Santhosh Kumar Gattu
SG6N6, 16211118

CS 490MT/5555: Software Methods and Tools

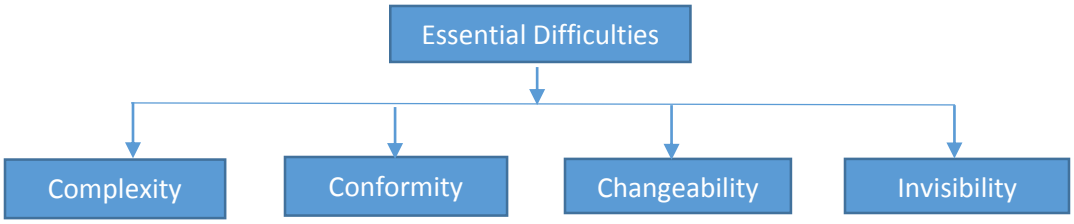
Assignment-1

1. **(1) What are the essential difficulties of software systems discussed in Fred Brook's paper? Explain each using your own words.**

   Ans.

   Fred Brook discussed the following essential difficulties of software systems in his paper:

   "No Silver Bullet – Essence and Accidents of Software Engineering"



   1. **Complexity**

      '*Complexity is the essential difficulty of the software systems but not an accidental one*.'

      Software elements are way larger in size and no two elements are alike. As the software entities increases in nonlinear fashion, the complexity of the software systems increases more than linearly. Software systems differ greatly from Computers, buildings and Automobiles. For example: Digital computers, having number of states which makes very hard to describe and test them.

      Fields like Mathematics and Physical sciences made revolutionary steps in modeling complex problems by ignoring the complexities as these are not required properties. On the other hand, this kind of model does not work in case of software systems. The problems encountered during the development of a software product, right from the Requirement Analysis gathering, discussion among team members, management problems, Designing, Implementation and Maintenance comes from the complexity itself. These results in faulty products, over costs and delay in schedule.

   2. **Conformity**

      Much of the software complexity comes from the conformation to other systems. Software must obey the rules and regulations as prescribed but it is not.

For example:

For developing a complex software as per the client requirement, the following series of steps needs to be followed:

1. Requirement Analysis.
2. Software design.
3. Implementation.
4. Testing.
5. Maintenance.

A situation comes when client changes the entire requirements and whole developed software should go through the above mentioned process but it is very difficult to follow as this involves manual effort, cost, time and it over burdens the entire team. If the developed software does not meet the client requirement it loses customer satisfaction.

3. Changeability

In the real world when we consider construction field where building designs are subjected to frequent changes which involves loss of huge budget, manual effort and precious time. This makes the construction company to initiate steps not to change the requirements now and then. Likewise, Software is also prone to changes over the time and these changes pressurizes it to adapt to new conditions and requirements. It is an easy process to change the software but recurring changes makes it difficult to redesign the entire software.

A successful software can be easily changed which involves two processes:

a) Software which is useful to large number of audience gains much customer satisfaction and subjects to frequent changes which are successful and it involves developing new functions within the existing ones.
b) The life expectancy of a successful software exceeds the normal human life.

4. Invisibility

Unlike building construction plan which helps an architect to build it, graphical sketches of mechanical parts to design a product, software is vast in nature that it is hard to imagine in space and difficult to represent by geometrical drawings. Even if we try to graphically represent the software structure, it cannot fit in one space as one graph overlaps with the other. Software designing is obstructed in spite of having much progress in simplifying the complex software structure. A solution to this problem is to bond a link between graphs in an orderly fashion.

Santhosh Kumar Gattu
SG6N6, 16211118

1. **(2). Pick one software method or tool that you used before and specifically explain whether or not you think this method or tool is a "promising attack" on the essential difficulties mentioned above.**

Ans.

"ADPART" is the tool that I have used before which I think is a "promising attack" on the essential difficulties faced by the software systems. It is an automatic testing tool in which we represent the entire system as flow diagrams, based on these flow diagrams it automatically generates all possible Test cases. Even for complex business systems it covers the entire system and serves as an excellent tool. Hence it can be used not only for simple systems but also for complex software applications. ADPART covers all possible scenarios to be tested, thereby it can handle huge systems, obeys rules and regulations, meets all the client requirements and finally wins customer satisfaction.

In most of the real time software projects there will be frequent requirement changes by the client, it is an easy process to alter the system design in ADPART for these projects. As the entire software system is represented as diagrams in this tool, it can be easily visualized. Complex software systems involves many number of flow diagrams which makes it even more complex to visualize but it is not at all a problem with ADPART.

All in all, based on my experience with this tool I would conclude ADPART as one of the "promising attack" on the essential difficulties faced by the software system.

2. **Make a class schedule for this course using Microsoft Project 2013.**

Ans.

WBS	Task Name	Start	Finish	Predecessors	Resource Names
0	**Software Methods and Tools**	**Mon 8/3/15**	**Tue 12/8/15**		
1	**Planning Course Schedule**	**Mon 8/3/15**	**Fri 8/14/15**		
1.1	Scheduling Lecture topics	Mon 8/3/15	Wed 8/12/15		Instructor,Teaching Assistant
1.2	Planning Assignments & Exams	Thu 8/13/15	Fri 8/14/15		Teaching Assistant
2	**Introductory class**	**Mon 8/24/15**	**Thu 8/27/15**		
2.1	Introduction to Software Systems	**Mon 8/24/15**	**Mon 8/24/15**		Instructor
2.2	Microsoft Project	**Thu 8/27/15**	**Thu 8/27/15**		Teaching Assistant
3	**UML Modeling**	**Thu 9/3/15**	**Thu 9/10/15**		
3.1	UML Modeling-1	Thu 9/3/15	Thu 9/3/15		Instructor
3.2	UML Modeling-2	Thu 9/10/15	Thu 9/10/15		
4	**Integrated Development Environments**	**Tue 9/22/15**	**Tue 9/29/15**		
4.1	Introduction to IDE and Eclipse	Tue 9/22/15	Tue 9/22/15		Instructor
4.2	Eclipse Plug-ins-1	Thu 9/24/15	Thu 9/24/15		
4.3	Eclipse Plug-ins-2	Tue 9/29/15	Tue 9/29/15		
5	**Software Architecture & Design**	**Tue 10/6/15**	**Thu 10/8/15**		
5.1	Software Architecture & Design-1	Tue 10/6/15	Tue 10/6/15		Instructor
5.2	Software Architecture & Design-2	Thu 10/8/15	Thu 10/8/15		
6	**Software Tools**	**Tue 10/13/15**	**Tue 10/27/15**		
6.1	ArchStudio	Tue 10/13/15	Tue 10/13/15		Instructor
6.2	Emacs	Tue 10/27/15	Tue 10/27/15		
7	**Software Testing**	**Thu 11/5/15**	**Thu 12/3/15**		
7.1	Introduction to Software Testing	Thu 11/5/15	Thu 11/5/15		
7.2	Junit	Tue 11/10/15	Tue 11/10/15		
7.3	GIT	Thu 12/3/15	Thu 12/3/15		
8	**Lectures/Labs**	**Tue 8/25/15**	**Thu 10/22/15**		
8.1	**Lectures**	**Thu 8/27/15**	**Thu 10/22/15**		**Instructor**
8.2	**Labs**	**Tue 8/25/15**	**Tue 10/20/15**		**Teaching Assistant**
9	Assignments	Fri 10/9/15	Tue 12/8/15	3	Students
10	Exams	Mon 8/17/15	Wed 10/7/15	3	Students