**SOFTWARE ENGINEERING**

**(IT-314)**

**EVENT PLANNER APPLICATION**

SDLC Models

**Team no: 14**

**Instructor: Prof. Asim Banerjee**

**Team Mentor: K Hemantha**

**Date: 14 March, 2016**

**Team Members**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Name** | **ID** |
| 1. | Patel Yash | 201301008 |
| 2. | Patel Tej Naginbhai | 201301018 |
| 3. | Anavadiya Yash Kumar | 201301052 |
| 4. | Chauhan Dwimitra Dharmendra | 201301079 |
| 5. | Rahul Shashank Saranjame (Team Leader) | 201301101 |
| 6. | Fatnani Varun Kumarbhai | 201301152 |
| 7. | Nikita Jain | 201301155 |
| 8. | Karri Kiran Satish Reddy | 201301168 |
| 9. | Raman Kumar Douchaniya | 201301202 |

**Version History**

|  |  |  |
| --- | --- | --- |
| **Date** | **Version no.** | **Authors** |
| 14/03/2016 | 2.0 | Rahul Saranjame |

**Table of Contents**

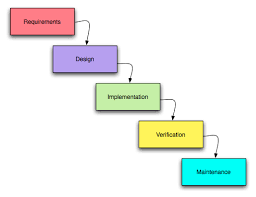
1. Classical Waterfall Model………………………………………………...4
2. Iterative Waterfall Model…………………………………………………4
3. Rapid Application Development Model………………………………….5
4. Prototyping Model………………………………………………………..6
5. Spiral Model……………………………………………………………...7
6. Evolutionary Model………………………………………………………8
7. Incremental Model………………………………………………………..9
8. Concurrent Development Model…………………………………………10
9. Conclusion………………………………………………………………..11
10. **Classical Waterfall Model**

**Merits:**

* Since it is a linear model, it is very simple to implement.
* Easy to manage since each phase is rigid and has a fixed set of deliverables.
* Works well for small level projects where requirements are very well understood.
* Phases are completed one at a time, i.e. only when one phase is done is the next started.

**Demerits:**

* This model does not work well when it comes to complex and time consuming projects.
* Even a small error in the previous stages causes a lot of work to be undone.
* This model does not accommodate scenarios where requirements are at a risk of changing.

****

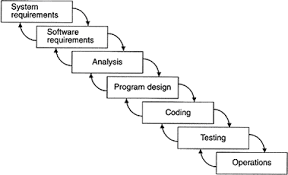
1. **Iterative Waterfall Model**

**Merits:**

* Apart from the merits of the classical waterfall model, it also has the potential to revert back to phases and correct then, in case an error is detected.
* Also, the error, once detected is contained in that particular phase due to which it becomes easy to rectify the error.

**Demerits:**

* The process of iterating can become time consuming.
* In case the requirements are not stable, it can create a lot of problems and chaos.
* The above said scenario mainly arises when the client changes the requirements frequently.



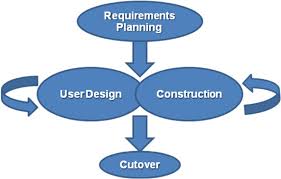
1. **Rapid Application Development Model**

**Merits:**

* RAD model delivers a software product rapidly and components are available and developers are working parallely.
* Changing requirements can be accommodated as planning occurs while coding.
* Progress can be measured.
* Work is highly productive in less time with fewer people.So reduced developement time
* Increases reusability of components as this model is all about using available tools and code pieces.
* Encourages customer feedback.

**Demerits:**

* RAD works well only if high skilled engineers are available and the customer is also committed to achieve the targeted prototype in the given time frame. If there is commitment lacking on either side the model may fail.
* Only system that can be modularized can be built using RAD.
* High dependency on modeling skills.
* Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.
* Management complexity is more.
* Requires user involvement throughout the life cycle.

****

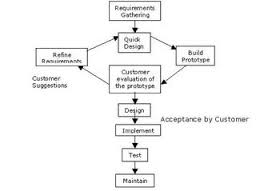
1. **Prototyping Model**

**Merits:**

* Users are actively involved in the development.
* Since users are very much involved if there is any error it can be corrected then and there itself by this we can save lot of time.
* Frequent reviews from the users will help us to accommodate the missing functionalities.
* Since user is involved he will be having better understanding regarding the project which we are building.

**Demerits:**

* Since it involves both user and development team the process of development will be slow due to many reasons lack of time of the client etc.
* Too many changes will disturb the flow of the development team.
* Since there is a chance for frequent changing of requirements the development process would take long time and cost will be high.

****

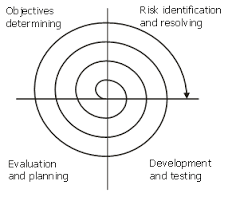
1. **Spiral Model**

**Merits:**

* Similar to the incremental model. Suitable for large and critical projects.
* In this model each requirement is implemented independent of each other so we can add new requirements at any point of time.
* Since we are taking one requirement at a time we are able to showcase the customer that we are able to do something regarding the project.
* The estimation in terms of schedule and cost become very easy as we proceed because we will know the capabilities of the persons who are working in the team.

**Demerits:**

* Cannot be used for smaller projects.
* Since this involves high risks the team who are undertaking this must be highly qualified.
* Cost involved will be high.
* Since it involves high risks rules and protocols should be followed properly in order to execute it efficiently.

****

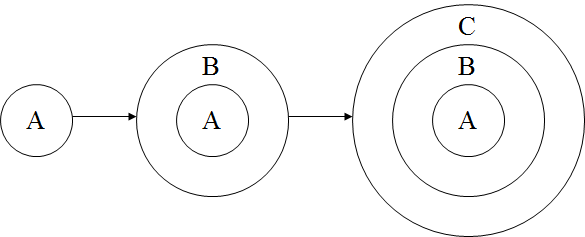
1. **Evolutionary Model**

**Merits:**

* This model is good when customer changes the requirements too often.
* Maintains involvement of customer after each iteration.
* Problems can be handled quickly and solving becomes easy as its defined for that particular iteration only.

**Demerits:**

* There is no clear definition of the project and can lead to too many iterations.
* Project has to be continuously managed and monitored.
* Costly as every member is too involved.



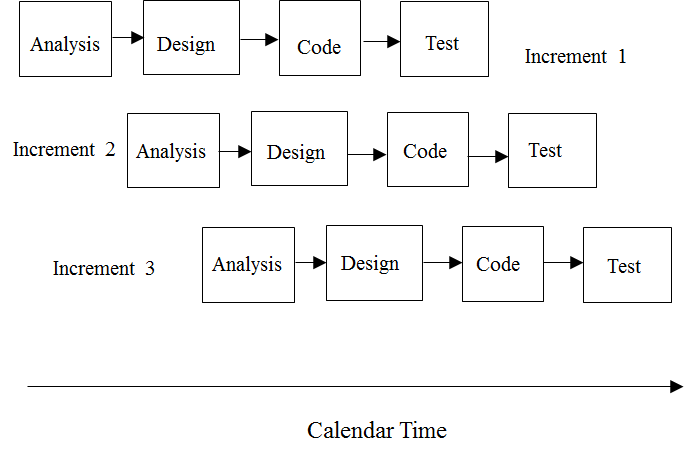
1. **Incremental Model**

**Merits:**

* Useful when not enough staffing is available to complete the project in the given timeframe.
* Early stages can be managed by a few people (this depends on the acceptability of the core product) and the subsequent stages can be managed by adding more resources.
* Increments can be planned to manage technical risks.

**Demerits:**

* Agreement on what features the core product should include is not easy..
* The client may not agree to getting solutions in bits and pieces.
* The process can be very time consuming.

****

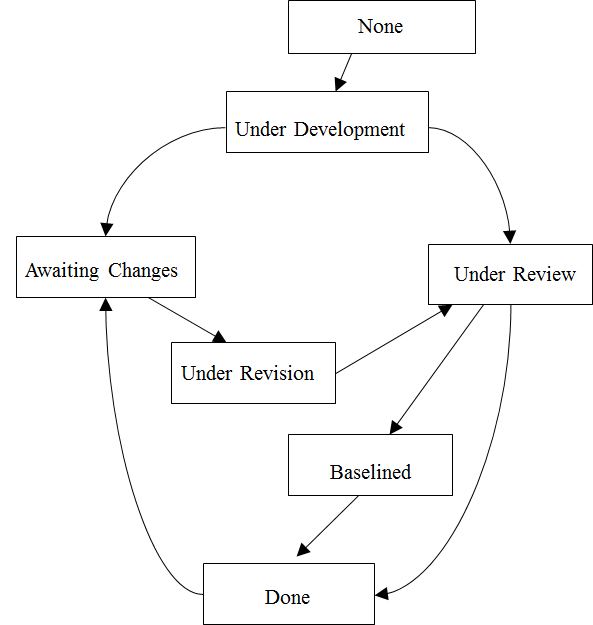
1. **Concurrent Development Model**

**Merits:**

* This model can adapt to all types of software development projects.
* The main advantage is the flexibility offered by this model.
* One can introduce new features in the project even in the later stages.

**Demerits:**

* The fact that the model provides enormous amount of flexibility means that if not used in a disciplined way it can lead to a lot of management problems.
* The requirements have to be continuously monitored since it is very much possible that some requirement will remain unfulfilled.



**9. Conclusion**

After thoroughly reviewing the merits and demerits of each Software Development Life Cycle Model we have decided to follow the Classical Waterfall Model. The main reason for following it is that we have a high amount of certainty about the requirements of the client and there is very less probability that the client will make amends. That considerably narrows down our choice of models. Further, due to time constraints it is preferable if we make use of Classical Waterfall Model since we will not have enough time to do different iterations. We will have to take care that error is minimised so that we do not have to undo a lot of work.