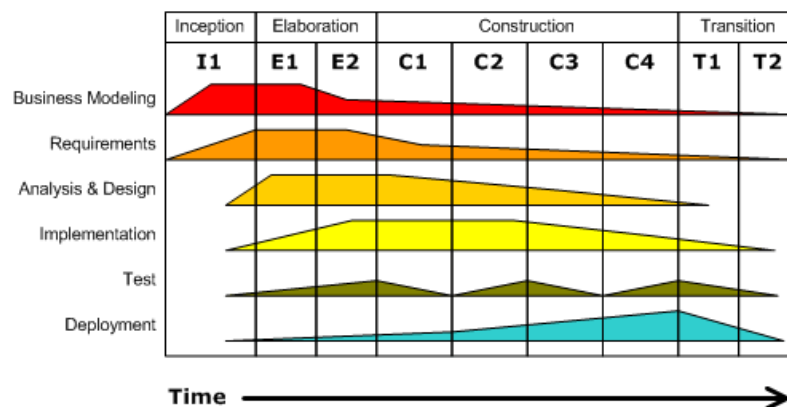
## Artifact sets

- Each artifact set has a different intention and uses different notations to capture the relevant artifacts.
- Management Set:
  - □Notation: Ad hoc text, graphics, textual use cases
  - □Goal: Capture plans, processes, objectives, acceptance criteria.
- Requirements set:
  - Notation: Structured text, models in UML (Use Case, Class, Sequence)
  - □Goal: Capture the problem in the language of the problem domain
- Design set:
  - □Notation: Structured text, models in UML
  - □Goal: Capture the engineering blueprints
- Implementation set:
  - □Notation: Programming language
  - □Goal: Capture the building blocks of the solution domain in human-readable format.
- Deployment set:
  - □Form: Machine language
  - □Goal: Capture the solution in machine-readable format.

## RUP

### **Iterative Development**

Business value is delivered incrementally in time-boxed cross-discipline iterations.



## Life of a Unified Process

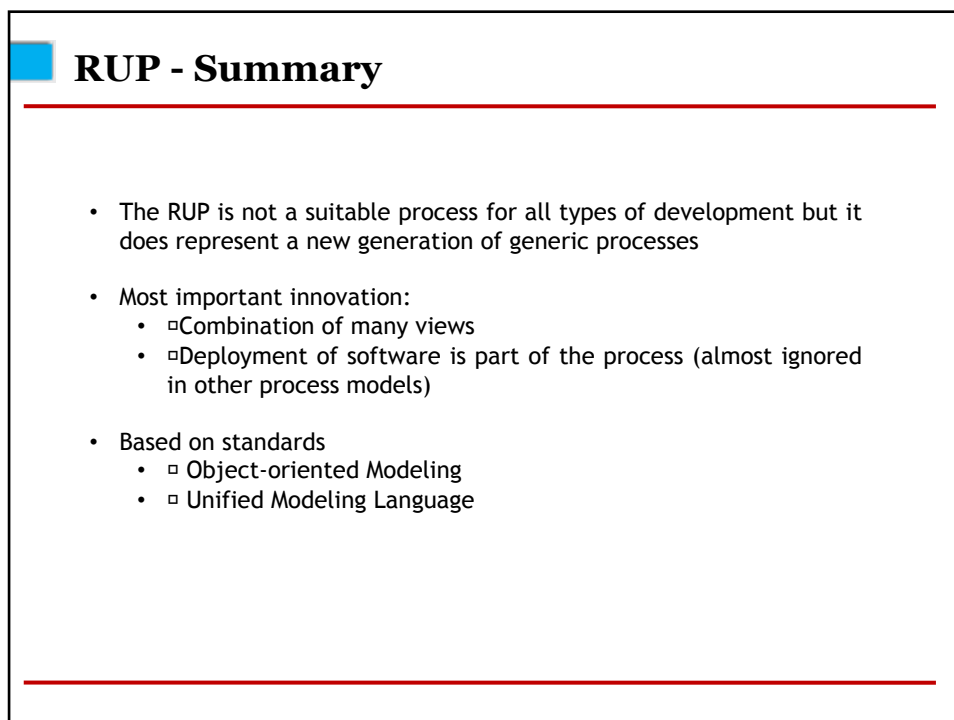
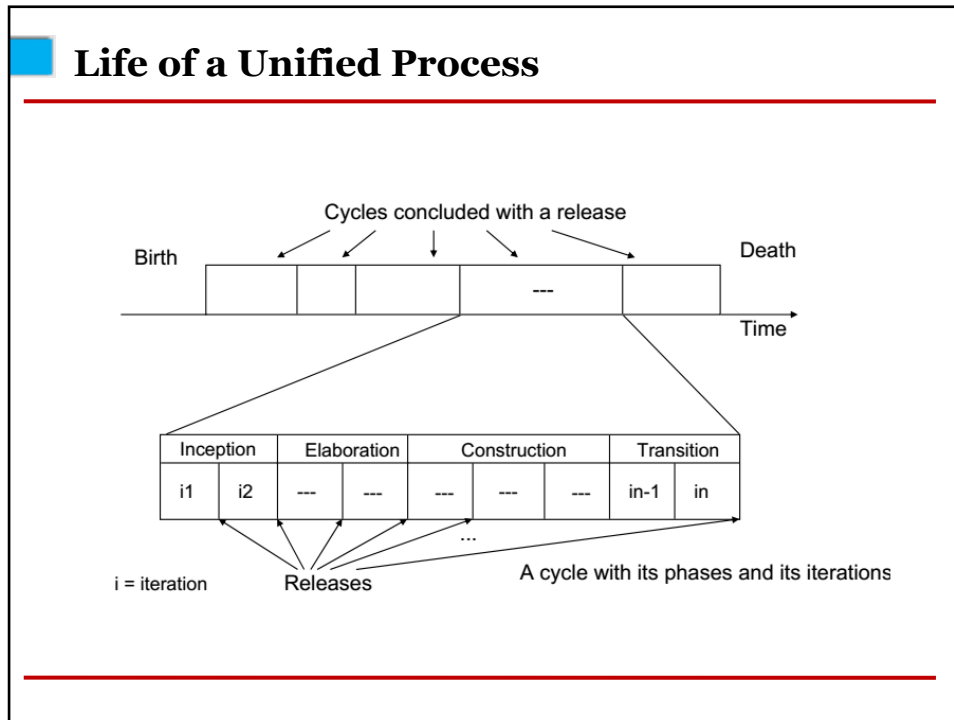
---

- Unified Process repeats over a series of cycles each concluding with a product release (increment) to the users
  - Cycles have no specific name but characterize the stage of maturity of the software system (like “birth” → “death”)
  - Each cycle has four phases (each with a number of iterations)
    - Inception, Elaboration, Construction & Transition
    - Phases have goals (→ result in artifacts or models)
  - Delivered products will be described by related models each with “trace” dependencies which chain backwards and forwards
    1. □ Use Case Model
    2. □ Analysis Model
    3. □ Design Model
    4. □ Deployment Model
    5. □ Implementation Model
    6. □ Test Model
- 

## Life of a Unified Process

---

- Unified Process repeats over a series of cycles each concluding with a product release (increment) to the users
  - Cycles have no specific name but characterize the stage of maturity of the software system (like “birth” → “death”)
  - Each cycle has four phases (each with a number of iterations)
    - Inception, Elaboration, Construction & Transition
    - Phases have goals (→ result in artifacts or models)
  - Delivered products will be described by related models each with “trace” dependencies which chain backwards and forwards
    1. □ Use Case Model
    2. □ Analysis Model
    3. □ Design Model
    4. □ Deployment Model
    5. □ Implementation Model
    6. □ Test Model
-



## XP (Extreme Programming)

XP is a lightweight methodology for small to medium sized teams developing software in the face of vague or rapidly changing requirements.

Kent Beck

## Agile

### WHAT IS XP?



Lightweight



Discipline



Humanistic



software  
development

## DEVELOPING IS LIKE DRIVING

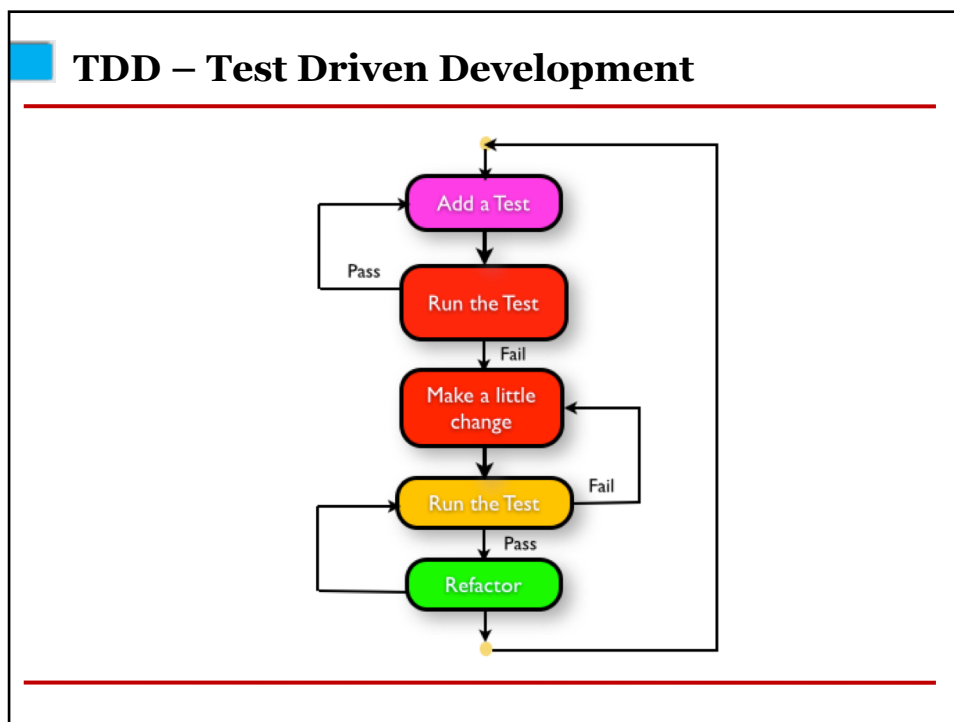
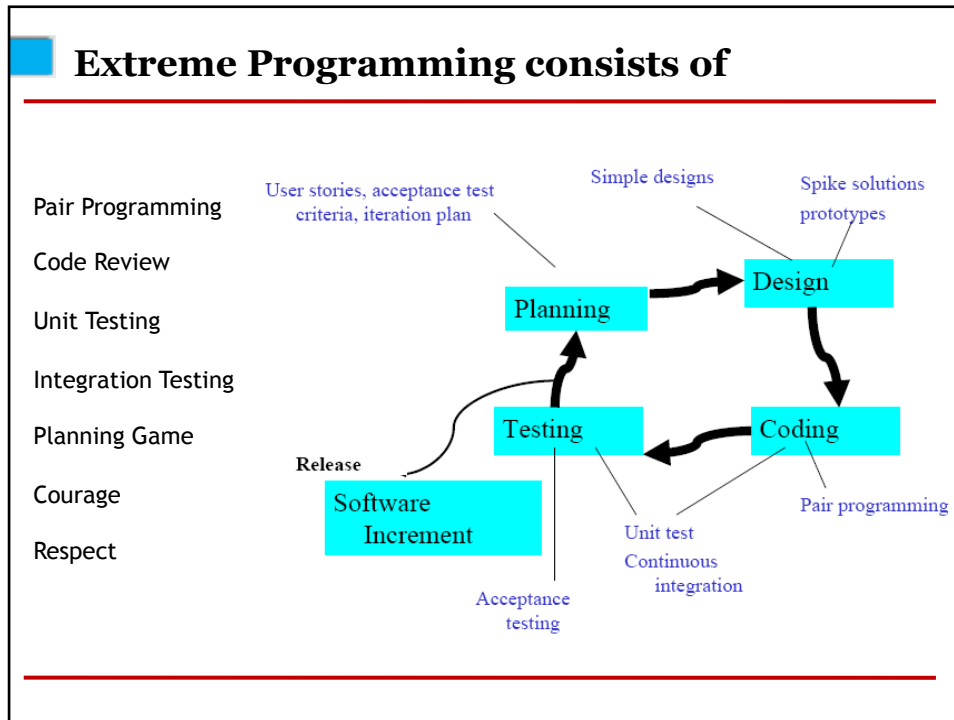


## MENTALITY OF SUFFICIENCY



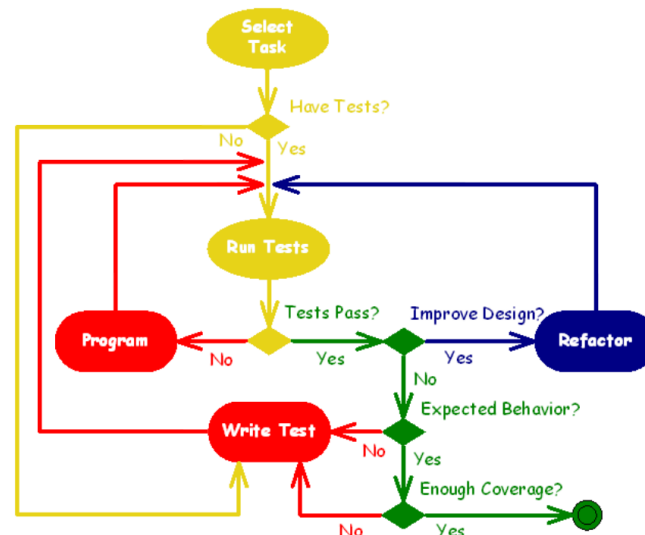
How would you program if you had all the time in the world?

- 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...  
Feasibility Studies...

---