Merging Prototyping with Agile Software Development Methodology

Sara Tanvir
National University of Sciences & Technology (NUST), H-12, Islamabad, Pakistan

Mamoona Safdar
National University of Sciences & Technology (NUST), H-12, Islamabad, Pakistan

Hanny Tufail
National University of Sciences & Technology (NUST), H-12, Islamabad, Pakistan

Usman Qamar
National University of Sciences & Technology (NUST), H-12, Islamabad, Pakistan

ABSTRACT
Agile software development is a vast and rising discipline in field of software engineering and has gained importance in software industry. Due to flexibility of agile development, it is adopted by various organizations to develop software projects and is also a research area for software engineers. This paper introduces the concept of merging prototyping with agile software development methodology. The paper introduces a concept to reuse components from agile development by developing prototyping that could be reused. Hence, the available components would be reused instead of developing a component from scratch.

Keywords
Agile methods; traditional software development; software development; component; versioning; prototype; UCSD

1. INTRODUCTION
Software development is going an evolutionary shift from traditional methodologies to agile methodologies [1]. Traditional software development follows series of steps sequentially; all requirements are elicited, documented and validated before the development process that makes the development process rigid and results in difficulty to manage change that is not possible in reality as whole system cannot be exactly defined before development. Traditional approaches are not preferable where requirements change dynamically.

Traditional software development lacks flexibility to accommodate change in a project that requires changes frequently. Agile software development emphasizes on change management and thus satisfying customers on basis of iterative and incremental development with contribution of stakeholders with development team. Agile is an adaptive methodology that responds to changing requirements. High quality with high flexibility is achieved using agile methodologies. Agile software development is different from traditional approaches in a way that it puts less emphasis on planning while more on managing change throughout the project [2]. According to the 10th Annual State of Agile Report, the chief aids of agile involve: administer changing priorities (87%), team productivity (85%), and project visibility (84%) [10].

Agile methods are used in software industry but limited only to fulfill the user requirements and customer satisfaction. The software itself must be Agile, lending itself to rapid incremental deliveries and must be architected accordingly [11]. These could be made beneficial by merging prototyping with agile methodologies and good collaboration among teams. Prototyping is an incomplete picture used to show a rough idea of an expected system so that user can validate the requirements before investing additional resources. This can be achieved by introducing reusability in agile software development processes so that solutions for same kind of problems can be shared among organizations to save time and other resources.

Change is dynamic; where there comes change there is a need to manage that change to accommodate with system requirements. Changes made in components throughout the project development if properly implemented, tested and documented could be easy to reuse. Customer requests for changes and wants final product on time, while there is no such time to plan for change and timely handling the system to user. To save time to add new functionality to system through agile development, changes if implemented, tested, documented and saved properly help in reusing components without any extra effort to work on the functionality.

The paper is divided into the following sections: Section II explains briefly about the research paper reviews relevant to the selected topic. The next Section III consists of methodology introduced in which a model is proposed explained through diagrams. In Section IV the proposed model is then validated using case study of PHR (Personal Health Record) application. The paper concludes and summarizes the main findings of study in Section V and Section VI includes the future work of our research.

2. LITERATURE REVIEW
In past papers, Marta Larusdottir conducted a research on integrating agile with User-centered systems design (UCSD) that allows reusing any context of use. This paper focuses on usability in development as well as throughout the software development lifecycle. To develop usability, it is important to know about the user and observe the user to get context of use. Rafaela Mantovani Fontana conducted a research and introduced a framework using maturity models for maturing in agile methodologies. It is a study on how agile teams introduce any practices or methodologies and how they get mature over time. It consists of analysis of nine models for structure of agile maturity model [5].

Amadeu Silveira Campanelli surveyed through Systematic Literature Review (SLR) of different research papers on tailoring agile methods. Many of introduced techniques in agile were independent of agile methodologies and some of the work is done on Scrum and Extreme Programming (XP) [6].

Eva-Maria Schon introduced a hybrid model using various development models. An artifact based model integration of development models with user centered design (UCD) and user experience (UX) to improve stakeholder’s involvement is proposed. Further the paper investigates on approaches for stakeholder collaboration and how to gather the requirements and manage them. What approaches allow the involvement of stakeholders for gathering requirements or any methodology
that help convey user’s perspective to stakeholders and how to manage those requirements using agile methodologies [7].

Two concepts: Prototyping and Agile motivated this study. Our objective was to merge both of these concepts that also helps reduce any limitations in agile or help developing the software better, like reusability of components.

3. MERGING PROTOTYPING WITH AGILE METHODOLOGIES

Prototyping is merged with agile methodology to reduce the limitations of agile methods with the introduction of concept of reusability with agile. It could be done by spending some time on documentation of components after developing a system through agile. The concept introduced is to gather the requirements from customer; search components in a repository to determine if a needed component already exists otherwise develop a prototype, design, implement, test and maintain the system. Repository contains component’s prototype along with its documentation. Our proposed model offers following advantages:

- Validation of end product
- Developers have design hints
- Testing team can generate test cases easily
- Allows components to be reused easily
- Easier to maintain
- Customers have an idea about what to expect as an end product.
- Prototypes are saved in a single repository
- Saves time, money and other resources of organization
- Same component does not need to be developed from scratch again and again

The phases in our proposed SMH Prototyping Model as shown in Figure 2 can be briefly described as:

3.1 Requirements Gathering

Functional, non-functional and domain requirements are gathered by collaboration with customers as performed in agile method. In addition to it, all requirements are grouped into components as shown in Figure 1, and then repository is searched for existing components to determine whether the required component already exists or needs to be developed to fulfill user requirements.

According to survey conducted by [12], requirements change is inevitable and the trends in requirements change is show in Table 1.

<table>
<thead>
<tr>
<th>Practitioner Practices</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modification of Existing Requirements</td>
<td>7.9%</td>
</tr>
<tr>
<td>Make version of existing requirement</td>
<td>36%</td>
</tr>
<tr>
<td>Record as New Requirement</td>
<td>36.1%</td>
</tr>
</tbody>
</table>

3.2 Design & Prototyping

Prototype is developed so that it could be used to validate the requirements, can also be considered as guidance for design phase and to provide idea about test case generation. Prototypes are stored in a repository and could be reused in future projects of organization.

The changes in software primarily reflect on changes in the business process in client organization. There are many tools and techniques available but the study shows that current trends in techniques and tools for change control don't tend to reduce the change but to make it more manageable and cost effective [9]. The most common change management technique, Versioning, allows documenting a change in a specific requirement as a new version of it and save the component (along with its documentation and rationale of change) for reuse. However, versioning has some limitations:

- Redundancy when creating a newer version as a result of a requirement change. There might be repetition of prototypes developed.
- Unnecessary space occupation i.e. creating prototypes from gathered requirements and forming versions for each prototype created after modifications might result in unnecessary space coverage. Hence, different versions created might be similar in functionality.

There is a strong need of a predefined procedure to overcome aforementioned limitations therefore a criteria should be set to overcome versioning discrepancies. The criteria may comprise of two steps:

- Measurement of the additional SLOC to accommodate changed requirement and the business value of that component
- Comparison of the SLOC and business value of previous requirement with the latest version of changed requirement

If there are more SLOC added to the changed functionality as compared to the original requirement, new version is created. The developed prototype is also validated from customer and stored in a repository. If a component is to be developed from scratch then it is stored in a repository for future use in other projects of organization.

Component based development makes it quite easier to accommodate changes in requirements since one wouldn’t have to develop from scratch. It also reduces the amount of time of implementation due to availability of documentation. If a component is amended and needs to be saved, it would be stored as a new version in a repository along with modification details and rationale.

3.3 Design & Development

This is a phase where the components of software are categorized in an effective way. The requirements gathered
during first phase i.e. Requirements gathering phase are converted to Design. The design explains the overall system in form of models. In Agile, design is normally considered a continuous activity and it assumes and even encourages the change in requirements during the entire project [8].
The system to be delivered to customer is implemented after validating the required functionality from customer. The design created also serves as input for development phase as well as testing phase of a system.

3.4 Testing & Maintenance
The developed components are tested thoroughly to make it error free that results in their seamless reusability as no further testing effort is required. Consequently, delivering the defect free functionality to the customer in time.

System is maintained by updating components, their documentation and rationale in a repository where prototypes of components have been saved for future use in projects.

This phase ensures that the product is according to customer’s needs and the product serves as solution to his problem. The system’s repository is also updated in case of corrective maintenance.

3.4.1 Software decomposition into Components
Using agile methodology, requirements are gathered, developed and delivered to customer without spending more time on documentation. Putting some effort on documentation would help to reuse prototype of components easily.

Software is divided into discrete independent components instead of developing as a whole. Software could also be divided in a way that each component represents a single requirement so one requirement does not affect any other component. Since agile caters the requirement change therefore decomposing whole system into components allows us to modify only those components where the change occurs without affecting others. In case if any of the components undergo requirement change and is not needed further could also be saved in a repository for use in some other project in future.

3.4.2 Proposed SMH Model
The Figure 3 shows the workflow of our proposed SMH prototyping model for a complete prototype development.

4. CASE STUDY
We have conducted a case study to validate the proposed model using Personal Health Record (PHR). PHR is an electronic application used by patients that allows them to log their diet and exercise routine information, manage and track their health progress, to check history from logs maintained, view and share past or current information about health[4]. With PHR, patients fill in the information and the information is stored on a patient’s computer or on Internet. PHR system helps patients add information about diet or exercise and can also help to track progress over time in form of graphs and charts[3]. A PHR application we have used is ‘ChemoWave’ application provided by health care institutions and providers. The system has following requirements:

Req 1: Patient registers by entering cancer information (Name, Cancer type, Cancer stage)

Req 2: Whenever the application goes into background, it should ask the user to enter a PIN code on next encounter. (Only if user was logged in before app went into background)
Req 3: On the very first tab (that would be visible by default) User should be provided with the Overview of his latest logged Symptoms, Condition and Mood.

Req 4: On this first screen, there should be a graphical representation of all the logged “Quick Logs” in past week.

Req 5: There should be an option to select graphs for past 7 days or 30 days.

Req 6: It should present a Daily View and a Monthly View.

Req 7: Application should show the logged items against its start date on Calendar.

Req 8: On tapping a logged item on a particular date would take us to the details of logged item, termed as logged history.

Req 9: User should be able to add Primary Contact information for his Hospital, Doctor and of Pharmacy.

Req 10: User should also be able to add friends’ contact information.

Req 11: There should be a section in the application for conducting a survey from time to time.

Req 12: Application should provide a way to create an appointment log entry on calendar with an option of reminder.

Req 13: There should be an option to log the medication time and the dose of a prescribed meds he took.

Applying the described case study using the proposed model, we gathered the 13 requirements for a PHR application with collaboration of stakeholders and customer that was a phase of requirement gathering and then grouped to form following components:

Requirements 1 and 2 are combined to form a Component 1 termed as Registration.

Requirements 3, 4 and 5 form another Component 2 termed as Insights.

Component 3, termed as Calendar, is formed from Requirements 6, 7 and 8.

Requirements 9 and 10 combine to form Component 4 i.e. Contacts.

Component 5, Survey, is formed from Requirement 11. Requirement 12 forms Component 6 termed as Appointment Creation.

Requirement 13 forms Component 7 termed as Medication. Hence, from 13 requirements we got 7 different components of a single application.

Next we should develop a prototype so that we, as development team, can validate requirements from customer whether it is according to their needs. Another task to be performed in the same phase is to search for any prototype of components that already exists or any component that was developed in past projects having same functionality or similar to the desired functionality. If a module exists, we do not need to develop the required functionality from scratch but we can use the existing component previously developed, tested and saved in a repository to save time. If there is not even a similar module in a repository, development team shall develop the prototype of component and save the prototype in a repository for future use in organization’s projects. Registration, Contacts and Calendar are one of the common components that are found in a repository while all other prototypes of components need to be developed. That helped save development time and cost for Registration, Contacts and Calendar components. All other that would be developed from scratch are saved in a repository.

Now a Prototype, a dummy application implementing requirements from user, is given to user to validate. In case, if a user asks for change in any requirement, like Requirement 12, asking that Req 12(a): User should be prompted about his appointment before it happens. Req 12(b): User should be provided with an option to select the time he wants to be reminded before due time of his appointment. To accommodate such change firstly, it must be decided to keep the previous prototype of component or not. Therefore, we apply versioning as explained before i.e. we decide about the saving the developed component in a repository using some criteria so that no extra space is occupied. Since it’s a modification and if prior prototype version were worth saving for the development organization then a new version of that prototype would be developed to accommodate new requirements.

After validation of requirements from user, application is then further developed, tested and maintained. When application is completely developed and validated from user and final product is ready, it is then delivered to customer.

5. CONCLUSION

Nowadays, agile is opted by many software industries across the world. This paper merges two main concepts from software engineering: prototyping and agile methodology. It helps to have benefits of both methodologies and it allows us to reuse the components in other projects across the organization and also to develop software and deliver it to the customer. Thus, development time is reduced in future projects.

6. FUTURE WORK

Agile is a vast area with so much research potential. A research could be done on techniques for storing and retrieving of components. Search mechanism for component in the repository that best meets requirement. Another area would be component storage mechanism to look for duplications in the repository. It would all greatly reduce agile limitations and make it an industry standard.

7. REFERENCES


“Agile methods tailoring – A systematic literature review,” Volume 110, December 2015


