Spring-2017

Software Methods and Tools

Assignment-1

Submitted by:

Moulika Chadalavada

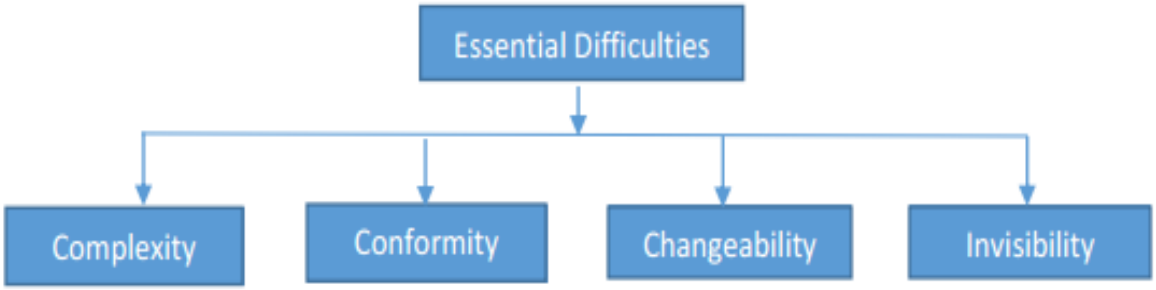
16234180

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

The paper '**No Silver Bullet: Essence and Accidents of Software Engineering'** defines some concepts related to software development. Fred Brook says there are two classes of qualities of software

i. **Essential qualities**
ii. **Accidental qualities.**

The essential qualities are extremely difficult to address or improve, whereas the accidental qualities can be improved through tools. The essential qualities all have to do with the design of software. Fred Brook discussed following essential difficulties of software system in his paper



i. **<u>Complexity:</u>**

Software systems are more complex of all our human constructions. Two parts are never alike; even if they are we will make it as single entity and make it complex again. In development of a software system no two parts are alike in hardware or software. Coming to digital computers they are very hard in construction (very difficult to build). There construction is very large and is very hard to test the system.

Coming to the scaling (Linear or Non-linear) of the system the parts or entities of the system are never alike. They interact with each other in non-linear fashion. So, it is being very difficult to scale them. So, the complexity of the whole system is increased. Software complexity is having more essence from many years. Some of the entities are experimented and tested to reduce the complexity without affecting the operation.

Many mathematical operations are done on this and they can reduce only minimum amount complexity which doesn't have essence. Many problems are developed to software because of the complexity such as difficulty of communication between members which in turn gave chances for production flaws and

missing the deadlines. From this there also problems like enumerating, not understanding the whole system, unreliability.

ii. 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, to develop complex software as per client requirements, the following series of steps needs to be followed:

1. Requirement Analysis
2. Design
3. Implementation
4. Testing
5. Maintenance

A situation may come where clients changes entire requirements and above process has to be repeated which results in increase of manual effort, cost, time and it over burdens the entire team. If the developed software does not meet the requirements, then the customer satisfaction is lost.

iii. Changeability:

Software is subjected to change as we are simply pushing bits around at last, changing the thoughts. Software is inalienably adaptable where physical items are definitely not. Despite the fact that product item is effective it may change.

A successful software can be changed easily involving below two processes:

a) Software that is useful to huge number of audience receives much customer satisfaction and also subjects to frequent changes in the product which are successful and that particular change involves in developing new functionalities within the existing ones.
b) A successful software life expectancy always exceeds the normal human life.

iv. Invisibility:

Unlike building construction plan that helps architect to build it, software is vast in nature which is hard to imagine and difficult to represent it geometrically. Even though we try to represent it graphically it might not fit into the space and will overlap with other. Software designing is obstructed in spite of having much progress in simplifying the complex software structure. A solution to this problem is to maintain a link between graphs in an orderly manner.

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.

There are many tools which I have used in my professional career to develop projects, one among them is "ADPART" which I found it as promising attack on essential difficulties Complexity, Conformity, Changeability, Invisibility.

"ADPART" is an automatic testing tool in which entire system is represented as flow diagrams. Based on this tool all the possible test cases are generated automatically. It works even for complex business systems excellently.

As ADPART covers all the scenarios to be tested all the bugs in the system are identified earlier maintaining all the rules and finally zero defect software is delivered to client. With this we can win customer satisfaction.

Any frequent changes in requirements can be handled easily with ADPART. Complex software systems involve many number of flow diagrams which makes it even more complex to visualize but it is not at all a problem with ADPART.

ADPART tool accepts changes and makes the system flexible by handling any of complex business systems so according to me it is a promising attack on of the essential difficulties 'Changeability', 'Complexity'.

3. Make a class schedule for SMT course using Microsoft Project 2013.

Microsoft Project 2013 is a project management tool used to schedule project. It helps in creating tasks, resources, WBS code, and also how to manage relationships between tasks. As part of this question class schedule of Software Methods and Tools is represented in Project 2013 as shown below.

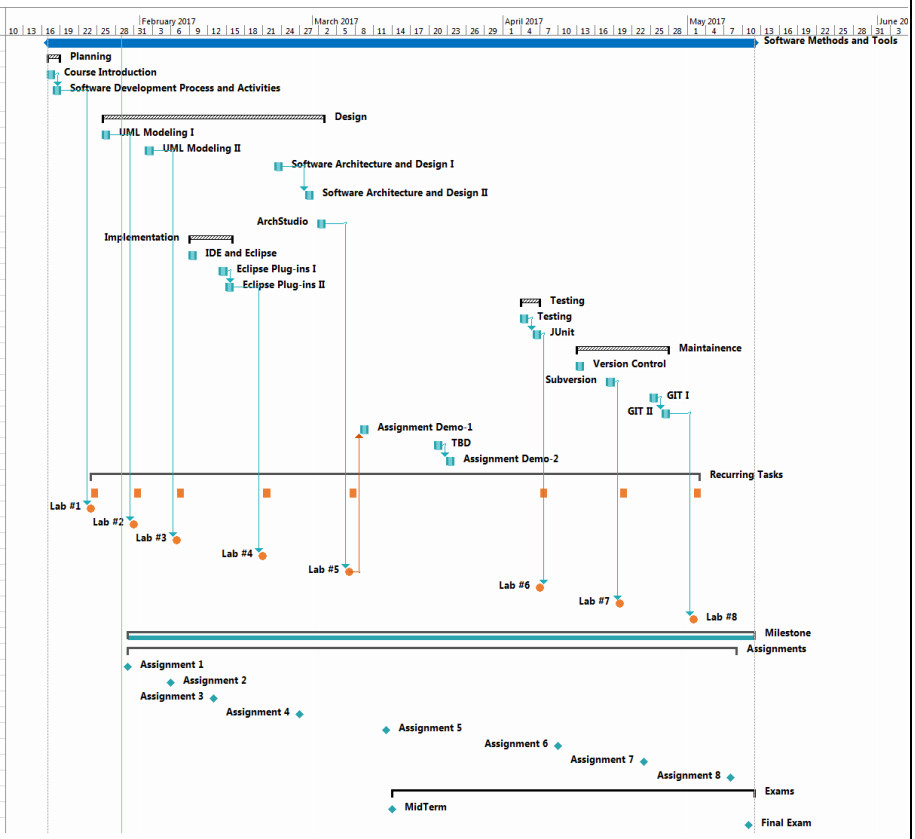
A Project Summary Task is created with name ➔ Software Methods and Tools

Tasks ➔ Planning, Design, Implementation, Testing, Maintenance

Recurring Tasks ➔ Labs

Milestones ➔ Assignments, Exams

ID	Task Mode	WBS	Task Name	Duration	Start	Finish	Predece	Resource Names
0		**0**	**Software Methods and Tools**	**83 days**	**Tue 1/17/17**	**Thu 5/11/17**		
1		**1**	**Planning**	**2 days**	**Tue 1/17/17**	**Wed 1/18/17**		**Professor,Student**
2		1.1	Course Introduction	1 day	Tue 1/17/17	Tue 1/17/17		Professor,Student
3		1.2	Software Development Process and Activities	1 day	Wed 1/18/17	Wed 1/18/17	2	Professor,Student
4		**2**	**Design**	**26 days**	**Thu 1/26/17**	**Thu 3/2/17**		**Professor,Student**
5		2.1	UML Modeling I	1 day	Thu 1/26/17	Thu 1/26/17		Professor,Student
6		2.2	UML Modeling II	1 day	Thu 2/2/17	Thu 2/2/17		Professor,Student
7		2.3	Software Architecture and Design I	1 day	Thu 2/23/17	Thu 2/23/17		Professor,Student
8		2.4	Software Architecture and Design II	1 day	Tue 2/28/17	Tue 2/28/17	7	Professor,Student
9		2.5	ArchStudio	1 day	Thu 3/2/17	Thu 3/2/17		Professor,Student
10		**3**	**Implementation**	**5 days**	**Thu 2/9/17**	**Wed 2/15/17**		**Professor,Student**
11		3.1	IDE and Eclipse	1 day	Thu 2/9/17	Thu 2/9/17		Professor,Student
12		3.2	Eclipse Plug-ins I	1 day	Tue 2/14/17	Tue 2/14/17		Professor,Student
13		3.3	Eclipse Plug-ins II	1 day	Wed 2/15/17	Wed 2/15/17	12	Professor,Student
14		**4**	**Testing**	**3 days**	**Tue 4/4/17**	**Thu 4/6/17**		**Professor,Student**
15		4.1	Testing	1 day	Tue 4/4/17	Tue 4/4/17		Professor,Student
16		4.2	JUnit	1 day	Thu 4/6/17	Thu 4/6/17	15	Professor,Student
17		**5**	**Maintainence**	**11 days**	**Thu 4/13/17**	**Thu 4/27/17**		**Professor,Student**
18		5.1	Version Control	1 day	Thu 4/13/17	Thu 4/13/17		Professor,Student
19		5.2	Subversion	1 day	Tue 4/18/17	Tue 4/18/17		Professor,Student
20		5.3	GIT I	1 day	Tue 4/25/17	Tue 4/25/17		Professor,Student
21		5.4	GIT II	1 day	Thu 4/27/17	Thu 4/27/17	20	Professor,Student
22		6	Assignment Demo-1	1 day	Thu 3/9/17	Thu 3/9/17	31	Professor,Student
23		7	TBD	1 day	Tue 3/21/17	Tue 3/21/17		Professor,Student
24		8	Assignment Demo-2	1 day	Thu 3/23/17	Thu 3/23/17	23	Professor,Student
25		**9**	**Recurring Tasks**	**71 days**	**Tue 1/24/17**	**Tue 5/2/17**		**Student,TA**
26		**9.1**	**Labs**	**71 days**	**Tue 1/24/17**	**Tue 5/2/17**		**Student,TA**
27		9.1.1	Lab #1	1 day	Tue 1/24/17	Tue 1/24/17	3	Student,TA
28		9.1.2	Lab #2	1 day	Tue 1/31/17	Tue 1/31/17	5	Student,TA
29		9.1.3	Lab #3	1 day	Tue 2/7/17	Tue 2/7/17	6	Student,TA
30		9.1.4	Lab #4	1 day	Tue 2/21/17	Tue 2/21/17	13	Student,TA
31		9.1.5	Lab #5	1 day	Tue 3/7/17	Tue 3/7/17	9	Student,TA
32		9.1.6	Lab #6	1 day	Fri 4/7/17	Fri 4/7/17	16	Student,TA
33		9.1.7	Lab #7	1 day	Thu 4/20/17	Thu 4/20/17	19	Student,TA
34		9.1.8	Lab #8	1 day	Tue 5/2/17	Tue 5/2/17	21	Student,TA
35		**10**	**Milestone**	**74 days**	**Mon 1/30/17**	**Thu 5/11/17**		**Student**
36		**10.1**	**Assignments**	**71 days**	**Mon 1/30/17**	**Mon 5/8/17**		**Student**
37		10.1.1	Assignment 1	1 day	Mon 1/30/17	Mon 1/30/17		Student
38		10.1.2	Assignment 2	1 day	Mon 2/6/17	Mon 2/6/17		Student
39		10.1.3	Assignment 3	1 day	Mon 2/13/17	Mon 2/13/17		Student
40		10.1.4	Assignment 4	1 day	Mon 2/27/17	Mon 2/27/17		Student
41		10.1.5	Assignment 5	1 day	Mon 3/13/17	Mon 3/13/17		Student
42		10.1.6	Assignment 6	1 day	Mon 4/10/17	Mon 4/10/17		Student
43		10.1.7	Assignment 7	1 day	Mon 4/24/17	Mon 4/24/17		Student
44		10.1.8	Assignment 8	1 day	Mon 5/8/17	Mon 5/8/17		Student
45		**10.2**	**Exams**	**43 days**	**Tue 3/14/17**	**Thu 5/11/17**		**Student**
46		10.2.1	MidTerm	1 day	Tue 3/14/17	Tue 3/14/17		Student
47		10.2.2	Final Exam	1 day	Thu 5/11/17	Thu 5/11/17		Student



Couse Schedule Project 2013 : https://github.com/cmoulika009/Software-Methods-and-Tools/blob/master/Assignment-1-Project%202013/Course%20Schedule%20-%20Assignment-1.mpp

GIF Image Link : https://www.dropbox.com/s/87q01b7h2x5f1gd/Assignment-1.gif?dl=0