CPSC 3720 - Project Details

Tues	12-Jan-2010	project overview; teams formed
Thurs	14-Jan-2010	interview and plan development, plus weekly meeting (in class); subr
Tues	19-Jan-2010	project plan due
Thurs	21-Jan-2010	user requirements v1 due (Mon); weekly meeting (in class); submit l
Tues	26-Jan-2010	independent review (in class)
Thurs	28-Jan-2010	revised user requirements v1 due; weekly meeting (in class); submit
Tues	2-Feb-2010	system requirements/plan v1 due
Thurs	4-Feb-2010	independent review (in class); weekly meeting (in class); submit log
Tues	9-Feb-2010	revised system requirements/plan v1 due
Thurs	11-Feb-2010	weekly meeting (in class); submit log
Tues	16-Feb-2010	READING WEEK
Thurs	18-Feb-2010	READING WEEK
Tues	23-Feb-2010	
Thurs	25-Feb-2010	weekly meeting (in class); submit log; code review; beta v1 due (Fri)
Tues	2-Mar-2010	final implementation v1 due
Thurs	4-Mar-2010	test plan and forms due (Wed); acceptance testing; re-assign teams
Tues	9-Mar-2010	
Thurs	11-Mar-2010	post mortem; weekly meeting (in class); submit log; project plan due
Tues	16-Mar-2010	final user requirements due
Thurs	18-Mar-2010	independent review (in class); weekly meeting (in class); submit log
Tues	23-Mar-2010	final system requirements/plan due
Thurs	25-Mar-2010	weekly meeting (in class); submit log
Tues	30-Mar-2010	alpha implementation due
Thurs	1-Apr-2010	code inspection; weekly meeting (in class); submit log
Tues	6-Apr-2010	beta implementation v2 due
Thurs	8-Apr-2010	weekly meeting (in class); submit log
Tues	13-Apr-2010	
Thurs	15-Apr-2010	project demos (in class); final implementation v2 due

This project will be implemented in two cycles At the end of cycle 1 I will reassign the teams.
All code will be available to all team members.
At the end of each cyle a mark will be given to each team based on the deliverables and the independent reviews that were performed on each.

PROJECT DESCRIPTION: the client would like a tool to allow her to draw Karnaugh maps of up to 6 variables. She requires a system that allows the entry of variable names and values of 1, 0 or X into each cell in the Karnaugh map. The tool must then minimize the given function to either a SOP or POS form, as specified by the user. The final version must have a GUI interface, although this is not essential in the first cycle.

Milestones - see above timeline. further details are below.

Project launch

At project launch you must plan how the project will proceed. This involves exchanging team information, identifying roles, and deciding on how documentation will be presented and developed. A project plan must be submitted. This must include: 1. introduction 2. project organization 3. risk analysis 4. hardware/software resources 5. work breakdown 6. project schedule and 7 monitoring and reporting mechanisms. For more details see Ch. 5 of Sommerville.

User Requirements

The user requirements should clarify, as much as possible, the requirements from a customer/client point of view. Your document must include three sections addressing functional requirements, non-functional requirements, and domain requirements. If possible measurable metrics (e.g. Fig. 6.5 p. 124) should be included for each requirements, particularly non-functional requirements. See Ch. 6 for more details on each of these.

System Requirements/Plan

Since it is difficult to separate requirements from a system plan, we will incorporate both into one document. This document must include behavioural models (both data processing models and state models) , object models (including inheritance models), class descriptions (possibly generated by doxygen), and all details that the programmer will need to develop the system. Pseudocode and/or doxygen output may be of use. For the development of GUIs pictures and/or screen snapshots are also of use.

test plan

When each version 1 implementation is handed in you will all spend a class being testers. For this you must come prepared with a test plan indicating what scenarios will be tested, under what priority, and also bring forms to record your test results. These will be used to provide feedback to the developers of the project you are assigned to test.

Implementation

Code must be managed using subversion. Code must be documented using doxygen. All of these aspects will be considered, as will ease of use, whether the final product meets the requirements, and any other relevant considerations when a final mark is given for the project.

Project Log/Notebook

I require that your team keep a project log/notebook. All submitted documentation must be duplicated in your project notebook, as must be any correspondence with the client. Copies of submitted logs also must be kept. It would be recommended that you supply all members of the team with a copy of the project notebook, although this may be done electronically if you prefer (e.g. a blog, wiki or other webpage would be a useful way to organize your materials). If this technique is chosen then all stored documentation must be in pdf form.