International Journal of Hybrid Information Technology
Vol.2, No.2, April, 2009

Story Card Based Agile Software Development

Chetankumar Patel, and Muthu Ramachandran
Leeds Metropolitan University, UK
c.patel@leedsmet.ac.uk

Abstract

The use of story cards for user stories in many Extreme Programming software development projects has been widespread. Several popular traditional methods for story cards (e.g., Cohen M, Kent B) have been used in successful fashion at some extent, but all lack of the powerful features for story cards guidelines, right sort of information on story cards and quality of user stories on story cards. They also do not involve anybody apart from customer on story writing workshop. This paper has described the INSERT model, new proposed frame work of story cards, and a new improved requirements elicitation process in XP. The experience with INSERT model and new framework of story cards indicates that it is feasible to contemplate improving user stories and story cards in Extreme programming. This paper also introduces best practices guidelines for agile requirements engineering to enhance the quality of requirements (story cards).

1. Introduction

Developing software that meets the customers or stakeholders' needs and expectation is the ultimate goal of the software development methodology. To meet their need we have to perform a requirement engineering step, which is one of the crucial steps in to software development methodology. Overall project success and failures of the project is depending on the user requirements. Requirements elicitation process is one of the challenging processes in the software development methods. In traditional software development methods end users or stakeholders predefined their requirements and sent to the development team to analysis and negotiation to produce requirement specification. Traditional software development has a problem to deal with requirement change after careful analysis and negotiation. This problem is well tackled by the XP, which is one of the agile software development methodologies.

Extreme (XP) programming is a conceptual framework of practices and principles to develop software faster, incrementally and to produce satisfied customer. It is a set of twelve practices and four principles, which makes XP successful and well known among all the agile software development methods. The goal of XP is to produce the software faster, incrementally and to produce satisfied customer [1],[2]. According to Bohem the cost of change grows exponentially as the project progresses through it lifecycle [3],[4]. The relative repair cost is 200 times greater in the maintenance phase than if it is caught in the requirement phase [5],[6]. XP maintain the cost of change through iterative software development methods and Refactoring.

In XP, Development starts with planning game where customer writes user stories on story cards. Those cards are estimated by the developer, based on those estimation customer priories them depends on their needs to establish a timebox of an iteration. Developers develop those story cards through pair programming and test driven development. At last customer provides acceptance test to accept the developed functionality. In between they consider all of the XP practices in mind to improve the quality of the software.

Story cards are one of the important aspects of the XP. They are playing vital role in XP. It describes functionality of system or software to be build that will be valuable to either purchaser or user of software. User stories are composed of three aspects [5]:

- A written description of the story used for planning and as a reminder
- Conversation about the story that serves to flush out the details of the story
- Tests that convey and document details and that can be used to determine when a story is complete

Story cards are written by the customer in XP to articulate their business needs. According to Cohn story cards must be testable, estimatable, valuable to the customer, small and independent [5]. These story cards must be written by the customer because they know their business need very well compared to developer.

XP strongly recommend an onsite customer to write their business need. Business is well understood by the customer. But generally customers have rarely a general picture of the requirements or system in their mind [1-9]. Traditional XP story card framework or template is not well defined for the requirements elicitation. It supports to write requirements or user needs in two to three sentences and it not discover any information rather than user functionality. Different stakeholders have different needs. End user has rarely a picture of a clear of system to write down the user stories. This will lead to problems related to requirements like requirements conflicts, missing requirements, and ambiguous requirements etc, and not address non-functional requirements from exploration phase. Due to this reason they hardly make a decision or predict wrong priority of the requirements or story cards. Two third of the projects are failed because of ambiguous and incomplete user requirements, and poor quality of the requirements. For small to medium organizations, proper requirements prioritization and selection can mean the difference in not only project success or failure but also overall company survivability [1]. Different users have different perspective of system in same organization. Different background can make problems to priories the requirements in XP. Different stakeholders have different needs and different requirements and different prioritization values so requirements conflicts there. A critical aspect of the requirements process is the selection of the an appropriate requirements set from the multitude of competing and conflicting expectation elicited from the various project stakeholders or from an onsite customers[7]. The CHOAS report published in 1995 shows that almost half of the cancelled projects failed due to a lack of requirements engineering effort and that a similar percentage ascribes good requirements engineering as the main reason for project success [10-13].

XP methodology highly relies on the onsite-customer interaction with the developer to identify or to tell which features to implement in a next release. XP builds software systems based on customer's domain knowledge and his/her expertise. In traditional XP or agile software development methodology story cards are written by the customers perhaps with the help of developer. Therefore each user story or story card must be split into unique and independent requirement.

Traditionally user story is written on the 2" X 3" Cards. Usually any size is acceptable to write user story on story cards. Following Figure 1 shows an example of the story card used in the real project as proposed in [2], which provides a traditional structure of story card.

According to general template of story card, it is really difficult to well tackle the user requirements expressed on the cards. Traditionally developed story cards are not providing enough information of user functionality. They express user functionality in single to couple of sentences, which is really difficult to do analysis, and they lead problem related to under or over estimate, and based on that estimation there is a possibility of wrong story cards prioritization as well. We apply this traditional story card method to the real project user story

Figure 4 shows the story cards written through a traditional way, which is just a short and vague statement of user requirement. This story card does not provide any information related to acceptance testing as well.

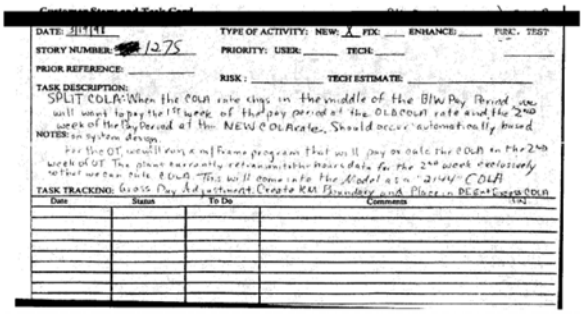

Figure 1. Traditional Story Card [2]

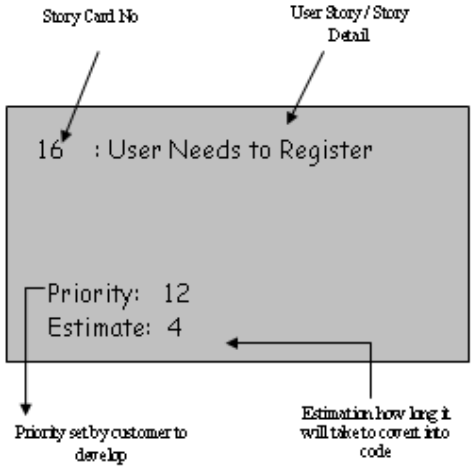

Figure 2. An Example of Traditional Story Card

This is a story card for an online e-commerce store. In this story cards user trying to express their requirement as

'Each and Every online purchaser needs to register with unique username and password before purchasing anything from the online store'

As a result of this investigation we propose a new prototype to improve requirement elicitation process in XP. This will help to customer and developer to improve the quality of the user stories or story cards, and to address functional and non-functional requirements on story cards based on the story cards and requirements engineering guidelines. We also propose an 'INSERT' model or methodology to perform requirements engineering in XP. This article compares, traditional requirement engineering and traditional XP requirements engineering approach with our new improved 'INSERT' technique to capture user requirements for agile software development environments. We also analyze commonalities and differences of both approaches and determine possible ways how agile software development team and customers can benefit from our improved requirements elicitation methods. In this paper we discuss about extreme programming and requirement elicitation process through story cards first and then after the challenges and problems on XP software

development methodology. This is followed by discussion of related research regarding to an 'INSERT' requirement elicitation method of the story cards for XP based projects.

2. An 'INSERT' Process in Extreme Programming (XP)

As a result of this investigation we propose a new 'INSERT' model to improve the quality of user story and to address customer requirements properly and on verifiable way the acronyms INSERT is as:

I: Independent

N: Negotiable

S: Small enough to fit into iteration

E: Estimatable or easy to Estimate

R: Representation of user functionality (Requirement)

T: Testable

2.1 Independent

Story cards must be independent or we have to take care as much as possible to avoid dependencies between story cards. Dependencies between story cards lead a problem related to estimation and prioritization. For example customer selected a high priority of story cards which is depend on low priority story cards, this situation make estimation harder than it suppose to be. In our insert model we do take care of dependencies between story cards. We take care as much as possible to flush out the dependencies between story cards. This type of story cards mostly captures atomic requirements which are directly transferable to a single use case and a design class.

2.2 Negotiable

Stories on Story card also be negotiable. Story cards are short description of the user requirements they are not working as a contract. User story is a reminder to have a conversation between developer and customer. If it is negotiable than only than it gives better chance to developer to understand customer needs, their business need and their domain knowledge as well. This type of story cards mostly captures complex requirements which relates to more than one used cases and scenarios.

2.3 Small

Stories on the story cards need to be small enough to fit into iteration. Too big story or too small story is not going to fit into the timebox of iteration. To solve this problem we suggest to write an acceptance test with the story it self. There is a direct co-relation between story and acceptance tests. If story many acceptance tests that means it is big to fit into iteration and needs to be split into two story cards based on the acceptance tests. If story is small on the story card then combine them with another small story to fit them into the same iteration. This type of story cards captures a part of a independent and negotiable requirements and may also represent a non-functional requirements.

2.4 Estimatable

Estimation is a crucial value of the story card. Based on the developer's estimation customer decide which functionality is going to be first and which one is next. On traditional XP cards estimation is complex. There are several reasons for that like developers do not have domain knowledge, or they are technically not sound, or story cards are too big. Our proposed model considers these all problems and tries to solve this problem by acceptance tests, which will help to bring domain knowledge and technical knowledge to customers and developers.

2.5 Representation of system functionality

This is an import and crucial part of the story cards. There isn't any tool or documentation that proves that the user story expressed on the story cards is valuable to the user or not. Stories on the story cards are written by the customer, so XP assume that requirement is correct, which leads problem related to requirement change and rework. To solve this problem we again focused on acceptance tests and strongly recommended to write them with the story cards. This acceptance test will help to write user stories on the verifiable ways.

2.6 Testable

Story cards must be testable. If it is difficult to test the story then that means story card is expressing non-functional requirements instead of user functionality. It is easy to write functional test or acceptance test for the functionality (functional requirements) in our model if you are able to write functional test or acceptance test that means the story is testable. Successfully passed all acceptance test means story card is fully developed.

This insert process is also based on the best knowledge based requirements engineering guidelines which are described in the section 3

Consider the following figure 3 which shows our new improved requirements elicitation process to capture user story on story cards based on the INSERT values.

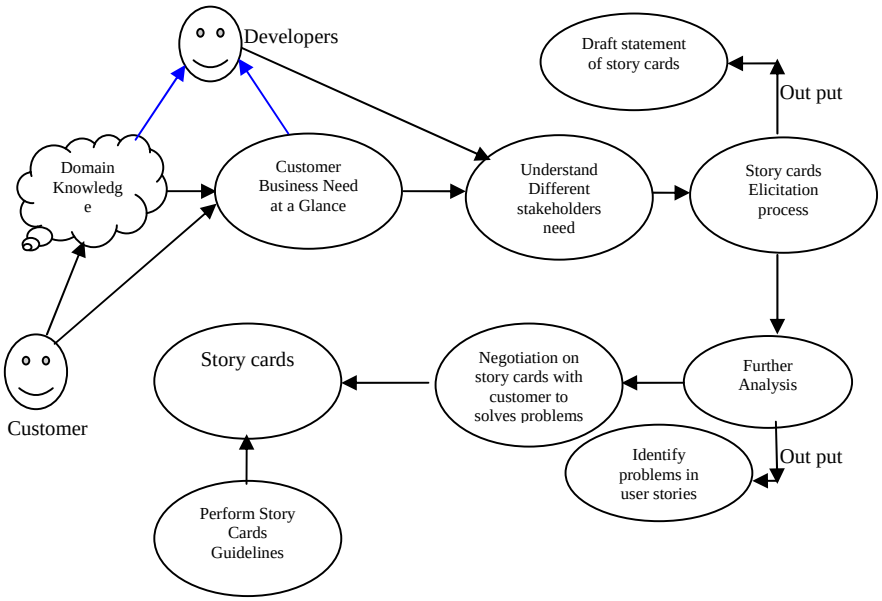

Figure 3. INSERT method for requirements elicitation process in XP

In our approach we recommended a customer or stake holder who is on site has comprehensive application domain and business knowledge to put a developer into the picture.

Application domain knowledge and customer business knowledge from customer will help developer to focus on stockholder's business needs and requirements and help them to cover any missing functionality. Customer business knowledge helps developers to understand how the system will affect and interact with the different part of the business, and help to identify and understand different stakeholders, who are directly or indirectly affected by the system.

At the end of this successful discussion, customer starts story elicitation process and write the draft statement of requirements on story cards. These story cards are further analyzed, which assist to customer and developer to identify any problems and missing functionality. Missing functionalities become defect in working software. This scenario will help developers to focus on non-functional requirements. Also helps to keep in mind what they have to do to improve all the aspects of the business through structured system. Identified problems are being negotiated between developers and customer to acquire story cards. Following figure 4 shows an example of story cards captured through the new improved requirement elicitation process based on the INSERT model.

Figure 3 INSERT story card method for requirements elicitation processes in XP, addresses the requirements capturing in XP with on-site customer. INSERT is based on a set of best practice guideline that helps both developers and customers to refine, clarify, and re-solve requirements conflicts with mutual discussion. The key benefits to apply the best practice guidelines are as

- Higher quality, lower cost requirements documents (Story cards).
- More understandable story cards compared to traditional story cards.
- Avoids misunderstandings among user requirements captured on story cards
- To reduce cost of changing the requirements at any stage of project life cycle
- to reveal technology we are using for a project is realistic
- Discovery of all likely sources of story cards
- Requirements are focused on core business needs
- Domain constraints often leads to critical requirements identification
- User finds easy to understand scenarios and to describe associated requirements.
- Easy to prioritize requirements
- Reveals ambiguities and inconsistency in the requirements
- Acceptance testing allows more stakeholders to participate in requirements validation.
- To support non-functional requirements

We also provide automated support tool to direct the processes of story card driven requirements capturing.

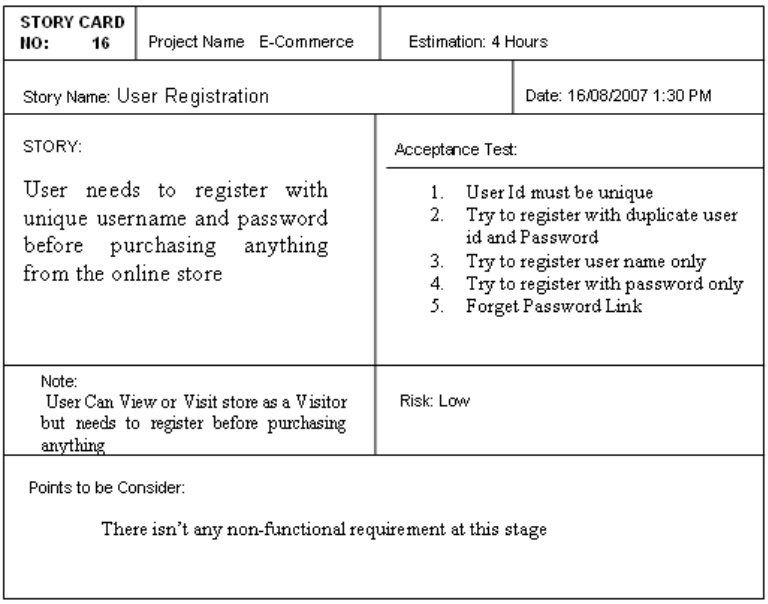

Figure 4. Story cards captured through the new improved requirement elicitation process based on the INSERT model.

Traditional method of story cards is still valuable and some of the guidelines are really crucial and unique. Onsite customer and simplicity of story cards are among them. In our proposal we also strongly recommended an onsite customer which is traditional story cards practice or guideline. In this section we tried to identify the improvement area of the story cards the following table will shows the commonality and variability between the story cards through traditional and INSERT model. This is just a prototype we still working on the story cards guidelines to extend them up to research level.

3. Best Practices Guidelines for Agile Requirements

Best practices guidelines for agile requirements practices can be categorised into many classes. The following figure 5 shows the best practices guidelines classification. Here these best practices guidelines are described into the following two categories:

Elicitation oriented guidelines: This describes the guidelines for the requirements elicitation, analysis and negotiation for agile software development environment. Existing agile requirements elicitation process does not go beyond existing available on-site customer. INSERT model for story cards elicitation process also consider these best practices guidelines.

Validation oriented guidelines: This describes the guidelines for Story cards validation, reuse, organizational and management guidelines. There are two type of validation on agile software development methodology mainly XP, unit testing and acceptance testing. There is a research issue about to test the requirements from the exploration stage.

The agile requirements guidelines are described as following.

3.1 Story cards must be small and complete

As we discussed into the section 2.1 Stories on the story cards need to be small enough to fit into iteration. Too big story or too small story is not going to fit into the timebox of iteration. This is a crucial guideline for the story cards based agile software development. the benefits to apply this guidelines are as 1) gives an ease in iteration planning 2) it is easy to understand.

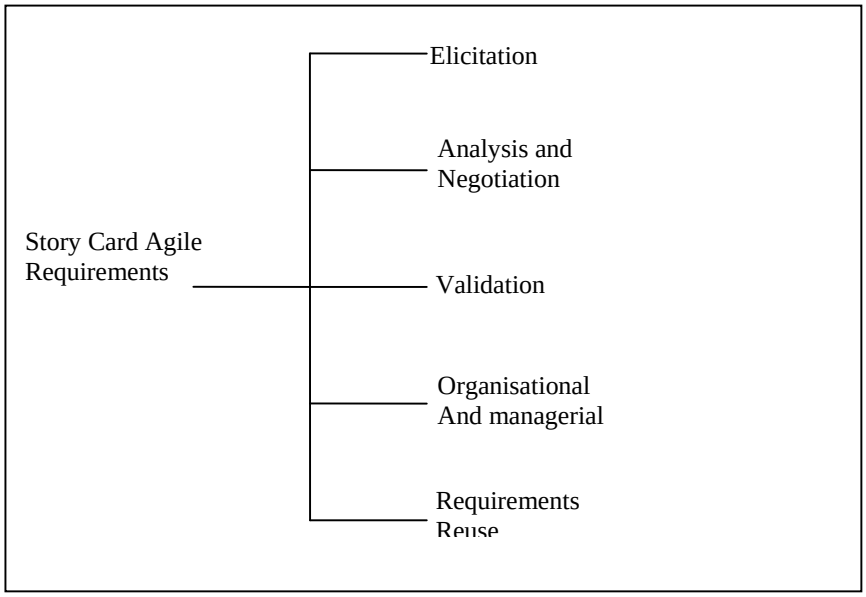

Figure 5. Classifying Best Practices Guidelines for Story card (Agile requirements)

3.2 Define a generic template or framework for story card

It is a good practice to have a standard and generic template or framework of story card, which should contain information regarding to on-site customer, developer, related story cards, risk, user story (requirements), time, non-functional constraint, domain name, and acceptance tests.

A standard structure for story cards means that any one can use their knowledge of previous story cards when reading a new story cards. It is also a good practice to have acceptance tests driven template of story card. Acceptance tests at the exploration phase which helps to find information more easily and understand the relationship between parts of the story cards

Acceptance test on acceptance test driven story card works as a checklist for developers and customer and reduces the chances of them accidentally omitting information. Ideally story cards should express the purpose of the user story or story cards, scope of story cards, user functionality, user characteristic, dependencies if exist and acceptance tests. It is easier to read, write and collect user requirements (user stories) on the standard template.

3.3 Dependencies of story cards

As we discussed into the section 2.1 the Story cards must be independent or we have to take care as much as possible to avoid dependencies between story cards. Dependencies between story cards lead a problem related to estimation and prioritization. For example

customer selected a high priority of story cards which is depend on low priority story cards, this situation make estimation harder than it suppose to be.

3.4 Concise story card (Write user requirement/user story on the story card in two to three sentences)

we should always write requirement on the story card in the format of the summary or abstract in two to three sentences long, which summarize the purpose and main functionality of the story cards. It is much easier for developer and customer to figure out user story if they have a broad description of user functionality want to develop. Summarizing the requirements allows story cards to make forward reference to other requirements.

3.5 Define simplicity on story cards (story cards must be Simple)

Simplicity is the core value of the agile software development or extreme programming. All Business terminology also needs to be predefined to make story cards as simple as possible. Simplicity helps or solves problems related to requirement change and priority.

3.6 Story cards must be Estimatable

The description of the user requirement to be developed as given in the story cards is a realistic basis for estimating story cards cost and development time. Estimation is a crucial value of the story card. Based on the developer's estimation customer decide which functionality is going to be first and which one is next.

On traditional XP cards estimation is complex. There are several reasons for that like developers do not have domain knowledge, or they are technically not sound, or story cards are too big. We consider these all problems and try to solve this problem by acceptance tests, which will help to bring domain knowledge and technical knowledge to customers and developers.

3.7 To define terminology used in the story card

Specialized terms used in the story cards must be predefined during the domain analysis. This will define the terms which are belonging to domain or system going to be developed. For example the system is concerned with mobile application, it would include terms such as Bluetooth, GPRS, and WAP etc.

3.8 Define an unambiguous story cards

Story card is unambiguous, if and only if, story card statement has only one interpretation. At a minimum, defined terms must describe functionality of the software. For unambiguous story card, it is a best practice to define key terms or terminology on the story card.

3.9 Valuable to the customer

Each story card is valued by the customer. It is really difficult to find out the story card is valuable to customer or not. The best way to ensure that each story is valuable to the customer or user is to have the customer to write the stories and to write acceptance test straightway at the exploration phase.

3.10 Define system boundaries during the domain analysis for Story cards

Domain analysis is a good practice to follow the successful agile requirement engineering process. To identify system boundaries is a crucial practice of the domain analysis. During the agile requirement engineering process, we should identify which story cards are business

story cards (presents system requirements) and which story cards are outside the scope of the business. As can be seen form the chapter 2, onsite customer often unclear about what should and what should not be in the system. They may suggest inappropriate requirements sometime. So it is a good practice to eliminate the requirements or story cards which are out side of the scope.

3.11 Story card must be negotiable

Stories on Story card also are negotiable. Story cards are short description of the user requirements they are not working as a contract. User story is a reminder to have a conversation between developer and customer. If it is negotiable than only than it gives better chance to developer to understand customer needs, their business need and their domain knowledge as well. This type of story cards mostly captures complex requirements which relates to more than one used cases and scenarios.

3.12 Story cards must be consistent

Consistent statement of story card does not include contradictory, incoherent and conflicting statements, so which will not lead to wrong discussions.

3.13 To link between story card requirements and stakeholder

We should always note the relationship between story cards and stakeholders related to that story cards. It will increase the traceability of the system. As we discussed into the chapter 2, user story or story card is a reminder to have a conversation on the particular functionality later on. If we do not link story card and relevant story card, it is really difficult to reference it. During the acceptance testing, it will help to validate the requirements by consulting the relevant stakeholder who is responsible for the particular story card.

3.14 Story card must present business values (requirements)

Business value is the primary function for story card' testing (acceptance testing), prioritization for the iteration. It is a best practice to identify weather story card (user story) presents a business requirements or value as a user story. This can be checked by the acceptance tests on the story cards. The story card should always express or explain why the functionality captured on story card is required and how it will contribute to the overall business objectives of the iteration. It is important to consider because when proposal or user story changes are made, we can use business case for the system to assess where or not the proposed changes are sensible.

3.15 Propose or Define validation plan for story cards (Write acceptance tests)

This is an import and crucial part of the story cards. There isn't any tool or documentation that proves that the user story expressed on the story cards is valuable to the user or not. Stories on the story cards are written by the customer, so XP assume that requirement is correct, which leads problem related to requirement change and rework.

To solve this problem we again recommended and focused on acceptance tests and strongly recommended to write them with the story cards. This acceptance test will help to write user stories on the verifiable ways. For each story card propose one or more acceptance tests to check if the system meet that requirement. Proposing possible tests is an effective way of revealing requirements problems such as incompleteness and ambiguity. If it is difficult to write acceptance test, then there is some kind of requirements problems there.

A story card is correct, if and only if, the iteration or software has to meet all the customer's requirements. Or a correct story card must accurately and precisely identify conditions and limitation encountered by the software. According to IEEE requirements standard there is no tool can automatically ensure correctness. But in the case of extreme programming it is easy to check correctness of story cards through acceptance testing from beginning.

3.16 Story card should welcome requirements changes and modification

Agile methods are well known for lower cost of change. Agile methods support requirements change during the any stage of the software development. They keep the cost of change as less as possible. Requirements change is supported by test driven development, iteration and release planning, acceptance testing, pair programming and refactoring, which are successful and well known agile practices.

It is always a good and handy to change story cards whenever necessary. The key benefit is it will reduce the cost of changing requirements. Producing or writing new story cards always expensive and time consuming. If story cards are not changeable then it is difficult to accommodate change during the test driven development and pair programming.

3.17 Consideration of domain analysis

It is a really good practice to record domain analysis. Domain analysis will help to collect missing or accidentally omitting user stories or requirements by reading it.

- Domain analysis also defines the following:
- Business analysis
- Domain Description
- Domain boundaries
- Sub-domains and their features
- Terminology used
- Domain stakeholders

We should study the domain and do domain analysis and understand constraint as a part of the planning game. Domain analysis generates system story cards and place limitation on story cards or planning game.

3.18 Identify and consider the system stakeholders from the onsite customer

Each system has a multiple stakeholders who are going to get benefit in a direct and indirect way from the system which is being developed. Different system has different stakeholders.

As a part of the story card workshop or user story elicitation process, we should consider all kind of the stakeholders and consult them to discover their specific needs for story cards. If we do not consider all the stakeholders who are going to affected by the introduction of the story cards, there is a chance of bugs into the system and likely to miss important requirements.

As a result cost of change will increase at the development stage. Identifying stakeholders and discussing their needs and requirements with them makes people happy. Again agile

methods are not process oriented, they are people oriented. People are in the heart of the agile process.

3.19 Story card prioritization for planning

Story card's prioritization for the iteration planning is a challenging issue. Traditionally it is prioritized by the customer according to their priority. Different stakeholder's have different priority, which lead problem related to prioritization.

It is a best practice to prioritize story cards based on the story card (agile requirements) and extreme programming core values. These are as following

- Business Functionality
- Customer Priority
- Core Values (INSERT and Best Practices Guidelines)
- Market Values
- Implementation Cost
- Business risk

3.20 Define story card in a natural human language

It is recommended, customer expresses a story card in natural language, write their requirement using simple, consistently, and concise language. Try to avoid complex sentences, long sentences, paragraphs and ambiguous terminology. When user story is written using simple language it is easier to read and understand. More people can understand story cards that are written in a simple way. It takes less time to explain the user story to developers.

3.21 Check the story cards meet the XP Principle and values

Before putting forward story cards into development stage, one of the developers should carry out a quick standards check to ensure that the story cards meet the XP principle and values.

3.22 Uniquely identify each story cards and user stories on story cards

Each story card should be assigned a unique no or story card no or unique constraint. If story card got unique identification, it is easy to reference story card during the planning game, TDD, Refactoring or at any agile practice. It is also being easy to reference story cards if we store the electronically.

3.23 User story writing technique

The best user story writing technique is face to face interview with pair wise story writing technique. (Business expert and developer).

3.24 On-site customer

This is the crucial practice of extreme programming to have a proper online customer. On-site customer helps to drive the story cards towards the business direction. It is recommended

to have a domain expert on site, who can help on domain analysis, to identify all possible stake holders and acceptance testing. Domain expert have a clear vision of the system.

3.25 Specify non-functional requirements also on story cards

This is a challenging issue for the traditional story card and research needs to consider this. It is a good practice; we should specify quantitative values to the story cards. This is most applicable to the non-functional requirements. Non-functional requirements are the requirements which are concerned with attribute of the system.

3.26 Use an electronic system to store story cards compared to write them on the card and destroy

Agile methods do not support documentation, which is sometime great practice and sometime challenging practice. It can be seen at the literature, sometime requirements documentation is required to facilitate user manuals or for organization polices.

It is a best practice, rather than maintaining requirements on cards and after implementation destroy them, to establish a story cards database or an electronic system and store the individual story cards in the database or electronically. Managing story cards in a database make easier to maintain the link between the story cards as well. It is also easy to print out and put them on the wall to follow traditional planning game.

3.27 Assess story cards risk

It is a good practice to do risk analysis for the story card. Risk analysis will help to identify the risk or problems those may arise in the implementation of the story cards. If we do risk analysis for the story cards and identify any problems and risks for story cards, it is easy and cost effective to modify or change story card at panning game. Acceptance test on story card also helps to identify the risk.

3.28 Story cards are defined by the on-site customer and developers

Establish the basis for agreement between developers and suppliers on what the system has to do. The enough and complete description of the functionality on the story cards to be performed by the software specified in the story card will assist the potential users to determine if the software specified meets their needs or how the software must be modified to meet their needs. Customer (Domain Expert, Business Analyst) knows very well their system or business. While developer will easily find out the risk involved into the requirements.

The following table 1 shows the guideline supported by the different approaches. Where SoBA is an automated tool for story cards based agile software development based on the ‘INSERT’ Methodology.

Table 1. Guidelines supported by the different approaches

Story cards feature and guidelines	Mike Cohn	INSERT (SoBA)	Kent Beck	3C Approach
Story cards must be small and complete	**X**	**X**	**X**	X
Define a standard story card structure.		**X**		
Include the summary or abstract in two to three sentences of the requirements	**X**	**X**		

Dependencies of story cards	**X**	**X**		
Define an unambiguous story cards. (unambiguous)		**X**	**X**	
Define simplicity on story cards. (Simple)	**X**	**X**	**X**	**X**
define consistency on story cards (Consistent)		**X**		
Make a business case for the requirement or Story card must present business requirements	**X**	**X**	**X**	
Define specialized terms used in the story cards		**X**		
Make the story cards easy to change		**X**		
Assume system feasibility		**X**		
Identify and consult the system stakeholders from the onsite customer		**X**		
Valuable to the customer.	**X**	**X**	**X**	**X**
Define system boundaries during the domain analysis for Story cards		**X**		
Use checklist to do requirements or system analysis		**X**		
Story card must be negotiable		**X**	**X**	
Prioritize story cards	**Limited**	**X**	**Limited**	**Limited**
Use language simply, consistently and concisely		**X**		
Check the story cards meet the XP Principle and values		**X**		
Uniquely identify each story cards and user stories on story cards		**X**		
User story writing technique.	**X**	**X**	**Limited**	
On-site customer	**X**	**X**	**X**	**X**
Look for domain analysis		**X**		
Write story cards from multiple viewpoints		**X**		
Use scenario to elicit requirements		**X**		
Provide a basis for estimation cost and schedule (Story cards must be estimatable).	**X**	**X**	**X**	X
Specify quantitative requirements also on story cards (For non-functional requirements) or story cards support non-functional requirements		**X**		

Try to links between story card requirements and stakeholder		**X**		
Propose or Define validation checklist for story cards (Write acceptance tests)		**X**		
Use a database system to store story cards compared to write them on the card and destroy		**X**		
Define modifiable story cards.		**X**		
Serves as a basis of functionality.	**X**	**X**	**X**	X
Story cards are in scope.		**X**		
Problem understanding.		**X**		
Assess story cards risk.		**X**		
Prioritize requirements based on the story card's value and XP values		**X**		
Assess story card's correctness		**X**		
facilitate transfer				
Learn from incident experience		**X**		
Use the past story cards estimation		**X**		
Reduced development Efforts.		**X**		
provides baseline for the validation and verification		**X**		
defined or representation of requirements by the developer and customer		**X**		**X**

4. Conclusion

Developing software that meets the customers or stakeholders' needs and expectation is the ultimate goal of the software development methodology. To meet their need we have to perform requirement engineering which helps to identify and structure requirements. In traditional software development methods end users or stakeholders predefined their requirements and sent to the development team to analysis and negotiation to produce requirement specification. In many cases it is risky or very difficult and not economical to produce a complete, verifiable set of requirements. Traditional software development has a problem to deal with requirement change after careful analysis and negotiation. This problem is well tackled by the XP as XP recommends an on-site customer to represents their requirements through user stories on story cards. Generally customers have rarely a general

picture of the requirements or system in their mind which leads problems related to requirements like requirements conflicts, missing requirements, and ambiguous requirements etc, and does not address non-functional requirements from exploration phase. This paper has described the INSERT model, new proposed frame work of story cards, and a new improved requirements elicitation process in XP. The experience with INSERT model and new framework of story cards indicates that it is feasible to contemplate improving user stories and story cards in Extreme programming.

References

[1] Azar, J., Smith, R., Cordes, R.(2007) Value-Oriented Requirements Prioritization in a Small Development organization, IEEE Software January/February 2007
[2] Beck, K (2000) Extreme programming Explained Embraced Change 2000 Addison-Wesley
[3] Beck, K and Fowler, M (2000) Planning Extreme Programming 2000 Addison-Wesley Press
[4] Bohem, B (1981) Software Engineering Economics 1981 Prentice Hall
[5] Cohn, M (2003) User stories applied for Agile Software Development 2003 Addison-Wesley
[6] Faulk, S (1996) Software Requirements: A Tutorial. Software Engineering 1996 M.Dorfman and R. H. Thayer Eds. pp 82-103.
[7] Karlsson, J (1996) Software Requirements Prioritizing. IEEE Proceeding of ICRE'96 1996
[8] Karlsson, J., Ryan K (1997) A cost-Value Approach for Prioritizing Requirements. IEEE Software 1997.
[9] Kotyana, G and Sommerville, I (1997) Requirements Engineering Processes and Techniques (1997) John Willy and Sons Ltd.
[10] Niu, N, Easterbrook, S. So You Think you know other's goal? A Repertory Grid Study IEEE Software March April 2007
[11] Paetsch, F., Eberlein, A. and Maure F (2003) Requirements Engineering and Agile Software Development. Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises 2003 (WETICE'03)
[12] Ramachandran, M (2005) A Process Improvement Framework for XP based SMEs. 6th International Conference on Extreme Programming and Agile Processes in Software Engineering, Sheffield, UK, June 2005
[13] Shine Technologies (2003) Agile Methodologies Survey, [internet] www.shinetech.com/agile_survey_results.jsp, Jan 2003
[14] Wiegers, K. First Things First: Prioritizing requirements software Development Vol. 7, no 9 Sept 1999;www.processimpact.com/pubs.shtml#requirements.

Authors

Chetankumar Patel is a research student in the software engineering research group at Leeds Metropolitan University,. His research interest is on Agile software development methodology, Extreme Programming, story (Requirement Engineering Process for XP) and agile process improvement. Contact him at c.patel@leedsmet.ac.uk

Dr. Muthu Ramachandran is a principal lecturer at Leeds Metropolitan university. Contact him at m.ramachandran@leedsmet.ac.uk