Chapter 1
Best Practices Guidelines for Agile Requirements Engineering Practices

Chetankumar Patel
Leeds Metropolitan University, UK

Muthu Ramachandran
Leeds Metropolitan University, UK

ABSTRACT

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 Agile Practices as it's 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 chapter introduces best knowledge based 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

DOI: 10.4018/978-1-60566-731-7.ch001

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 (Beck 2000). According to Boehm the cost of change grows exponentially as the project progresses through it lifecycle (Boehm 1981). The relative repair cost is 200 times greater in the maintenance phase than if it is caught in the requirement phase (Faluk 1996). 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.

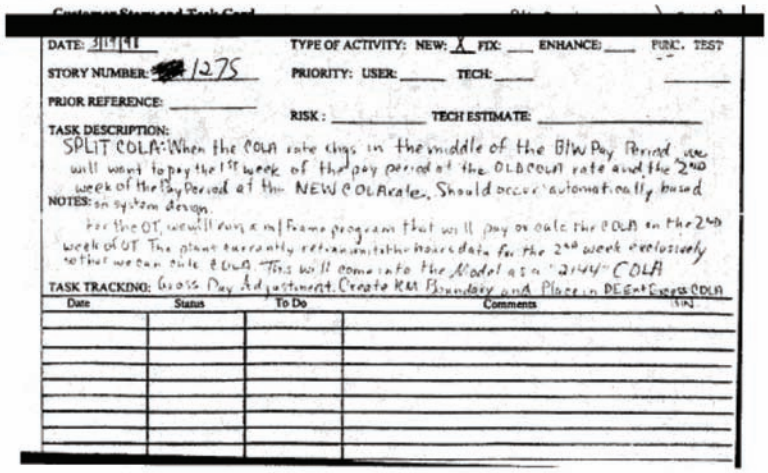
Story cards are one of the important aspects of the XP. They are playing vital role in XP. It describes functionality of system or software to be build that will be valuable to either purchaser or user of software. User stories are composed of three aspects (Cohen 2004):

- 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 (Cohn 2003). 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 (Kotyana and Somerville 1997). 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 (Azar, J. et. al. 2007). 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

Figure 1. Traditional Story Card (Back 2000)



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 (Wiegers, K. 1997). 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 ([Chaos 95]

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 (Beck 2000) which provides a traditional structure of story card.

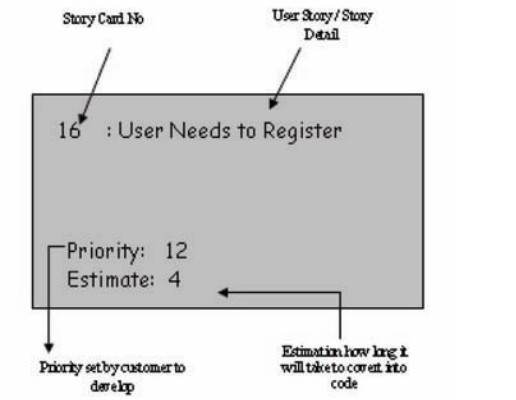
According to general template of story card, it is really difficult to well tackle the user requirements expressed on the cards. Traditionally developed story cards are not providing enough information of user functionality. They express user functionality in single to couple of sentences, which is really 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 2 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.

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

Figure 2. An example of traditional story card



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 chapter 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: IndependentN: NegotiableS: Small enough to fit into iterationE: Estimatable or easy to EstimateR: 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 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.

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

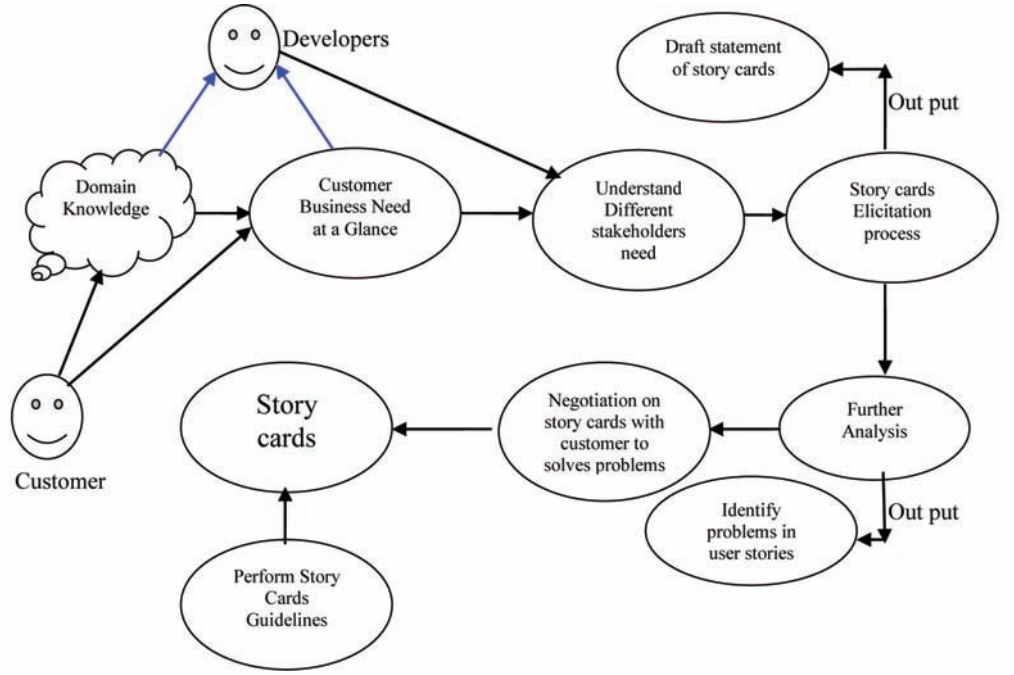
Consider the following Figure 3 which shows our new improved requirements elicitation process to capture user story on story cards based on the INSERT values.

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

Figure 3. INSERT method for requirements elicitation process in XP

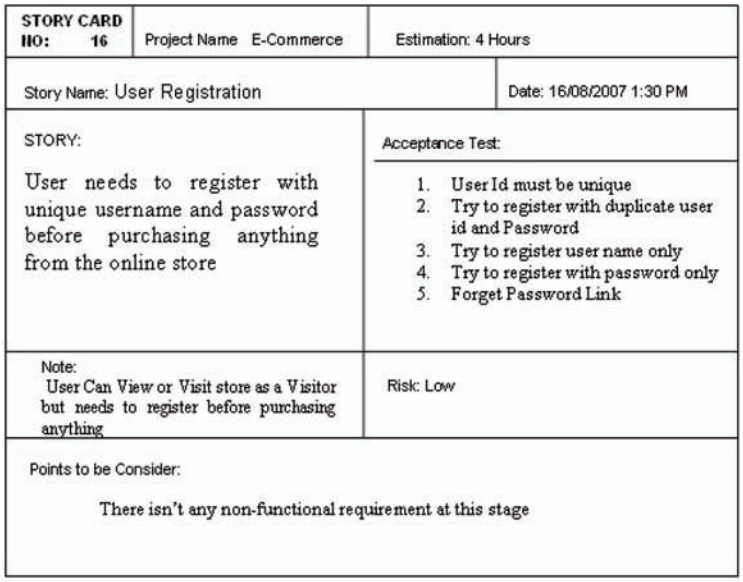


to focus on non-functional requirements. Also helps to keep in mind what they have to do to improve all the aspects of the business through structured system. Identified problems are being negotiated between developers and customer to acquire story cards. Following Figure 4 shows an example of story cards captured through the new improved requirement elicitation process based on the INSERT model.

Figure 3 INSERT 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

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



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

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

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.

3.2. Define a Standard Story Card Structure

Story card should have a common structure 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 anyone can use their knowledge of previous story cards when reading a new story cards. They can find information more easily and understand the relationship between parts of the story cards. The standard structure of the story cards 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, product or function perspective, user characteristic, and assumptions and dependencies if exist. Standards make story card easier to read. Standards make story cards (Requirements) easier to collect, and standards make story cards easier to write as well.

3.3. Write Requirement on the Story Card in the Format of the Summary or Abstract in Two to Three Sentences Long

You should always write requirement on the story card in the format of the summary or abstract in two to three sentences long, which summarise the purpose and principle of the story cards. It is much easier for developer and customer to figure out user story if they have abroad picture of what they are trying to understand. Summarising the requirements allows story cards to make forward reference to other requirements.

3.4. 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.

3.5. 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.

3.6. 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.

3.7. 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.

3.8. Story Card Must Present Business Requirements

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.

3.9. 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.

3.10. Define Specialised Terms used in the Story Cards

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

3.11. Story Card should Easy to Change or Modify

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.

3.12. investigation of System Feasibility

It is beneficial to carry out feasibility study, before investigating efforts and allocating expenses on requirement engineering for story cards work shop. This will tell us the suitability of the implemented given existing technology and effectively integrated with our current way of working.

3.13. 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, we are likely to miss important requirements. The system may have to rework to include these requirements on the later date. Identifying stakeholders and discussing the system with them makes people fell that they are the part of the requirement elicitation process

3.14. 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.

3.15. Define System Boundaries during the Domain Analysis for Story Cards

Domain analysis is a good practice to follow the successful requirement engineering process. During the requirement engineering process, we should identify which story cards are system requirement story cards, which are presenting the operational process associated with system, and which story cards are outside the scope of the system 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 outside of the scope.

3.16. 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.

3.17. Prioritise Story Cards Based on the Different Factors

This is one of the challenging issues for the extreme programming. Story cards are prioritized, if each requirement has an identifier to indicate, for example, its importance or volatility. In terms of importance, we can classify story cards into essential, conditional, and optional. The iteration or software needs to fulfil essential requirements in an agreed manner to be acceptable. While conditional requirements enhance the software, their absence does not make it unacceptable.

3.18. Use Language Simply, Consistently and Concisely

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.

3.19. 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.

3.20. Uniquely Identify each Story Cards and User Stories on Story Cards

Each story card should be assigned a unique identifier or story card no. unique identifier may helps to make a reference to related requirements for tractability. If we store story cards on a database system the story card identifier works as a unique columns to identify story card on database

3.21. 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)

3.22. 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.

3.23. Consideration of Domain Analysis

Domain constraint is the system requirements which is retrieve from the domain analysis of the system. We should study the domain and do domain analysis and understand constraint as a part of the story card elicitation process. Domain analysis generates system story cards and place limitation on other requirements. If we take domain considerations into account, there may be insuperable legal, organisational or physical obstacles to implementing the proposed story cards.

3.24. Write Story Cards from Multiple Viewpoints (Collect Requirements from Multiple Viewpoints)

There are many influences on the story card for a XP based projects. These includes the stakehold-

ers, end-user of the system, whor are involved in the process. These stakeholders are potential source of system requirement or story cards and they have their own viewpoint on the service that the system. If you collect requirements from the single point of the view, we unlikely to meet the needs of other stakeholders in the system. Collecting story cards or requirements from their multiple viewpoint is a useful way of prioritising requirements.

3.25. Specify Non-Functional Requirements also on Story Cards

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. 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.

3.27. 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 meete 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.

3.28. Use a Database System to Store Story Cards Compared to Write them on the Card and Destroy

Rather than maintaining requirements on cards, establish a story cards database and store the individual story cards in the database. Managing story cards in a database make easier to maintain the link between the story cards

3.29. Assess Story Cards Risk

For each story cards, carry out a risk analysis in which we suggest possible problems which may arise in the implementation of that story cards. Explicit risk assessment means identifying story cards which are likely to cause particular difficulties to the system developers. If these can be identified at this stage, it may be possible to modify the story cards to reduce the risks to the development process. Assessing risk to story cards often reveals that insufficient information about the requirement is available and that further details should be discovered from requirement stakeholders.

Table 1. Story card 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	√	√	√	√
Define a standard story card structure.		√		
Include the summary or abstract in two to three sentences of the requirements	√	√		
Dependencies of story cards	√	√		
Define an unambiguous story cards. (unambiguous)		√	√	
Define simplicity on story cards. (Simple)	√	√	√	√
Story cards must be consistent		√		
Story card must represent business requirements	√	√	√	
Define specialised terms used in the story cards		√		
Make the story cards easy to change		√		
Investigation of system feasibility		√		
Identify and consult the system stakeholders from the onsite customer		√		
Valuable to the customer.	√	√	√	√
Define system boundaries during the domain analysis for Story cards		√		
Story card must be negotiable		√	√	
Prioritise story cards based on different factors	**Limited**	√	**Limited**	**Limited**
Use language simply, consistently and concisely		√		
Check the story cards meet the XP Principle and values		√		
Uniquely identify each story cards and user stories on story cards		√		
User story writing technique.	√	√	**Limited**	
On-site customer	√	√	√	√
Domain analysis		√		
Write story cards from multiple viewpoints		√		
Specify non-functional requirements also on story card.		√		
Try to links between story card requirements and stakeholder		√		
Propose or Define validation checklist for story cards (Write acceptance tests)		√		
Use a database system to store story cards compared to write them on the card and destroy		√		
Assess story cards risk.		√		
Prioritize requirements based on the story card's value and XP values		√		
Reduced development Efforts.		√		
defined or representation of requirements by the developer and customer		√		√

3.30. Reduced Development Efforts

The preparation of the story cards forces the various concerned group in the customer organisation. To consider rigorously all of the requirements before design begins and reduce later redesign, recording and retesting. Careful review of the story cards can reveal omissions, misunderstanding, and inconsistency early in the development cycle when these problems are easier to correct.

3.31. Defined or Representation of Requirements by the Developers and Customer

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.

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 and best practices guidelines for story card.

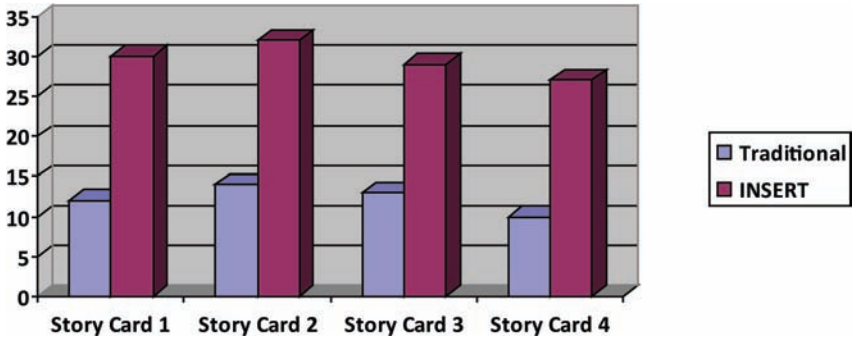
4. CASE STUDY

We apply INSERT technique based on the best knowledge based guidelines to the few functional and unique requirements of the same project called e-commerce online store. Table 2 and Figure 5 show a result, this shows that the INSERT model increased quality of the user story on story cards

Table 2. Comparison of traditional methods and INSERT method on key story cards

Story Card No	User Story Title	No Of Passed Guidelines		Quality Percentage of user Story	
		Traditional story cards	INSERT Story Cards	Traditional Story Card	(Improved) Quality by INSERT Story Cards
1	Admin Login	12	30	35	85
2	User Registration	14	32	40	91
3	Payment Method	13	29	38	83
4	Shipping Products	10	27	29	77

Figure 5. Comparison graph of traditional methods and INSERT method on key story cards



compared to the traditional methods. We apply this INSERT model to the key requirements or key story cards of the project to set up a prototype of this model.

5. CONCLUSION

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 (eg, 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 chapter 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.

6. REFERENCES

Beck, K. (2000). *Extreme programming Explained Embraced Change*. Reading, MA: Addison-Wesley.

Beck, K., & Fowler, M. (2000). *Planning Extreme Programming.* Reading MA: Addison-Wesley Press.

Boehm, B. (1981). *Software Engineering Economics.* Englewood Cliffs, NJ: Prentice Hall.

Cohn, M. (2003). *User stories applied for Agile Software Development.* Reading, MA: Addison-Wesley.

Faulk, S. (1996). Software Requirements: A Tutorial. In M. Dorfman & R. H. Thayer (Eds.) *Software Engineering*, (pp. 82-103). Washington, DC: IEEE Computer Society Press.

Karlsson, J. (1996). Software Requirements Prioritizing. *IEEE Proceeding of ICRE'96.*

Karlsson, J., & Ryan, K. (1997). A cost-Value Approach for Prioritizing Requirements. *IEEE Software*.

Kotyana, G., & Sommerville, I. (1997). *Requirements Engineering Processes and Techniques.* Hoboken, NJ: John Wiley and Sons Ltd.

Niu, N., Easterbrook, S. (2007). So You Think you know other's goal? A Repertory Grid Study. *IEEE Software, March/April*.

Paetsch, F., Eberlein, A., & Maure, F. (2003). Requirements Engineering and Agile Software Development. In *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* 2003 (WETICE'03).

Ramachandran, M. (2005). *A Process Improvement Framework for XP based SMEs*. Paper presented at 6th International Conference on Extreme Programming and Agile Processes in Software Engineering, Sheffield, UK, June 2005.

Wiegers, K. (1999, September). First Things First: Prioritizing requirements. *Software Development, 7*(9). Retrieved from www.processimpact.com/pubs.shtml#requirements