Contents

[] GUS Process Proposal

by Colby Blair, cblair@vandals.uidaho.edu October 23rd, 2010

[] Introduction

In our group planning and in my own research, it seemed clear to me which subset of Agile Development would probably be most popular. The choice is Scrum. Scrum seems to be most adapted to our needs, and fits nicely with a lot of the tooling that exists. Indeed, it seems that there may be a greater amount of tools and documentation to go with Scrum development, because the industry and development world may actually use it more as well. It also incorporates many of the requirements of this class nicely, although perhaps not as explicitly in the wikipedia definitions as one may like. There may be a necessity, therefore, to create additions or extensions to our final process, resulting in a hybrid thatâ€™s foundation is based on Scrum.

[] Problem Statement

For CS 383, we must decide formally on an Agile Development process in which to use for our year long project. The code portion of the project will happen in the second semester, which will be about 4 months long. This is a relatively short development time of a reasonably big project, so it is worth

while here to determine a good process to stick to for the next year.

[] Consideration for Requirements

The first general thing I wanted to say about Agile Development is that I thought Agile was inherently an iterative process. Not so, at least not as defined in some Agile Methods. Some methods suggested some sort of Agile but linear approach where each stage is done once, and some suggested skipping around from stages as desired. Both seemed to have little advantage over unstructured development, so, at best, some of their stages were actually good, and could be used well as substages in Scrum, perhaps.

[] Agile Modeling

The first method to consider was Agile Modeling. I liked part of the idea, with the bulk of the process going into practicing trying to conceptualize the software model with the customer. This takes a lot of fore site, which is actually good. In the definition, it mentions Agile Modeling is used to create a good model and documentation system. The second is problematic for me, because the Agile Manifesto promotes working software over lots of documentation. But Agile Modeling could easily generate a lot of documentation. Still, the author suggests that Agile Modeling is an attempt to correct the shortcomings in Agile. But I buy into the working software instead of documentation, so Iâ€™ ve vetoed this method.

[] AUP

Agile Unified Process was actually my runner up choice to Scrum. Its only downfall to me was that there was only 2 iterations, the development iteration and the production iteration. But, it details its stages well (better than Scrum), and they may align very well with the control and data management structure we end up using. It seems nowadays that web apps end up using model-view-control frameworks (MVCâ€™ s), such as Ruby on Rails or the plethera of PHP MVCâ€™ s (Code Igniter, etc). AUPâ€™ s Model, Implementation, Deployment, Configuration Management, and Environment stages are nicely automated with MVCâ€™ s. Testing and Project Management would still need other tools, but these tools exist and are highly used (PHPUnit, whatever-Unit tool, Trac + Git/SVN, etc).

AUP did have some more drawbacks. One of the philosophies was that â€˜your staff knows what theyâ€™ re doing.â€™ Not us. Also, constant customer involvement is essential, and perhaps beneficial in many situations. We, however, had a hard time getting initial contacts with customers, so constant involvement is probably not suitable for us either. All the rest of the philosophies are laid out nicely and I believe fit nicely with the project, so this is my proposal as the inferior to get hybridized with Scrum.

[] DSDM

Dynamic Systems Development Method seemed the least attractive of the group. Again, user involvement is emphasized, but will be hard for us. It also states the principle of entities making design decisions without higher approval. Again, probably hugely beneficial with a team of experts

that communicate well. Not great for our group with such a large range of talent in different fields. Frequent delivery of products seems like a great idea, and I would recommend making this a requirement of our own as well. The iterative aspects of DSDM are great, but can't really be driven well by our customers. All changes being reversible is a good principle, as well as base lined scopes and requirements, and constant testing. I would propose these make it into our hybrid as well.

[] EssUP

Essential Unified Process seems more dynamic, but all the positive aspects I think are highlighted in AUP, another RUP derivative.

[] Extreme Programming

Extreme Programming seems like it has good properties as well, but may not be enough as an entire process alone for our needs. Two problem with it are good communication with the customer, and very frequent releases. I agree that frequent releases are good, but our group may not need as frequent of releases as Extreme Programming calls for. Good properties of Extreme Programming include: "programming in pairs, doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, simplicity and clarity in code, and expecting changes in the customer's requirements as time passes and the problem is better understood." [1] Some of these are repeats, but still good highlights of what we should implement.

[] FDD

Feature Driven Development is a good method, but falls short. It doesn't mention an iterative portion in its process. It would work great for a customer who is present and would like to see constant progress (not us). Most of the features that were positive here seem to be covered in the Test stage of most of the other Agile methods, and is about all we need. Milestones are a good property of FDD, and should be implemented well in our Project Management stage of our method. Since we are not yet experts or even really sure what features exactly will look like or be included, designing based of them is probably not appropriate for us.

[] OpenUP

Open Unified Process didn't seem to offer more over AUP, and is yet another RUP derivative that I didn't find gave much more towards our needs.

[] Scrum

Finally, Scrum is an Agile Method that uses backlogs over short periods of time to produce working increments of software. This seems to already be a hybrid of other things I liked in other Agile Methods. A master product backlog is maintained throughout the project, and pieces are chipped off to form a sprint backlog. Each iteration of the process are a sprint, and there can be a relatively flat management structure to maintain sprints (2 to 3 tier). Daily meetings of 15 minutes in length are held

for members to update each other on what they have done since yesterday. A sprint planning meeting is held at the beginning of each sprint to plan. Each sprint itself happens every 7-30 days. Review meetings are held at the end of the sprint to reflect on the processes.

Scrum seems like it is very capable of implementing an Agile Spiral. In which, we can stick any extra processes we like to create our own hybrid that still fits the structure of Scrum.

[] Requirements

It is then my proposal by the Consideration of Requirements that the following requirements be considered (properties taken from other methods end with parenthesized citing):

- A Scrum process be used as a foundation
- The process implement an Agile Spiral with main stages:
    - Plan
        - Model/conceptualize the structure and functionality of the software (Agile Modeling)
        - Project Management (AUP)
            - Division of Labor
            - Milestones (FDD)
    - Prototype
        - Model software (AUP)
    - Implement/Develop
        - Pair Programming per Planning stage (Extreme Programming)
        - Implement (AUP)
        - Deployment (AUP)
        - Configuration Management
    - Test and Next Sprint Planning
        - Testing (AUP)
        - Environment (AUP)
        - Spring Review Meeting
            - Random Code Reviews
- The software be developed as an MVC or some sort of entity-boundary-control to fit the development process and the Agile Spiral stages as closely as possible.
- Constant testing in development as often at makes sense (i.e. completion of each function?) (DSDM)
- The process stays as simple as possible, from planning all the way through to testing. (AUP, Extreme Programming)
- Scrum sprints occur every 2 weeks to allow enough time for work, but enough iterations to see whether progress is being made. (Scrum, AUP)
- Tool independence, meaning we should use unit testing and any necessary aides that are written in / specialize in the languages and frameworks we will use (Java uses junit, php uses phpunit, etc) (AUP)
- Delivery of working product by the end of every Scrum sprint (Scrum, DSDM)
- Git is used, and developers use it as best as they can to make changes reversible (DSDM)
- Scrum Master and Managers assign pair programming or let team members volunteer when it

seems beneficial. It shouldn't be something we do only if we feel like, but something we make a point to do as a requirement. (Extreme Programming)
- Random Code Reviews are done in Sprint Review meetings. Everyone needs to be involved in the process. (Extreme Programming)
- The ACM Code of Ethics [2] is adhered to, as well as the Interpersonal Communications Rules of Engagement outlined by Dr. Jeffery. Everyone needs involvement in the entire process to make the boat float.

[] Methodology

[] Evaluation

Evaluation of the process will occur in our Scrum spring review meetings, of which code reviews will also be able to show where we succeed and where we fail. Upon the re-division of labor on the next Scrum sprint meeting, it can become clear which milestones we achieve, and what needs to change in the process for some/all of the members to make the process more productive for them.

[] Dissemination

Everyone in the software team will use the process on some level. Managers and Scrum Master(s) will use the process to help divide tasks and evaluate production, and team members will use the process to gauge where they are and what they still need to achieve.

[] Conclusion

Overall, it seems some sort of hybrid with Scrum is definitely appropriate. None of the methods are all inclusive, but they are all Agile, and Agile is arguably dynamic. So we should be able to somehow combine all the parts we want, and change the process until is works closest to max productivity. We won't have a lot of time to get the process straightened out in battle, so having a good foundation, proposed here as Scrum, is the best battle plan.

[] External Links

[1] http://en.wikipedia.org/wiki/Extreme_Programming

[2] http://www.acm.org/about/code-of-ethics

[3] http://www2.cs.uidaho.edu/~jeffery/courses/383/lecture.html#lecture14

[] References

See External Links