Good case study about software engineering

Business, Company

Question 9. 1

Classical architecture refers to the period from the European history, covering through the rise of Greek Empire through to the foremost power cultural period until the collapse of the Roman Empire. This classical architecture is particularly referred to as Roman and Greek architecture mainly describing this ancient era of design architecture. Whereas software architecture in essence is distant from the concept of building and construction it is the process of computer systems engineering.

Similarities

The implementation and planning of software architecture and classical architecture are both characterized by the distinctiveness of the objectives and the available resources thereby referred to as a project. Both the software architecture and classical architecture are similar in phases of development for instance;

There is the blueprint plan in the work phases of classical architecture whereas in the phases of software engineering there is the involvement of software architecture design. In classical architecture, there has to be implementation planning involved whereas in the phase of software architecture there is the implementation of the design.

In the phases of software architecture there is the maintenance of the system, in classical architecture warrantee claims and documentation processes are undertaken.

While in classical architecture there is the approval planning involved, in software architecture involves design of components and classes. In software

architecture there is the integration and test while in classical architecture there is the preparation and award of contract.

The phases in classical architecture involve basic evaluation while in the software architecture it involves analysis. In the phase of classical architecture, there is the definition of requirements while in software engineering there is the preliminary planning involved.

Differences

The implementation of classical architecture is undertaken by external laborer at the construction site, whereas software architecture is carried out by a software developer. The other distinct difference is that in classical software, maintenance involves technical facilities repair during the warrantee period while in the software architecture, maintenance is done throughout the life of the software system.

Question 9. 8

A computer with a distinct transform flow characteristics is an example of one that has a series of computational or manipulative components into output data. A pipe and a filter pattern have set of components that are referred to as filters, linked with pipes that transmit data from a single component to the next. If the data flow reprobates into a single line of transforms, then it is referred to as batch sequential. It is this structure that accepts the single line of transforms and then applies a series of sequential filters to transform it.

A mapping technique that is referred to as structural design is particularly characterized as a design that is data flow oriented technique because of the

convenient transition from a data flow diagram to software architecture. There is a six part process involved in the transition from information from flow as represented as a data flow diagram to program structure includes; establishment of information flow, indication of flow boundaries, mapping of the data flow diagram into the program structure, definition of the control hierarchy, refinement of the resultant structure using heuristics and design measures, and elaboration of the architectural description.

Data flow mapping is for instance presented by a step by step transform mapping for a safe home security function. The determination of the information flow has to be done and one type of information flow is the transform flow which exhibits a linear quality. Transform mapping is an establishment of design steps that allow a data flow diagram which has transform flow characteristics to be mapped into specific architectural style.

Question 14. 8

A good enough software is on with compromises and the compromises may be due to the resource constraints of a company developing the software. Good enough software particularly delivers high quality functions and features that user’s desire and specify but at the same time there is the provision of specialized functional errors and features that is contained known as bugs.

It is prohibitively expensive to produce software that is bug free and it most important that the software serves its functionality and benefits the consumer as long as the benefit provided outweighs the problems that might be associated with the bugs. The customers may overlook the bugs since they are satisfied with its functionality.

A company that has a budget in marketing which is large and there is the capacity of company to convincing consumers in purchasing a version inferior, therefore it has waylaid the market and by having a more improved version development in subsequent version development then it has the market in the lockdown.

A well established a known company is Valve. This particular company develops and distributes non-free computer games that come with Digital Restrictions Management (DRM). The purpose of the non-free popular games on the Linux/GNU operating system is to increase adoption of the software system. The objective is also to bring freedom to the users of the software.

References

Ding, Wei, and Xia Lin. Information Architecture: The Design and Integration of Information Spaces. San Rafael, Calif.: Morgan & Claypool Publishers, 2010. Internet resource.

Horowitz, Ellis, and SartajSahni. Fundamentals of Data Structures. Woodland Hills, Calif.: Computer Science Press, 1976. Print.

Humphrey, Watts S. A Discipline for Software Engineering. Reading, Mass: Addison-Wesley, 1995. Print.

Pressman, Roger S. Software Engineering: A Practitioner's Approach. Boston: McGraw-Hill Higher Education, 2010. Print.

Damiani, Ernesto, L C. Jain, and M Madravio. Soft Computing in Software Engineering. Berlin: Springer, 2004. Print.

Hitchins, Derek K. Systems Engineering: A 21st Century Systems Methodology. Chichester, West Sussex, England: John Wiley, 2007. Print.

Lamsweerde, A . Requirements Engineering: From System Goals to Uml

Models to Software Specifications. Chichester, England: John Wiley, 2009. Print.