Software Process

Week 2

Agenda (Lecture)

- What is the definition of software process and a software process model and why these are important in software development?
- Are there any drawbacks about process in software development?
- What are the main activities of software processes?
- What are the main software process types?
- How would a team such as a student team go about selecting a process model (or developing a process)?

Agenda (Lab)

- Use cases
- One-page project proposal
- Weekly progress report
- Hour 1 quizzes (page 29)
- Hour 6 quizzes (page 100)
- Submit the report, proposal and the answers of the quizzes by the end of the Wednesday lab session.

Weekly Progress Report

- From now on, each team is required to submit a weekly project progress report to the instructor by the end of the Wednesday lab session. The report should be typed up and should include
  - The team name and a list of team members' names
  - A list of activities that have done in the previous week and the names of the corresponding contributors
  - A list of activities that will be conducted next week

Team Lab Assignment #1

- Finalize the topic of a group project
  - Submit an one-page description of your project topic.
  - Make slides for presentation
- Due date
  - The beginning of the 2/7 lab session

Team Homework Assignment #2

- Study Spiral model, Unified Process (UP), Agile, and XP and prepare for presentation slides.
- Presentation slides should include, description, visual representation (figure), advantages and disadvantages of each process model
- Due date is by 1:00 pm on February 7th

Software Process

- A set of activities whose goal is the development or maintenance of software
- Process (life cycle) models and processes

A Specific Process for a Specific Project and Software Process Models

- A specific process is used for committing a specific project
  - Be familiar with the process models
  - Customize the process (models) for a project
  - Apply the process for the project with project management
  - IEEE 1074
- A software process (life cycle) model depicts the significant phases or activities of a software process from conception until the product is retired.

IEEE 1074

- The product of this standard is the SLCP that is required for a specific software project. The SLCP is based that is selected for the project based on the following:
  a. An SLCM that is selected for the project
  b. The Activities that are provided in Annex A
  c. The OPAs that are selected for the project

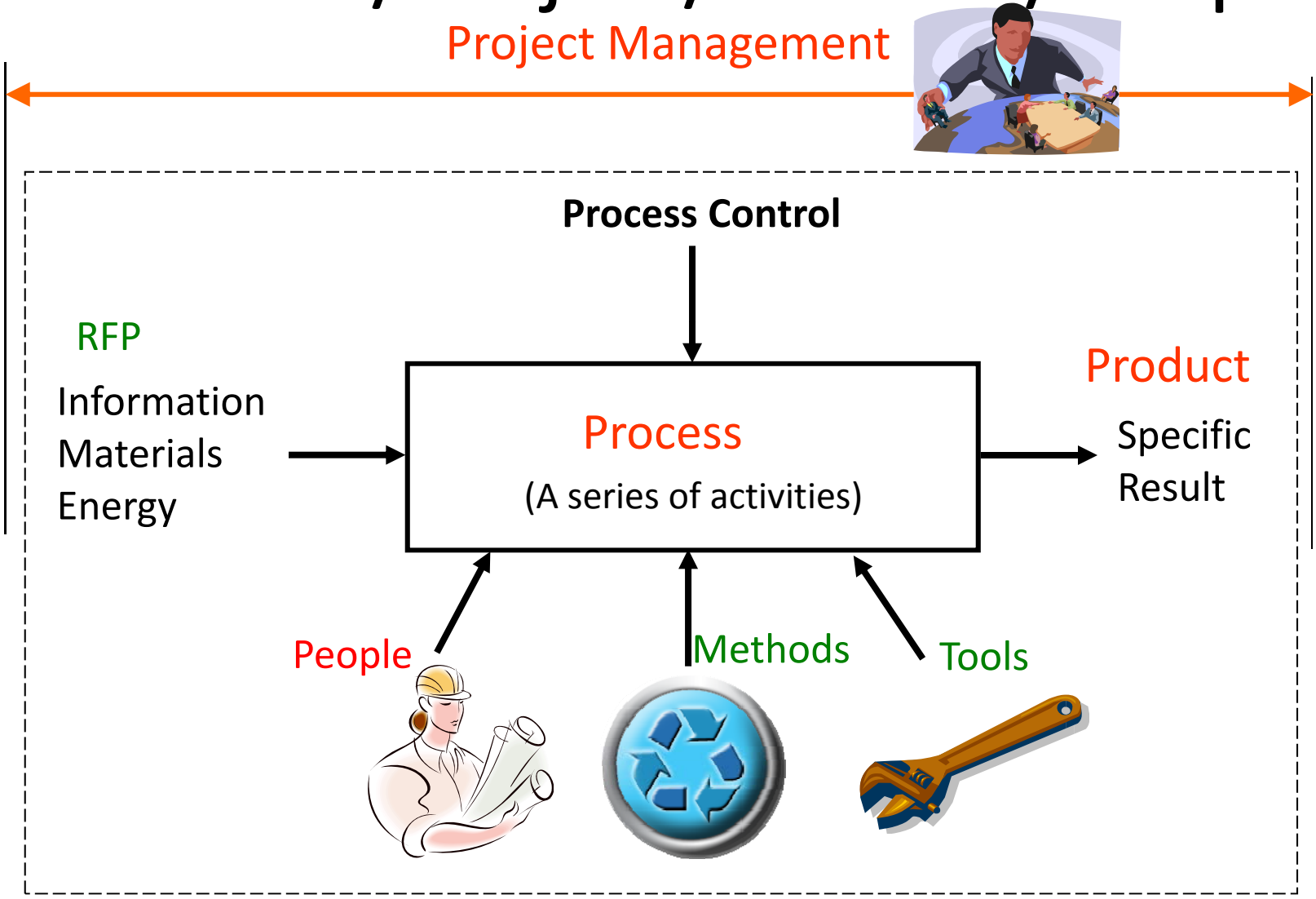
Possible Adverse Impact

- Process may be associated with overhead, unnecessary paperwork, longer schedules, etc.
- In other words, some people feel that process doesn’t add value, or worse, that it adversely affect the success of a project.

The Activities of Software Process

- Most software process models include a similar set of phases and activities
- The difference between models is the order and frequency of the phases
- The main phases of software process models
  - Requirements analysis, design, implementation, testing, maintenance

Process/Project/Product/People



Use Case

- Use cases are a way to capture system functionalities (i.e., functional requirements)
- Based on use case diagrams and their associated user case descriptions,
  - The rest of UML diagrams are developed.
  - The functions of software products are tested.
- Components
  - Diagrams
  - Descriptions

Use Case Diagrams / Descriptions

- Use case diagrams show use cases, actors and relations among them.
- Use case descriptions address in details what the system (software product) shall do for the actor to achieve a particular goal (functionality).

Use Case Development Process (1)

1. Find actors and use cases, and draw a draft of a use case diagram
   - GUI might be helpful for identifying interfaces between user(s) and the system, which initiate functions (use cases)

Use Case Development Process (2)

2. Refine iteratively a use case diagram by considering relationships between use cases and actors, and between use cases, and between actors
3. Develop each use case (starting with the priority ones) by creating its use description

Use Case Tutorial

- Hour 6, 7 and 19

Use Case Tutorial - Use Cases

- Represent a distinct functionality for a system
- Each use case must have a name describing the function
- Use an oval with the name of the use case

Use Case Tutorial - Actors

- A use case must be initiated by someone or something outside the scope of the use case
- An actor does not need to be a human user; any external system or element outside of the use case may trigger the use case
- An actor can be shown with a stick figure with the name of the actor written near the icon

Use Case Tutorial - Relationships (1)

- An actor is associated with one or more use cases
- A relationship between an actor and a use case indicates the actor initiates the use case, the use case provides the actor with results

Use Case Tutorial - Relationships (2)

- An association is shown as a solid line between an actor and a use case
- Other types of relationships
  - Actor and use case ***generalization***
  - Use case ***include***
  - Use case ***extend***

Use Case Descriptions (1)

- Use case name with a use case ID
- Characteristic information (goal, pre-condition, successful end condition, primary actors)
- Main (primary) scenario (“normal” messages flows between an actor and a use case)

Use Case Descriptions (2)

- Alternative scenario (“exceptional” or “conditional” workflows between an actor and a use case)
- Utilizing other use cases, if necessary

Use Case Description (1)

UC1: Startup

Characteristic Information

Goal	Power-up and initialize the ATM
Pre-Condition	ATM must be in the OFF mode
Success End Condition	ATM is powered up and has been initialized
Primary Actor	Operator

Use Case Description (2)

Primary Scenario

Step	Actor/System	Action Description
1	User	Push the power on button
2	ATM	Perform a self-test
3	ATM	Set the ATM in IDLE mode
4	ATM	Run the clock

Alternative Scenario

Step	Condition	Action Description
2a	self-test fails	Set an alarm and notify the operator to correct the problem
3a	Mode setting failure	Set an alarm and notify the operator to correct the problem
4a	Clock failure	Set an alarm and notify the operator to correct the problem

UML Drawing Tools

- Pacestar UML Diagrammer
- Microsoft Office Visio
- NetBeans 6.0+
- Visual Paradigm
- Others including open source software