Software Engineering

Group coursework marking criteria

Summary

The marking scheme is designed to reflect the fact that this is a software engineering project just as much as a coding project. The emphasis is on the process as much as the code. So it may the case that a well-designed and managed project that has is less strong in implementation terms scores just as well as a poorly designed project with a lot of fancy code. Bear in mind that a well -documented project has the potential for another development team to pick up and carry on where the original development team finished. But this is not the case for poorly documented work.

Incremental development process

It is intended that you will conduct your development in a series of incremental phases (sprints in the Agile terminology). Each development phase (sprint) should result in some kind of working prototype. So we will be drawing on the Agile process model, but you can also incorporate elements of the Waterfall process model as you see fit, resulting in a hybrid model. We will discuss these ideas more in lectures. At each stage, you will need to identify some functionality that you aim to deliver. This implies that you need to make steady progress throughout the semester, rather than leave everything to the last minute and then not have enough time to a decent set of development phases.

Marking scheme

Marks are calculated as the sum of a base mark and a peer marking. The rules for peer marking are detailed on Canvas. Each team will need to submit a set of peer marks for all of the members of your group. Remember that if peer marks award zero to a team member, we will interpret this as a signal that the team member contributed nothing to the project. If that is the case then we reserve the right to award the ghost member zero overall for the assignment. Remember to submit your peer marks as part of your overall submission. Failure to produce a set of peer review marks will results in you being given zero for that part of the marking scheme overall. It is perfectly acceptable to award the same peer mark to all group members.

The base mark will be assessed out of 100 for convenience, and then the base mark and the peer marks will be further combined to give an overall grade out of 100. Exactly how they will combined will depend on the range of peer marks, but as a general guide the bias is significantly in favour of the base mark. More information on this is given on Canvas.

Base mark criteria

Deliverable elements		**Max marks available**	**What makes for a good score?**
Planning elements	Project plan (PERT or Gantt chart or similar)	5	Clear sense of individual tasks, order of achievement and assignment of team members to tasks.

	Copies of notes, actions and minutes to evidence team organisation	5	Clear evidence of setting short terms goals and achievement of goals. Evidence of distribution of tasks among team members and decision making.
Process documentation	Process document	15	Clear evidence of organisation into incremental development phases. Setting of targets with relative priorities, analysis of risk factors, determination of requirements.
Design documentation	High level designs (e.g. high level components and use cases)	5	That the design permits the achievement and delivery of the high level requirements whether expressed through textual requirements, use cases. Evidence of validation of the design using sequence diagrams, structured walkthroughs.
	Lower level designs (e.g. class and sequence diagrams)	5	
Software documentation	Electronic documentation (e.g. Javadocs or other text descriptions of classes and relevant APIs)	10	Completeness of documents covering key sections, describing key aspects of core components. Javadocs is one very good way of doing this – there are other ways also.
Code	Full code in electronic formats	40	General good coding practice and that the code produced reflects any design documentation.
Testing documentation	Testing strategies Details and evidence of unit level testing Details and evidence of system level testing Summary of know issues or bugs Screen shots and evidence that the software is working	10	Demonstration of understanding of unit level and system level testing. Identification of edge type test cases. Linking system level tests back to requirements documentation, user stories or the original user requirements. Linking unit tests back to design documentation.
Report	Short report describing any other issues	5	Clear summary of what works and what does not, and description of relative successes and failure of the project as a whole.

A small element (typically no more than 5-10%) of each section is reserved for those teams that deliver to an exceptional standard adding relevant value above and beyond the original specification.

Dr Kingsley Sage
khs20@sussex.ac.uk
January 2020

END