Project Plan

Mavis

Helsinki 7th of February 2006
Software Engineering Project
UNIVERSITY OF HELSINKI
Department of Computer Science

Table Of Contents

Introduction

The Geophysics Department of the University of Helsinki uses a SQUID magnetometer for measuring the magnetism of rock samples. Previous OhTu-project built a new and improved program for using the machine based on old software but the client has need for some improvements and bug fixes. Client also has need for an analyzing software that can be run under newer windows platforms.

Goal of the project is to improve the existing program and find a way to run analyzing software in newer windows platforms, either by building a new one, or by improving the old one.

Organization

Project Group

Kimmo Holm	Project leader
Juho Julkunen	Requiment specification, Project leader
Rami Jarvinen	Testing
Janne Laukkarinen	Software architect
Joel Linden	Documentation
Jan Wagner	Coding

Project Masters

Joni Salmi	Tutor
Juha Taina	Head of Project

Clients

Tomas Kohout	Client
Fabio Donadini	Client

Homepage

http://www.cs.helsinki.fi/group/mavis/

Workflow

Project will have two concurrent processes: one for fixing and improving the current program and another for the analyzing software. Exact details for the analyzing software process depend on the course of action; whether we try to make the current program run under windows, or build an entirely new software for it. That has yet to be determined.

Main Process: Bugs And Improvements

Main process will follow iterative model and development of the program will happen in short, concise blocks, of which each will contain definition, design, production and testing of the current test version. Next block will start when client is supplied with a new test version for testing. Through this testing (and internal testing) requirements list will be kept up-to-date to current problems and design features. Next test version will be designed based on the requirements list. Concurrently with this designing/coding, will run testing of the current test version to update requirements list.

Rationale for this approach is that we have no need to design and build a whole new software from scratch, but rather to improve and fix current software. On top of the sections explained below, each block will also include short time to re-evaluate project plan.

Definition

Requirement analysis will be conducted prior to fixing the issues. During the project, requirement document (requirements list) will be kept up-to-date of the bugs/features to be fixed and each new test version will be designed based on the requirements list. Team will work closely with the client to keep requirement list updated and well defined.

Design

Goal of the design is to have as accurate implementation of the requirements as possible, without sacrificing efficiency, security and stability. Designing will happen concurrently with other activities and will partially overlap between test versions, depending on how much work is required and based on priorities defined by the client. Due to the nature of the project, designing will mainly be done by the inviduals working on problems. This is because generally speaking bugs/features to be implemented are well defined and usually isolated in nature.

Production

Programming will be done based on the design. Usually by 1-2 inviduals.

Testing

Testing of current test version will happen concurrently with definition/design and production of the new test version. The result will be updated project list for next test version and testing document. On top of this, each new feature/bug fix will be unit tested by programmers.

Secondary Process: Analyzing Software

Problem with this process is that client has need for an analysis software to run under windows XP platform. Client already has software that runs under DOS. Making current software run under windows XP is sufficient for client and team researched this possibility. For this to happen we needed accurate information of problems with this from client. First step, then, for analysis software project was to determine wheter we will attempt to get the old software working with modern systems, or produce a new software.

Fairly late in the project we determined that there was a real need for new analysis software and decided to implement it. With only five weeks remaining, waterfall model is the way to go, with definition, design, production, and testing following each other in quick succession. Also, we will try to get a workable version delivered to the client as fast as we can for acceptance testing, so final implementation phase will overlap with testing process somewhat.

Estimated Size

Exact size is due to the hazy nature of the project impossible to guess even remotely accurate, since most work is done on isolated blocks and implementing small features and fixing well defined bugs. Not to mention analyzing software may not need any new code at all. It is, however, my rough estimate that new lines of code will be in the 1000 - 1500 range, without taking analyzing software into account. With the new software, we might be talking about figures like 2000 to 3000 new lines of code (both processes combined).

Timeline

Since we have decided to follow iterative model, project is chopped into small blocks. Due to the nature of the project I have decided to keep the blocks small and concice. Roughly 2 weeks per block will enable us to have 7 iterations with some room to spare. Due to the size of the pics, it was impossible to include them here. Hence, only hyperlinks are offered.

Project starts 16.1. and ends 5.5.2006. Lasting roughly 14 weeks.

16.1. - 28.2.

See Attachment 1

15.2. - 28.3.

See Attachment 2

15.3. - 2.5.

See Attachment 3

Important Days

- Monday prior to releasing new test version.
  - Evaluation of the progress
  - Assignment of jobs for new phase
  - Feedback/discussion

- Monday middle of phase
    - Evalution of the progress
    - re-adjustment of schedule if necessary
    - Feedback/discussion

- Wednesday, end of phase
    - Testing of the new test version commences and hence starts a new iteration

Risk Analysis

Some of the risk analysis was done by previous group working on same project. As our goal is to improve same project, some of the risks are same.

Team

Risk: Communicating in foreign language (english).
Propability: Low
Seriousness: Low
Countermeasures: Have people with sufficient english skill to communicate with the client.

Risk: A team member gets sick.
Propability: Medium
Seriousness: Medium
Countermeasures: Good communication and documentation, so that people can take over if necessary.

Risk: A team member quits the project.
Propability: Low
Seriousness: High
Countermeasures: Socialize and familiarize with other members of the team.

Project Management

Risk: The estimations (size of work, skills) are incorrect.
Propability: High
Seriousness: Medium
Countermeasures: Keep and eye on the project plan and update it as necessary.

Risk: The project will not stay in schedule.
Propability: High
Seriousness: High
Countermeasures: Keep an eye on the project list and if necessary, re-prioritize/refactor jobs. If it seems like the goals set forth at beginning will not be met within reasonable timetable, prepare it so that another team can continue, if necessary.

Risk: Communication is not adequate.
Propability: Medium
Seriousness: Medium
Countermeasures: Arrange regular meetings between the team members. Follow closely what everybody is doing at the moment and what the results are. Encourage asking questions and raising up issues.

Risk: Participants are not interested in the project.
Propability: Low
Seriousness: High
Countermeasures: Keep everybody informed about the status of the project to keep their interest up. Show the current results for the client when there is something to show. Arrange interesting work for the team members.

Risk: Some personal interest takes the time of a team member.
Propability: Medium
Seriousness: Medium
Countermeasures: Follow closely what everybody is doing. If somebody does not show progress, find out what is wrong. Tell others if you know that you will not have enough time.

Technology

Risk: The old source code and the operation of the machine is not being understood well enough, and there will be problems in the reuse of the code.
Propability: Medium
Seriousness: High
Countermeasures: Allocate more people to have a look at the source code. Discard the old source and start from a scratch, if the inclusion of legacy code proves to be too risky.

Risk: The SQUID gets broken.
Propability: Low
Seriousness: High
Countermeasures: Sufficient testing capabilities need to be arranged, so that program can be tested even without SQUID. Test the code well before trying it with the real machine.

Risk: New programming languages and obstacles in learning them.
Propability: medium
Seriousness: low
Countermeasures: One language only, that is determinated at the beginning. Vigorous testing and analyzing of the (new) code. Directing jobs to people who have familiarity with the topic.

Risk: If some need to learn using new tools, they will be slow in using them.
Propability: Medium
Seriousness: Low
Countermeasures: Take the learning curve into consideration and adjust the schedule. Make one or two people learn the new stuff well, so that others won't need to spend as much time on it.

Risk: A person is not skilled enough in the task that has been assigned to him.
Propability: Low
Seriousness: Low
Countermeasures: Assign that person to do some other work where he is better. Try to utilize the special skills of the team members as much as possible, especially in case of a difficult task.

Client

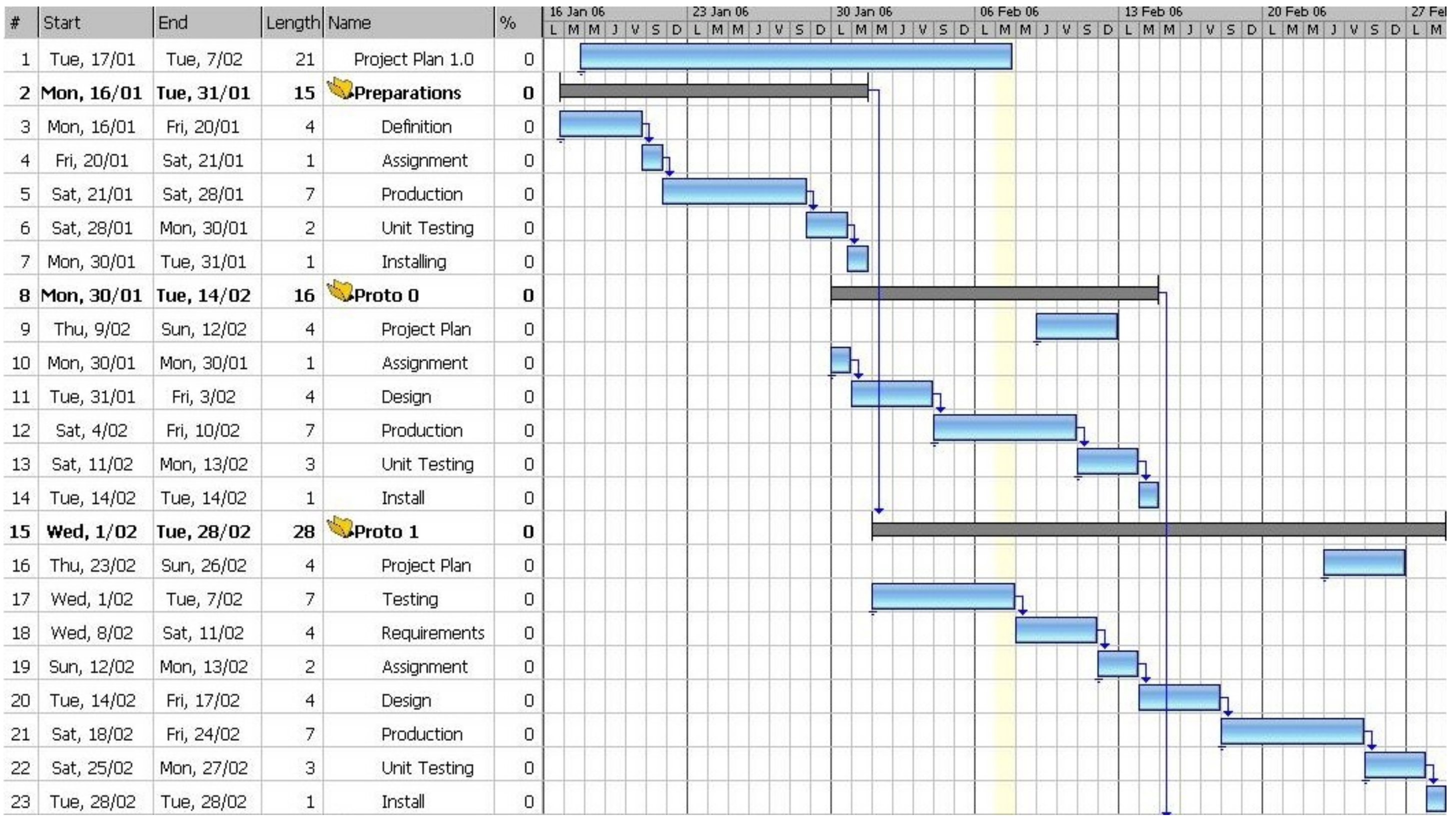
Risk: The project team will not understand how the magnetometer is being used.
Propability: Medium
Seriousness: High
Countermeasures: Check all decisions concerning the program design with the client. Build an accurate prototype of the system and make the client accept it before writing code.

Risk: Client unavailable when needed.
Propability: Low
Seriousness: low
Countermeasures: Have clients inform the team of schedules and unavailability. Have team as well informed as possible. Maintain independancy as much as possible and try to make each client meeting as fruitful as possible.

Risk: The client will not be satisfied with the finished product or the product will be useless.
Propability: Medium
Seriousness: High
Countermeasures: Build prototypes and write documents that will be accepted by the client. Test the program thoroughly. Have clients test the program during the project, so that any new problems we face, we'll be able to accomodate to.

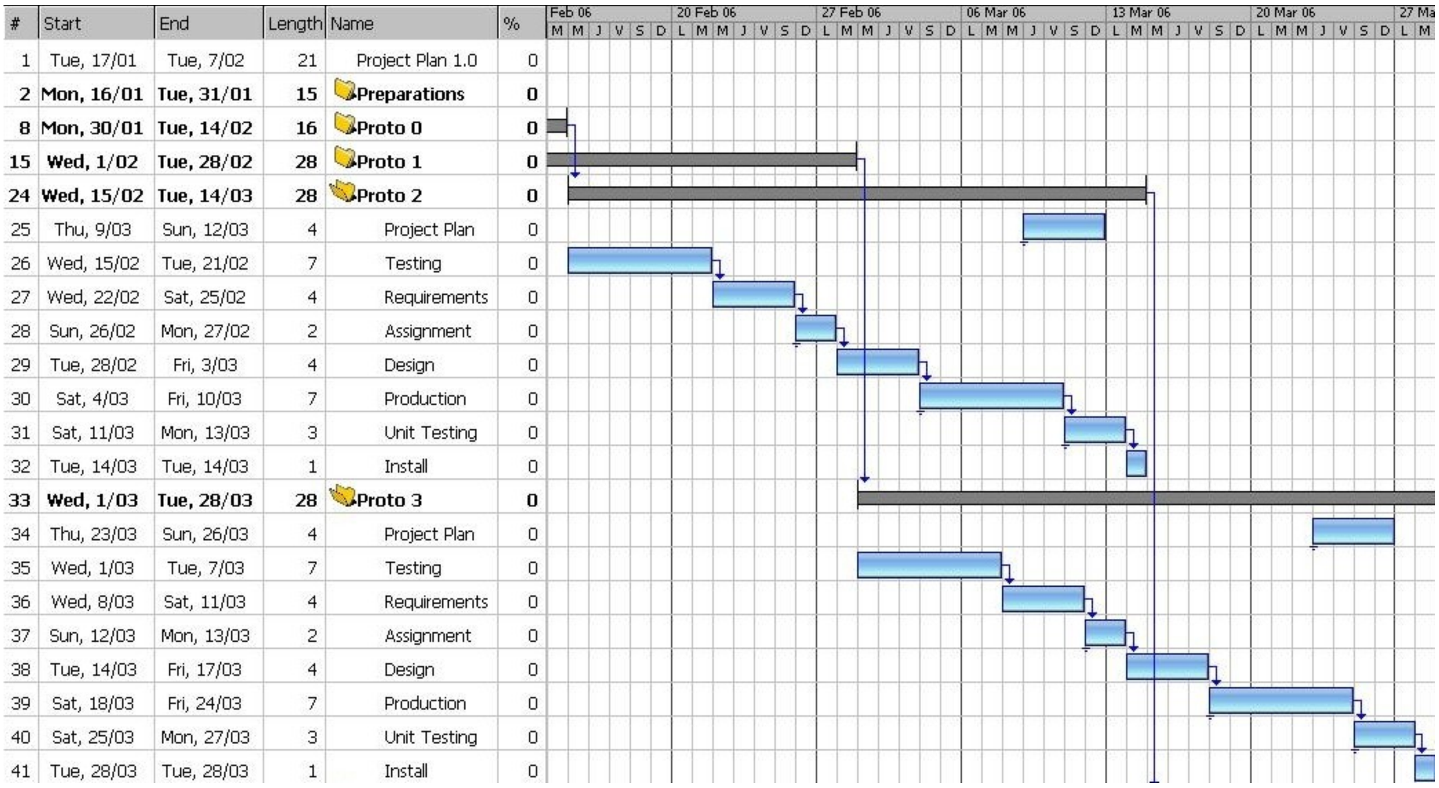
Risk: The client does not understand the prosess of making software.
Propability: High
Seriousness: Low
Countermeasures: Explain to the client what we are doing. Remind the client not to expect too much.

ATTACHMENT 1: **16.1. - 28.2.**



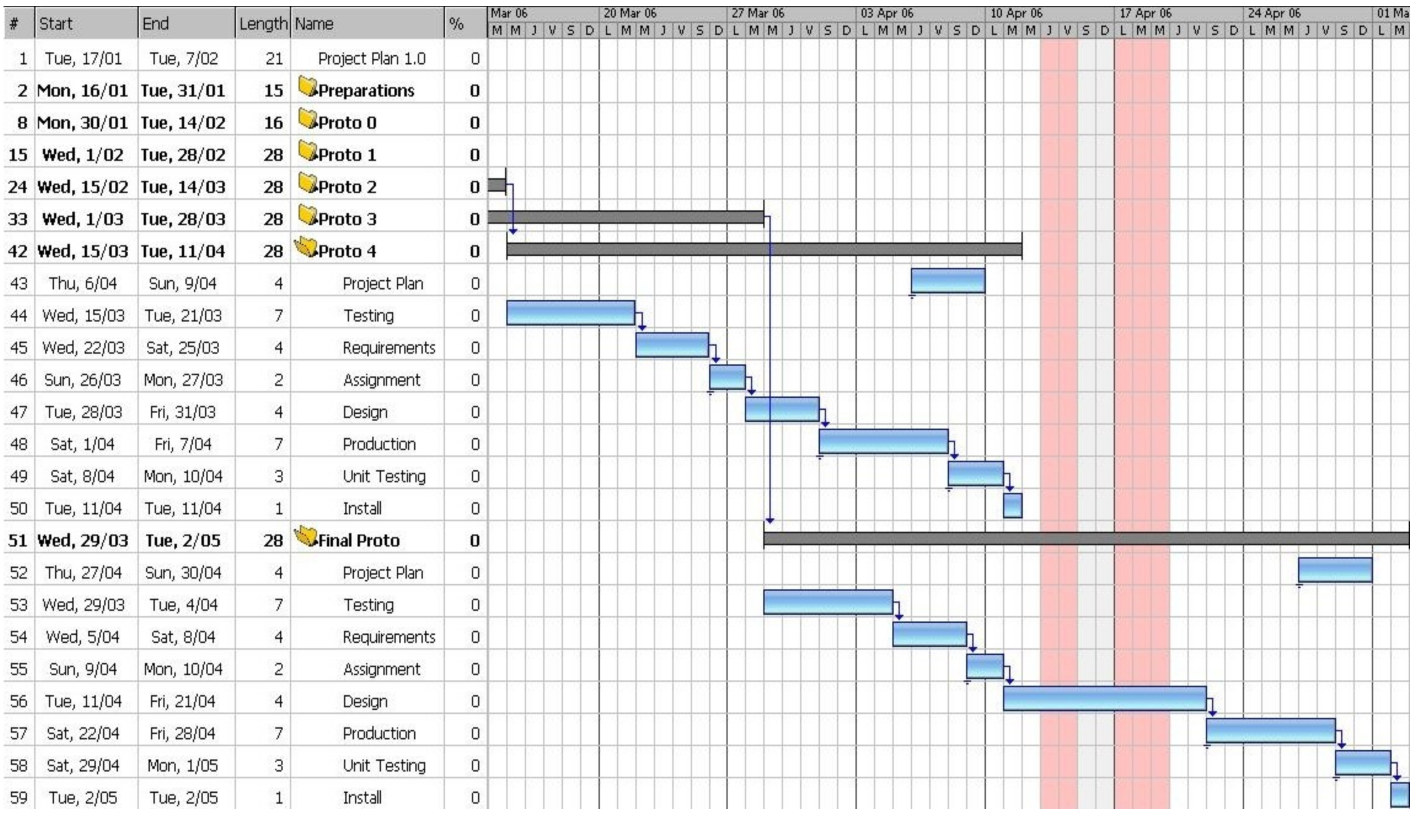
#	Start	End	Length	Name	%
1	Tue, 17/01	Tue, 7/02	21	Project Plan 1.0	0
2	**Mon, 16/01**	**Tue, 31/01**	**15**	**Preparations**	**0**
3	Mon, 16/01	Fri, 20/01	4	Definition	0
4	Fri, 20/01	Sat, 21/01	1	Assignment	0
5	Sat, 21/01	Sat, 28/01	7	Production	0
6	Sat, 28/01	Mon, 30/01	2	Unit Testing	0
7	Mon, 30/01	Tue, 31/01	1	Installing	0
8	**Mon, 30/01**	**Tue, 14/02**	**16**	**Proto 0**	**0**
9	Thu, 9/02	Sun, 12/02	4	Project Plan	0
10	Mon, 30/01	Mon, 30/01	1	Assignment	0
11	Tue, 31/01	Fri, 3/02	4	Design	0
12	Sat, 4/02	Fri, 10/02	7	Production	0
13	Sat, 11/02	Mon, 13/02	3	Unit Testing	0
14	Tue, 14/02	Tue, 14/02	1	Install	0
15	**Wed, 1/02**	**Tue, 28/02**	**28**	**Proto 1**	**0**
16	Thu, 23/02	Sun, 26/02	4	Project Plan	0
17	Wed, 1/02	Tue, 7/02	7	Testing	0
18	Wed, 8/02	Sat, 11/02	4	Requirements	0
19	Sun, 12/02	Mon, 13/02	2	Assignment	0
20	Tue, 14/02	Fri, 17/02	4	Design	0
21	Sat, 18/02	Fri, 24/02	7	Production	0
22	Sat, 25/02	Mon, 27/02	3	Unit Testing	0
23	Tue, 28/02	Tue, 28/02	1	Install	0

ATTACHMENT 2: **15.2. - 28.3.**



#	Start	End	Length	Name	%
1	Tue, 17/01	Tue, 7/02	21	Project Plan 1.0	0
2	**Mon, 16/01**	**Tue, 31/01**	**15**	**Preparations**	**0**
8	**Mon, 30/01**	**Tue, 14/02**	**16**	**Proto 0**	**0**
15	**Wed, 1/02**	**Tue, 28/02**	**28**	**Proto 1**	**0**
24	**Wed, 15/02**	**Tue, 14/03**	**28**	**Proto 2**	**0**
25	Thu, 9/03	Sun, 12/03	4	Project Plan	0
26	Wed, 15/02	Tue, 21/02	7	Testing	0
27	Wed, 22/02	Sat, 25/02	4	Requirements	0
28	Sun, 26/02	Mon, 27/02	2	Assignment	0
29	Tue, 28/02	Fri, 3/03	4	Design	0
30	Sat, 4/03	Fri, 10/03	7	Production	0
31	Sat, 11/03	Mon, 13/03	3	Unit Testing	0
32	Tue, 14/03	Tue, 14/03	1	Install	0
33	**Wed, 1/03**	**Tue, 28/03**	**28**	**Proto 3**	**0**
34	Thu, 23/03	Sun, 26/03	4	Project Plan	0
35	Wed, 1/03	Tue, 7/03	7	Testing	0
36	Wed, 8/03	Sat, 11/03	4	Requirements	0
37	Sun, 12/03	Mon, 13/03	2	Assignment	0
38	Tue, 14/03	Fri, 17/03	4	Design	0
39	Sat, 18/03	Fri, 24/03	7	Production	0
40	Sat, 25/03	Mon, 27/03	3	Unit Testing	0
41	Tue, 28/03	Tue, 28/03	1	Install	0

ATTACHMENT 3: **15.3. - 2.5.**



#	Start	End	Length	Name	%
1	Tue, 17/01	Tue, 7/02	21	Project Plan 1.0	0
2	**Mon, 16/01**	**Tue, 31/01**	**15**	**Preparations**	**0**
8	**Mon, 30/01**	**Tue, 14/02**	**16**	**Proto 0**	**0**
15	**Wed, 1/02**	**Tue, 28/02**	**28**	**Proto 1**	**0**
24	**Wed, 15/02**	**Tue, 14/03**	**28**	**Proto 2**	**0**
33	**Wed, 1/03**	**Tue, 28/03**	**28**	**Proto 3**	**0**
42	**Wed, 15/03**	**Tue, 11/04**	**28**	**Proto 4**	**0**
43	Thu, 6/04	Sun, 9/04	4	Project Plan	0
44	Wed, 15/03	Tue, 21/03	7	Testing	0
45	Wed, 22/03	Sat, 25/03	4	Requirements	0
46	Sun, 26/03	Mon, 27/03	2	Assignment	0
47	Tue, 28/03	Fri, 31/03	4	Design	0
48	Sat, 1/04	Fri, 7/04	7	Production	0
49	Sat, 8/04	Mon, 10/04	3	Unit Testing	0
50	Tue, 11/04	Tue, 11/04	1	Install	0
51	**Wed, 29/03**	**Tue, 2/05**	**28**	**Final Proto**	**0**
52	Thu, 27/04	Sun, 30/04	4	Project Plan	0
53	Wed, 29/03	Tue, 4/04	7	Testing	0
54	Wed, 5/04	Sat, 8/04	4	Requirements	0
55	Sun, 9/04	Mon, 10/04	2	Assignment	0
56	Tue, 11/04	Fri, 21/04	4	Design	0
57	Sat, 22/04	Fri, 28/04	7	Production	0
58	Sat, 29/04	Mon, 1/05	3	Unit Testing	0
59	Tue, 2/05	Tue, 2/05	1	Install	0