Formal Approaches to Change Management:

To minimize the amount of time that was spent learning the implementation used in the inherited code, we chose a project that was written in C# and built in the Unity engine, which we already had experience working with.

After deciding which project to work on for this assessment, the current state of the software's implementation was thoroughly examined and compared to the project's requirements to determine exactly which features had yet to be implemented. From this analysis, a list of implementation tasks was created. Each task was given a priority, and any dependency between tasks was identified. As development proceeded, this list of tasks was consulted to coordinate work on various parts of the software.

Because the inherited code encapsulated the various behaviours of game elements using different scripts for each behaviour, we decided to do the same for the implementation of new features, despite leaning more toward a class-based implementation in previous assessments. This was done to maintain internal consistency in the code, and because overhauling the code to operate using classes would take more time than we could afford to spend refactoring the code.

To minimise the impact that new features had on the functionality of the inherited code, most new features were implemented in new scripts rather than by extending existing scripts. Some features, however, were much easier to implement by modifying part of an inherited script. To facilitate such extensions of the inherited code, much of the code was heavily refactored to better suit the programming conventions (i.e. using familiar naming conventions for methods and variables) that the team was already accustomed to.

While our approaches to the implementation of the code was greatly affected by the change in source code, our strategies for planning, task allocation, and risk management remained largely unaffected. Since the inherited project is similar to our original project in terms of subject matter, timeline, and scope, most of our team and project management strategies could be carried over from the previous project.

Changes to GUI Report:

Original GUI Report from Assessment 2: [link]
Updated GUI Report for Assessment 3: [link]

Since taking ownership of the project, we have carried forward the GUI for the main game, with some notable improvements which are reflected in the updated GUI report. The unit assignment buttons on the edge of the screen were removed, as these were for debugging purposes and had no place in the functional game. A photo was added for each sector, which is displayed in corner of the screen under the selected sector's name. We have also implemented a timer for each turn, which is displayed in the bottom-right corner. The chance card counter and GUI element are now functional: the card changes colour according to which player's turn is in progress and the number of cards that they hold is displayed on it. Also, a text field was added to indicate which of the possible chance card effects occurs when one is used.

The game now has a main menu, which allows for loading a previously saved game, and lets the user choose the number of players when starting a new game. We have also changed the in-game menu slightly, removing the options to change video and audio settings, as these have not been implemented.

Changes to Testing Report:

Original Testing Report from Assessment 2: [link]
Updated Testing Report for Assessment 3: [link]

The inherited project included a number of hand run tests instead of unit tests written in C# within unity. Rather than completely rewriting all of our testing, we decided to continue testing in the style of the previous team, adopting black and white box testing done manually. The testing plan was reorganised to group tests for specific aspects of the game together. Some tests that we deemed unnecessary were also removed from this plan, such as a testing that a sector is on the map for every single sector. These tests were removed as it was also being tested in other tests involving unit allocation and conflict resolution which would fail if the sector was not there.

Various tests were added to the plan to include our new features such as adding tests for 3 and 4 player games, tests for the PVC and minigame. We also expanded a number of tests into multiple tests like testing each button (such as the help, settings, save and exit buttons) in the game instead of a single test for all buttons in the game. A main menu for the game was also added in this assessment which also had tests written for. We also changed how landmarks worked in the game so they would no longer give bonuses and so tests surrounding the bonus mechanic had to be altered too. Chance cards were not implemented when the project was received so this feature was added and tests were also written for this feature too.

Changes to Methods and Plans:

Original Methods Selection and Planning from Assessment 2: [link]
Updated Methods Selection and Planning for Assessment 3: [link]

Most of our methods and plans did not change after moving on assessment 3, taking on a different team's project. Our team worked as it had throughout this project, following our adoption of the agile development method and working in scrums. We felt there was no need to make any changes to our methods because the team had operated very well so far on our current method.

The project that was chosen had also opted to use Unity as their game engine and development platform which meant that no new tools would have to be learned or to adapt our method or plan for new software. Another reason for not making changes to our methods is that the project we chose was also developed using and agile approach and scrums. Having the same method also meant it was simple to move across to this new project.

We also decided that there was no need to alter our established task schedule for assessment 4, as shown in the Gantt Chart below.

Project section	Priority	Spr Week 7	Spr Week 8	Spr Week 9	Spr Week 10	Eas Week 1	Eas Week 2	Eas Week 3	Eas Week 4	Sum Week 1	Sum Week 2	Sum Week 3	Sum Week 4
Assessment 4 - Sum Week 3 Wed + presentation													
Phase 1 - select other product	high												
Phase 2 - requirements change	high												
Implementation	medium												
documented code													
summarise software changes for new requirements													
Architecture and traceability report	medium												
models, explanations and justification													
changes from design													
provide URLs													
Evaluation and testing report	high												
justify approach to evaluation and testing													
product meets/does not meet requirements													
provide URLs													
Project review report	high												
summarise approach to mangement and structure													
summarise SE development methods and tools													
Presentation	high												
create presentation													
practise presentation													
give presentation													

Changes to Risk Assessment

Original Risk Assessment Document from Assessment 2: [link]
Updated Risk Assessment Document for Assessment 3: [link]

The changes made to the risk assessment document that we inherited have been fairly minimal, as it was a fairly comprehensive document to begin with and all of the previously identified risks also apply to our team. Risk ownership of course had to be reassigned to the relevant members of our team, rather than members of the previous team.