# Software Engineering G6046

# Group coursework marking criteria

**Summary**

Several teams have asked for some guidance on the level of functionality required in the Software Engineering project to achieve a grade. This document provides some further clarification of the marking scheme. The document is provided for guidance only. There are other possible factors that can influence your overall mark, such as quality and ease of use and general good code craft. But it should give you an idea of how your team is performing. This guidance only applies to the determination of the project team base mark. The peer assessment process is used to calculate your individual mark, in accordance with the principles published already for this project.

| **Element: Planning docs (out of 5)** | |
|---|---|
| 0 | No evidence of any planning presented. |
| 1-2 | A basic list of tasks has been produced, but there is no attempt to organise them into a coherent project plan. |
| 3 | At this level, there will be a project plan in the form of a PERT/Gantt chart with some attempt to determine the critical path and delivery dates. |
| 4 | At this level, there will likely be evidence of an attempt to identify risk factors using a formal risk analysis methodology. |
| 5 | Excellent evidence of planning. There will be a clear sense of individual tasks, order of achievement and assignment of team members to tasks. There will also be some attempt to measure how well the team managed the plan in reality. |

| **Element: Planning – meeting notes (out of 5)** | |
|---|---|
| 0 | No evidence of any meetings, discussions or agreed actions |
| 1-2 | A basic list of meetings has been produced, that records who was present at them (this is helpful where there is an issue with the peer review process). |
| 3 | At this level, a useful basic set of meeting notes has been produced recording essential actions that were agreed at each meeting. |
| 4 | At this level, the notes will likely observe whether actions were completed. |
| 5 | A full set of notes that demonstrate an organised team, reflecting on progress on the project plan, and monitoring of risk factors identified, with appropriate mitigation through decision making. |

| **Element: Process documentation (out of 10)** | |
|---|---|
| 0-2 | Limited evidence of any application of a meaningful Agile process. At this level, it is likely that a team just “went for it” without any real process model at work. |
| 3-4 | At this level, there is some evidence of a series of organised sprints, but the amount of detail is limited. The purpose of each sprint is vague, and there is no real sense of the journey from user stories and requirements through to sufficient design. |
| 5-6 | At this level, sprint documents will show evidence of user stories, use cases or similar that can be linked back to the original set of user requirements. Each sprint will have a clear set of objectives and it can be seen to what extent the sprint met its objectives. |
| 7-8 | At this level, the sprint documentation is well organised and there is a clear sense of progression through a series of prototypes. There will be clear evidence of the design process at each sprint (likely linked to the design documentation). There is clear evidence that testing and evaluation of the prototype has occurred at the end of each sprint cycle. |
| 9-10 | A full and proper set of sprint documents have been produced that give clear and transparent oversight of the work of an organised team following an Agile process. There will be a proper evaluation at the end of each sprint, reflecting how the process will be improved for the next sprint cycle. |

| **Element: Design documentation – high level (out of 5)** | |
|---|---|
| 0 | No evidence of any design. The team has likely pursued a “burst activity approach” to solving the problem. |
| 1-2 | At this level, there should be some attempt to provide a high level design in terms of major functional components. |
| 3 | At this level, there is a discussion of the role of each high level component in the design. |
| 4 | At this level, key interactions between high level components will be identified. |
| 5 | At this level, there will be clear evidence of high level design expressed as use cases, activity diagrams and similar for key processes and procedures in the project design. |

| **Element: Design documentation – low level (out of 5)** | |
|---|---|
| 0 | No evidence of any design. The team has likely pursued a “burst approach” to solving the problem. |
| 1-2 | At this level, there is likely some attempt at a class diagram, but the content is limited and does not correspond well to the delivered codebase. |
| 3 | At this level, there should be a clear class diagram (that corresponds with the codebase), and a clear sense that OO principles (e.g. high cohesion, low coupling, polymorphism) have been applied effectively. For non OO implementations there should be a clear description of the high level functionality required, and details of key Application Programming Interfaces (APIs) for those key components. |
| 4 | At this level, there should be some attempt to validate the high level design. |
| 5 | At this level, there will be evidence of proper validation of the design using sequence diagrams, structured walkthroughs and similar. There will be clear evidence of the design in the codebase i.e. the class diagrams and sequence diagrams reflect properly how the codebase operates, |

| **Element: Software documentation (out of 5)** | |
|---|---|
| 0 | No attempt has been made to provide useful software documentation. |
| 1-2 | There is limited code commenting, and little or no structured code documentation. |
| 3 | At this level, each method/function has a proper commented description. |
| 4 | At this level, there will be proper structure documentation e.g. JavaDocs for Java, or DocString for Python, although the documentation may not be 100% complete. |
| 5 | Full and substantive structured code level documentation is provided that would help further maintenance of the software at a later date, |

| **Element: Testing documentation (out of 10)** | |
|---|---|
| 0-2 | Little or no testing has been performed. |
| 3-4 | At this level, there should be some attempt to show that the software meets the original user requirements. |
| 5-6 | At this level, there will evidence of unit and/or system level testing across more than one sprint. |
| 7-8 | At this level, there will be properly documented testing showing validation against the original system requirements / use cases. |
| 9-10 | Full testing documentation has been provided, noting outstanding issues where relevant. There will be evidence of unit level and system level testing with Identification of edge type test cases.<br>System level tests will be fully linked back to requirements documentation, user stories or the original user requirements.<br>Unit tests will be fully linked back to design documentation. |

| **Element: Code (out of 50)** | |
|---|---|
| 0-10 | Little progress has been made. At this level, there may be some useful code, but it likely does not compile or run correctly. |
| 11-20 | At this level the code should compile and/or run, but there is unlikely to be a game that can be played in a manner recognisable as The World Conquest. At the upper end of this band, there will be evidence of organisation of code into classes/other key high level functional units. |
| 21-30 | At this level there will be a basic working game. At the lower end, it may be command line driven. At the higher end there will be a Graphical User Interface (GUI). A basic game should include a board, players, tokens, the ability for players to take a turn, roll and collect game cards. The board should be recognisable as a game of World Conquest. At the higher end, the ability to attack and do battle should be implemented. |
| 31-40 | At this level, there is a complete working game for human players. There may be some incomplete elements of the game (documented in the testing notes). At the upper end, there may be a basic option for an AI player, even if it makes decisions on a random basis, and some basic options for customisation e.g. change the player names and token designs. There will also be substantive implementations of some of the more challenging game aspects and variations e.g. secret missions The code will be well organised and commented, and will likely reflect properly the design documentation. |
| 41-50 | At this level, there is a fully featured game that meets most/all of the user requirements. There should be an AI player with some aspects of more sophisticated decision making. At the upper end, the game may work across multiple devices, have an animated visual appearance and incorporate significant options to customise the game. |

| **Element: Group report (out of 5)** | |
|---|---|
| 0 | No group report provided. |
| 1-2 | A basic report that summarises work presented is provided, but with little other observation or reflection. |
| 3 | A useful report that summarises key achievements and outstanding issues, but with not much more critical analysis (e.g. what could be done about these issues). |
| 4 | A useful report that summarises key achievements and outstanding issues, combined with some proper critical analysis (e.g. what could be done about these issues). |
| 5 | A full account of the work of the group is provided, showing how the team was organised, who was responsible for each task (making the peer assessment easy to agree with), and considers what went well, and how the team would do things differently next time. |

**END**