Course: Software Engineering  
Multimedia University Cyberjaya,  

ANSWERS TO ASSIGNMENT NO. 5

1. In the Good Book under core principles for software engineering practice (Chapter 5), the author stated one principle of the many principles (which to me is the most important) as the activity: Think!

Instruction: Please give five concrete, specific and detailed examples (not SE principles) that you would think about in SE practice in accordance with what the author meant by his “Think” principle.

ANSWER

The Good Book mentioned ‘Think’ as the seventh (last) core principle in software engineering practice (Page 133). To relate the last principle, we need to recall the first six core principles:

(1) Reason for software to exist – provide value to its users (not a useless product)
(2) Design the software system – keep it simple and follow design standards
(3) Keep the vision of the software – do not stray from the purpose of the software
(4) Maintenance of the software – specify properly, design properly, document properly
(5) Openness in system design – future requirements may be added, new hardware
(6) Plan for reuse – future use of existing components, design patterns, objects etc

So the principle of “Think” here is meant for you to carefully think about those first six principles. You are strongly urged to find those answers (i.e. to the 6 principles) before you jump into doing your software engineering project. If you do not know some of he answers, go and do research on it, ask somebody, look into previous completed software documents, study, plan forward what you will do in the project. Essentially, “think before you do” because some people just “do without thinking”. So when you “think before you do, you will do things with clear reasons, objectives and goals”

So for 5 concrete examples: Consider the case that I am just being appointed to be the Project Manager for a software project. I do not know anything about it yet, so I must find the most important information for me to strategize and plan for the project. In this case my “think” principles will cover the following:

1. What is the project all about? Information regarding the high level view, the scope of work and deliverables of the project.

2. Who is the client (owner) of the project? What is the business case for the client to implement this project? Basically why the client wants to implement this project.

3. When is the project to commence? How long is the duration of this project? Is it a straight through implementation or a phased and staged out implementation?

4. Where is the project going to be implemented (installation sites)? Local, remote and/or including the client’s third party sites.

5. How much is the contract worth? What are the values of the various components (breakdown of major deliverables) of the contract worth?

IMPORTANT: Your answers to this question are not like the 5 examples above (That is about me being appointed as a new Project Manager, not knowing anything to start with so I have to “think” and I came up with those 5 examples). In your case, you must think
about the software you are about to build according to the 6 core principles in software engineering. I hope you understand ("Think").

2. The author of the Good Book at one point mentioned the "W5HH Principle" as one of the excellent principles to be adopted for software engineering processes and projects. Directly related to this principle is a statement, quote: “The customer, users, and other stakeholders also have responsibilities.”

Instruction: Please list five specific responsibilities of the customer that would provide significant contributions to the eventual success of the software engineering project.

ANSWER

The Good Book mentioned the "W5HH principle" twice (following the Index) on page 138 in Chapter 5 on Software Engineering Practice and page 644 in Chapter 21 on Project Management. This principle comes from the famous Barry Boehm [BOE96] not from the author of the Good Book. Note: You must remember this principle all the time because it is used everywhere in your lives. (What, Why, When, Who, Where, How, How much)

The "W5HH Principle" is about organizing your project to meet and satisfy (not strayed, nor delayed, nor slipped) your project objectives, milestones and schedules, responsibilities, management and technical approaches, and required resources.

The following examples list five specific responsibilities of the customer that would provide significant contributions to the eventual success of the software engineering project.

1. After you finished the documented user requirements document (the what), you must get the customer's involvement (the who) to agree on the terms in the document. If you do not get agreement, you may be implementing a project not to the user's requirements (the why).

2. On purchasing, you must list in the project organization as to what is to be purchased, why is it purchased, when is the purchased required, who is responsible for issuing the purchase order, who is responsible to approve the purchase order, etc. Normally a work flow for purchasing includes the communications for the purchasing activity, just like the UML2 activity diagram. For a properly implemented project this information is inside in the Project Quality Manual, with an organization that includes representatives of the customer (why) as it involves payment.

3. After you finished your design and you made some software screen prototypes, you need the users to be involved to see your mock-up screens so that they agree with the functionality and others as they see on your prototype screens etc.

4. When you plan to do the user software testing, you need users to execute the testing following your test protocol sheets (documents). You need the users to agree on the testing schedule (dates) and persons (users) of the right category.

5. On the software deployment, you need the customer to agree on your deployment architecture, the network, the people who get access to the correct functionality etc. Otherwise, you will get havoc.

3. In our Software Engineering class (version WRY), we have essentially completed the User Requirement Specifications (URS) at this point. The next logical step is to analyze the contents of the URS. On that issue, the Good Book (Chapter 5) stated that one of the two classes of models that represent the user requirements is the 'Analysis Model'. This model depicts the software in three different domains: the information domain, the functional domain and the behavioral domain.
Instruction: In your own words, what do you understand by the difference the author made between the functional domain and the behavioral domain? Repeat, in your own words. (Hint: Look at the tasks for Analysis Modeling)

ANSWER
The Good Book on page 141 (Task Set) in Chapter 5 addressed Item 4 (Functional Domain) and Item 5 (Behavioral Domain).

The difference between the functional domain information and the behavioral domain information is in its definition.

The functional domain information for the software is the functionality requirements of the software, while the behavioral information is the changes in states, events, modes of the software with respect to users input, system events, timing etc.

A use-case diagram provides the typical functional domain information. A data flow diagram (DFD) provides both structural and functional domain information. Refer to the Good Book Chapter 8 in Figures: 8.9, 8.10 and 8.11. A class diagram also provides both structural and functional domain information, like in Figure 8.14.

A state (statechart) diagram provides the typical behavioral domain information. Refer to the Good Book Chapter 8 in Figure 8.12.

Also note in Chapter 8, page 226 “Although the data flow diagram (DFD) and related diagrams are not a formal part of UML, they can be used to complement UML diagrams and provide additional insight into system requirements and flow”.

4. In the later part of the discussions the author of the Good Book uses a lot of abbreviations for terms that should have been understood earlier to avoid unnecessary repetitions. For the first time reader, this can be confusing. On that note, in Chapter 6, explain what is meant by the following abbreviations:

- CLSS
- SCD
- AFD
- SFD
- UML
- BPE
- NC

ANSWER
CLSS – Conveyor Line Sorting System
SCD – System Context Diagram
AFD – Architecture Flow Diagram
SFD – System Flow Diagram
UML – Unified Modeling Language
BPE – Business Process Engineering
NC – Numerical Control

Important note: You may be curious as to why this assignment question was so simple and straightforward, which is not a typical habit of WRY. Think. (Hatley-Pribhai Modeling in Chapter 6 on page 165). The idea is for you to read the text document and understand something about it while you look for the answers to the abbreviations.

It is actually about your perspective when you go about solving large complex problems. You begin from the ‘world view” or big picture (Fig 6.4), then each component in the big picture is broken down into smaller pictures (Fig 6.5), narrow down your view to the
"domain view". You must have realized this when you had difficulties in your design assignment (project) based on the User Requirements Document. Ha..ha..ha..

5. In your own words, what do you understand by the term 'Analysis Patterns'? Why would analysis patterns speed up development in the software engineering process? (Hint: Do not reinvent the wheel)

ANSWER

The Good Book in Chapter 7 Requirements Engineering, on page 200 devoted a section on Analysis Patterns. It is about certain recurring things in software engineering (e.g. a class, a function, a behavior etc) that people experienced when they have done several projects.

When you analyse your software requirements (system or functional), you may find some kind of 'pattern' or similarity in the solution to a specific problem, i.e. referring to information gathered in your previous experiences. If you adopt the pattern in your design, you sort of have a proven solution for it done by somebody else earlier. Just use it and you do not need to reinvent the wheel. This speeds up development in the software engineering process.

The Good Book in Chapter 9 on Software Engineering Practice, on page 266 also discussed design patterns. It states "The intention of each design pattern is to provide a description that enables a designer to determine

1. whether the pattern is applicable to the current work
2. whether the pattern can be reused (hence, saving design time)
3. whether the pattern can serve as a guide for developing similar, but functionally or structurally different pattern. "