ASSIGNMENT NO. 4

1. According to Dennis Ludwig (Required Reading No. 9), what are the three (3) prerequisite steps that must be decided before choosing a programming language for a software project? Explain why at the outset these decisions are important. Please give examples in your arguments.

Note: This question does not ask for the things or issues that must be considered before choosing a programming language.

ANSWER

(1) Decide on the computer hardware and operating system to be used.
If the software is for control and monitoring of instruments and devices, then the operating system must be capable of real-time processing. If the software is for information storage, processing and retrieval, the operating system must support whatever language selected to write the software. In some instances, the client may dictate the decision on the operating system and hardware for reasons like: being familiar with a particular hardware, or software, or having a special commercial arrangement with a particular hardware and software vendor or simply the client wants to ‘synchronize’ the application with their existing systems thus making it easier for their support and maintenance staff to work on as they are already familiar with the existing systems.

(2) Consider the support and maintenance of the hardware and operating system
The availability of both current and future support and maintenance of the selected hardware and operating system is crucial. It is not only important for the client own staff, it is also important for the vendor (contractor/subcontractor etc). Sometimes, decisions are made on the basis of: very reputable hardware and software vendors, very good customer support, strong R & D, vendor will exist for a long future period (financially strong company), provides guarantees of hardware (spare parts) and software upgrades up to a certain period. For example, in 2003 Microsoft announced that it will no longer support Windows 98. The selection of the operating system thus directly affects the selection of the programming language because most operating systems support their own set of programming languages.

(3) Decide on the methods to be used to design the software.
If the architecture of the software is going to be object-oriented, then the language selection will be restricted to object oriented programming languages. If the software design method is going to be of top down procedural method, the decision will open up the choices to almost all standard programming languages.

2. If the activities of user requirements gathering, user interviews and reviews, elaborations and negotiations with users, documentation, validation and agreement with users lead to the Software Requirements Specifications (SRS), then what are the similar activities that lead to the Software Design Specifications (SDS)?

ANSWER

There is no single correct answer for the series of activities that lead to the production of the Software Design Specifications (SDS) because everybody has and believes in their own ‘best way’ to conduct activities leading to it. However, there are certain standard activities that most practitioners agree to:
• Analyze the Software Requirements Specifications (SRS) to turn and translate it into some suitable software component models
• Analyze and create models for the data requirements, data flow, and data control and restrictions etc,
• Analyze and create models for the functional behavior of the software
• Analyze and create models for the components (procedural functions or object OO components) that represent the functional behavior of the software
• Look into the interface, synthesis and integration of the various components
• Analyze and create models (screens) for the user interface
• Evaluate the various strategies in software design from the created and modeled components based on both current and future requirements (coding efforts, technologies adopted - advantages, disadvantages, efficiency, maintainability, etc)
• Look into the requirements on how the final software will be deployed in the production environment (local, remote access, internet access, security, etc)
• Analyze the effort, capability and resources needed to execute the implementation based on the various software designs alternatives
• Evaluate the various strategies and methods, decide on the most optimum design strategy and write out the Software Design Specifications (SDS). The completed specification document or set of documents must include all selected and agreed information and diagrams which are outputs of the activities leading to the SDS.

3. On the subject of software project tracking and control, the highly recommended software tool for learning and use in this course is the Microsoft Project application. In your own words state at least five (5) reasons to that? (Hint: Required Reading No. 21)
ANSWER
The Microsoft Project application has been recommended as the software tool for learning project management, project tracking and control for the following reasons:

(1) Has all the major tools for project tracking and control like: Track progress, manage schedule, manage resources, manage costs, manage scope, manage risks, report project status etc.

(2) Is a proven tool and has been used in many real projects successfully. When reputable companies use this tool in their major projects, it gives a typical user confidence and assurance that the software tool is a right choice.

(3) Is easy to use (GUI based) and provides tutorials, useful help and hints. This facilitates self learning on the subject of management, tracking and control of projects.

(4) Has various perspectives (representations) towards project management, tracking and control like: Gantt charts, Network diagrams, critical path, slacks, milestones, resource allocation and planning, budgets and costs, activity progress charts (percentage completions) etc.

(5) Provides integration (import and export) of information with standard office automation tools like: Microsoft Office (Word, Excel, Power Point, Access etc).

4. The Good Book at one point states that "software does not wear out". While another writer believes that "If software does not wear out, then software once built must be perfect". The writer provided arguments that software cannot be perfect (Required Reading No. 20).

In another occasion the Good Book states "During its life, software will undergo change." Explain what you understand by this second statement. If you agree, explain why you think there will be changes. If you disagree, explain why there cannot be changes. In both cases, argue whether the changes or no changes are beneficial or detrimental.
ANSWER

In most cases, software during its life will undoubtedly undergo change, albeit for many reasons. The first obvious reason is to correct for 'bugs or errors' in its design. The next obvious reason is to incorporate new functionalities, either user requested or vendor upgrade versions. The next reason is incorporation of new and more efficient technologies into the software which did not exist during its initial production. And the list goes on.

For that changing reason, an international standard exists (ISO/IEC 12207) - Software Life Cycle Processes. Among the many things in this standard is the coverage on the management and engineering of software that include the activities and tasks that need to be undertaken to manage changes in software during its life.

5. In your own words, what are the differences between a PERT (Network) diagram and a CPM diagram? In which instances will you choose one representation over the other? Please answer to the questions asked.

ANSWER

A critical path method (CPM) diagram shows the 'longest path' in the project by joining the applicable tasks to project completion, then displays it on the Gantt Chart (a chart plotted for the tasks against a time based calendar). A network diagram (PERT) does not have the perspective to show the 'longest path' because its representation is activity or node based and not time or calendar based.

A CPM diagram representation of a project is primarily chosen to monitor and track the progress of work against the 'critical activities or tasks' such that the entire project do not go over the agreed completion date. A CPM is normally a Gantt Chart with specific milestones and project deliverables. We therefore choose a CPM diagram when we want to monitor project progress.

A PERT or network diagram looks more towards the dependencies of tasks and subtasks and how they relate to one another. For example: one task must be finished before the start on another, because the other depends on the first one. Its primary goal is not towards the tracking of the overall project completion date as the CPM does. We therefore choose a network diagram when we want to look at tasks or group of tasks and their dependencies.