Advanced Software Engineering

CS-511

Fast-National University of computer & Emerging Sciences

Who am I?

Dr. Khubaib Amjad Alam

- Assistant Professor Fast-NU
- Ph.D (University of Malaya, Kuala Lumpur, Malaysia)
- MS (Comsats Institute of Information Technology, Pakistan)
- Orginated from Wah Cantt (Ancient town of Taxila near Islamabad)

Course objectives

- To further develop your understanding of the concepts and methods required for the construction of large software intensive systems. It aims to develop a broad understanding of the discipline of software engineering.
- Equip the students to conduct and evaluate software engineering research
  - Identifying software engineering research problems
  - Conduct comprehensive literature reviews
- To familiarize students with the techniques and approaches to develop better software. Evaluating approaches for a particular context.
- To develop students' skills for planning and managing real life software projects successfully

Learning outcomes

- Understand the issues in software development
- Familiarize with the current software engineering practices in industry
- Understand the state of the art of research in software engineering
- Understand the professional and technical literature on software engineering.
- Understand the advance topics in software engineering

Agenda

- What is software engineering?
- Why software engineering?
- What is software process?
- Software process models (i.e. waterfall, spiral, agile)
- Fundamental activities and sub-activities of software process

Software Engineering

Software engineering: The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software

Software Engineering is a branch of systems engineering concerned with the development of large and complex software intensive systems. It focuses on:

- – the real-world goals for, services provided by, and constraints on such systems,
- – the precise specification of systems structure and behavior, and the implementations of these specifications,
- – the activities required in order to develop an assurance that the specifications and real world-world goals have been met,
- – the evolution of these systems over time, and across systems families,
- – It is also concerned with the processes, methods and tools for the development of software intensive systems in an economic and timely manner.

Why software engineering?

- The number, size, and application domains of computer programs have grown dramatically; hundreds of billions are being spent on software development.
- There are serious problems in the cost, timeliness, and quality of many software products. The reasons for these problems are many and include the following:

1. Software products are among most complex of man-made systems, and software by its very nature has intrinsic, essential properties (e.g., complexity, invisibility, and changeability) that are not easily addressed [Brooks 95].
2. Programming techniques and processes that worked effectively for an individual or a small team to develop modest-sized programs do not scale-up well to the development of large, complex systems (i.e., systems with millions of lines of code, requiring years of work, by hundreds of software developers).
3. The pace of change in computer and software technology drives the demand for new and evolved software products. This situation has created customer expectations and competitive forces that strain our ability to produce quality of software within acceptable development schedules."

Software Myths – Management Myths [1/2]

- We already have a book that's full of standards and procedures for building software. Won't that provide my people with everything they need to know?
  - The book of standards may very well exist, but is it used?
    - Are software practitioners aware of its existence?
    - Does it reflect modern software engineering practice?
    - Is it complete? Is it adaptable?
    - Is it streamlined to improve time to delivery while still maintaining a focus on Quality?

    In many cases, the answer to these entire question is no.

Management Myths [2/2]

- If we get behind schedule, we can add more programmers and catch up.
  - Not a manufacturing process
  - Need to trained people
  - Communication problems
- If I decide to outsource the software project to a third party, I can just relax and left that firm build it.
  - Organization needs good management skill
  - If an organization does not understand how to manage and control software project internally, it will invariably struggle when it out sources software project.

Customer Myths

- A general statement of objectives is sufficient to begin writing programs – we can fill in the details later.
  - Ambiguous requirements: recipe of disaster
  - Effective and continuous communication
- Software requirements continually change, but change can be easily accommodated because software is flexible.
  - Time of change is important

Practitioner’s Myths [1/2]

- Once we write the program and get it to work, our job is done.
  - 60-80% of all effort expended after software delivery
- Until I get the program "running" I have no way of assessing its quality.
  - SQA practices from very beginning

Practitioner's Myths [2/2]

- The only deliverable work product for a successful project is the working program.
  - Other work products
- Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.
  - It is not about documents only
  - Creating quality products
  - Ultimately reduced work and faster delivery time

- What is your understanding of a “Software Process”?
- Have you used any “Software Process Model” in your practice?

Software process and Process model

A structured set of activities required to develop a software system

- – Specification;
- – Design;
- – Validation;
- – Evolution.

A software process model is an abstract or simplistic representation of a process. It presents a description of a process from some particular perspective.

Four fundamental activities of software Process

- Any software process must include the following four activities:
- **Software specification** (or requirements engineering): Defining what the software should do. Main functionalities of the software and the constrains around them are specified.
- S**oftware design and implementation**: Defining the software and data organization and implementing the system. The software is to be designed and programmed in this activity.
- **Software verification and validation**: Testing the system for bugs and conformance to requirements. The software must conforms to it's specification and meets the customer needs.
- **Software evolution** (software maintenance): Changing the system after it has gone into use. The software is being modified to meet customer and market requirements changes.

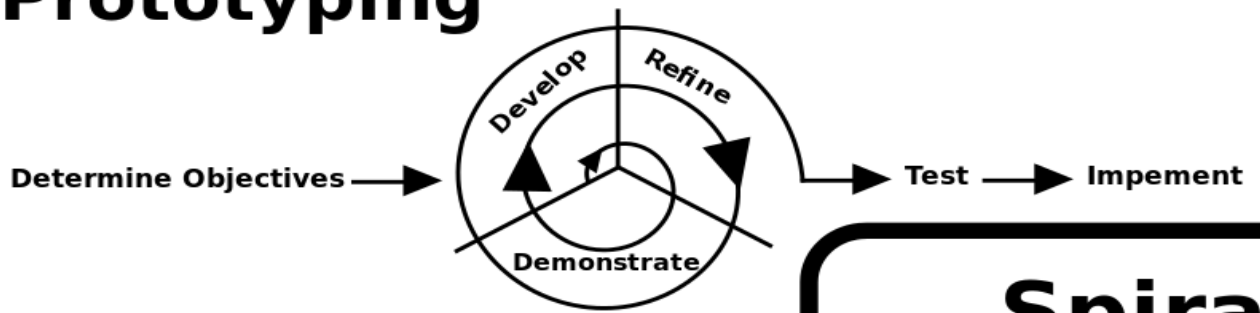
Organization of software process activities

- These four fundamental activities of specification, design & implementation, Validation and evolution are organized differently in different development processes.
- For example in waterfall model, they are organized in sequence, whereas in incremental development (agile processes) they are interleaved .
- In practice, they include ***sub-activities*** such as *requirements validation, architectural design, unit testing*, …etc.
- There are also ***supporting activities*** such as configuration and change management, quality assurance, project management, user experience.

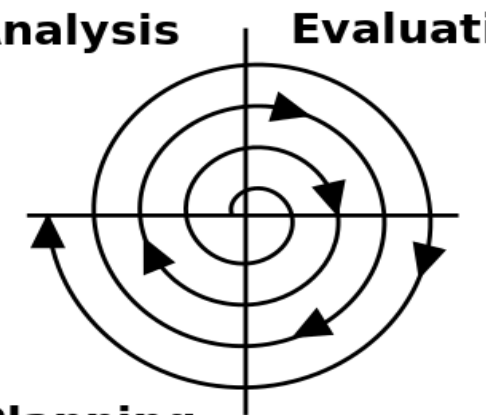
What constitutes a software process?

- When we talk about a process, we usually talk about the activities in it. However, a process also includes the process description, which includes:
- **Products**: The outcomes of the an activity. For example, the outcome of architectural design maybe a model for the software architecture.
- **Roles**: The responsibilities of the people involved in the process. For example, the project manager, programmer, etc.
- **Pre and post conditions**: The conditions that must be true before and after an activity. For example, the pre condition of the architectural design is the requirements have been approved by the customer, while the post condition is the diagrams describing the architectural have been reviewed.

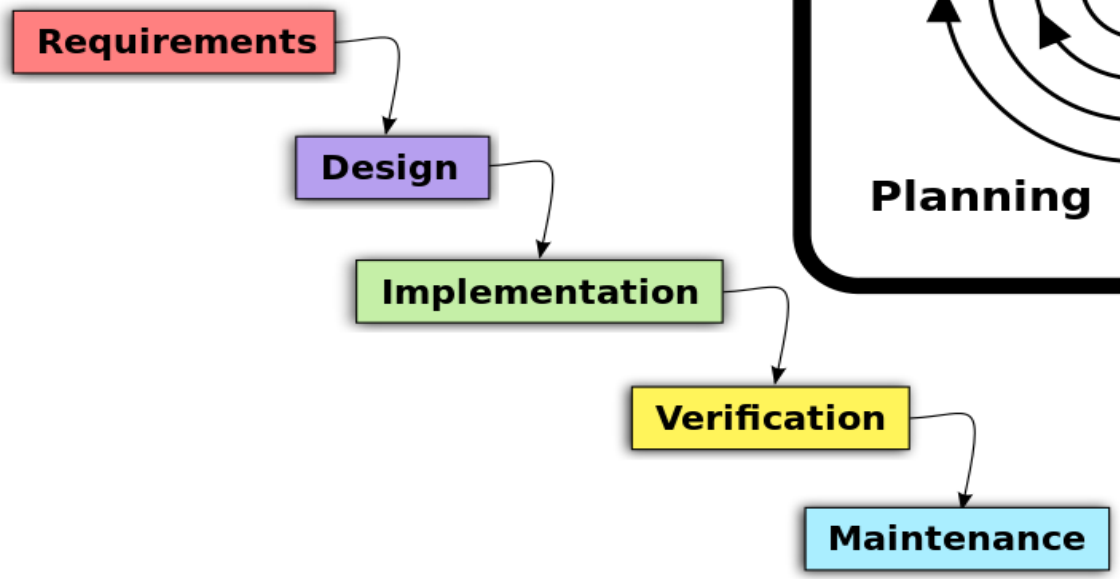
Prototyping



Spiral



Waterfall



- **Waterfall Model:** a sequential approach, where each fundamental activity of a process represented as a separate phase, arranged in linear order.
- **Prototyping:** a version of a system or part developed quickly to check the customer's requirements or feasibility of some design decisions. In prototyping, the client is involved throughout the development process.
- **Incremental Development:** developing an initial implementation, exposing this to user feedback, and evolving it through several versions until an acceptable system has been developed.
- **Spiral Model:** a risk-driven where the process is represented as spiral rather than a sequence of activities. best features from the waterfall and prototyping models, and introduces a new component; risk-assessment. Each loop represents a phase.
- **Iterative Development** Iterative development model aims to develop a system through building small portions of all the features, across all components.

Agile methods

Agility is flexibility, it is a state of dynamic, adapted to the specific circumstances.

The agile methods refers to a group of software development models based on the incremental and iterative approach, in which the increments are small and typically, new releases of the system are created and made available to customers every few weeks.

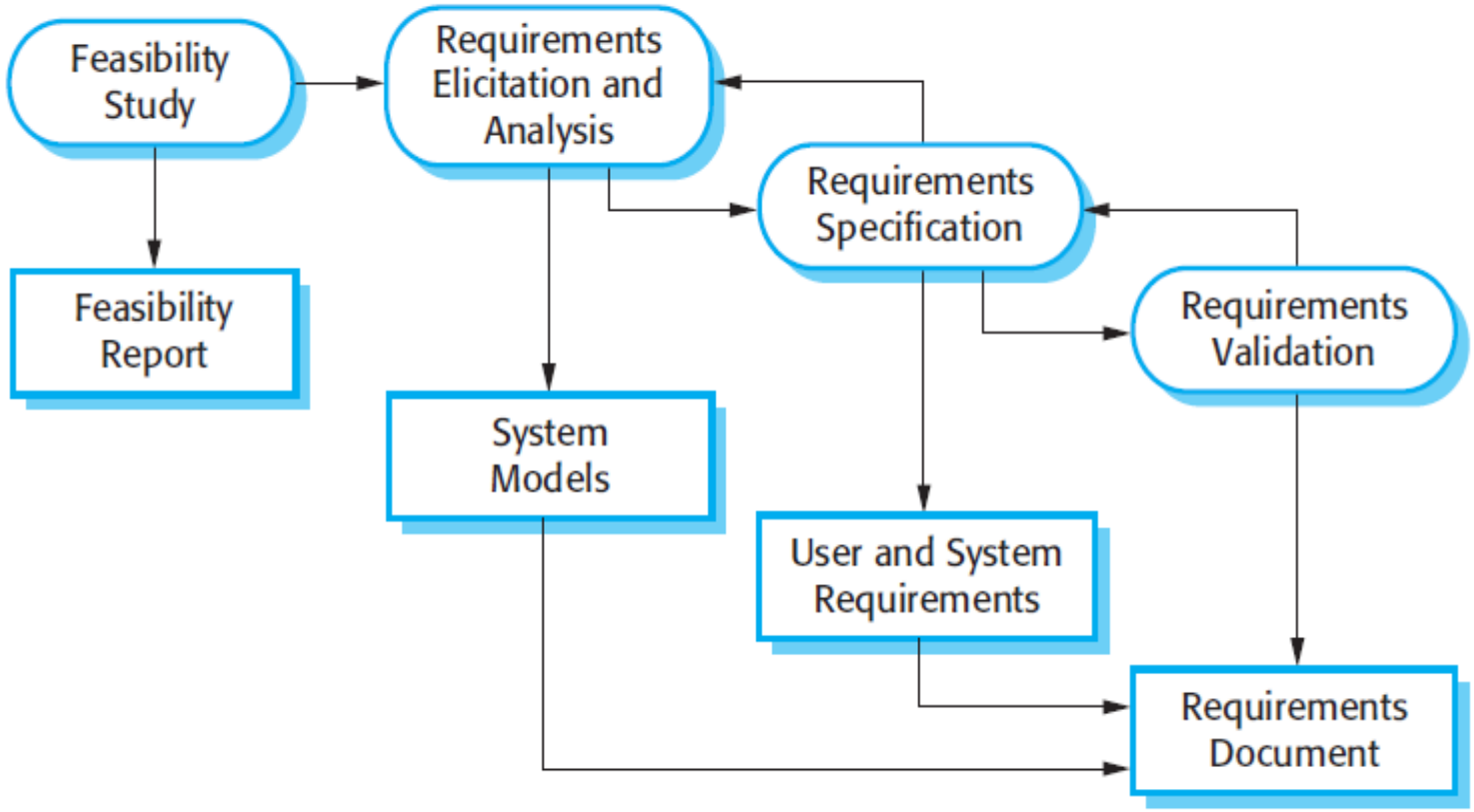
- involve customers in the development process to propose requirements changes. They minimize documentation by using informal communications rather than formal meetings with written documents.
- best suited when requirements change rapidly during the development.
- There are a number of different agile methods available such as: *Scrum, Crystal, Agile Modeling (AM), Extreme Programming (XP)*, etc.

Increment Vs Iterative Vs Agile

- Each increment in the incremental approach builds a complete feature of the software, while in iterative, it builds small portions of all the features.
- An agile approach combines the incremental and iterative approach by building a small portion of each feature, one by one, and then both gradually adding features and increasing their completeness.

Requirement Engineering

- the process of understanding and defining what services are required and identifying the constraints on these services.



1. **Feasibility study:** An estimate is made of whether the identified can be achieved using the current software and hardware technologies, under the current budget, etc. whether or not to go ahead with the project.
2. **Requirements elicitation and analysis:** This is the process of deriving the system requirements through observation of existing systems, discussions with stakeholders, etc. This may involve the development of one or more system models and prototypes that can help us understanding the system to be specified.
3. **Requirements specification**: It's the activity of writing down the information gathered during the elicitation and analysis activity into a document that defines a set of requirements. Two types of requirements may be included in this document; user and system requirements.
4. **Requirements validation:** It's the process of checking the requirements for realism, consistency and completeness. During this process, our goal is to discover errors in the requirements document. Errors must be rectified.

Of course, the activities in the requirements process are not simply executed in a strict sequence, but, they are interleaved. For example, analysis activity continues during the specification as new requirements come to light.

In agile methods, requirements are developed incrementally according to user priorities and the elicitation of requirements comes from users who are part of the development team.

Software Design And Implementation

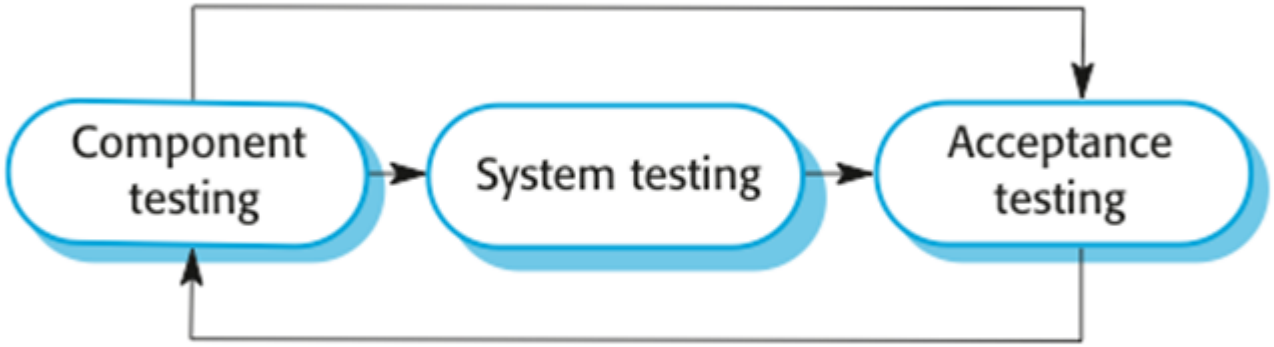
process of converting specification into an executable system. A software design is a description of the structure of the software to be implemented, data models, interfaces between system components, and maybe the algorithms used.

- **Architectural design:** defines the overall structure of the system, the main components, their relationships.
- **Interface design:** It defines the interfaces between these components. The interface specification must be clear.
- **Component design:** Take each component and design how it will operate.
- **Database design:** The system data structures are designed and their representation in a database is defined.

Software Verification And Validation

intended to show that a system both conforms to its specification and that it meets the expectations of the customer.

- Validation may also involve checking processes, such as inspections or reviews at each stage of the software process, from defining the requirements till the software development.
- Testing is an important validation technique.



- **Development (or component) testing:** The components (functions or object classes) making up the system are tested independently, without other system components. Automation tools i.e. **JUnit** commonly used.

- **System testing:** System components are integrated to create a complete system. This process is concerned with finding errors that result from interactions between components. It is also concerned with showing that the system meets its functional and non-functional requirements.

- **Acceptance testing:** This is the final stage in the testing process before the system is accepted for operational use. The system is tested with data supplied by the system customer rather than using simulated test data. It may reveal errors in the system requirements definition.

Software Maintenance and evolution

- Requirements are always changing, even after the system has been put into its operating environment.
- Historically, a split between software development process and evolution process (software maintenance).
- However, this distinction is increasingly irrelevant. It is more realistic to think of software engineering as an evolutionary process where software is continually changed over its lifetime in response to changing requirements and customer needs.

Recommended Resources

- Text books
  - I. Sommerville, Software Engineering, 9th Edition, Pearson Education, 2011.
  - R. S. Pressman, Software Engineering: A Practitioner's Approach, 7th Edition, McGraw Hill Education, 2010.