**Software Manager**

The software manager is responsible for leading the design and implementation of the product. They should fully understand the customer requirements and the company’s functional specification of the project in order to design and develop the product. The software manager will work closely with the Deputy Manager and Documentation Manager to ensure the quality of both the software and any design documentation produced during the project.

Software manager’s tasks include:

* Decide on design and development methods.
* Lead the development of user stories from the customer requirements to produce test driven code
* Develop the architecture of the product from the specification, identifying class hierarchy, objects, data types.
* Produce appropriate documentation of the design process.
* Collate product requirements for inclusion in the Project Wide Standards
* Coordinate the coding efforts of other group members and monitor its implementation.
* Ensure consistency in the code and design practice carried out through the project.

|  |  |
| --- | --- |
| Risk | Mitigation |
| Code failure at some point during the project | Ensure code is backed up to allow reverts to previously working versions of the code. Conduct regular reviews of the code and work with the Test and Integration manager to provide test cases to indicate failures. |
| Software Manager is unavailable | Ensure all designs and user stories are documented and accessible to other members of project. |
| Programmers are unavailable during the project | When user stories are assigned to an iteration allow time for delays. |
| Disagreement between programmers about implementation of code | Have a clear implementation standard to provide a consistent code design strategy. Software manager will mediate any disagreements that do arise, and have the final decision on the code implementation. |

**Development Manager Roles**

The Development Manager's role is to be the lead programmer on the project and is responsible for the technical implementation of the user stories created for the project. The Development Manager is also to set an example to other programmers on the project of the coding standards and practices required. If the Software Manager becomes unavailable the Development Manager will take over their role and lead the development of the project.

**Development Manager Responsibilities**

* Lead spikes (investigations) into new technologies or methods that may be required during the project.
* Provide language/syntax based support to other programmers within the company.
* Initial field queries relating to the product code from programmers.
* Should the Software Manager become unavailable step up to fill the role.
* Actively lead the technical implementation of the product code.
* Assist the Software Manager with the design and specification process.
* Verify the quality of product code produced by programmers.
* Provide some level of defect and work item tracking on a weekly basis to interested parties, and formulate actions to improve metrics if required.

**Risks**

|  |  |
| --- | --- |
| Risk | Mitigation |
| Code inconsistencies | Produce a coding standards document and ensure that all team members are briefed on it before coding. |
| Errors or delays due to new technologies required (e.g. HTTP in Java) | Research and conduct spikes into new technologies. |
| Code produced does not meet standards | Conduct regular audits of code produced in order to give advice and instructions about code which needs to be improved. |

**Software Metrics**

|  |  |
| --- | --- |
| Metric | Measured |
| Time spent coding vs estimated time for code | Record time spent working on each user story |
| Completed user stories | Record how many stories are completed/in progress/waiting to begin |
| New Defects within code per week | Record unresolved defects with the date recorded |
| Defects Resolved per week | Resolved defects to be reported and archived when finished. |
| Quality Assured Code | Record when a piece of completed code has been quality assured by either QA managers or Software Manager (Check for code consistency, passes all tests according to Test Standards document, meets user story) |
| Integration test passes | Record the number of test passes each time an integration test is run |

**Requirements & specification Phase**

The customer provides a product idea or concept to the company, who then create a functional specification based on this. The customer should then review the functional specification, identifying areas of confusion or where they feel the specification deviates from their concept. This feedback should be taken into consideration by the company, and the functional specification amended to reflect this. This loop should be iterated over until both the customer and company are agreed upon a final version of the functional specification. The functional specification provides the starting point for the design phase.

The process used to generate the functional specification, is that all members of the company should attend a planning meeting where the customers product idea or concept is outlined. The company then uses a process similar to that of Behavioural Discovery to identify the features that are required to achieve the customers end goal. This consists of rather than looking for what the customer wants from the product, looking at how the final user will use and interact with the product, and then looking for what is missing to make this work. In other words based on the features already found, why won’t the product work? What is missing? What will cause problems? What vulnerabilities are created or haven’t already been accounted for. This should generate quite a comprehensive list of features desired by the customer, and to some extent suggestions made by the company.

**Design Phase**

Based on the functional specification, a series of user stories will be created which describe the functionality of the product as the user would interact with it. The whole company should be present for this stage, as people with different roles often view things from different angles, which provides a much wider perspective from the users’ angle. By creating user stories this allows the work to be broken down into manageable chunks, which can then be assigned appropriately and easily tracked.

The process of generating the user stories is again very similar to that of behavioural discovery described above, however now looking at the next level of detail down the chain. This produces a series of high level user stories, from which it should be possible to implement the product. Each user story is first given an ignorance level from 1-5.

|  |  |
| --- | --- |
| **Ignorance Level** | **Meaning** |
| 1 | This task is extremely easy, and is the kind of task undertaken regularly on a daily basis. |
| 2 | This task is relatively easy, and undertaken now and again. |
| 3 | This task will provide a small challenge, but is similar to those undertaken before. |
| 4 | This task will be will challenging, and is extremely rarely undertaken. |
| 5 | This task or similar have never been undertaken before, and consequently unable to define a starting point for it. |

If an ignorance level of 5 is given to any story, this then becomes a spike (investigation) as the company does not have the required knowledge to implement this feature/use this technique, so are also unable to put an estimate on how long it will take. Meaning this story should then become the highest priority as it is extremely difficult to plan around it. Based on the ignorance level of each story, a SWAG (Silly Wild Ass Guess) or estimate of time required to implement it should be given to each story. This can vary in format depending on the working style adopted by the company. For example if pair programming is to be adopted, then the estimates should be given in “pair days”, if it is a task for an individual to complete then it should be given in “man hours”.

If for some reason the company feels unable to provide a SWAG or ignorance level for a story, then this should be further broken down into sub-stories that are able to be graded. The aim of this method is to spend as little time as possible planning upfront, but to still have planned to a fine enough grain that everyone in the company is confident of what the feature is, that the user story describes.

**Implementation Phase**

The company will use a Kanban style board to track the progress of work items and defects, for all members. The initial version of this is attached in APPENDIX ?, however following agile principles this will change over time, as the company feels that it needs to adapt to the working practices that best suit its employees. A Kanban board allows all employees but especially managers to get a thorough overview of the current state of each employee, and progress on items. It also allows for easy tracking of metrics such as team velocity, and reduces the need for progress documentation.

User stories will be assigned to iterations based on their ignorance level, the highest ones should be done first as they are more likely to over-run. However time estimates must also be taken into consideration, so that all stories proposed for an iteration will actually fit within the iteration timescale. Also some form of priority needs to be considered at this stage so that features with the highest value but also the most complex to implement are done first, down the scale, to features that give very little value to the product so will be implemented last.

User stories will be assigned to programmers based on experience. Depending on the working style adopted, for instance pair programming, a user story should be assigned to the individual with the most experience in this area, who are then paired with an individual who has little or no experience in this area. Although this can take slightly longer to complete the user story, it ensures that knowledge and experience is shared between programmers, which improves the team’s ability to work in the long term.

A coding standards document has been created, and is attached in APPENDIX ? This should guide all programmers as to the quality and style of coding that should be adopted.

**Test & Integration Phase**

Testing of each feature should take place during its implementation phase, as each function or method should be unit tested. The company is adopting the style of Test Driven Development, meaning that a failing unit test is written first, followed by the product code that makes the test pass. In this way features will be thoroughly tested before reaching the integration phase. Further guidance on this can be found in the test standards document attached in APPENDIX ?.

Once a feature is complete the integration testing should take place. This is in the form of a deterministic test pass (DTP), meaning that the core functionality of the product is tested using a set of standard tests with pre-determined outcomes. Should any test from the DTP not achieve the pre-determined outcome, then the integration or new feature has broken the existing product, and will be removed and referred back to the original programmers for further investigation and testing. Results of DTPs should be recorded and provided to the QA Manager at the earliest possible time.

**Flow Summary (to be added beneath 4.0 – project manager diagram)**

* Customer concept or product idea provided to company.
* Functional specification written.
* Function specification checked and amended with customer.
* User stories created using Behavioural Discovery.
* User stories ranked with ignorance level.
* User stories given SWAG (time estimate).
* User stories assigned to pairs or individuals.
* For each user story;
  + Tests written (TDD)
  + Product code written
  + Integration testing takes place – deterministic test pass
* Extra documentation generated if required.
* Ship product to customer.

**Example Kanban Board**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ready** | **Writing Tests** | **Writing Source Code** | **Ready For Integration** | **Integration Testing (DTP)** | **DONE!** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |