Extreme Programming Method: XP is a software development methodology which is intended to make software quality better by adapting to changing customer requirements.

Scenario: I the author of this document and the owner of the companyoversees creating an application for the university registration system. We will have a team of 19 people including 12 software engineers. Our assignment for this project is to merge the BYUI grad plan with the registration system. We need to find an optimum schedule that will decide the number of sections needed for any specific class in that semester.

**Meetings:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Who | Agendas | Purpose | Event |
| **Iteration plan meeting** | Everyone except secretaries and custodian | To assign everyone their role and try to find the optimum process for doing this project.  To plan for the whole iteration and to select *pair programmer* for people who will be writing code. | The main purpose of this meeting is to introduce the user’s requirement to the developer. All the requirements will be kept in index card which can be assigned to different developers later. Feature to be developed in each iteration will be discussed. Also, the eligibility criteria to pass the acceptance testing, which is done after the feature is integrated, will be drafted. | This only happens once at the beginning of each iteration (An iteration consists of planning, designing, coding, testing, and releasing of software.) |
| *Designing meeting* | Software engineer and developer | To draft the design of the software.  To come up with a simple prototype if the project is too complex. | In the meeting, developers will come with the design which will be implemented in coding phase. The design will be very abstract and will be drawn in whiteboard in workplace. | This happens only once in an iteration. It happens after the iteration plan meeting. |
| **Open conversation** | Everyone except secretaries and custodian | Communication among developers | To enhance collaboration among developers. To discuss solutions and ideas. Enhanced pair programming. | This happens throughout the project/iteration. |

**Documentation:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Authors | Audience | Purpose | Deadline and Usefulness |
| *Design* | Programmers and software engineer | Everyone except custodian and secretary. | To understand the connection in coding different properties/classes to get a result.  To provide better clarification and enhances communication among developers and tester. | Until the feature is integrated into the system or until the software is ready. |

**Roles:**

* *Custodian*: As regular custodian will be Chrissy. She will clean the office in the evening.
* *Secretaries:* Stan and Sally will be secretary and will help Rochak for office work. They will be very friendly and help to fill the bridge between Rochak and his worker.
* *Project manager*: The project manager will be Rochak: the owner of the company. To be the project manager one mush have a good management background and business-minded. Should be able to take hard and difficult decisions when needed. Rochak must do all the regular talking with the customer as well as present our product to our customers. Rochak should be good at managing his workers.
* *Software Engineer:* Abe will be a software engineer. One needs to have long experience in software engineering to be in this position. The software engineer will listen to user stories and come with different solutions that can solve the customer requirements. And it designs the best solution and does necessary prototyping if needed.
* **Customer:** A customer will be one who will record users’ stories. To be in this role one needs to be good in understanding user’s requirements. The person in this position will talk to Peter and Patricia regularly. People with experience in technical writing will be the best fit for this position. Teri will be working as a customer. She will be in our team as usual, but she will be working as a customer of our final product.
* **Programmer:** One needs to be good at programming to be in this position. People in this role need to program the feature design by software engineer. People in this role also need to collaborate with software engineers and themselves while designing the feature. Ursula, Xavier, Britney, Claire, Emily, Grace, Ingrid, and Larry will be working as a programmer.
* **Tester:** One needs to good in catching error in code as well as good in analyzing the product and feature developed. People in this role will create automatic testing, do unit testing, integrated testing, and do QA analysis. Doug, Frank, Holly, Jack, and Keith will be our Tester.

**Checkpoints:**

|  |  |  |
| --- | --- | --- |
| Name | Time Estimation | Exit Criteria |
| **Planning** | The total time for planning phase should not cost more than 8 hours. This happens once an iteration. | The following work must be done to successfully exit this phase:   * Features will be developed in the small iterative process so if there are lots of features, features will be priorities according to customer requirements. * The feature developed in the current iteration will be chosen. * Eligibility criteria for the software to pass the integration test should be planned. * These all planning will be taken to iteration plan meeting; with the meeting, this planning phase will be over. |
| **Designing** | This is a very short checkpoint to make sure everybody will be on the same page during coding phase. This takes place after iteration plan meeting. | Following things should be completed to exit this checkpoint:   * A simple design would be explained to developer. A diagram of different classes like flowchart or URL diagrams can be made. * Spike solutions should be explored. * If the software is too complicated a very simple prototype will be made for ease of understanding and enhancing communication among developers.   After doing the required job we have fulfilled the criteria for exiting design phase. |
| **Coding** | Takes most of the time of an iteration. This phase is a repetitive phase. The code generated needs to be unit tested and pass refactoring. Since the integration of new code should not affect the whole system this will continue as long as we don’t pass the integration testing. | The exit criteria for this phase is simple.   * A lot of coding and feature development. * Making sure the code we added does not affect the whole system. * Coding all the features designed in designing phase. * Fixing the error code.   It's easy to exit this phase, just code a feature, but we might need to repeat the process if not passed the testing. |
| **Testing** | It starts with the coding of new feature and ends with successful integration testing | **The exit criteria are simple:**   * Pass each unit testing for the coding done in the previous phase. * Pass the refactoring. * And finally pass the integration testing, and we can exit the testing phase. |
|  |  |  |

**Reflection:**

Since we have small team of 18 including secretaries. I personally think XP methodology will be very good for our project. In a small team like ours, all the developers know each other so it will be easier for pair programming, collaboration and better conversation among the developer. Developers can get feedback faster which will be more effective. In the small team XP can increase courage among developers and can give them the feeling of ownership.

There are some strengths as well as weakness of the XP model; some are listed below:

Strength

* Fewer documentation required.
* Better collaboration with the customer
* Focus on timely delivery of final product.
* Flexibility in the development process and flexibility to developer
* Easy to manage.
* Suitable for a small team
* Better collaboration and communication among the developer
* Optimum solution is obtained because of developer collaboration

Weakness

* It depends heavily on customer interaction.
* Transferring technology to new members and replacing team members can be quite challenging due to lack of documentation.
* This methodology will be difficult to implement if developers are located remotely.

Resources:

* R. Juric, "Extreme Programming and its Development Practices" *22nd Int. Conf. Information Technology ITI 2000*, Jun. 2000   
  [Online] Available: [http://ieeexplore.ieee.org/xpls/abs\_all.jsp?arnumber=915842&tag=1](https://content.byui.edu/file/fb36352f-44a4-473d-bb81-1e5a2ce36646/1/CS%20432%20PDFs/Extreme%20programming%20and%20its%20development%20practices.pdf)
* R. Jensen, "A Pair Programming Experience," *CrossTalk*, vol. 16, no. 3, pp. 22-24, March 2003  
  [Online] Available:[http://static1.1.sqspcdn.com/static/f/702523/9292400/1289016242200/200303-0-Issue.pdf?token=tZDyBy7a9JOdod%2B2GFJ1YEPwVgc%3D](https://content.byui.edu/items/fb36352f-44a4-473d-bb81-1e5a2ce36646/1/?.vi=file&attachment.uuid=15fd2e3c-7950-4c68-b13a-b79898638fd9)
* *Essential XP: Documentation*. [Online]. Available: https://ronjeffries.com/xprog/articles/expdocumentationinxp/. [Accessed: 13-Oct-2019].
* “Extreme Programming.” *Wikipedia*, Wikimedia Foundation, 7 Sept. 2019, en.wikipedia.org/wiki/Extreme\_programming.
* Raman, Sanjeev. “EXtreme Programming The Methodology.” *InfoQ*, InfoQ, 2 Apr. 2014, [www.infoq.com/articles/implementing-xp-methodology/](http://www.infoq.com/articles/implementing-xp-methodology/).

**Self Grading:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Exceptional 100% | Good 90% | Acceptable 70% | Developing 50% | Missing 0% |
| Accuracy 40% | It is completely obvious which development methodology is described. Any knowledgeable person would be able to identify the methodology based on this document. | There is nothing to add and nothing wrong; the development methodology is completely described. One part of the plan may be misclassified as **bold** or *red/italic* | There exists one small problem (factual error or missing component). | There exists one large or multiple small problems (factual errors or missing components). | Large parts of the development methodology are inaccurately described or missing. |
| Application 30% | It is obvious that real thought went into the application (*the red/italic part*) of the plan. | The development methodology is applied to the scenario in an uncontrived way. | Every aspect of the scenario is incorporated into the development methodology. | Large parts of the plan are overly vague, do not appear to be related to the scenario, or do not appear to be related to the development methodology. | No attempt was made to apply the development methodology to the scenario. |
| Reflection 20% | The reflection cuts to the heart of the strengths and weaknesses of the development methodology. | The strengths and weakness of the development methodology are clearly communicated. | One strength and one weakness is mentioned in the reflection. | Little thought or effort was put in the reflection part of the paper. | The reflection part of the paper is missing. |
| Professionalism 10% | The paper is easy to read and ideas are clearly communicated. | Everything is properly cited, there are no grammar or spelling errors, and writing style is "professional." | One instance of a spelling error, grammar error, incomplete citation, overly verbose, poor formatting, or poor writing. | A citation is missing where one is needed (plagiarism alert!). |  |