Computer Sciences
University of Technology

Date: / / 2012
Time: 3 hours
Lecturer: Anmar Ali

Final Exam. 2011-2012
Term

Subject: Software Engineering
Class: 3rd
Branch: Software

Notes:

1. **Answer only four questions.**
2. **Question No. two must be answered.**

Q1. Define the following:

Computer Software, Risk identification, Software project scheduling, Software reliability, Compartmentalization.

(12.5 Marks)

Q2. Explain in detail the Spiral model; support your answer with a neat diagram.

(12.5 Marks)

Q3. Answer both:

- What are the attributes of "good" software? Explain them.
- List metrics guidelines? **(Only four)**

(12.5 Marks)

Q4. Explain software prototyping? What are the various prototyping methods and tools?

(12.5 Marks)

Q5. State the difference between:

- Proactive strategies::Reactive strategies.
- Process indicators::Project indicators.
- Software characteristics::Hardware characteristics.

(12.5 Marks)

Good Luck

Solutions 3rd class first trail

Q1.

Computer software: It is the product that software engineers design and build. It encompasses programs that execute within a computer of any size and architecture, documents that encompass hard-copy and virtual forms, and data that combine numbers and text but also includes representations of pictorial, video, and audio information.

Risk identification: Is a systematic attempt to specify threats to the project plan (estimates, schedule, resource loading, etc.).

Software project scheduling: Is an activity that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering tasks.

Compartmentalization: The product and process must be decomposed into a manageable number of activities and tasks.

Software reliability: Defined as the probability of failure-free operation of a computer program in a specified environment for a specified time period.

Q2. The Spiral model

Spiral model, originally proposed by Boehm, is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model. It provides the potential for rapid development of incremental versions of the software. Using the spiral model, software is developed in a series of incremental releases. During early iterations, the incremental release might be a paper model or prototype. During later iterations, increasingly more complete versions of the engineered system are produced.

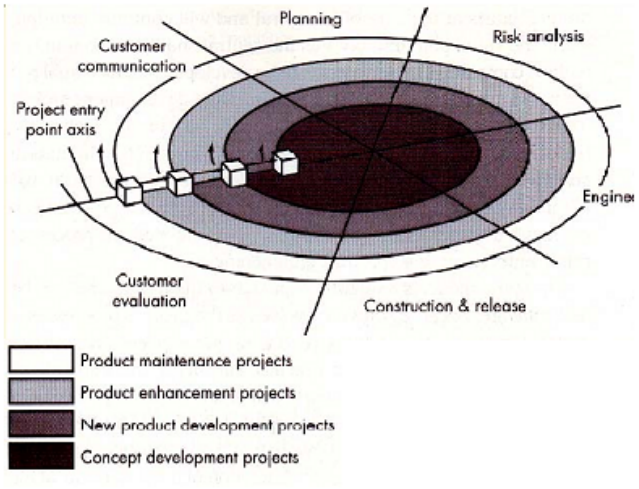


Figure (2.5): The Spiral model

A spiral model is divided into a number of framework activities, also called task regions. Figure 2.5 depicts a spiral model that contains six task regions:

1. **Customer communication-tasks** required to establish effective communication between developer and customer.
2. **Planning-tasks** required to define resources, timelines, and other project-related information.
3. **Risk analysis-tasks** required to assess both technical and management risks.
4. **Engineering-tasks** required to build one or more representations of the application.
5. **Construction and release-tasks** required to construct, test, install, and provide user support (e.g., documentation and training).
6. **Customer evaluation-tasks** required to obtain customer feedback based on evaluation of the software representations created during the engineering stage and implemented during the installation stage.

As this evolutionary process begins, the software engineering team moves around the spiral in a clockwise direction, beginning at the center. The first circuit around the spiral might result in the development of a product specification; subsequent passes around the spiral might be used to develop a prototype and then progressively more sophisticated versions of the software. Each pass through the planning region results in adjustments to the project plan. Cost and schedule are adjusted based on feedback derived from customer evaluation. In addition, the project manager adjusts the planned number of iterations required to complete the software. Unlike classical process models that end when software is delivered, the spiral model can be adapted to apply throughout the life of the computer software. An alternative view of the spiral model can be considered by examining the project entry point axis, also shown in Figure 2.8. Each cube placed along the axis can be used to represent the starting point for different types of projects.

But like other paradigms, the spiral model is not a panacea, for many reasons:

1. It may be difficult to convince customers (particularly in contract situations) that the evolutionary approach is controllable.

2. It demands considerable risk assessment expertise and relies on this expertise for success. If a major risk is not uncovered and managed, problems will undoubtedly occur.

3. The model has not been used as widely as the linear sequential or prototyping paradigms. It will take a number of years before efficacy of this important paradigm can be determined with absolute certainty.

Q3.

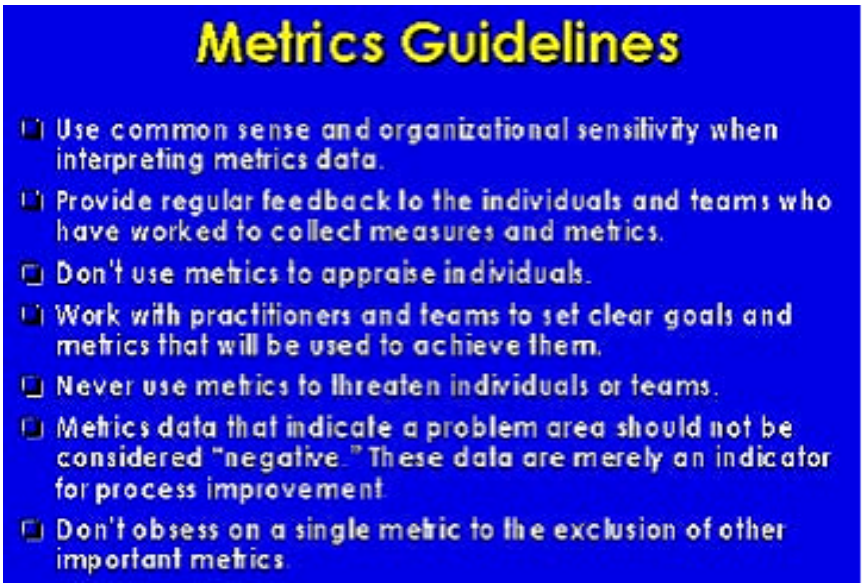
a) 1- Maintainability: software should be written in such a way that it may evolve to meet the changing needs of customer. This is critical attribute because software change is an inevitable

   2- Dependability: software dependability has a range of characteristics, including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure.

3- efficiency: software should not make wasteful use of system resources, such as memory and processor cycles. Therefore efficiency includes responsiveness, processing time, memory utilization etc…

4- Usability: software must be usable, without under effort by the type of user for whom it is designed. This means that it should have an appropriate user interface and adequate documentation.

b)



Q4.

Software prototyping: Throwaway prototyping (prototype only used as a demonstration of product requirements, finished software is engineered using another paradigm) Evolutionary prototyping (prototype is refined to build the finished system) Customer resources must be committed to evaluation and refinement of the prototype. Customer must be capable of making requirements decisions in a timely manner.

Prototyping Methods and Tools

- Fourth generation techniques (4 GT tools allow software engineer to generate executable code quickly).
- Reusable software components (assembling prototype from a set of existing software components).
- Formal specification and prototyping environments (can interactively create executable programs from software specification models).

Q5.

- **Reactive strategies** - very common, also known as *fire fighting mode*, project team sets resources aside to deal with problems and does nothing until a risk becomes a problem.

  Proactive strategies - risk management begins long before technical work starts, risks are identified and prioritized/ranked by importance, then team builds a plan to avoid risks if they can or minimize them if the risks turn into problems (Not all risks can be avoided) .

- **Process indicators:** enables S/W Engineering organization to gain insight into the efficacy of an existing process (SE task, Paradigm …). They enable managers to assess what works and doesn't.

  Project indicators: enable S/W project managers to: assess project status, track potential risks, detect problem area early, adjust workflow or task, and evaluate team ability to control product quality.

- Software is developed or engineered, not manufactured in the classical scene.
  Software doesn't "wear out", while hardware do "wear out".