AGILE SOFTWARE DEVELOPMENT METHODOLOGY

Charles Edeki, Ph.D
Bronx Community College, City University of New York
Department of Business and Information System
2155 University Avenue
Bronx, New York 10453
Charles.Edeki@bcc.cuny.edu

ABSTRACT

The agile software development methodology has recently become one of the most commonly used software development techniques. Rather than the long drawn out release cycles in the previously popular waterfall methodology, the agile technique suggests regular short sprint release cycles. This allows the customers and stakeholders to have more involvement within the software development process. This helps promote a higher quality final product because it combats the difficult task of a customer fully understanding and identifying all requirements in the software project planning phase. This also allows for the stakeholders to adjust the priorities of remaining tasks easily throughout the entire software development process. This study described the agile software development methodology and specifically targeted the iterative approach, and stakeholder management.

INTRODUCTION

The iterative approach has become vastly effective in helping software developers improve their skills in estimating schedule for remaining tasks. Schedule estimation is one of the most difficult responsibilities for developers because software issues are common and are unpredictable by nature. By breaking the large requirements down into more manageable sub requirements, the agile process naturally promotes better estimation.

In today's software development world, it is becoming more important than ever to keep the software stakeholders as involved as possible. With any software project there can be countless stakeholders who all can affect or can be affected by the outcome of the software. It is important for the software development team to identify the important stakeholders and find ways to connect with them individually to help promote stakeholder interest and involvement in the product. As agile promotes constant short release cycles, the stakeholders can see continual progress and make suggestions and develop new improved ideas for the software.

Iterative Approach

The agile software development methodology is focused around a short iterative software release cycle. This design is geared toward heavily involving the stakeholders and constantly showing them demonstrations of the current state of the software. This allows for the stakeholders to make recommendations and suggest changes while the software is being actively developed so that the software can track what the customers actually desire. Agile also helps software projects improve the expectations of when software will be completed, determine what items can feasibly go into a release cycle, and provide the ability to easily track overall project progress. The agile philosophy suggests that software development teams attempt to break down large difficult requirements into many small and simple requirements. This allows for easier estimation of time and helps present any potential road blocks in completing the tasks. Decomposing the requirements into very small tasks also provides the benefit of being able to easily determine the percentage of the software that has been completed at any instant in time. This allows the project managers to determine if the project is keeping pace with the expected schedule. If the project does begin to fall behind, the agile process naturally presents the risk immediately which allows for the managers to work with the stakeholders to fine a potential risk mitigation.

Project Timeline

According to Ghule (2014), software development is notorious for poorly predicting the amount of time required to complete the provided requirements (Ghule, 2014, p. 546). One of the leading causes of software schedule slip is due to poor up front planning. When project schedules are first determined in the project planning phase, developers are forced to make estimations on how long a complete program will take to implement. The development team should attempt to break the large overarching requirements down into much smaller tasks in order to estimate each task individually. This will help improve their ability to estimate completion time. However, it is human nature to think about the best-case scenario when determining the estimation. However, the best-case scenario is often an unrealistic expectation.

When planning project timelines, developers should build in room for error when conducting their calculations. The development team should decompose the initial requirements into sub-requirements that they can develop a better estimation on how long each individual sub-requirement will take. Once the software has been broken down into smaller requirements the development team and the stakeholders should attempt to build a rough monthly release schedule. Each month should contain the expected features or enhancements for that month's release.

After all of the top level requirements have been divided up and designated into a planned sprint cycle, the customer must look at the timeline and the expected total man hours and determine if the project's schedule is sufficient and if the budget allows for the project to be completed. If either the project's schedule or budget does not meet the allocated time or resources then the stakeholders should sit down with the development team to attempt to determine if any non-functional requirements can be omitted from the planned development to allow for the software to meet schedule and budget.

Sprint Planning

The agile software development methodology is designed to help software developers improve their ability to estimate the amount of time required to complete a task. This is accomplished multiple ways. The first is through sprint planning meetings. Sprint planning meetings are held at the beginning of each sprint in order to determine which features, enhancements, and fixes should be implemented in the next release cycle. The development team should sit down with the customer and ask them to prioritize the importance of the remaining open tasks for the project.

After the tasks have been prioritized, the team should sit down and as a group and estimate how much effort is required for each task. All team members should be asked to judge the amount of effort based on a relative scale compared to the other tasks. One common technique is to have the developers use shirt sizes as reference points. For example a developer may decide one task is a “medium” and another may be an “extra-large”. Whichever method the team decides to implement, it is important that it is on a relative scale. It is also important to have all team members rate each task rather than just the engineer that it is assigned to. This typically results in discussion amongst team members if there are any discrepancies between their ratings. There may be scenarios that one team member may know of an easier solution to a problem than the engineer assigned did. The is also the possibility of one team member pointing out a potential or known road block to a solution that other team members did not account for. These discussions between team members will help minimize the error associated it estimating software timelines.

The agile methodology also helps train software developers on estimating time after a task is completed. When the task is first assigned to a developer, the developer should write in an estimated time they believe the solution to the task will take. After a task is completed the developer should then enter in the actual time spent on the task and see how accurate their estimation was. If they were off, they will see the difference and should account for that the next time a similar task is assigned to them.

Project Progress

The agile software development methodology helps promote keeping track of the software development process. According to Wysocki (2013), the agile process excels at tracking project progress by having daily or bi-weekly status meetings which keeps the managers constantly informed of the overall progress (Wysocki, 2013, p. 91). The managers then have the ability to inform all relevant stakeholders and ensure that the project is meeting expected schedule and budget.

The project manager can use tools such as earned value management to track the progress of the program. Earned value management estimates the amount of time and money required to finish the project and creates a burn down graphs of the percentage of the software that is complete and the amount of money remaining. However, in order for this to be effective it is crucial that the software development team determine a way to accurately and subjectively determine a formula to determine the percentage complete and the percentage remaining.

Stakeholder Management

The agile development methodology strongly focuses on the best practices on how to involve key stakeholders throughout the entire software lifecycle. Stakeholder management represents the process of identifying, prioritizing, and communicating with stakeholders. Any given software project will likely have a large number of stakeholders who will have different interests within the project. It is important to attempt to identify as many stakeholders as possible for the project as the different interests can produce different views and new ideas. After identifying as many organizations and individuals who are vested in the success of the program, it is important to prioritize the stakeholders. With the large number of potential stakeholders it is not feasible for a software development team to completely satisfy everybody. Therefore, the software development team should attempt to rank the stakeholders in a way that shows which stakeholders should receive special attention. After determining which stakeholders need to be tightly involved with the program, the software development team needs to determine the best way to communicate with each individual stakeholder. It is important to understand that different stakeholders prefer to be contact in various ways, and the software development team needs to ensure they inform each stakeholder in the most efficient and preferred method.

Identifying Stakeholders

Whenever a new software development project begins, it is important for the development team to attempt to identify as many of the project's stakeholders as possible. A study conducted by Eskerod and Huemann in 2013, stated that a stakeholder is any group who is affected by the project or has the ability to influence it (Eskerod and Huemann, 2013, p. 37). Stakeholders on projects can come from many different backgrounds and can have widely different interests from the program. Stakeholders who can affect the project are typically easier to identify because they are either management or provide funding and other influential power of that nature. The more difficult stakeholders to identify are those who can be directly or indirectly affected by the outcome of the project.

Software development project stakeholders can come from vastly different backgrounds and from geographic locations around the world. Therefore it can be nearly impossible to get all stakeholders in the same place at the same time, even in the early planning stages of the project. This increases the difficulty for the software development team to identify all stakeholders of the project. A study conducted by Boonstra (2009) suggests that the best way to identify stakeholders is to develop a methodical system tailed to the project (Boonstra, 2009, p. 1). An example of this would be to list geographical locations, and then spend time within each location to determine all stakeholders within that location. The goal of this system is to break the problem of identifying stakeholders down into multiple smaller and more manageable sub problems.

Once the software development team has successfully identified all of the project's stakeholders the team should create and maintain a stakeholder registry. The stakeholder registry is intended to keep track of all stakeholders and their relevant information. The development team should maintain contact information, relation to the program, importance to the program, and any other information that the team deems important. The information needs to be constantly updated and organized in an easy to navigate and read format. The stakeholder registry should be made readily available for all members of the software development team so they can get any questions answered quickly.

If the software development team fails to identify stakeholders the project's success can be put into jeopardy. Stakeholder satisfaction is often times a major component to a project's overall success. Without identifying all stakeholders the team may not satisfy potentially important unknown stakeholders which can cause an otherwise successful project to become a failure.

Prioritizing Stakeholders

Stakeholders within a software development project can have varying level of influence on a program. With the large number of stakeholders it is important that the development team speed the time required to understand who their stakeholders are and which ones have the most importance to the program. With the large number of potential stakeholders it is often unrealistic to expect that the development team satisfy all stakeholders throughout the entirety of the project. Therefore, the team must decide which stakeholders warrant the extra time to ensure that they are satisfied.

The development team should determine the priority of the stakeholders based on how influential and powerful the stakeholder can be. This can be a challenging task thinking about each stakeholder compared to the rest. The team should attempt to think about each stakeholder individually and then rate them on a common scale. One common technique is to use a chart with one axis being influence and the other axis being impact. The chart can be as simple as dividing the graph into quadrants with a low and high section for each axis. Then taking advantage of the stakeholder registry the development team can individually account for all stakeholders and place them into a quadrant. After completion the development team can then determine if the number of highest priority stakeholders is manageable or if another round of filtering must be completed. Following the agile development methodology it is crucial to compare the stakeholders in a relative manner.

Prioritizing stakeholders should be done as a team rather than a single manager completing the task. With multiple team members weighing in on each stakeholder, more information about each stakeholder can be shared from the team's personal experience. Furthermore, the team can decide on how important any given stakeholder is to the program. Depending on their contribution to the program some team members may think the stakeholder is more valuable than other team members do.

Failing to prioritize the project's stakeholders can cause the program to fail. If the team does not determine which stakeholder to take the extra time to satisfy then they can potentially please less impactful or influential stakeholders yet fail to satisfy the ones who can kill a program or keep it alive.

Stakeholder Communication

Communication can be a challenging task within any field, and can be especially difficult during software development. A study conducted by Clark (1999) determined that stakeholder communication can be troublesome, yet it is necessary in order to satisfy the stakeholders (Clark, 1999, p. 219). As some stakeholders are responsible for making executive decisions for the software, it is extremely important to keep them as up-to-date as possible to ensure that they have all the information they need to make an informed and correct decision. As the development team becomes more familiar with each stakeholder, they should determine and log the best method of contact for each. A stakeholder might propose email as the best form of contact; however, they may be unresponsive to email and contacting via phone could prove more successful. Furthermore, as each stakeholder's personal ideas and preferences become known, they should be logged within the stakeholder registry. It can be beneficial for the development team to know the stances of the stakeholder when issues arise in order to tailor their messages accordingly.

The software development team should make a conscious effort to become familiar with each of the key stakeholders personalities. Each stakeholder can have different levels of communication and it is important to attempt to relate to each stakeholder on an individual level. This will strengthen the working relationship with the stakeholder and therefore make is easier to keep the stakeholder involved in the project.

When dealing with a large number of stakeholders and in particular a large number of key stakeholders, it can be useful to group the stakeholders into categories of the type and amount of effort required to keep them informed. Similar to the prioritizing of stakeholders described in the previous section, the software development team can create a grid which allows them to group the stakeholders. For the communication grid, the development team should use one axis as the stakeholder's potential power over the program and the other axis should be the stakeholder's level of interest over the program. If each axis is divided into high and low sections the development team can place the stakeholders accordingly. The stakeholders who have high power and high interest need to be continually informed throughout the entire lifecycle of the project.

CONCLUSION

The agile software development methodology is being widely accepted within the software development community. Agile provides multiple benefits over the previously used waterfall methodology. Agile attempts to simplify the software planning and estimation process by decomposing large requirements into small individual tasks. Analyzing small tasks allow the software development team to more accurately predict the level of effort required in order to implement the change. This allows the project manager to accurately depict the percentage complete of the software which allows them to continually track overall project progress against the originally planned progress. The agile process also is designed to help train developer in their schedule estimating skills throughout the lifecycle. For each task the developer should be required to make an estimation of how long they believe they will need to complete the task, after the task is completed they should enter in the actual time spent on the task. This will show the developer the delta between their estimation and their actual time spent.

Agile's other main focus is how to identify and involve stakeholders throughout the lifecycle of the software. Given the large number of potential stakeholders, the development team needs a method to determine the major players and find a way to connect with them in order to promote stakeholder involvement and interest in the program. This will lead to new and improved ideas from the stakeholders and will create a higher quality product at the end of the development cycle. With improved planning and stakeholder involvement agile results in an efficient and effective software development methodology. The incremental process is an idea that keeps the software on track even if the destination changes frequently.

REFERENCES

Boonstra, A. (2009). Identifying and managing stakeholders in enterprise information system projects. *International Journal of Enterprise Information Systems, 5*(4), 1-16. Retrieved from http://search.proquest.com/docview/921601543?accountid=8289

Clark, C. E. (1999). Developing stakeholder communication. *American Marketing Association.Conference Proceedings, 10,* 219. Retrieved from http://search.proquest.com/docview/199480361?accountid=8289

Eskerod, P., & Huemann, M. (2013). Sustainable development and project stakeholder management: What standards say. *International Journal of Managing Projects in Business, 6*(1), 36-50. doi:http://dx.doi.org/10.1108/17538371311291017

Ghule, S. (2014). Risk analysis and mitigation plan in software development. *International Journal of Engineering, Sciences and Research Technology, 3*(8), 546-548.

Wysocki, R. K. (2013). Effective project management : Traditional, agile, extreme. Hoboken: Wiley.