LEC # 01 & 02

SOFTWARE ENGINEERING

Waris Ali

Objectives

- Become enable to evaluate software engineering techniques and approaches.
- Become enable to exercise professional judgment in selecting an approach for a particular project based on an understanding of:
  - How the present state of software practice came about
  - What was tried in the past
  - What worked and what did not work
  - Why

Course Policies, Schedule, Syllabus

- Lectures are electronic and will be available after class.
- The quizzes will be announced and normally last for ten minutes.
- It will be the instructor's will to choose the number of quizzes for evaluations purposes.

Assignment work

- All work submitted by a group must be solely the work of that group
- All work submitted by an individual must solely be the work of that individual
  - This is not to mean that you may not consult with others, but:
    If you receive any help, you must specifically acknowledge that person in your submitted work
  - If any student or group of students submits work which is not their own, they will be disciplined according to the University and Department policies

Grading Criteria

Quizzes	10 %
Assignments	5-10 %
Mid Term Exam	30 %
End Semester Exam	45-50 %
Project	5-10%

Recommended Reading

- There is no prescribed text. The following books cover the basic material in the unit:
  - Pressman, R., Software Engineering: A Practitioner's Approach
  - Bernd Duit, Software Engineering using UML, Patterns and java
- The lecture slides are long and detailed—the intention is to give you the material you will need
- A list of further useful books is provided in the course outline.

Contact Information

- Instructor: Waris Ali
- Email: a.waris@gmail.com

- Counseling hours: to be announced later
- Class group: Whatsapp Group

Engineering

-- Application of systematic, disciplined, quantifiable approach to some process

Software

-- Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system

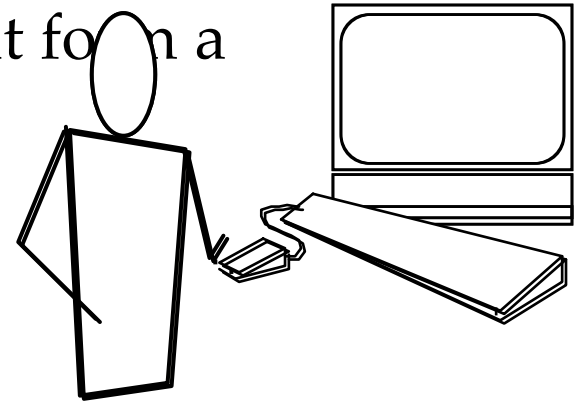
Software's Dual Role

- Software is a product
  - Delivers computing potential
  - Produces, manages, acquires, modifies, displays, or transmits information

- Software is a vehicle for delivering a product
  - Supports or directly provides system functionality
  - Controls other programs (e.g., an operating system)
  - Effects communications (e.g., networking software)
  - Helps build other software (e.g., software tools)

What is Software?

- Software is a set of items or objects that form a "configuration" that includes
  - Programs
  - Documents
  - Data ...



What is Software?

- software is engineered
- software doesn't wear out
- software is complex

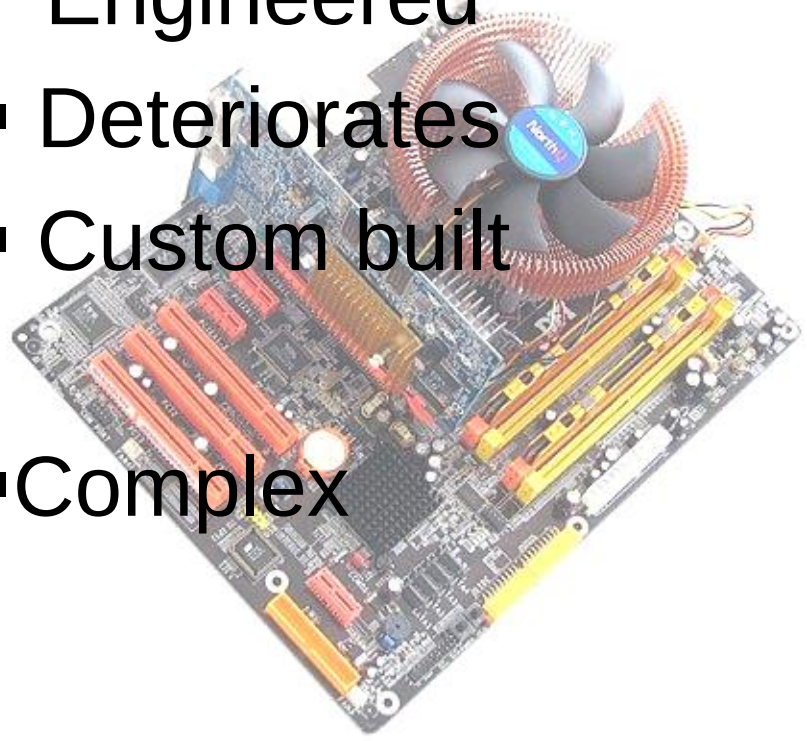
Hardware vs. Software

Hardware

- Manufactured
- Wears out
- Built using components
- Relatively simple

Software

- Developed/ Engineered
- Deteriorates
- Custom built
- Complex



Software Nature

- **Software is intangible**
  - Hard to understand development effort
- **Software is easy to reproduce**
  - Cost is in its *development*
  - In other engineering products, manufacturing is the costly stage
- **The industry is labor-intensive**
  - Hard to automate

Software Nature

- **Untrained people can produce something together**
  - Quality problems are hard to notice
- **Software is easy to modify**
  - People make changes without fully understanding it
- **Software does not 'wear out'**
  - It deteriorates by having its design changed:
    - erroneously, or
    - in ways that were not anticipated, thus making it complex

Types of Software

- **Custom**
  - For a specific customer
- **Generic**
  - Sold on open market
  - Often called
    - COTS (Commercial Off The Shelf)
    - Shrink-wrapped
- **Embedded**
  - Built into hardware
  - Hard to change

Types of Software /2

- **Differences among custom, generic and embedded software**

	Custom	Generic	Embedded
Number of *copies* in use	low	medium	high
processing power devoted to running this type of software	high	medium	low
Worldwide annual *development effort*	high	Medium	low

Types of Software /3

- **Real time software**
  - E.g. control and monitoring systems
  - Must react immediately
  - Safety often a concern
- **Data processing software**
  - Used to run businesses
  - Accuracy and security of data are key
- ***Some software has both aspects***

PROBLEMS — SOFTWARE DEVELOPMENT

1. **schedule and cost estimates often grossly inaccurate**
   - over schedule
     - 16% in time
   - over budget
2. **productivity of software developers has not kept pace with demand for their services**
3. **quality of software is sometimes less than acceptable**
   - unreliable — Ariane 5 rocket
   - unsafe — London Ambulance System; Therac-25
   - inflexible — hard to change/maintain
   - abandoned — London Stock Exchange

PROBLEMS — SOFTWARE DEVELOPMENT

Ariane 501 whose maiden flight on June 4, 1996 ended in the launcher being exploded because of a chain of software failures “The failure of the Ariane 501 was caused by the complete loss of guidance and attitude information 37 seconds after start of the main engine ignition sequence (30 seconds after lift- off). This loss of information was due to specification and design errors in the software of the inertial reference system.”

London Ambulance System where because of a succession of software engineering failures, especially defects in project management, a system was introduced that failed twice in the autumn of 1992. Although the monetary cost, at “only’ about £9m, was small by comparison with other examples, it is believed that people died who would not have died if ambulances had reached them as promptly as they would have done without this software failure.

Therac-25 where between 1985 and 1987 six people (at least) suffered serious radiation overdoses because of software-related malfunctions of the Therac-25 radiation therapy machine. Three of them are thought to have died of the overdoses. An important root cause was a lack of quality assurance, which led to an over-complex, inadequately tested, under-documented system being developed, and subsequently to the failure to take adequate corrective action.

Taurus a planned automated transaction settlement system for the London Stock Exchange, The project was canceled in 1993 after having lasted more than five years. The project cost was around £75m; the estimated loss to customers was around £450m; and the damage to the reputation of the London Stock Exchange was incalculable.

PROBLEMS — SOFTWARE DEVELOPMENT

- lack of historical data on software development process
- no good way to measure productivity
- poor communication with customer/other developers
- no systematic approach to software development
- vague/unclear specification of customer requirements
- lack of systematic and complete software testing
- little emphasis on software maintenance

What is Software Engineering?

- Many Definitions
  - "… the establishment and use of sound engineering principles in order to obtain economical software that is reliable and works efficiently on real machines." (Bauer 1969)
  - "The application of science and mathematics by which the capabilities of computer equipment are made useful to man via computer programs, procedures, and associated documentation." (Boehm 1981)
  - Designing, building and maintaining large software systems in a cost-effective way.

Software Engineering

-- The application of systematic, disciplined, quantifiable approaches to software development, operation and maintenance.*

*** - IEEE Standard 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, IEEE Standards Collection Software Engineering, IEEE (1997).**

Why learn Software Engineering?

- It is a means to an end, not an end in itself!

  "You lift weights so that you can knock over a defensive lineman, or carry your groceries or lift your grandchildren without being sore the next day. You do math exercises so that you can improve your ability to think logically, so that you can be a better lawyer, doctor, architect, prison warden or parent. MATH IS MENTAL WEIGHT TRAINING. It is a means to an end, (for most people), not an end in itself."

  "Made to Stick"

  Chip Heath and Dan Heath

- Similarly you are not necessarily learning software engineering to engineer softwares. By the end of this course, you will be trained on how to carry out a project successfully, how to communicate with the stakeholders & how to transform this communication into powerful models which will lead to a winning product. Moreover you will also learn how to test these models & how to evaluate risks at every stage of the development process.

Why bother with Software Engineering?

- Many very successful projects don't use software engineering, e.g.
  - early Microsoft
  - ID Software's Doom
  - Sausage's Hotdog

  BUT they are often not repeatable
- Many more projects fail because they don't use software engineering. Failures occur because:
  - of the size of the project relative to previous efforts
  - key personnel have left
  - of failure to understand requirements
  - the project delivers, but lacks the required quality
  - of the introduction of new technology
  - of many, many other reasons

WHAT IS SOFTWARE ENGINEERING

What are your thoughts ?

WHAT IS SOFTWARE ENGINEERING?

- Software engineering is an engineering discipline which is concerned with all aspects of software production
- Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available

WHAT IS SOFTWARE ENGINEERING?

"The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines." — Frtiz Bauer

"... multi-person construction of multi-version software." — Dave Parnas

- engineering principles -> disciplined effort: methodology, tools
- economically…reliable…efficiently -> built-in quality: metrics
- multi-person -> team effort: management, training
- multi-version -> not a "one-shot" effort: documentation, maintenance

WHAT IS SOFTWARE ENGINEERING?

- Modeling
- Problem solving
- Knowledge acquisition
- Rational driven

Problem Solving

- Engineering is problem solving activity
- Engineering methods include:
  - Formulate the problem
  - Analyze the problem
  - Search for solution
  - Decide on the appropriate solution
  - Specify the solution

Problem Solving

- Software development:
  - Requirements elicitation & analysis: Problem formulation and Analysis
  - Search for Solutions, decide on the appropriate solution: System Design; Object Design
    - Breaking the system into pieces
    - Selecting the general Strategies
  - Specify the solution: Implementation

Software Engineering Concepts

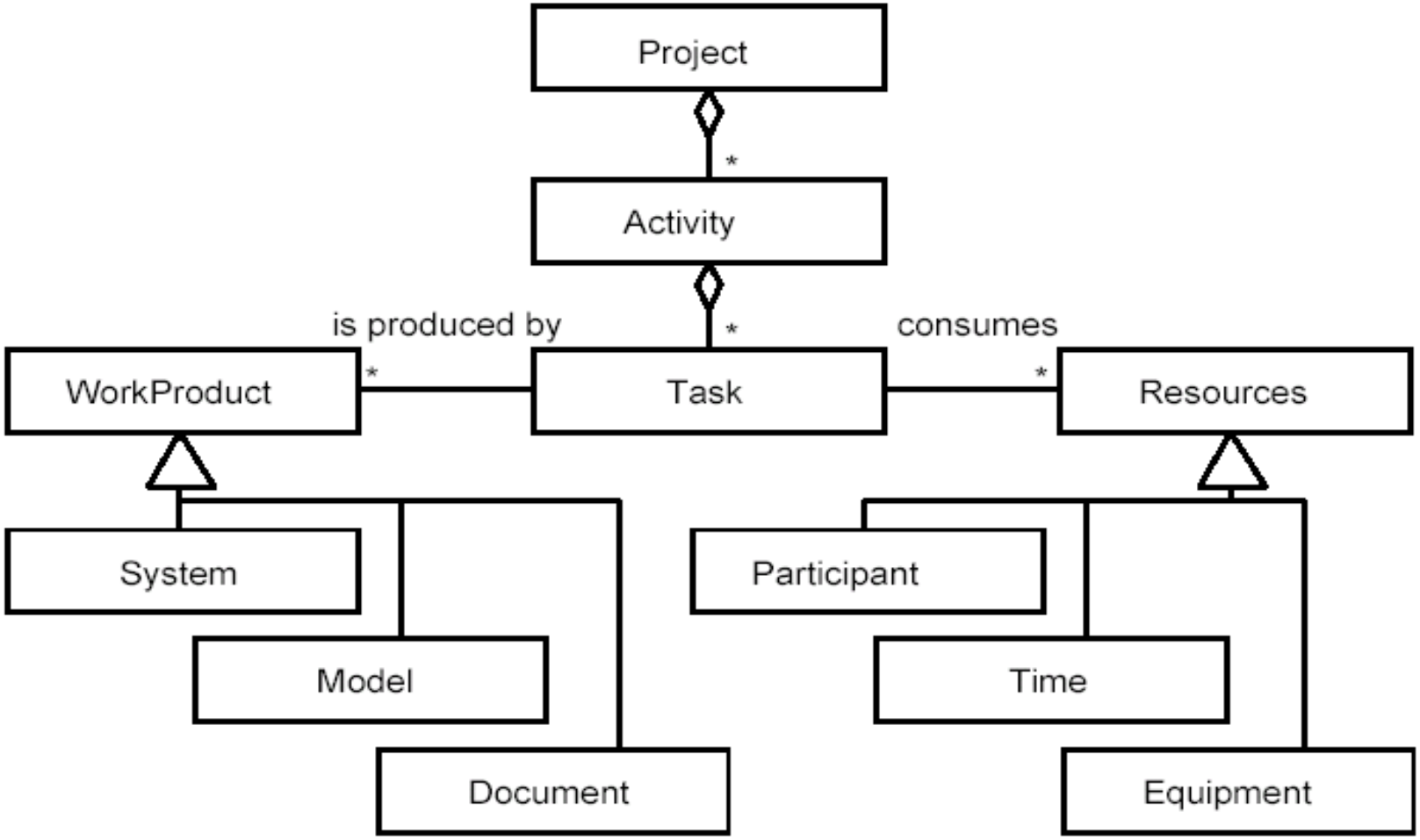
- **Project:** Purpose is to develop software system
- **Activities:** Activity is a set of task that is performed towards a specific purpose e.g. Requirement Elicitation; Activities sometimes also called **Phases**
- **Tasks:** An Atomic unit of work that can be managed
- **Resources:** Assets that are used to accomplish work Participant, Time, Equipment;
- **Work Product:** System, Model or a Document;
- **Participants:** Stakeholders
- **Roles:** Set of responsibilities associated with a set of task and is assigned to a participant
- **System & Models:** Reality; Representation of Reality

Software Engineering Concepts

- **Work Products:** An Artifact Produced during development of the system e.g.
  - Documents, Code
  - Internal Work Product
  - Deliverable
- **Goal:** High level principle that is used to guide the project e.g. safety for space shuttle guidance software development
- **Requirements:** Features that a system must have
  - Functional Requirement: Functionality that a system must support
  - Non-Functional Requirement: Constraint on the operation of the system
- **Notation :** Graphical of textual representation (e.g. UML)
- **Method :** Repeatable technique for solving problems.
- **Methodology:** Collection of methods.

Software Engineering Concepts

Software engineering concepts, depicted as a UML class diagram.



Software Engineering Development Activities

- Project definition
- Requirements Analysis
- System design
- Object design
- Implementation
- Testing

Process Activities (1)

- **Project Definition**
  - States the purpose of the project
  - Makes initial decision on political and technical feasibility of the project
- **Requirements Analysis**
  - High level definition of the functionality of the system, primarily from the point of view of the users
- **Design**
  - Looks at the software requirements of the system and the architecture of the system
  - Lower level design activities - data structures, interface representations, procedural (algorithmic) details

Process Activities (2)

- **Implementation**
  - Writing or generating the code to build the system
- **Testing**
  - Component Testing
    - Testing of the individual components while they are being built and after they have been completed
  - Integration Testing
    - Testing of the way individual components fit together
  - System Testing
    - Testing of the whole system usually in concert with the users (acceptance testing)

Process Activities (3)

- **System Delivery**
  - Implementation of the system into the working environment and replacement of the existing system
- **Maintenance**
  - Corrective
  - Adaptive
  - Perfective

WHAT IS A SOFTWARE PROCESS MODEL?

- A simplified representation of a software process, presented from a specific perspective

- Generic process models (examples)
  - Waterfall
  - Evolutionary development
  - Integration from reusable components