Study Guide for Test 2

**Cost Estimation**

1. **Types of cost**
   1. Time – from start date to end date of project. Aka *elapsed time*.
   2. Effort – explained below
2. **What is effort (in S.Engineering)** – Effort is also called development effort. Amount of labor needed for a project expressed inperson-months mostly.
3. **How to convert efforts to money** 
   1. Findaverage salaryin company
   2. Calculate weighted average cost by multiplying the average salary by a weight, determined by expenses on employees (usually between 2 and 3)
   3. Cost = effort \* weighted average cost
4. **COCOMO models – *Constructive Cost* Model. Considers complexity of project.**
   1. **Types of projects(organic semidetatched, embedded)**
      1. Organic – Relatively small SW team develop SW in a familiar, in-house environment. People in the project have extensive experiences working with related systems within the organization, and have thorough understanding of how the system under development will contribute to the organizations objectives. Very few organic-mode projects have developed products with more than 50 thousand delivered source instructions (KDSI).
      2. Semidetached – Intermediate between organic and embedded. Can mean intermediate level of project characteristic, OR intermediate of organic and embedded mode characteristics. The size range generally extends up to 300 KDSI
      3. Embedded – Distinguished by a need to operate within tight constraints. Must operate within (embedded in) a strongly coupled complex of software, regulations, and operational procedures, such as electronic funds transfer system or an air traffic control system.
   2. **Basic,intermediate, and detailed COCOMO models (features, advantages and disadvantages**
      1. Basic – Only uses a single variable (size in DSI) and three development modes. Good for quick, early, rough order of magnitude estimate of SW cost, but accuracy is limited because doesn’t include other factors that have large influence on SW costs.
      2. Intermediate – Estimates SW effort by using 15 cost driver variables besides the KDSI used in basic. Applied across entire SW product for easy and rough cost estimation during early stage, or be applied at the SW product component level for more accurate cost estimation in more detailed stages.
         1. Uses Effort Adjustment Factor (EAF) and slightly different coefficients for effort equations. EAF is the product of Effort Multipliers corresponding to each of the cost drives for the target product.
         2. Attributes are based on product, computer, personal, and project attributes.
      3. Detailed
         1. Uses same equations for estimation as intermediate, uses a very complex procedure to estimate cost of the product. Allows to estimate staffing, cost, and duration of each of the development phases, subsystems, and modules, allowing you to experiment with different development strategies to find the best plan for your needs.
         2. Differs from intermediate in two ways:
            1. The effort multipliers for cost drivers are different during each software development phase. These are used to determine effort to complete each phase.
            2. The software product is estimated in three level hierarchical decomposition. The 15 EAF are related to module or subsystem level.
5. **Function Points Method** , uses FP instead of DSI. Computed by Unadjusted function points (UFP), multiplied by complexity
   1. **5 parameters (data) needed for calculating unadjusted function points**
      1. Data functions
         1. # user inputs
         2. # user outputs
         3. # user inquires
      2. Transactional functions
         1. # of files
         2. # external interfaces
   2. **14 technical complexity factors (don’t need to remember all, but know how to use them)**
      1. FP = UFP \* (0.65 + 0.01 \* TCF). Sum TCF for each of the 14 TCF from 0-6 based on their influence. Examples of TCF :
         1. Reliable backup/recovery, data communication, performance, multiple screens for input, Reusability, online update, interface or process complexity, ease of use.
   3. **What are function points**
   4. **What’s the relationship between function points and programming languages**
      1. Functional points have different ratio to LOC per FP based on the language.
   5. **Compare with COCOMO models, what is the main advantage of the function points method**
      1. Todo: Find good answer for this question.

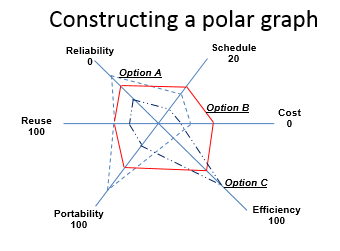
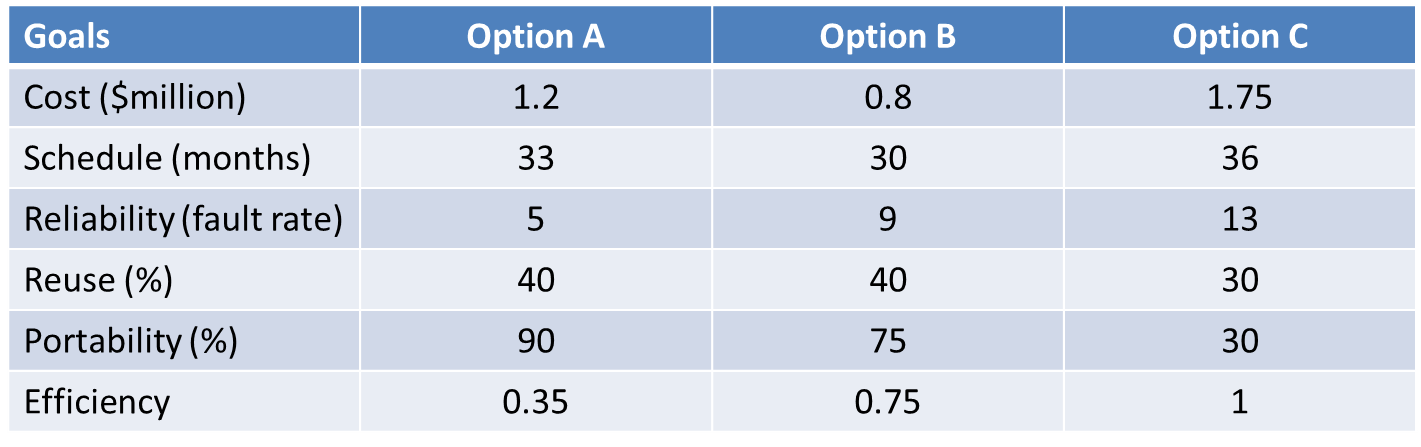
***~~Project Management (Slides 1 to 9 only)~~***

1. ***~~Needs for Software Project Management~~***
   1. *~~SW systems are large, complex, intellectual products~~*
   2. *~~SW are intangible~~*
   3. *~~SW systems are developed by many S.Engineers working together~~*
   4. *~~SW projects need to accomplish PQCT goals.~~*
2. ***~~Three team structures (what are they, and when/how to use them to form a team)~~***
   1. *~~Egoless team structure~~*
      1. *~~The team leader is coordinator, not commander~~*
      2. *~~Team members make technical decisions~~*
      3. *~~Is non-hierarchical, flat structure~~*
   2. *~~Hierarchical team structure~~*
      1. *~~Team manager, senior software engineers, each supervises a number of Jr. software engineers~~*
   3. *~~Chief programmer team structure~~*
      1. *~~One chief programmer manages the project, makes all decisions, and supervises the other team members~~*

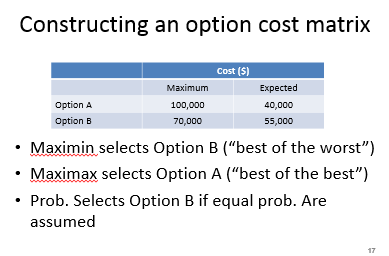
**Project Planning**

1. **Planning errors may lead to what?**
   1. Schedule slippage
   2. Cost overruns
   3. Poor SW quality
   4. High maintenance costs
2. **Procedure for planning a project (list and explain the major tasks of each step)**
   1. **Define the problem**
      1. Define / prioritize the project and product goals
      2. Develop definitive statement of the problem
      3. Justify a computerized solution
      4. Identify the functions and constraints
      5. Relate the functions and constraints to HW/SW/Human subsystems
      6. Define high level acceptance criteria
   2. **Develop a solution strategy**
      1. Develop several solution strategies
      2. Conduct a feasibility study for each strategy solution
      3. Recommend a solution strategy, record why rejecting the others
      4. State / Prioritize the features of the solution strategy
   3. **Plan the development process**
3. **Project goals vs. product goals** – Explained below in examples (a.i vs a.ii, b.i vs b.ii)
4. **Qualitative goals vs. quantitative goals**
   1. Qualitative – related to qualitative features of items
      1. Project goal – project should enhance professional skills of SQA personnel
      2. Product goal – System should make user’s job more interesting
   2. Quantitative – related to quantitative features of items
      1. Project goal – the project should be completed within 12 months
      2. Product goal – The system should reduce transaction cost by 25%
5. **Explain the following terms:**
   1. **Needs** – things that need to be done to resolve problems
   2. **Requirements** – specify capabilities that a system must provide
   3. **Constraints** – specify the acceptable solution space
6. **You need to know how to use a Polar Graph to solve a problem**
   1. Select goals to be satisfied
   2. Score the options against the goals
   3. Construct a polar graph for the options
   4. Select the one which enclose the greatest area

Example for scoring the options against the goals:



1. **How to use payoff matrices to solve a problem (Maximin strategy vs. Maximax strategy)**

****

**Software Metrics**

1. **Purpose of software metrics**
   1. Get insight into the complexity, maturity, sufficiency, and successfulness of a product, project, or process.
   2. Allow to continuously improve the quality of product, project, and process from the past experiences.
   3. Make more accurate estimates on cost, effort, schedule
   4. Make Project management easier
2. **Measure, measurement, metrics, and indicator**
   1. Measure – quantitative indication of a measurable item (amount, dimension, size)
   2. Measurement – the act of determining a measure
   3. Metric – a quantitative measure of the degree to which a system, component, or process possesses a given attribute.
   4. Indicator – a metric or combination of metrics that provide insight into the desired property of a process, project, or product.
3. **4 important technical metrics (Read the terms used in the metrics. For examples, KLOC, MTTC, threat, security, … you don’t need to memorize the formulas.**
   1. **Correctness** – the degree to which SW performs its required function. (defects per KLOC)
   2. **Maintainability** – Tasks? Error correction + change modification + function enhancement. MTTC (mean time to change)
   3. **Integrity** – A system’s ability to withstand attacks to its security. The probability that the attack of a specific type will be repelled.
   4. **Usability** – The ease for users to use the software; the degree of “user-friendliness”. The skill required to learn system, time required to learn well the operation of the system, the net increase on productivity, and a subjective assessment of users attitudes toward the system.
4. **Characteristics of OO, and WHY these characteristics make OO better** (infer from below details)
   1. **Localization** - the manner in which information is concentrated in the program.
   2. **Encapsulation** – the packaging of a collection of items
   3. **Information hiding –** hides the operational details of a program component, and provide only the necessary access of the component to others
   4. **Inheritance** – enables the responsibilities (attributes and operations) of one object to be propagated to other objects.
   5. **Abstraction** – enables the designer to focus on the essential details of a program with little concern for lower-level details.

**Software Quality Assurance**

1. **Purpose of SQA (or quality control)** – variation control (actuality/expectation, differences, surprises)-core of quality control.
2. **Types of quality**
   1. **Quality of design**  - characteristics design for an item – ex. Performance specifications).
      1. Requirements
      2. Specifications
      3. Design
   2. **Quality of conformance (conformity)** - Implementation
3. **V&V, and when to do V&V – TODO: when to do**
   1. Verification – The process of checking that the “implementation” conforms to “specification”
   2. Validation – Aimed at checking the correspondence between a model and real world. That the functional specification corresponds to the real needs of the customer.
4. **Static software inspection vs. dynamic software testing**
   1. **Static** – review and inspection are concerned with analysis of the static system representation to detect problems (static verification)
   2. **Dynamic** – Testing is concerned with executing the product and observing its behavior (dynamic verification)
5. **The cost for detecting/correcting an error at different phases of a process model.**
6. **The precondition of doing a formal software review (Inspection)s**
7. **The procedure of a formal review (inspection)**
8. **Metrics for software quality control**
9. **Software testing (refer to the testing PowerPoint file)**

**Software Configuration Management**

1. **What is software configuration management**
   1. **Baseline management**
   2. **Software configuration item management**
   3. **Motivated by characteristics of software**
2. **What is a baseline (example?)**
3. **What is a (software) configuration item (example?)**
4. **Software configuration management is a discipline for:**
   1. **Systematically identifying and labeling software configuration items**
   2. **Controlling changes to software configuration items**
   3. **Tracking implementation of the changes**
   4. **Reporting configuration status**
5. **SCM functions (from (4)):**
   1. **SC Identification: define the SCIs and baselines**
   2. **SC control: control changes to the SCIs and baselines**
   3. **SC status accounting: report status of SCIs and baselines**
   4. **SC auditing: ensure changes are made properly and timely**
6. **Software configuration control procedure**
7. **What is SCC**
8. **What is CCB**

***~~Agile Software development~~***

1. ***~~Why rapid software development models?~~***
2. ***~~The characteristics of rapid software development models~~***
3. ***~~The principles of agile methods~~***
4. ***~~The problems with agile methods~~***
5. ***~~The XP model (basic concepts, goods and bads)~~***
6. ***~~The Scrum model (basic concepts, goods and bads)~~***