A USER STORY QUALITY MEASUREMENT MODEL FOR REDUCING AGILE SOFTWARE DEVELOPMENT RISK

Sen-Tarng Lai

Department of Information Technology and Management, Shih Chien University, Taipei, Taiwan

ABSTRACT

In Mobile communications age, the IT environment and IT technology update rapidly. The requirements change is the software project must face challenge. Able to overcome the impact of requirements change, software development risks can be effectively reduced. Agile software development uses the Iterative and Incremental Development (IID) process and focuses on the workable software and client communication. Agile software development is a very suitable development method for handling the requirements change in software development process. In agile development, user stories are the important documents for the client communication and criteria of acceptance test. However, the agile development doesn't pay attention to the formal requirements analysis and artifacts tracability to cause the potential risks of software change management. In this paper, analyzing and collecting the critical quality factors of user stories, and proposes the User Story Quality Measurement (USQM) model. Applied USQM model, the requirements quality of agile development can be enhanced and risks of requirement changes can be reduced.

Keywords

Agile development, user story, software project, quality measurement, USQM

1. INTRODUCTION

In software development process, it is often necessary to face challenge of requirements change. In order to reduce development risks, software project must effectively overcome the impact of requirements change [1], [2], [3]. Requirements change often affects software development operations and makes the development flow need back to earlier development phases for revising related artifacts. It not only need invest extra resource and cost, but also may cause the schedule delay [1], [2]. In requirements change process, affected design and development documents unable to effectively isolate, will increase software development risk. In addition, affected design and development documents unable completely and correctly modify, will greatly reduce project success ratio. There are many factors may affect the software project failure. One of critical issues is software design and related documents can't immediately revise and effectively adjust with the requirements change. For this, software development process must have high isolation capability and modification flexibility for reducing requirements changes risks.

The paper discusses and surveys mutual relationship between the software development models and system requirements. Because of the requirements change often is the critical factor to cause project fail. Iterative and Incremental Development (IID) model has been widely used by many software methodologies (ex. Unify process, Spiral model and Agile Process). Iterative technology makes the requirements have many change opportunities. Incremental concept can effectively decrease the requirements complexity.

DOI: 10.5121/ijsea.2017.8205

IID model can reduce the partial risk of software requirements change. Agile development model is a new software development methodology [4], [5], [6] that uses iterative and incremental development model to reduce the impact of requirements change [7], [8]. Agile development model has high adaptability and high tolerability for requirements change. In addition, agile development model uses IID to reduce the requirements complexity, refactoring to increase the requirement modified flexibility, do not rely on the documents can reduce affected items of requirements change. In requirements change, agile development model greatly decreases schedule delay, cost out of budget and quality unsatisfied requirement events, requirements change risk can be effectively reduced. However, the agile software development neglects analysis phase operations and documents, emphasizes workable products, does not pay attention to follow-up maintenance operations that are major shortages [4], [5], [6], [9].

In agile software development, formal requirements documents don't be regarded as the necessary items. User story is a short description customer's need. User story also is commonly used in the agile software methodologies of requirements description. User story becomes an important basis for agile development process and handling requirements change. In order to reduce agile software development risks, in this paper, analyzing and collecting the major quality factors of user story. Based on quantified quality, the paper proposes the User Story Quality Measurement (USQM) model. In USQM model, discussable, controllable and confirmable qualities will be measured and combined. Applied the quantification mechanism of USQM model, user story quality defects can be timely identified. Applied rule-based correction and improvement manner can concretely enhance user story quality and reduce agile software development project risks. In Section 2, survey critical factors of software project failure and describe the relationship between system requirement and development model. Many factors may affect user story quality, in Section 3, discusses user story quality factors which directly affect the development risks of agile software project. In Section 4, proposes the USQM model, and defines the rule-based user story quality defects identification and improvement manner. In Section 5, draw up the USQM based user story quality improvement procedure. Finally, in Section 6, describes the advantages of USQM model and does conclusions of this paper.

2. REQUIREMENTS CHANGE RISKS AND CRITICAL DEVELOPMENT MODEL

Requirements change is one of major critical factor to cause project fail. The section discusses the more suitable development model for overcoming requirements change.

2.1. Critical factors of software project failure

According to the Standish group study report which investigated large volume software project, the success rate of software project only approach one third [10], [11]. 80% failure software projects suffer from the thorny problems which include cost over budget, schedule delay and not compliance requirements. High failure risk of software project comes from the schedule delay, insufficient budget and unfinished requirements etc. events [3], [12], [13]. Four critical events are major reason to cause the software project with high risks (shown as Table 1):

(1) In requirement phase, system analysts acquire and collect incomplete documents and data to cause system requirements and user requirements existed inconsistent situations.
(2) In software development process, new IT technologies and IT environments are innovated continuously. It causes the existing plans can not completely adapt to new technologies and environments.
(3) Entering development phases, the client proposed to adjust, modify or delete the existed requirement items. Some new requirements are even required to append into the system. These requirements changes will greatly impact to follow up software development operations.

(4) Each phase operation of software development needs different resource which includes developers, hardware devices, software tools and development environment. Resource allocation that can't adjust to requirements change may increase software development risks.

Table 1. Four major reasons of software project failure

Occurred development phase	Failure reasons
Requirement phase	Occurred inconsistency between system requirements and user requirements
Design and implementation phases	Can't quickly fit the new technologies and environments.
All development phases	Can't meet the continuous requirements change
All development phases	Can't timely adjust the resource allocation

Summary the above description, incomplete system requirements, technology and environment evolution, client change requests and resource reallocation are four major events of increasing project risks. These events often cause requirements change. In software development process, the requirements change events can't be avoided or excluded. Requirements change is a critical reason of software project failure. Therefore, for reducing the project failure ratio, software requirements should have discussible, estimable, controllable, manageable, confirmable capability to handle each kinds of requirements change. It is because that new requirements of software system will be continually proposed until system be terminated or phased out.

2.2. Advantages and shortages of agile process

Most software development models have high relationship with the user requirements. Water fall model defines clear phase mission and very concerns the phase documents [14, 15]. The quality of requirement specification can't reach correctness, completeness and consistency, development operation will be denied to enter the following phase. Recently proposed development models have modified requirement specification style. In a time, it is not necessary describe all kinds and complete requirement items. By iterative development model, user can provide the requirement items incremental. Software development risk can be greatly reduced. Rapid prototyping model quickly develops prototype product and becomes a well communication channel with user. The prototype can help recognize the incomplete, inconsistent or incorrect requirement specification. Spiral development model very concerns the development risk and has high flexibility. In development process, spiral can adjust suitable development method for encountering different risks. However, the development risk can't effective be improved or reduced, the spiral recommends to terminate or abort software project. Each development model has an adjustment strategy for the project requirements change. The adjustment strategy can't effectively reduce the requirements change risks that will impact success ratio of software project.

In February 2001, seventeen software developers met at a Utah (ski resort) of USA for two days and produced the Manifesto for the agile software development. Many of participants had previously authored their own software development methodologies, including Extreme programming, Crystal, and Scrum [4, 6, 9]. Agile software development proposes several critical viewpoints:

(1) In development process, does not concern analysis and design phase operations and documents.
(2) As soon as possible to enter programming phase, workable software is more practical than development document.
(3) Support and provide high changeability software system.
(4) Enhance the cooperative relationship between developer and user, user can fully attend the development team.

In time management side, agile software development applies time-boxing approach to control process schedule [4, 9]. Requested software project must release a new version in two or three weeks. Let client clearly understand the development progress and test, audit the requirement of new version. In each day, a fifteen minutes stand up meeting is fixedly held to effectively reach the fully communication between client and developers. In addition, agile process uses iterative and incremental development to reduce the requirement complexity, refactoring to increase the requirement modified flexibility, non-document oriented can reduce the cost of requirements change. In requirements change, agile development greatly decreases schedule delay, cost out of budget, and quality unsatisfied user requirement situations, software requirements change risk can be effectively reduced. However, agile process does not concern analysis and design phase operations and documents, applies non-document oriented development, does not pay attention to follow-up maintenance operation that are major shortages. The advantages and shortages of agile software development process are shown as Figure 1. The shortages of agile development should be improved or enhanced to overcome the impacts of requirements change.

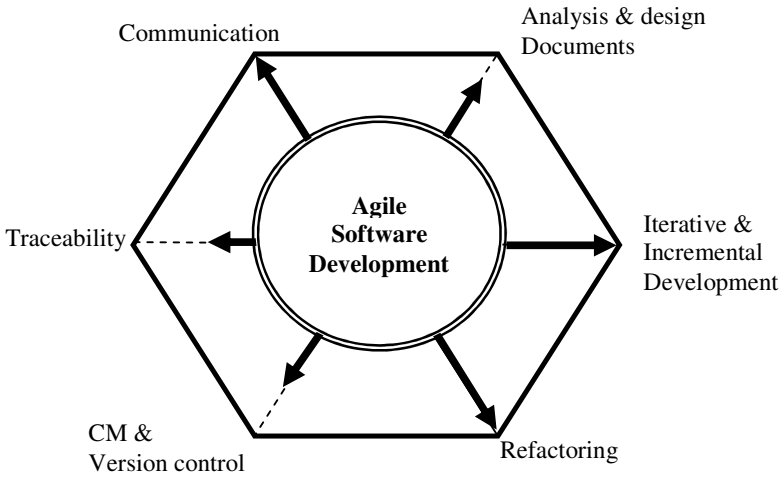

Figure 1. Advantages and shortages of agile software development

3. USER STORY QUALITY

Agile software development has high correlation with user stories. Therefore, quality of user stories becomes a critical indicator to impact the agile software development project risks.

3.1 Importance of user stories

User story is a short description customer's requirement. In agile software methodologies and frameworks such as Extreme Programming or Scrum, user stories are commonly used as a way of gathering requirements [16]. A user story is a form of software system requirement that has become the necessary items and quite popular in agile methodologies. Unlike the traditional methods such as a system requirements specification or use case diagrams, user story emphasis is

simplicity and changeability. Card, conversation, and confirmation are three critical representations of user stories [17]. In agile development process, formal requirements documents don't be regarded as the necessary items. Therefore, user story becomes an important basis for software development and requirements change. A user story describes a small piece of module functionality, in a simple and easy to read sentence. A well written user story will describe what the desired functionality is, who use it for, and why it is useful. A high quality user story should have independent, negotiable, valuable, estimable, small and testable characteristics [18, 19]. According to development needs, the user story description should have the uniform format. For reaching the requirements consistency, the similar and duplicate user stories should be removed. For avoiding the requirements conflict, the high correlation user stories should be merged.

In agile development process, user story becomes an important basis for software development and requirements change. User story should have five major quality characteristics:

- Discussable: Clear and simple contents of user story can provide the easy discussion among client, developers, end users and stakeholders.
- Estimable: Low complexity and high modularity user story can provide developers easy to estimate development time and cost.
- Controllable: Applied configuration management and version control can timely identify the difference between the story revision versions.
- Manageable: Establishing the cross-reference table can conveniently manage integrated relationships among the user stories.
- Confirmable: Correct, complete and consistent user story contents can help generate good test cases for unit testing, integration testing and product acceptance testing.

3.2. Quality factors of user stories

User story quality is the key problem to affect software development efficiency and requirements change effort. User stories should consider basic, management, and acceptance three levels quantified quality. Basic quality includes discussable and estimable characteristics, management quality includes controllable, manageable characteristics, and acceptance quality includes confirmable characteristics. Based on three levels quality, user stories quality measurement should consider the follow quality factors:

(1) Basic (discussable and estimable) quality: In agile software development process, user stories quality can affect directly the iterative and incremental operation. Ambiguous, complex, high coupling and low cohesion user stories often become the troubles of software development especially in requirements change. For enhancing discussable and estimable characteristics, user stories should have quality factors:

- Clarity documents: User stories are the discussion basis of software development features. Clarity is a necessary factor for documents discussion and communication. Clarity documents must have correctness, completeness, and consistency and uniform documentation format.
- Low complexity: User stories complexity can be measured with size, logic complexity [20] and data structure complexity [21]. User stories complexity should have the feature to determine the development schedule and cost.
- High modularity: User stories have an important mission of iterative and incremental operation. The modularity user stories make developers to build more understandable systems that are easier to configure and extend to meet the evolving needs of stakeholders [22]. Modularity is a major quality factor for requirements changes and incremental development. Low coupling and high cohesion can create a high modularity user stories.

(2) Management (Controllable and manageable) quality: Configuration Management (CM) system can manage all development phase documents, control product versions and documents cross-reference relationship [14]. Configuration management mechanism should consider follows three factors:

- CM system: A well and practical CM procedure is used to manage the related operations for phase documents check-in and check-out [14].
- Version control: CM system need combine the suitable and practical version control tools to handle, record and storage the difference among the user story versions [9, 14].
- Cross-reference tables: Cross-reference tables of development phase documents are critical item to concretely establish the interconnection relationships and traceability of user stories.

(3) Acceptance (Confirmable) quality: One of important mission of user story is to help accomplish the acceptance testing. Therefore, user story need the qualified assures contents to help well communication, high testability to help acceptance testing, and verifiability to help product verification.

- Assured contents: In order to establish effectively a common consensus among clients, developers and stakeholders, user story should have correct, complete and consistent contents.
- Testability: In order to confirm the product features, user stories should have high testability [23]. Testable user story can guide testers to draw up the test plan and complete test cases to assure the acceptable product features.
- Verifiability: In agile software development, user stories are the base of BDD and IID. Verifiable user stories can assistant assure the behavior correctness, integrated functionality completeness and system acceptance validation [23].

The architecture of user stories quality factors is shown as Figure 2.

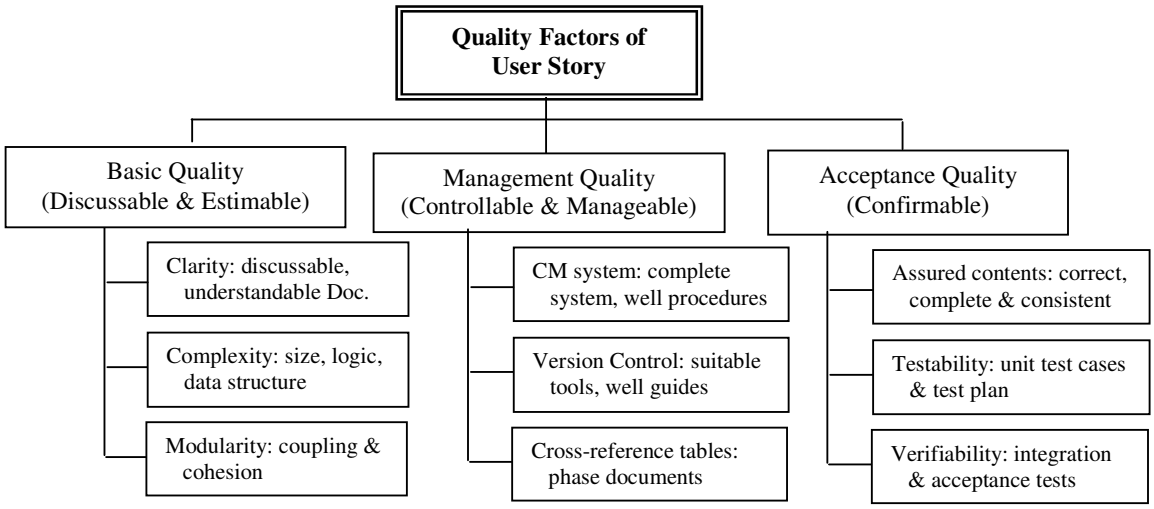

Figure 2. The architecture of user story quality factors

3.3 Quality factors collection and normalization

For collecting useful factors, it is necessary to develop a set of complete and clear inspection checklists to conduct the user story discussion activity. Checklists can detect and identify the

quality and incomplete defects of user story. User story discussion and implementation can help collect the discussible, estimable, controllable, manageable and confirmable quality factors of user story. However, individual factor or measurement can only measure or evaluate the specific quality characteristic. In order to effectively monitor and assess the maintenance process defects, individual measurements should make the appropriate combination [24, 25, 26]. Two kind of metric combination models are Linear Combination Model (LCM) [24, 25], and Non-Linear Combination Model (NLCM) [24, 26]. NLCM has higher accuracy measurement than LCM. However, LCM has high flexibility, more extensible and easy formulation than NLCM. For this, in this paper, LCM is applied to maintenance process quality measurement. The different level development activities have different quality metrics be shown. Therefore, before applying the linear combination model, the quality factors must be normalized. The normalized value is between 0 and 1. The best quality quantified value approaches to 1 and the worst quality approaches to 0.

4. USER STORIES QUALITY ENHANCEMENT

Enhancing user stories quality can effectively decrease the impact of requirements change and concretely reduce the development risk of software project.

4.1 User story quality measurement model

User stories should consider three levels quantified quality that include basic, management, and acceptance quality. Using the LCM, related quality factors can combine into the primary metric, and then the related primary metrics can combine into the quality measurements. Finally, combines with three critical quality measurements and generates a user story quality indicator. Four combination formulas described as follows:

(1) Discussable and Estimable Quality Measurement (DEQM) is combined clarity documents, low complexity and modularity three metrics. The combination formula is shown as Equation (1):

DEQM: Discussable and Estimable Quality Measurement

CC: Clarity Contents	W_1*: Weight of CC*
LC: Low Complexity	W_2*: Weight of LC*
HM: High modularity	W_3*: Weight of HM*

$$DEQM = CC * W_1 + LC * W_2 + HM * W_3 \qquad W_1 + W_2 + W_3 = 1 \qquad (1)$$

(2) Controllable and Manageable Quality Measurement (CMQM) is combined CM system, version control tools and cross-reference table three metrics. The combination formula is shown as Equation (2):

CMQM: Controllable and Manageable Quality Measurement

CMS: CM System	W_1*: Weight of CMS*
VCT: Version Control Tools	W_2*: Weight of VCT*
CRT: Cross-Reference Table	W_3*: Weight of CRT*

$$CMQM = CMS * W_1 + VCT * W_2 + CRT * W_3 \qquad W_1+W_2+W_3=1 \qquad (2)$$

(3) Confirmable Quality Measurement (CFQM) is combined assured contents, testability and verifiability metrics. The formula is shown as Equation (3):

CFQM: Confirmable Quality Measurement

AC: Assured Contents	W_1*: Weight of AC*

TM: Testability Metric — *W_2: Weight of TM*
VM: Verifiability Metric — *W_3: Weight of VM*

$$CFQM = AC * W_1 + TM * W_2 + VM * W_3 \qquad W_1+W_2+W_3=1 \qquad (3)$$

(4) Finally, combine DEQM, CMQM, and CFQM into a User Story Quality Indicator (USQI). The formula is shown as Equation (4):

USQI: User Story Quality Indicator
DEQM: Discussible and Estimable QM — *W1: Weight of DRQM*
CMQM: Controllable and Manageable QM — *W2: Weight of CMQM*
CFQM: Confirmable QM — *W3: Weight of CFQM*

$$USQI = DEQM *W_1 + CMQM *W_2 + CFQM * W_3 \qquad W_1+W_2+W_3 = 1 \qquad (4)$$

In agile software development, user stories which affect requirements change were collected are major resource of software development and divided into 9 groups. In first layer, 9 groups basic quality factor are combined into 9 quality metrics. In second layer, 9 quality metrics are combined into three quality measurements. In third layer, three quality measurements are combined into a User Story Quality Indictor (USQI). With three layer combination formulas to generate a USQI, the process is called a User Story Quality Measurement (USQM) model. The USQM model architecture is shown as Figure 3.

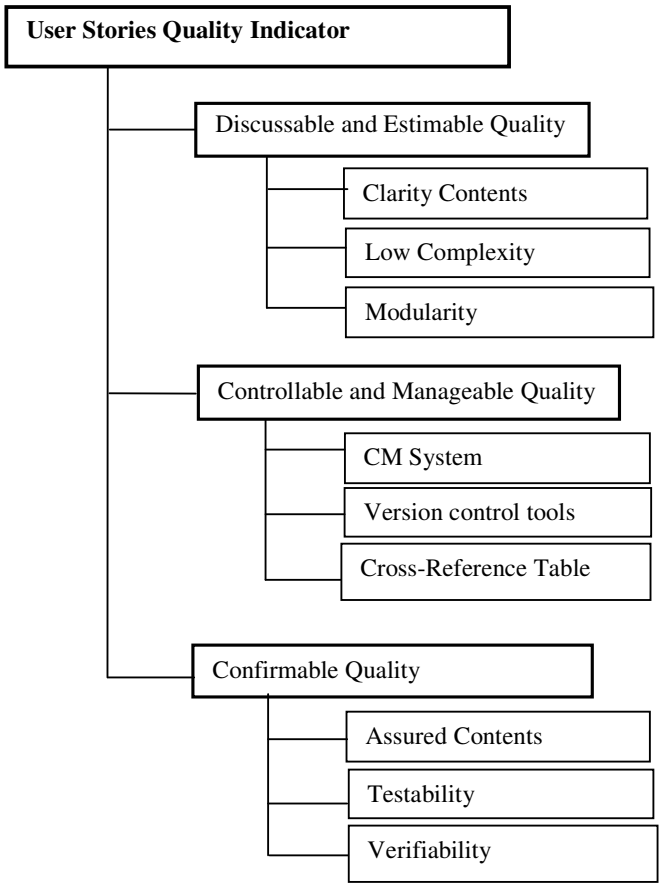

Figure 3. Architecture of USQM model

4.2. Rule-based quality correction manner

USQI is a relative judgment mechanism also is a basis to identify problems or defects of user stories. In USQM model, basic quality factors are combined into high level quality measurements. High level quality measurements are combined into a USQI. Therefore, if the USQI does not satisfy quality criterion, it represents user stories existed some related defects. According to the combination formulas, quality measurements map to three measurement equations and several quality factors. Related activities should be rigorously inspected to identify the problem or defect and propose the corrective action. This paper proposed the rule-based improvement activity for increasing user stories quality, described as follows:

<Rule 1> IF USQI does not satisfy user story quality criterion

THEN DEQM, CMQM and CFQM should be analyzed to identify the problems and defects of related documents or activities.

<Rule 2> IF DEQM does not satisfy discussable and estimable quality criterion

THEN clarity contents, low complexity and high modularity quality factors should be analyzed to identify the defects of related documents or activities.
<Action 1> Identify the activities and reasons to cause user stories basic quality defects.
<Action 2> Confirm the defects revision or correction has accomplished.

<Rule 3> IF CMQM does not satisfy controllable and manageable quality criterion

THEN CM system, version control tools and cross-reference table quality factors should be analyzed to identify the defects of related documents or activities.
<Action 1> Identify the activities and reasons to cause user stories management quality defects.
<Action 2> Confirm the defects revision or correction has accomplished.

<Rule 4> IF CFQM does not satisfy confirmable quality criterion

THEN communication capability, testability and verifiability quality factors should be analyzed to identify the defects of related documents or activities.
<Action 1> Identify the activities and reasons to cause user stories assurance quality defects.
<Action 2> Confirm the defects revision or correction has accomplished.

5. USER STORY QUALITY IMPROVEMENT PROCEDURE

The requirement items change is the agile software development project must face challenge. Able to overcome the impact of requirements change, software project development risks can be effectively reduced. Based on USQM model, the paper develops the user story (requirement items) quality improvement procedure (USQIP) for timely correcting user story quality defects and concretely improving agile development and change management quality. The improvement procedure divides into five phases (shown as Figure 4)

- User stories selection:
  - First step of improvement procedure is user stories selection.
  - From cards of user stories select the critical and suitable story contents as the version functions
  - The selected user story has to pass the user stories selection procedure.

- BDD based user story adjustment:
  - For reaching the consistency, the similar and duplicate user stories should be removed.
  - For avoiding the conflict, the high correlation user stories should be merged.
  - According to development needs, the user stories description should have the uniform format.
- Quality measurement:
  - Collect quality factors of user story in the user story discussion and BDD operation.
  - Quantify the quality factors of user story.
  - Applied the USQM model to measure user story quality.
- Defects identification:
  - Based on predefined quality criteria to determine qualify user story quality.
  - Identify the quality defects of user story.
  - Confirm the quality defects of user story.
- Quality defects revision:
  - Applied the rule-based quality correction manner to revise the user story.
  - The revised user story has to pass the formal review procedure.
  - Save the new version user story to the source repository for version control.

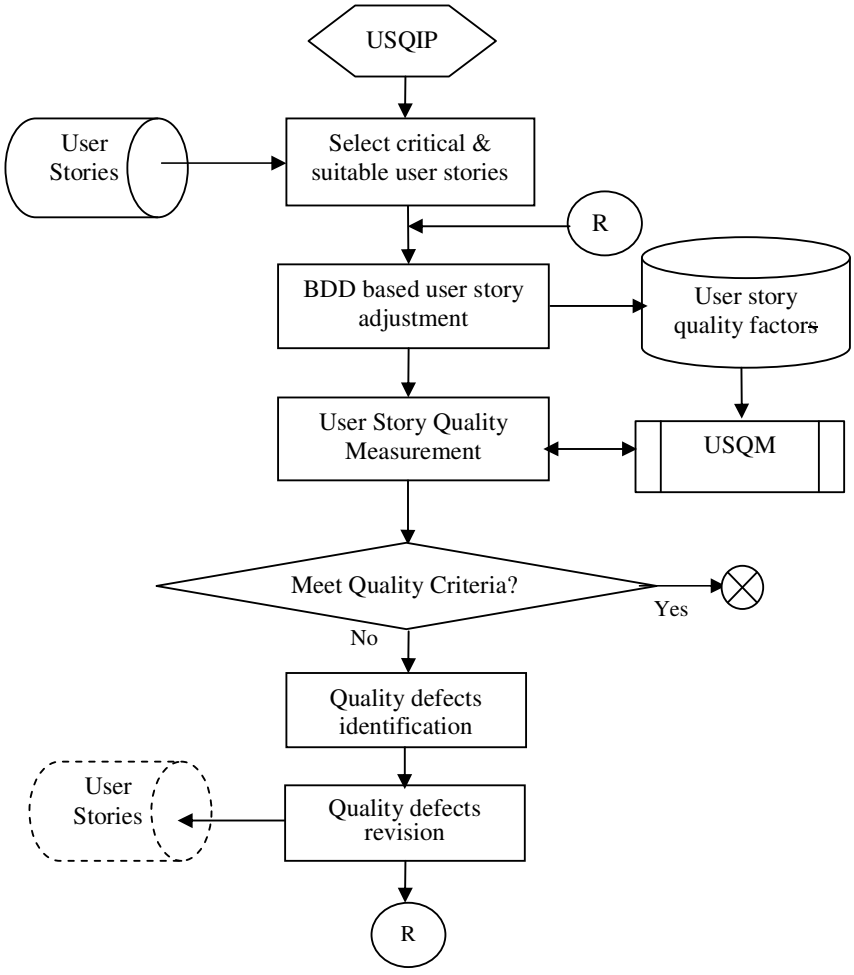

Figure 4. Flowchart of user story quality improvement procedure

6. CONCLUSIONS

Requirements changes often cause software project failure. In order to reduce software project risks, software development must conquer the challenges of requirements changes. Major advantages of agile software development are that agile methodologies very concern communication, workable software, and refactoring mechanism. IID and refactoring mechanism can reduce potential risk of requirements change. However, agile software development doesn't stress formal requirements documents and requirements traceability. Causing the user story becomes an important basis for agile software development and requirements change management. In order to handle requirements change and reduce software development risk, in this paper, analysing and collecting the critical quality factor of user story and proposes the User Stories Quality Measurement (USQM) model. Applied USQM model can enhance development requirements quality and efficiently handle requirements change to reduce software development risk. The advantages of USQM model are described as follows:

(1) Timely identify user story quality defects and provide correction suggestions.
(2) Effectively improve user stories quality for handling requirements change.
(3) Combination formula of USQM model has precise, simple and flexible adjustment attributes.
Based on USQM model, the user story quality improvement procedure is applied for improving user story quality and reducing agile software project development risks.

ACKNOWLEDGEMENTS

The research was supported by Ministry of Science and Technology research project funds (Project No.: MOST 105-2221-E-158-002)

REFERENCES

[1] S. A. Bohner and R. S. Arnold, 1996. Software Change Impact Analysis, IEEE Computer Society Press, CA, pp. 1-26.

[2] S. A. Bohner, 2002. Software Change Impacts: An Evolving Perspective, Proc. of IEEE Intl Conf. on Software Maintenance, pp. 263-271.

[3] B. W. Boehm, 1991. Software risk management: Principles and practices, IEEE Software, 8(1), 1991, pp. 32-41.

[4] A. Cockburn, 2002. Agile Software Development, Addison-Wesley.

[5] M. Cohn and D. Ford, 2003. Introducing an Agile Process to an Organization, IEEE Computer, vol. 36 no. 6 pp. 74-78, June 2003.

[6] V. Szalvay, “An Introduction to Agile Software Development,” Danube Technologies Inc., 2004.

[7] C. Larman and V. R. Basili, 2003. Iterative and Incremental Development: A Brief History, IEEE Computer, June 2003.

[8] C. Larman, 2004. Agile and Iterative Development: A Manager's Guide, Boston: Addison Wesley.

[9] S. R. Schach, 2010. Object-Oriented Software Engineering, McGraw-Hill Companies.

[10] J. L. Eveleens and C. Verhoef, 2010. The Rise and Fall of the Chaos Report Figures,” IEEE Software, vol. 27, no. 1, pp. 30-36.

[11] The Standish group, 2009. "New Standish Group report shows more project failing and less successful projects," April 23, 2009. (http://www.standishgroup.com/newsroom/chaos_2009.php)

[12] B. W. Boehm, 1989. "Tutorial: Software Risk Management," IEEE CS Press, Los Alamitos, Calif.

[13] R. Fairley,1994. "Risk management for Software Projects," IEEE Software, vol. 11, no. 3, pp. 57-67.

[14] R. S. Pressman, 2010. Software Engineering: A Practitioner's Approach, McGraw-Hill, New York, 2010.

[15] David S. Frankel, 2003. Model Driven Architecture: Applying MDA to Enterprise Computing, John Wiley & Sons.

[16] Mike Cohn, 2004. User Stories Applied: For Agile Software Development, Addison-Wesley Professional; 1 edition.

[17] Ron Jeffries, 2001. "Essential XP: Card, Conversation, Confirmation," Posted on: August 30, 2001. (http://xprogramming.com/index.php)

[18] Bill Wake, 2003. "INVEST in Good Stories, and SMART Tasks," Posted on August 17, 2003, (http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/)

[19] Bill Wake, 2012. "Independent Stories in the INVEST Model," Posted on: February 8, 2012, (http://xp123.com/articles/independent-stories-in-the-invest-model/)

[20] T. J. McCabe, 1976. A Complexity Measure, IEEE Trans. On Software Eng., Vol. 2, No 4, pp.308-320.

[21] M. H. Halstead, 1977, Elements of Software Science, North-Holland, New York.

[22] Ivar Jacobson and Pan-Wei Ng, 2004, Aspect-Oriented Software Development with Use Cases, Addison-Wesley Boston, 2004.

[23] Ralph Young, 2001, Effective Requirements Practices, Addison-Wesley, Boston, 2001.

[24] S. D. Conte, H. E. Dunsmore and V. Y. Shen, 1986. Software Engineering Metrics and Models, Benjamin/Cummings, Menlo Park.

[25] N. E. Fenton, 1991, Software Metrics - A Rigorous Approach, Chapman & Hall.

[26] D. Galin, 2004. Software Quality Assurance – From theory to implementation, Pearson Education Limited, England.

AUTHOR

Sen-Tarng Lai was born in Taiwan in 1959. He received his BS from Soochow University, Taiwan in 1982, master from National Chiao Tung University, Taiwan in 1984 and PhD from National Taiwan University of Science and Technology, Taiwan in 1997. His research interests include software security, software project management, and software quality. He is currently an assistant professor in the Department of Information Technology and Management at Shin Chien University, Taipei, Taiwan.