Group Report on the

CS7CS3 Advanced Software Engineering Group Project

Group: ***5***

18301719	**Aman Ray**
18304203	**Ashutosh Sharma**
18301407	**Bhavesh Mayekar**
18304211	**Lal Singh Dhaila**
18300989	**Roman Shaikh**
18301733	**Udita Retharekar**

This report is intended for you to provide a reflection on the use of eXtreme Programming in your group project. Please give a "real" assessment of the actual benefits and drawbacks of using XP for your project under each of the 12 core practices of XP. Indicate when each category was not applicable, and why.

1. Whole Team

"All the contributors to an XP project – developers, business analysts, testers, etc. – work together in an open space, members of one team…."

How was this applied in your team?

We set up a plan for every iteration for 12 weeks with the stipulated time slots allotted from Mon-Friday. We would book discussion rooms beforehand and work in pairs during these hours.

Benefits

1. Knowledge Sharing – Every member brings something new to the table. The collaboration helps in sharing knowledge and finding creative solutions.
2. Quick Problem Solving – Due to a variety of expertise and skills in the team, solving problems was faster and easier
3. Team Spirit – Working together as a team every day builds trust and promotes healthy risk-taking.

Drawbacks

1. Managing everyone's busy schedule
2. Too much time spent discussing solutions or approach due to different inputs from different members
3. Sometimes difficult to reach a conclusion due to conflicts

Lessons learned

1. Conflict resolution – Learned how to resolve conflicts that arise in a team
2. Collective Ownership – Ownership is not individual but collective. All team members are responsible for all parts of the software lifecycle.
3. Awareness – Each team member had full knowledge about every module/component

2. Planning Game

"Planning is continuous and progressive. Every two weeks, for the next two weeks, developers estimate the cost of candidate features, and customers select those features to be implemented based upon cost and business value"

How was this applied in your team?

We had stand up meetings every day for 15 mins and also at the end of every week. We also maintained a project diary which was updated at the end of every week. At the end of 2 iterations, we would have a meeting to discuss any delays so far and steps that need to be taken to overcome this.

Benefits

1. Time Management – Helped in managing time efficiently based on product backlog
2. Prioritizing tasks – Helped in prioritizing tasks
3. Timely bug fixing – Weekly meetings would help in identifying and fixing bugs on time and avoid surprises

Drawbacks

1. Short iterations led to errors in the estimation of tasks
2. Less predictability as it is difficult to quantify the efforts in short cycles
3. Going off track - Since the iterations have short term goals or no finite end, developers tend to go off track and focus on new deliverables

Lessons learned

1. Plans are not rigid. They change over time and hence one needs to adapt to the changing requirements and plan.
2. Breaking down large tasks into smaller iterations ensures each module is well tested adhering to coding standards and integrated continuously.

3. Customer Tests

"As part of selecting each desired feature, the customers define automated acceptance tests to show that this feature is working"

How was this applied in your team?

As there was no direct access to customer, our TA and module instructor's inputs and suggestions were considered as customer acceptance tests.

Benefits

1. Real world scenario – UATs help in identifying and ensuring the application adheres to real-world situations or scenarios
2. Clarification of goals and objectives – Makes sure all features as per requirements are implemented
3. Goals specified are clear - UAT goals ensures business needs are satisfied in every test.

Drawbacks

1. Changing requirements – Sometimes, customers expect a different or new approach and hence the changing requirements need to be handled and delivered within the given time frame.

Lessons learned

1. Customer inputs and suggestions are always beneficial in ensuring quality deliverables
2. Avoid and identify risks – Risks related to requirements can be avoided.

4. Simple Design

"The team keeps the design exactly suited for the current functionality of the system. It passes all the tests, contains no duplication, expresses everything the authors want expressed, and contains as little code as possible".

How was this applied in your team?

To ensure the code was simple, high-level programming language such as python was used that was easily understandable. To avoid duplication, the code was refactored after every 2 iterations into functional or logical blocks. This also made the code more reusable. Python's PyUnit feature was used to test all the code.

Benefits

1. Readability and easy to understand – Makes the code simple to understand for future releases
2. Reduces code complexity

3. Easy to enhance – Modifying complex code can be onerous and hence simple design makes it easy to enhance or modify existing applications

Drawbacks

1. None

Lessons learned

1. The Django framework is simple to use however, adding new features and languages to achieve tasks led to us making it more complex which could have been avoided.
2. Reusability of code helped in later iterations as it was easy to add new features to the existing application.

5. Pair Programming

"All production software is built by two programmers, sitting side by side, at the same machine".

How was this applied in your team?

A pair programming plan was made and followed through all iterations. Each pair was assigned a different task during each iteration

Benefits

1. Mentoring – Each member of the team had a different skill and hence the collaboration helped in finding a new or improved solution
2. Communication – Improves communication between team members
3. Fresh outlook – In situations where a developer in the pair was stuck on some piece of code, a fresh pair of eyes always helped in solving the problem.

Drawbacks

1. Time-consuming – Sometimes an individual can finish a particular task quicker due to his/her skill in that area, however pair programming can sometimes slow down the task when there are conflicts of ideas between individuals of the pair
2. More effort spent on simple tasks – Certain tasks that are easy and be completed in a short span of time doesn't require the effort of 2 programmers

Lessons learned

1. It helped in making programming more interactive as it encourages a collaborative effort
2. As pairs worked on different tasks during each iteration, the knowledge was equally shared among all team members
3. Each pair developed a sense of responsibility over time

6. Test-Driven Development

“The programmers work in very short cycles, adding a failing test, then making it work”

How was this applied in your team?

The team used python’s PyUnit framework for carrying out test driven development. Before writing any function, a test case for the same was written and tested which would fail and then it was made to work by writing the correct solution.

Benefits

1. TDD ensures that code is tested at every step
2. This in turn improves the overall quality of code
3. Leaves little room for error at the last stage

Drawbacks

1. Test designing is difficult as it is sometimes a bit puzzling to write a test before writing its logic.
2. Test design is time consuming as it is a test first and not code first approach

Lessons learned

1. TDD helps in reducing the overall number of bugs as they are identified even before the code is written
2. TDD also helped in making the application more reliable from the get go. It helped us to concentrate on the next task rather than focus our efforts on testing and fixing a component after it has been implemented,

7. Design Improvement

“Don’t let the sun set on bad code. Keep the code as clean and expressive as possible”

How was this applied in your team?

Refactoring was carried out every 2 weeks where complex piece of code was identified and refactored to make it more easy to understand and implement. Most code was commented to keep it expressive.

Benefits

1. Easy to modify – Simple design is easy to modify when a new developer or programmer tries to add a new feature
2. Easy to read and understand
3. Time saving – A good design or code saves a lot of effort than could have been spent on rectifying bad code later on in the lifecycle.

Drawbacks

1. None

Lessons learned

1. Design or code standard should be decided in the first stage and revisited after a few iterations and not at a later stage to avoid collisions or complexity of code
2. Segregating the code into functional and logical blocks helped speed up coding tasks as the code was highly reusable

8. Continuous Integration

"The team keeps the system fully integrated at all times"

How was this applied in your team?

Git version control tool was used to maintain and track changes in code. Automated deployment scripts were used to update code on server at the end of each day.

Benefits

1. Reduce risk of failure – Continuous integration reduces risk of system failure due to faulty deployment
2. Avoid integration issues in future

Drawbacks

1. Synchronizing commits – It is difficult to integrate commits from different pairs in case

Lessons learned

1. Continuous integration helped in maintaining code uniformity
2. Automated deployment ensured that there were no manual errors in deployment from local to production website
3. Continuous integration ensures all code merged is tested

9. Collective Code Ownership

"Any pair of programmers can improve any code at any time"

How was this applied in your team?

Since pairs was assigned different tasks in different iterations, any pair could easily resolve and improve code as each pair had enough knowledge about the application.

Benefits

1. Minimize bottleneck – Any issues that may cause bottle neck situations (such as failure of the developer to code due to illness) can be minimized as any one on the team can make changes to the code and fix it
2. Balances load – No single individual is responsible for a module or piece of code. This reduces the load on any single individual

Drawbacks

1. Different solutions may be implemented by different individuals which cause code instability
2. Sometimes bad code may also be implemented by someone on the team who may not be well versed with that module

Lessons learned

1. Before changing any code, every team member should be made aware of the changes
2. Traditional approaches put more load and dependency on individual programmers however, this was eliminated due to collective code ownership.

10. Coding Standard

"All the code in the systems looks as if it was written by a single – very competent – individual"

How was this applied in your team?

To ensure all code followed particular standards the PEP8 – python's guide for best practices in coding. Along with this, we conducted code reviews before every integration to ensure all coding standards were followed.

Benefits

1. Uniformity – Ensures that the code is uniform and avoids any inconsistencies
2. Reduce vulnerability – Reduces the vulnerability of code by reducing the complexity

Drawbacks

1. Coding standards enforce rigidity in developing code which may be difficult if people have different coding styles

Lessons learned

1. Ensures consistency and makes it easy to read the code by all the developers.

11. Metaphor

"The team develops a common vision of how the program works"

How was this applied in your team?

We had frequent brainstorming sessions where everyone would discuss their individual ideas and would decide on a solution as a group. This made sure that the best or most feasible idea was chosen.

Benefits

1. Clarity of goals - Every team member has a clear vision of the project goal
2. Time Management - If the goals are set out clearly, it gets easier to manage time and complete tasks accordingly.

Drawbacks

1. Conflicts - There may be many conflicts within the team regarding the approach that can be taken to solve a solution or design a task.
2. Time Consuming - Reaching a common goal can sometimes be time-consuming if the group as a whole does not reach a consensus.

Lessons learned

1. It is always beneficial to have a common goal that has been set out at an early stage in the project.
2. Open-ended discussions that lead to new and innovative solutions and ideas.

12. Sustainable Pace

"The team is in it for the long term. They work hard, at a pace that can be sustained indefinitely. They conserve their energy, treating the project as a marathon rather than a sprint"

How was this applied in your team?

We tried out best to stick to the timetable or schedule set in the first week of the project. The schedule was deisgned in a way that every team member spends approximately 5 hours a day only for the project and overall 20 hours a week. We would also book discussion rooms to avoid any distraction and achieve high productivity in those hours. Therefore we never had to put in extra effort over the weekend.

Benefits

1. Productivity - Proper time management ensured that no member was over worked
2. Delegation of tasks -In case any of the pairs completes their task for the sprint earlier, they can pick a new task or solve any issues in the current sprint and so on to maintain a sustained pace.
3. A fresh perspective is always beneficial which can be achieved if the pace is sustained.

Drawbacks

1. None. As all team members belonged to the same strand the schedule was made to match everyone's time.

Lessons learned

1. Having a good pace ensures there's is no time or schedule slippage which may cause stressful situations in future.
2. Coding or development done in a haste is often faulty or has errors.
3. A fresh mind is always more productive. Therefore the work done is of good quality and finished faster than usual.

13. Overall Project

Benefits

The ASE module helped in understanding the importance of good software principles and practices that need to be followed and how they affect the overall development process. It also helped in understanding how software engineering is implemented in real life organizations.

Drawbacks

None.

Lessons learned

1. Time Management
2. Process Improvement
3. Planning and Organizing
4. Communication
5. Understanding software methodologies and practices
6. Problem Solving