A MAINTAINABILITY ENHANCEMENT PROCEDURE FOR REDUCING AGILE SOFTWARE DEVELOPMENT RISK

Sen-Tarng Lai

Dept. of Information Technology and Management, Shih Chien University,
Taipei, 104, Taiwan

ABSTRACT

In mobile communications age, environment changes rapidly, the requirements change is the software project must face challenge. Able to overcome the impact of requirements change, software development risk can be effectively decreased. In order to reduce software requirements change risk, the paper investigates the major software development models and recommends the adaptable requirements change software development. Agile development applied the Iterative and Incremental Development (IID) approach, focuses on workable software and client communication. In software development, agile development is a very suitable approach to handle the requirements change. However, agile development maintenance existed many defects that include development documents control, user story inspection and CM system. The maintenance defects of agile development should be improved. Analysing and collecting the critical quality factors of agile development maintainability, in this paper proposes the Agile Development Maintainability Measurement (ADMM) model. Based on ADMM model, the Agile Development Maintainability Enhancement (ADME) procedure can be defined and deployed for reducing the risk of requirements change.

KEYWORDS

Agile development, maintainability, requirements change, quality measurement, ADMM

1. INTRODUCTION

In software development process, it is necessary to face challenge of requirements change. Software project must overcome the impact of requirements change to effectively reduce development risk [1], [2], [3]. Requirements change often affects software development operations. Requirements change makes the development flow need back to earlier development phases for revising related artefacts. 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 adjustment capability and modification flexibility for reducing requirements changes risk.

DOI : 10.5121/ijsea.2015.6403

The paper discusses and surveys mutual relationship between the software development models and user 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)[4] . Iterative technology makes user requirements have many change opportunities. Incremental concept can effectively decrease the requirements complexity. IID model can reduce the partial risk of software requirements change. Agile development model is a new software development methodology [5], [6], [7] that uses IID model to reduce the impact of requirements change [8], [9]. Agile development model has high adaptability and high tolerability for requirements change. In addition, agile development model uses IID to reduce the requirement complexity, refactoring to increase the requirement modified flexibility, and 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, agile process neglects analysis and design phase operations and development documents, emphasizes workable products, does not pay attention to follow-up maintenance operations that are major shortages [5], [6], [7], [10].

In order to make up the disadvantages of agile development, development maintenance quality should be enhanced. There are many development maintenance quality factors which include development document system, user story (requirement item) inspection mechanism and configuration management system may affect requirements change operations. For overcoming the challenge of requirements change, in this paper, analysing and collecting the critical quality factor of development maintenance. Based on the metric combination model, in this paper presents the Agile Development Maintainability Measurement (ADMM) model. In ADMM model, development document control, user story inspection mechanism and configuration management system etc. qualities will be measured and combined. Based on ADMM model and applied rule-based defects identification and improvement manner, the Agile Development Maintainability Enhancement (ADME) procedure can be defined and developed for reducing the risk of requirements change. In Section 2, surveys the critical factors of software project failure, the relationship between system requirement and development models, and describes the advantages of agile development. Many factors may affect maintenance process quality, in Section 3, discusses maintenance process quality factors which are affected by requirements change. In Section 4, proposes the ADMM model, and develops the rule-based maintenance process quality defects identification and improvement manner. In Section 5, defines and deploys the Agile Development Maintainability Enhancement (ADME) procedure. Finally, describes the advantages of ADMM model and ADME procedure, and does a conclusions in Section 6.

2. REQUIREMENTS CHANGE RISK AND CRITICAL DEVELOPMENT MODELS

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]. Four critical events are major reason to cause the software project with high risk:

(1) In requirement phase, system analysts acquire and collect incomplete documents and information to cause software system requirements existed the inconsistent, incomplete and incorrect situations. Seriously impact the software system development follow operations.
(2) In software development process, requirements change, new technologies and operating environment are innovated continuously. It causes the existing development plans can not completely adapt to new situation.
(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 risk.

Summary the above description, incomplete system requirements, technology and environment evolution, client change requests and resource re-allocation are four major events of increasing project risk (shown as Figure 1). 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 process should have high adaptability and maintainability to handle many kinds of requirements change. It is because that new requirements of software system will be continually proposed until system be terminated or phased out.

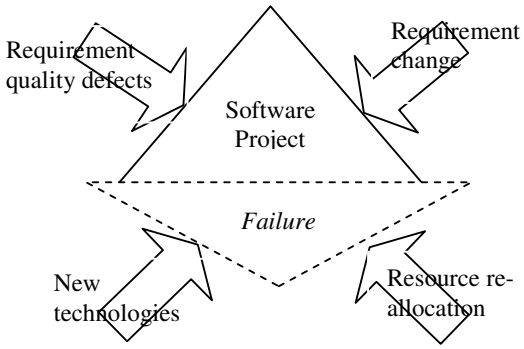

Figure 1. Four critical factors of software project failure

2.2. Advantages and Shortages of Agile Development

In 1967, the concept of software engineering was proposed and discussed in the NATO international conference [13]. In the period, with the growth of information technology, operational environment and user requirements, software development methodologies continuously progress. From early Fix-code to recently Model-Driven Architecture (MDA) [14] and agile development model about have a dozen development models. Most software development models have high relationship with the user requirements. Water fall model defines clear phase mission and very concerns the phase documents. The quality of requirement documents can't reach correctness, completeness and consistency, development process 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 specifications. 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 user requirements change. The adjustment strategy can't effectively reduce the software development risks that will impact success ratio of software project.

In February 2001, seventeen software developers met at the Utah (ski resort) of USA for two days and produced the Manifesto of 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 IID and user stories [15] 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 is shown as Figure 2. The shortages of agile development should be improved or enhanced to effectively overcome the impacts of requirements change.

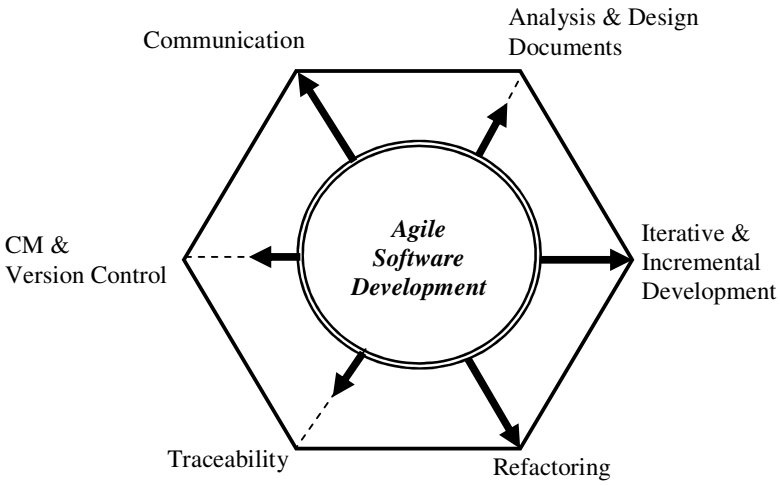

Figure 2. Advantages and shortages of agile development

3. DEVELOPMENT MAINTAINABILITY

Software development model evolution has high correlation with user requirements. Can appropriately handle user requirements, the software development risk can be reduced concretely.

3.1. Quality factors of development maintainability

Software development maintainability is a key point to affect change risk of user requirements. In software development, the development documentation tasks, requirement items inspection mechanism and Configuration Management (CM) system are critical activities of maintenance quality. Development maintenance quality should consider the critical factors that described as follows:

(1) Development documents control: In requirements change process, the documents quality is critical factor to directly affect the effort of requirement changes. Agile development should draw up a document review mechanism and consider follows three factors:

- Documents quality criteria: development documents must have correctness, completeness, consistency and standard documentation format. Agile development should provide the document review checklists to control documents quality. High quality documents can reduce the impaction of personnel mobile.
- Documents review procedure: agile development needs draw up a useful and practical review procedure to define the documents review steps and detailed tasks.
- Documents cross-reference table: document cross-reference tables is critical item to concretely create the interconnection relationships and traceability.

(2) User story (requirement item) inspection: Ambiguous, complex, high coupling and low cohesion requirement items often become the troubles of software development especially in requirements change. Therefore, user story quality mechanism should consider follows three factors:

- Clarity inspection: user stories are the discussion basis of software development. Clarity is a necessary factor for user story discussion and communication. Qualified user story should have high clarity to requirement description.
- Complexity inspection: requirement item complexity can be measured with size, logic complexity [16] and data structure complexity [17]. Qualified user story should have low logic and data structure complexity to requirement description.
- Modularity inspection: requirement items with high modularity can increase development maintenance quality [4]. Qualified user story should have the low coupling and high cohesion to requirement description.

(3) CM system: CM system can manage all kind development documents, control product versions and documents cross-reference relationship [13]. Configuration management mechanism should consider follows three factors:

- CM procedure: CM procedure: a well and practical CM procedure is used to manage the related operations for documents check-in and check-out.
- Version control tools: CM system need combine the suitable and practical version control tools to handle, record and storage the difference among the document versions.

- Change control procedure: for control requirement change, agile development should set up a Change Control Board (CCB) to audit and evaluate change requests [13]. And identify the affected development documents and artefacts in change operations.

The architecture of agile development maintenance quality factors is shown as Figure 3.

3.2. Quality factors collection and normalization

In CMMI, software process improvement is critical task for increasing software product quality. Collecting and quantifying the critical software activities factors is first step to improve software process quality. Each phase documents review and inspection activities can help collect the quality factors of document cross-reference. For collecting useful factors, it is necessary to develop a set of complete and clear inspection checklists to conduct the process audit activity. Audit checklists can detect and identify the quality and incomplete defects of software process. Software process audit can help collect the procedure quality factors of documents review, qualified user story and configuration management. Procedure items and steps checklists can help collect some the detailed tasks and criteria quality factors. Collecting documents review, user story inspection and configuration management mechanisms quality factors are the necessary task for quantifying the maintenance process quality.

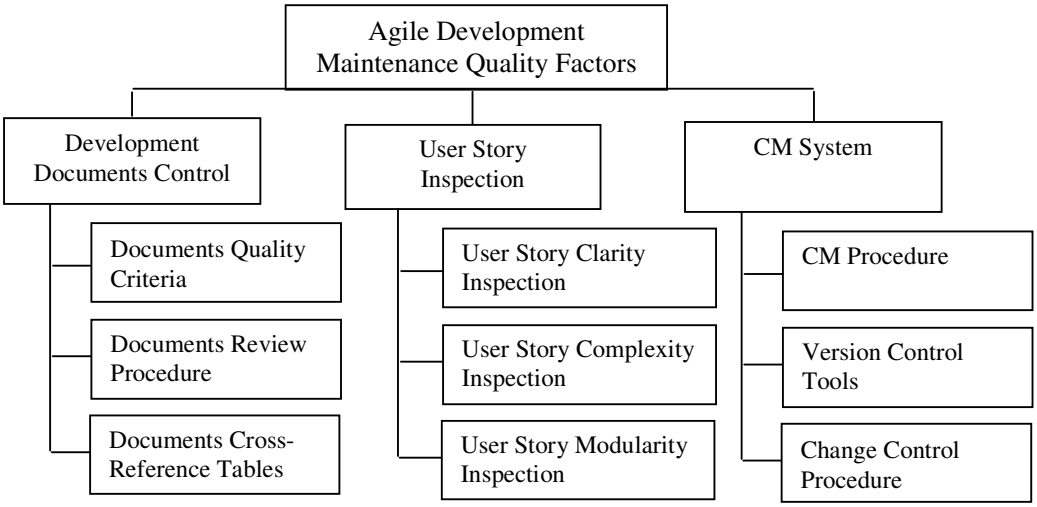

Figure 3. The architecture of maintenance process quality factors

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 [19], [20], [21]. Two kind of metric combination models are Linear Combination Model (LCM) [19], [20], [21] and Non-Linear Combination Model (NLCM) [19], [20], [22]. NLCM has higher accuracy measurement than LCM. However, LCM has high flexibility, more extensible and easy formulation than NLCM (Shown as Table II). For this, in this paper, LCM is applied to maintenance 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. MEASUREMENT MODEL AND DEFECTS IDENTIFICATION

Enhancing maintenance process can effectively decrease the impact of requirements change and concretely reduce the development risk of software project.

4.1. Maintainability measurement model

Maintenance process activities should consider three quantified quality that include documents traceability, configuration management and requirement items changeability. Using the LCM, the basic quality factors can be combined into the primitive metric, and then the related primitive metrics can be combined into a quality measurement. Finally, combines with three critical quality measurements and generates an agile development maintainability indicator. Four formulas described as follows:

(1) Development Documents Control Quality Measurement (DDCQM) is combined document review criteria, document review procedure and document cross-reference table three metrics. The combination formula is shown as Equation (1):

DDCQM: Development Documents Control Quality Measurement

DQC: Document Quality Criteria	*W1: Weight of DQC*
DRP: Document Review Procedure	*W2: Weight of DRP*
DCRT: Document Cross-Reference Table	*W3: Weight of DCRT*

$$DDCQM = DQC * W1 + DRP * W2 + DCRT * W3 \qquad W1 + W2 + W3 = 1 \qquad (1)$$

(2) User Story Inspection Quality Measurement (USIQM) is combined clarity, low complexity and modularity three determination capabilities. The combination formula is shown as Equation (2):

USIQM: User Story Quality Measurement

CDC: Clarity Determination Capability	*W1: Weight of CDC*
CDC: Complexity Determination Capability	*W2: Weight of CDC*
MDC: Modularity Determination Capability	*W3: Weight of MDC*

$$USIQM = CDC * W1 + CDC * W2 + MDC * W3 \qquad W1+W2+W3=1 \qquad (2)$$

(3) Configuration Management System Quality Measurement (CMSQM) is combined CM system, version control tool and change control procedure metric. The formula is shown as Equation (3):

CMSQM: Configuration Management Quality Measurement

CMP: CM Procedure	*W1: Weight of CMP*
VCT: Version Control Tools	*W2: Weight of VCT*
CCP: Change Control Procedure	*W3: Weight of CCP*

$$CMSQM = CMP * W1 + VCT * W2 + CCP * W3 \qquad W1+W2+W3=1 \qquad (3)$$

(4) Finally, combine DDQM, USQM, and CMQM into an Agile Development Maintainability Indicator (ADMI). The formula is shown as Equation (4):

ADMII: Agile Development Maintainability Indicator

DDCQM: Development Documents Control Quality Measurement	*W1: Weight of DDCQM*
USIQM: User Story Inspection Quality Measurement	*W2: Weight of USIQM*
CMSQM: CM System Quality Measurement	*W3: Weight of CMSQM*

$$ADMI = DDCQM *W1 + USIQM *W2 + CMSQM * W3 \qquad W1 + W2 + W3 = 1 \qquad (4)$$

In agile development, CM system, user stories and development documents inspection activity identified the factors which affect requirements change were collected 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 an Agile Development Maintainability Indictor (ADMI). With three layer combination formulas to generate an ADMI, the process is called an Agile Development Maintainability Measurement (ADMM) model. The ADMM model architecture is shown as Figure 4.

4.2. Rule-based maintenance quality defects identification

ADMI is a relative judgment mechanism and also a basis to identify problems or defects of maintenance quality. In ADMM model, basic quality factors are combined into high level measurement. High level quality measurements are combined into an ADMI. Therefore, if the ADMI does not satisfy quality criterion, it represents maintenance process existed some related defects and problems. According to the combination formulas, quality measurement mapping to three major quality measurements and some quality factors. The affected development activities and documents should be rigorously inspected to identify the problem or defect and propose the corrective action. This paper proposed the rule-based improvement activities for increasing maintenance quality of development, described as follows:

<*Rule 1*> IF ADMI does not satisfy "quality criterion"
THEN DDCQM, USIQM and CMSQM should be analysed to identify the problem and defect of related documents or activities.

<*Rule 2*> IF DDCQM does not satisfy "quality criterion"
THEN document quality criteria, document review procedure and document cross-reference table quality factors should be analysed to identify the defects of related documents or activities.
<Action 1> Identify the activities and reasons to cause development document management defects.
<Action 2> Confirm the defects revision or correction has accomplished.

<*Rule 3*> IF USIQM does not satisfy "quality criterion"
THEN clarity inspection, complexity inspection and modularity inspection quality factors should be analysed to identify the defects of related documents or activities.
<Action 1> Identify the activities and reasons to cause user story quality mechanism defects.
<Action 2> Confirm the defects revision or correction has accomplished.

<*Rule 4*> IF CMSQM does not satisfy "quality criterion"
THEN CM procedure, version control and tools, and cross-reference table quality factors should be analysed to identify the defects of related documents or activities.
<Action 1> Identify the activities and reasons to cause CM system defects.
<Action 2> Confirm the defects revision or correction has accomplished.

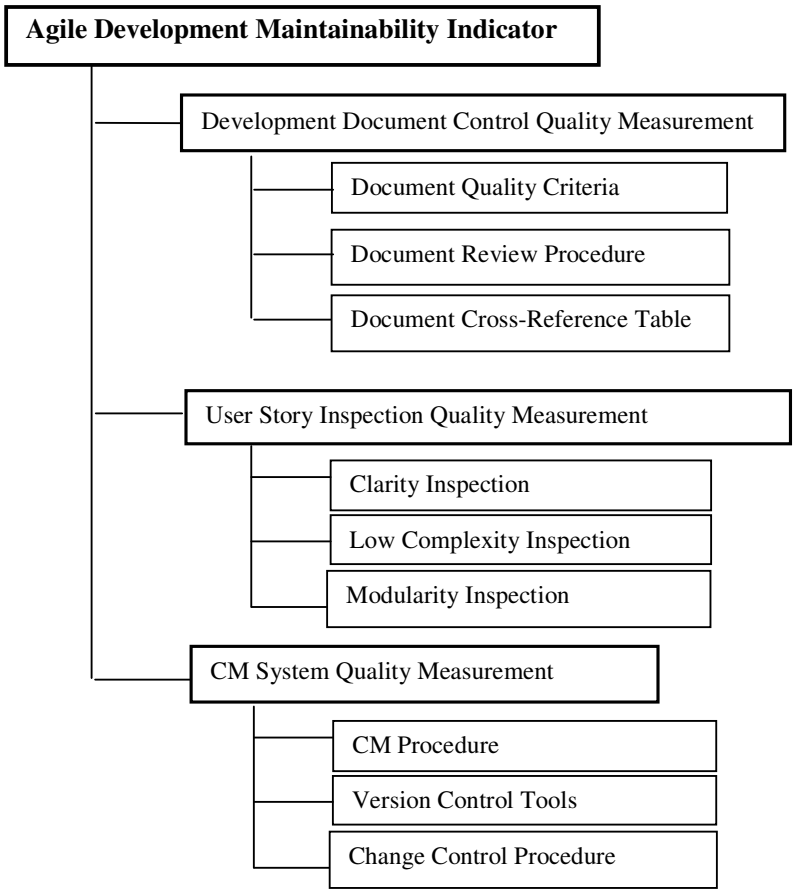

Figure 4. Architecture of ADMM model

5. AGILE DEVELOPMENT MAINTAINABILITY ENHANCEMENT PROCEDURE

For overcoming the requirements change, the agile development should strengthen three major shortages that include development documents control, user story inspection mechanism and CM system. In this paper, the ADMM model is proposed for enhancing the maintainability of agile development. Based on ADMM model, the Agile Development Maintainability Enhancement (ADME) procedure is defined, deployed and shown as Figure 5. ADME procedure develops six phases which are described as follows:

- Routine auditing phase: In order to understand and improve the maintenance process, agile development should plan the routine auditing operation. Routine auditing can help inspection maintenance activities and collect the influence factors.
- Factors collection phase: In auditing phase, critical maintenance quality factors should be checked, collected and normalized to assist identify the defects of maintenance activities.
- Quality measurement phase: Based on the normalized quality factors and ADMM model, the Agile Development Maintainability Indicator (ADMI) can be computed and generated for identifying the quality defects.
- Defects identification phase: According to ADMI and acceptable quality criteria, the major and minor defects of maintenance process can be identified.

- Revision phase: Analysing the reasons of identified defects, agile development should develop a revision plan to adjust the current quality activity or append the new quality activities for enhancing agile development maintainability.
- Continuous Improvement phase: Finally, based on the environment and application change, ADMM model should continuously be adjusted and modified for accomplishing the target of quality continuous improvement.

Software development uses the agile development methodology and applies the ADMM model enhancing its maintainability can concretely reduce software development risk. Agile development enhanced maintainability can accomplish three major missions, description as follows:

(1) Complete, correct and consistent development documents:

- Based on documents quality criteria, the development documents have a set of quality regulations.
- Based on documents review procedure, the defects of development documents can be identified and corrected.
- Based on documents cross-reference table, the cross-relationship of development documents have high interconnection to enhance documents consistency, completeness and maintainability.

(2) Clarity, low complexity and modularity user story:

- Based on clarity inspection, requirement items can increase the communication capability of requirements change.
- Based on complexity inspection, requirement items can increase the extension and adjustment capability of requirements change.
- Based on modularity inspection, in requirements change, affected requirement items and documents can be timely isolated.

(3) Manageable, controllable and traceable development documents:

- Based on CM procedure, the development documents and user stories can be identified and managed.
- Based on version control and tools, the development documents and user stories versions difference and revision contents can be preserved and controlled.
- Based on change control procedure, the change requests can careful control and management to reduce change risk of software development.

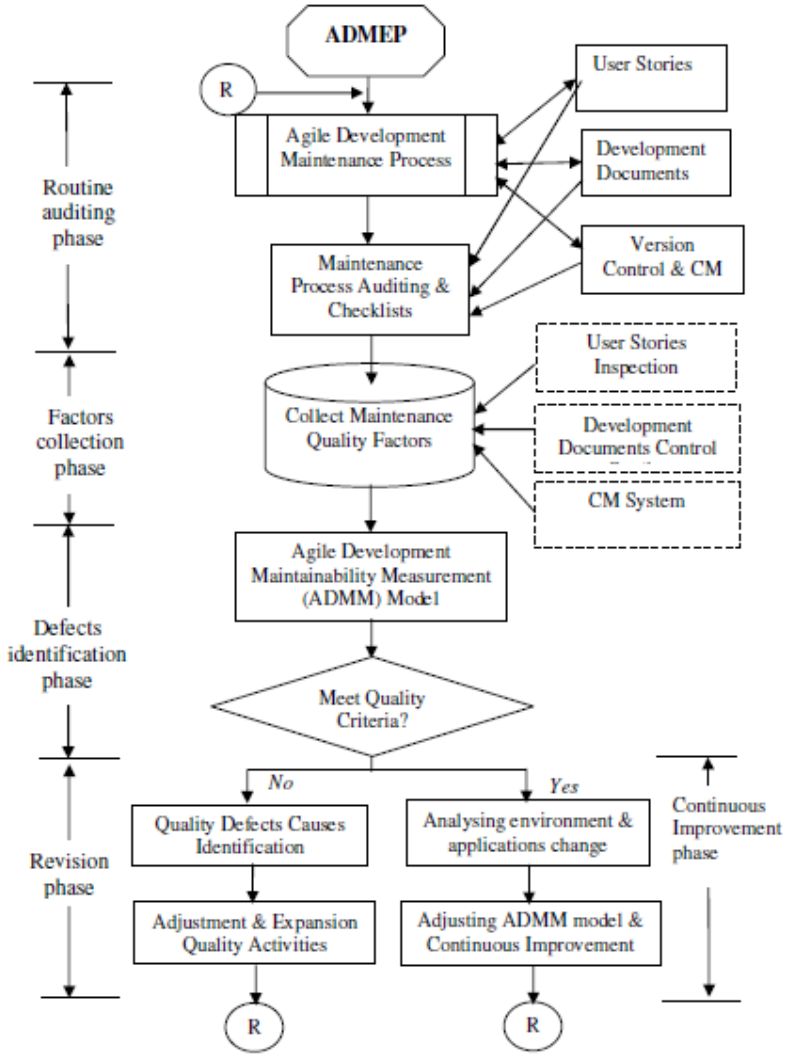

Figure 5. Flowchart of Agile Development Maintainability Enhancement (ADME) Procedure

6. CONCLUSIONS

Success software project must conquer requirements change challenges. Any requirements change always affects to software development operations and add software development failure ratio. Requirements change needs invest extra resource and cost, and cause project delay. Accepted requirements change, the affected development documents and tasks can't be effectively isolated will increase software development risk. Agile development applies IID model and very cares about the communication, working software, and refactoring features. IID and refactoring can reduce partial risk of requirements change. However, agile development existed some maintenance quality defects that include development documents quality, user story inspection mechanism and CM system. Agile development should enhance maintenance quality to reduce requirements change risk. For improving the maintenance quality of agile development, the paper proposes the Agile Development Maintainability Measurement (ADMM) model. Based on ADMM model, the Agile Development Maintainability Enhancement (ADME) procedure is defined and deployed for reducing the risk of requirements change. Agile development methodology enhancement maintainability can easy handle requirements change to reduce

software development risk. The advantages of ADMM model and ADME procedure are described as follows:

(1) Concretely identify agile development maintenance quality defects and provide correction suggestions.
(2) Effectively improve agile development maintenance quality can concretely handle requirements change and reducing development risk.
(3) Combination formulas of ADMM model have precise, simple and flexible adjustment features. Based on user or product features, development team can easy adjust or modify the weights of combination formulas.
(4) Combined with ADME model, ADME procedure has continuous improvement capability on the changeable applications environment, and keep up with the times.

REFERENCES

[1] Bohner, S.A. & Arnold, R. S., (1996) "Software Change Impact Analysis," IEEE Computer Society Press, CA, pp. 1-26.
[2] Bohner, S. A., (2002) "Software Change Impacts: An Evolving Perspective," Proc. of IEEE Intl Conf. on Software Maintenance, pp. 263-271.
[3] Boehm, B. W., (1991) "Software risk management: Principles and practices," IEEE Software, 8(1), pp. 32-41.
[4] Schach, S. R. (2011) Object-Oriented and Classical Software Engineering, McGraw-Hill Companies.
[5] Cockburn, A., 2002, Agile Software Development, Addison-Wesley.
[6] Cohn, M. & Ford D., (2003) "Introducing an Agile Process to an Organization," IEEE Computer, vol. 36 no. 6 pp. 74-78,
[7] Szalvay, V. (2004) An Introduction to Agile Software Development, Danube Technologies Inc., 2004.
[8] Larman, C. & Basili, V. R. (2003) "Iterative and Incremental Development: A Brief History," IEEE Computer.
[9] Larman, C., (2004) Agile and Iterative Development: A Manager's Guide. Boston: Addison Wesley.
[10] Eveleens J. L. & Verhoef, C. (2010) "The Rise and Fall of the Chaos Report Figures," IEEE Software, vol. 27, no. 1, pp. 30-36.
[11] The Standish group, "New Standish Group report shows more project failing and less successful projects," April 23, 2009. (http://www.standishgroup.com/newsroom/chaos_2009.php)
[12] Fairley, R. (1994) "Risk management for Software Projects," IEEE Software, vol. 11, no. 3, pp. 57-67.
[13] Pressman, "R. S. (2014) Software Engineering: A Practitioner's Approach, McGraw-Hill, New York.
[14] David S. Frankel, (2003), Model Driven Architecture: Applying MDA to Enterprise Computing John Wiley & Sons, January.
[15] Cohn, Mike, (2004) User Stories Applied: For Agile Software Development, Addison-Wesley Professional.
[16] McCabe, T. J. (1976), "A Complexity Measure," IEEE Trans. On Software Eng., Vol. 2, No 4, pp.308-320.
[17] Halstead, M. H. (1977) Elements of Software Science, North-Holland, New York.
[18] Fairley, R. (1985) "Software Engineering Concepts," McGraw-Hill, Inc.
[19] Conte, S. D., Dunsmore H. E. & Shen, V. Y. (1986) Software Engineering Metrics and Models, Benjamin/Cummings, Menlo Park.
[20] Fenton, N. E. (1991) Software Metrics - A Rigorous Approach, Chapman & Hall.
[21] Lai S. T. & Yang, C. C. (1998), "A Software Metric Combination Model for Software Reuse," Proceedings of 1998 Asia-Pacific Software Engineering Conference, Taiwan, Taipei, pp. 70-77, 1998.
[22] Boehm B.W. (1991) Software Engineering Economics, Prentice-Hall, New Jersey.

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.