A PROCESS QUALITY IMPROVEMENT MECHANISM FOR REDUCING THE RISK OF CI ENVIRONMENT

Sen-Tarng Lai

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

ABSTRACT

In the age of fast evolution, software development project must accept many challenges of unpredicted requirements change and new technology environment. Software development processes should have adjustable and extendable features to meet the multifaceted needs of the users. Iterative and Incremental Development (IID) is a practical approach to overcome the various challenges of software development. However, continuous testing and building new versions need to spend more time and human resources that is a major obstacle of IID. The other, the iterative operations must have a sound communication skills. Lack of standard version control and intercommunication manner often lead to failure of software project. High quality Continuous Integration (CI) environment can effectively make up the defects of IID. In this paper, CI environment and advantages are deeply surveyed. In order to overcome the defects of IID, CI environment needs combine the perfect procedures and qualified tools, and concretely enhance the quality of CI environment. Based on the process quality measurement model, this paper proposes the Process Quality Improvement Mechanism (PQIM). Applying PQIM, in software development, the processes problems and the CI environment quality defects can identify timely and indeed revise to reduce the risk of CI environment.

KEYWORDS

Continuous Integration, IID, development risk, CI environment, quality improvement

1. INTRODUCTION

According to the Standish Group software project survey report in 2009, only 32% belong to the successful software projects that budget and requirement quality almost fully met the project planning. In development process, 24% projects were cancelled and the rest 44% owing to development time delay, over budget and quality unsatisfied the requirement situations were regarded as the failure projects [1]. In 2014, the latest research report of the Standish Group further pointed out that 31.1% of software projects are cancelled, the development cost of 52.7% cancelled projects exceed 189% of the original budget. Investigating the major cause of software project failure is over plan change frequency. In software development process, many unexpected events will occur successively to impact the software development activities. Forcing the software project plan must continue to change. Project plan modification often is not able to effectively and timely reflect to the important sub plans that include budget planning, development time, quality management and resource allocation. Therefore, project plan changes make expansive gap between plan items and the software development work, gradually leading to the project plan completely unable to control the budget execution, development time, quality management and resource allocation. Finally, software project becomes a failure project of budget overruns, schedule delays and poor quality [2, 3]. Therefore, in order to handle high changeable software project, software development process must have high adjustability and changeability to reduce the function extension and requirement change risks.

DOI:10.5121/ijcsit.2017.93010

Internet widespread age, the pursuit of a variety of high efficiency and high effective trading activity must be combined with the network technologies. In order to overcome the ever changing hardware facilities, advanced IT technology, services requirement change and continuous expansion functions, software projects should cautiously select the appropriate development model. Agile development model uses IID (Iterative and Incremental Development) process, each repeated development work required to complete a workable version, immediately test and evaluate the new version, early detection the processes problems and product defects, assist timely improvement measures, reduce the software development risk [3, 4, 5]. IID mode gradually by the software engineer attention, many of the software development models have combined with IID. Among which the continuous integration (CI) with agile development model is cited and discussed in [6, 7, 8]. CI is a software implementation concept that was proposed in 1998. CI have combined testing tools, integration tools, software version management system and other aspects of technology and become an important member of the agile development method. The advantages of CI can identify software development problems and product quality defects in software initial development stage to reduce the software development risk [3, 6, 7, 8, 9].

Rapidly changing technology and environment makes software project must have the flexible and changeable features. Therefore, software development process is very suitable to use IID methodology and apply CI practical concept. In early development phase, coding errors and quality defects can be quickly identified, corrected and revised that make the risk of software project can be greatly reduced. However, version integration test need expend more manpower and resource that is major obstacle of IID. In addition, IID must have well intercommunication capability. Lacking formal version control system and stakeholder communication channel often led the software project failure. High quality Continuous Integration (CI) environment can effectively make up the defects of IID. In this paper, CI environment and advantages are deeply surveyed. In order to overcome the defects of IID, CI environment needs combine the perfect procedures and qualified tools, and concretely enhance the quality of CI environment. Based on the process quality measurement model, this paper proposes the Process Quality Improvement Mechanism (PQIM). Applying PQIM, in software development, the processes problems and the CI environment quality defects can be identified timely and indeed revised to reduce the risk of CI environment. In section II, software project failure factors and software development risks management are discussed. In section III, advantages of IID method and major features of CI are investigated. In section IV, based on the process quality measurement model, this section proposes the Process Quality Improvement Mechanism (PQIM). In Section V, plans the CI-based IID process and evaluates the benefits of CI-IID. In Section VI, emphasizes the importance of CI environment processes quality and IID to reduce the software development risk and does the paper conclusion.

2. SOFTWARE DEVELOPMENT RISKS

In this section, software project failure factors and software development risks management are discussed.

2.1. MAJOR FAILURE FACTORS OF SOFTWARE PROJECT

According to the Standish Group survey report of large scale software project, software project success ratio is under 30%. And 80% of failure projects are time delay, over budget and quality not satisfied the requirements [1]. Many papers have discussed the causes of software project failure [10, 11, 12]. Development time delay, over budget and quality unsatisfied the requirements etc. three situations always cause to software project with high development risk. The reasons of the high risk is high relationship with the following four key events:

- In beginning of development, system and requirement analyst collect and analysis incomplete data to cause system requirement specifications and user requests existed incorrect, incomplete and inconsistent defects. The defects can't be revised immediately. It must cause development time delay, over budget and product quality unsatisfied the requirements.
- In software development process, new technology or development environment evolution makes the project plan must to be adjusted to match new technology and development environment. Before planning items unable be re-vised immediately, the follow development operations will be affected.
- In initial development stage, user submits many kinds of requirements. These requirements often are requested to expand, modify, or remove in software development process. These requirements changes become the major impaction for the follow phase development.
- Each software development phase need involve difference resource that include developers, computer hardware, software tools, and development environment. Resource allocation defects may cause the project plan resource management need to readjustment and allocation. Imperfect and unchangeable resource management often makes the high development risk of software project.

Four key events of software project risk can summarize as incomplete requirement analysis, evolution technology and environment, user requested requirements change, and changes of resources allocation. In software development process, these high risk events almost cannot be avoided or excluded. In order to effectively reduce the risk of software development project. Only way is to overcome the challenges of these risk events.

2.2. SOFTWARE DEVELOPMENT RISKS MANAGEMENT

In order to reduce project failure risk, software development process must have high flexibility and changeability. According to the situations encountered, development operations should timely adjust and effectively change, until software projects can overcome existed risk. Four software project change factors and the affected project planning and response measures are discussed as follows:

- Incomplete requirements analysis: generally occurs that software requirements are not clear or missing. For this, the project plan need to give the additional cost and resource, delay the development time, and adjust the requirement items and quality assurance mechanism. In order to reduce the impact of incomplete requirements analysis of software project, firstly, user can pick and choose the major and clear requirement and postpone the unclear or ambiguous requirements.
- Evolution technology and environment: introducing new development technologies or new operating environment need to take an additional acquisition cost and staff training costs. For adjusting the software project plan, it may affect the cost estimation, schedule and resource management, and increase development risks. Incremental development approach can gradually introduce new technologies and the environment to avoid the extra cost, schedule and resource impaction and reduce the software project risks.
- User requirements change: in software development process, user requirement changes are the unavoidable challenge. However, the unexpected requirement changes is bound to affect the project plans and the completed phase documents. Frequent user requirements change not only may impact the development cost, schedule, product quality and other planning projects, but also cause the software project failure risks. In order to reduce the impact of requirement changes, iterative development can be used to gradually adjust for the requirement changes.

- Resources allocation changes: personnel high mobility, organizational downsizing, development equipment late delivery, major equipment repairing or updating are bound to affect the software development process and schedule. In order to avoid the impacts of development schedule, the part of requirements should consider to shift to the next version.

Based on the above four items, IID is an appropriate development method to reduce the software development risk.

3. AGILE DEVELOPMENT AND CONTINUOUS INTEGRATION

3.1 AGILE SOFTWARE DEVELOPMENT AND IID

In recent twenty years, the growth of information technology, operational environment and user requirements, software development methodologies continuously progress. Most of software development models have high relationship with user requirements. Waterfall 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. The requirement items are allowed to propose by incremental manner, not necessary to decision at same time. Using iterative development model, user can incrementally provide the requirement items. Software development risks can be greatly reduced. Each development model should has the adjustment strategy to handle the change of user requirements. The adjustment strategy can't effectively apply to reduce software development risks, then may impact success ratio of software project.

In 2001, seventeen software developers met at the Utah of USA 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 [5, 13, 14]. Agile software development proposes several critical viewpoints:

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

Agile software development neglects the formal analysis and design phases, uses non-document oriented development, and doesn’t pay attention to follow-up maintenance operations that are major drawbacks (shown as Figure 1). For effectively reducing the software development risks, the drawbacks of agile model should be concretely improved or enhanced. However, agile software development uses IID to reduce the requirement complexity, refactoring to increase the requirement modified flexibility, and non-document oriented can reduce the cost of requirements change [15]. Requested software project must release a workable version during two or three weeks. Letting the client can detailed test, validate requirement items of new version and clearly understand the development progress. In each day, a fifteen minutes stand up meeting is held to effectively reach full intercommunication between client and developers. In development change impacts, IID greatly decreases schedule delay, cost over budget and quality unsatisfied user requirement situations, software development change risk can be effectively reduced. However, version integration test need expend more manpower and resource that is major drawback of IID.

3.2 ADVANTAGES OF CONTINUOUS INTEGRATION

CI is a software engineering practice that the functions of add or change are immediately tested and verified before committing to a source repository. The goal of CI is to quickly identify the coding errors and quality defects, and to make the errors and defects can be corrected and revised as soon as possible. In 1994, Grady Booch mentioned the CI phase in his book of Object-Oriented Analysis and Design with Applications [16]. In 1997, Kent Beck and Ron Jeffries proposed XP which included CI concept [6], [17]. In 1998, Kent published a paper of CI which valued and supported the importance of directly communication technology [18]. Based on IID mechanism, agile development combines XP, TDD (Test Driven Development) and BDD (Behaviours Driven Development) methods, and makes CI to become a main stream and important trend of software development. Focus on the automatic and repeat process are the major characteristics of CI. Therefore, integrated with related automated development tools is first mission of CI environment.

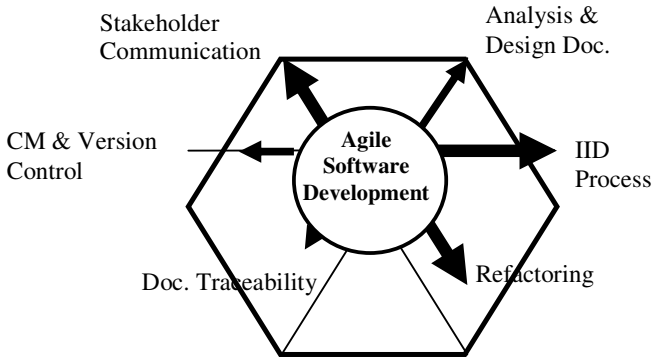

Figure 1. A schematic diagram of agile process advantages and drawbacks

In software development process, software project management applies modularization to enable teamwork development and make complex system manageable. Modularization decomposes the complex software system into many manageable program units. Each program unit just handle a specific function, therefore the program units have to work together and integrate into a module function, subsystem and system. CI through more times integration and testing makes the errors and defects can be identified quickly. Errors correction and defects revision can be timely handled and effectively avoid to errors and defects to transfer to the follow phases. Summarize five major advantages of CI and discuss as follows (shown as Figure 2):

High efficiency:CI combines many automated tools and procedures can increase the development and operation efficiency.
Reduce development risks: According to development requests, CI can more times integrate in a day to reduce software project development risks.
Smooth communication: CM system [19], source repository and workable software can assist build common understanding among stakeholder.
Improvement quality: CI follows the systematic procedures and combines useful automated tools to improve software development quality.
Increased morale: CI increases communication frequency, dramatically reducing communication barriers and software development risks. The development team confidence can be established and morale can be increased.

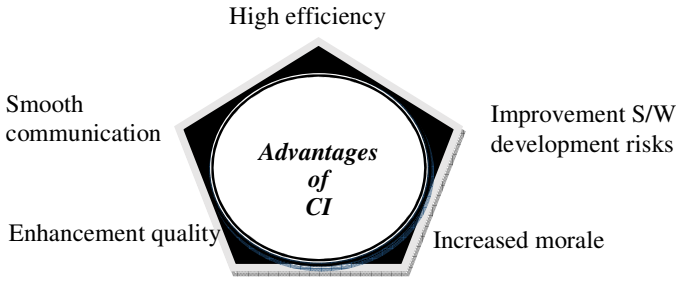

Figure 2. Five major advantages of CI

4. QUALITY MEASUREMENT MODEL

4.1 FIVE PROCESSES OF CI ENVIRONMENT

CI pays attention to increase software development productivity and quality, concretely reduce development risk and continuously monitor coding errors and integration defects. CI environment must suitably combine qualified CASE tools and perfect control procedures to reduce human resources expense and manual operation defects. This section applies the CM environment to IID software development project. CI environment is divided into five process items to describe as follows:

(1) CM Process: Development documents and source code, passed review activity, must have a storage and management mechanism. Based on the Work Breakdown Structure, CM system and version control tools can effectively manage and control the documents and source code check-in and check-out. In incremental development, CM system is a necessary process which can assist to record and control continuous development process problems and defects. Version control tools can detailed record documents and source code modification time, personnel, causes and difference between the versions.
(2) Unit Testing Process: Unit program is the primitive element of software system. Complete and correct unit testing can increase the follow development efficiency and quality. Unit testing conforms the test criteria and passes the review activity, the source code and test cases can commit for the CM system. In order to increase the efficiency and quality of unit testing, the norm of program language related unit testing tool should be defined to assist unit testing automation.
(3) Integrated Testing Process: Test phase should cover function, integration and system testing steps. However, before testing, each test steps must complete test cases design and planning. Selecting the suitable test approaches (ex. Equivalence partition, cause effect graph, symbolic execution and mutation) and assistant tools can quickly generate high efficiency and quality test cases. With suitable test approach, integration interface errors and design defects can be identified in time.
(4) Build and acceptable testing Process: Modularization design decomposes the system into several program units. Using make and make-file tools can handle tens or hundreds of unit programs and modules to clearly and quickly define the relationships between unit programs and modules. Based on the produced or modified time of files, the make and make-file tools can judge the source code whether recompile or relink to be a new image file. Effectively reduce human possible misses.
(5) Deployment and validation process: passed acceptance testing activity, According to the user requirements and assigned environment, new version software must execute deployment and

validation activities. Using batch process command on specific operation platform can quickly accomplish automatic system deployment. Finally, the deployed system should execute verification and validation activities by the stakeholder.

CI environment combining five process items includes CM and VC system, automatic unit testing, repository management, automatic integration testing, automatic deployment, and errors and defects identification etc. five major elements (shown as Figure 3).

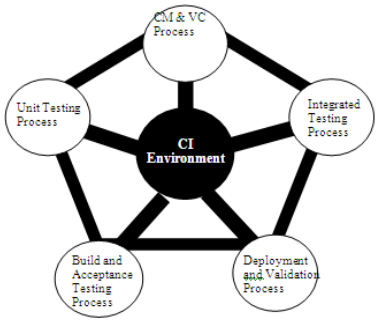

Figure 3. CI environment and five development processes

4.2 QUALITY FACTORS OF PROCESSES

CI environment quality is important critical factor to affect IID software project success. In order to ensure high quality of CI environment, in this section use check lists to collect quality factors of CI environment. The meta-process of CI environment should measure perfect procedure, qualified CASE tools and process interoperability three quality metrics to assure CI environment quality. Five processes (shown as Figure 4) are instanced from meta-process and described as follows:

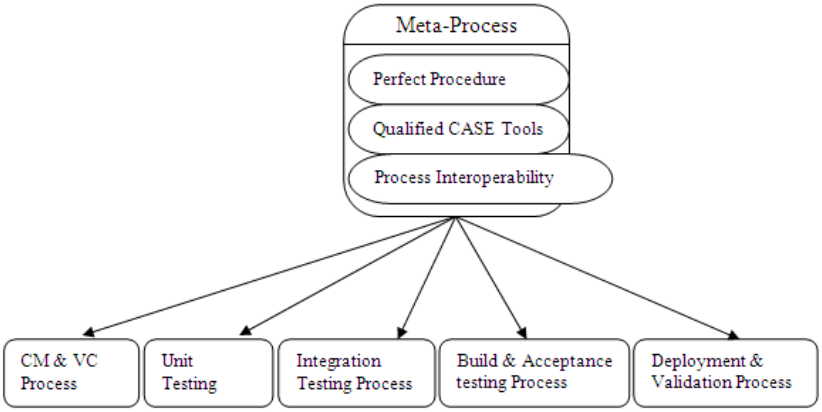

Figure 4. Meta-process and five instanced processes

(1) CM and VC Process (CMP) quality factors:

- Does CMP have the complete and correct control procedure to manage and control development documents?
- Does CMP use the suitable CM assistant tools and integrated with version control system?
- Does CMP consider the interoperability between the other processes?

(2) Unit Testing Process (UTP) quality factors:
- Does UTP have the complete and correct unit testing operations procedure?
- Does UTP use the suitable automated unit testing assistant tools?
- Does UTP consider the interoperability between the other processes?

(3) Integration Testing Process (ITP) quality factors:
- Does ITP have the complete and correct integration testing operations procedure?
- Does ITP use the suitable automated integration testing assistant tools?
- Does ITP consider the interoperability between the other processes?

(4) Build & Acceptance testing Process (BAP) quality factors:
- Does BAP have the complete and correct operations procedure of build and acceptance testing?
- Does BAP use the suitable automated build assistant tools?
- Does BAP consider the interoperability between the other processes?

(5) Deployment & Validation Process (DVP) quality factors:
- Does DVP have the complete and correct operations procedure of deployment and validation?
- Does DVP use the suitable automated deployment and validation assistant tools?
- Does DVP consider the interoperability between and other processes?

4.3 PROCESS QUALITY MEASUREMENT MODEL

For collecting useful quality factors, it is necessary to develop a set of complete and clear inspection checklists to conduct the CI environment items audit. Checklists can collect the quantified quality of CI environment items. Checklists discussion and implementation can help collect the completeness, correctness, and usability and interaction degree quality factors of CI environment. However, individual factor or measurement can only measure or evaluate the specific quality characteristic. In order to effectively monitor and assess the CI environment quality defects, individual measurements should make the appropriate combination [20], [21], [22]. Two kind of metric combination models are Linear Combination Model (LCM) [20], [21], and Non-Linear Combination Model (NLCM) [21], [22]. 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 CI environment 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.

CI environment should consider three quantified quality that include basic, usability and interaction degree quality. Using the LCM, related quality factors can be combined into the primitive metric, and then the related primitive metrics can be combined into the quality measurements. Finally, combines with three critical quality measurements and generates a CI environment quality indicator. The formulas described as follows:

Each one Process Quality Measurement need combine the operation procedure quality, the tools applicability and the processes interoperability three metrics. The combination formula is shown as Equation (1):

PQM: Process Quality Measurement
POP: Process Operation Procedures Metric *W1: Weight of CMPQ*
PTA: Process Tools Applicability Metric *W2: Weight of CMTU*
PI: Processes Interoperability Metric *W3: Weight of CMPI*

$$PQM = W1 * POP + W2 * PTA + W3 * PI \qquad W1 + W2 + W3 = 1 \qquad (1)$$

Based Equation (1), five process quality measurements (CMP, UTP, ITP, BAP and DVP) can be sequentially generated. Finally, combining CMP, UTP, ITP, BAP and DVP quality measurements into a CI Environment Quality Indictor (CIEQI). The combination formula is shown as Equation (2):

CIEQI: CI Environment Quality Indictor
CMPQM: CMP Quality Measurement *Wcm: Weight of CMPQM*
UTPQM: UTP Quality Measurement *Wut: Weight of UTPQM*
ITPQM: ITP Quality Measurement *Wit: Weight of ITPQM*
BAPQM: BAP Quality Measurement *Wba: Weight of BAPQM*
DVPQM: DVP Quality Measurement *Wdv: Weight of DVPQM*

$$CIEQI = Wcm * CMPQM + Wut * UTPQM + Wit * ITPQM + Wba * BAPQM + Wdv * DVPQM$$
$$Wcm + Wut + Wit + Wba + Wdv = 1 \qquad (2)$$

IID process of agile development model, CI environment quality is the major cause to effect software project development results. For measuring and improving the quality of CI environment, the major quality factors of each CI sub-processes were collected and divided into 3 groups. In first layer, three groups basic quality factor are combined into three quality metrics. In second layer, three quality metrics are combined into a process quality measurements. In third layer, five process quality measurements are combined into a CI Environment Quality Indictor (CIEQI). With three layer combination formulas to generate a CIEQI, the process is called a Process Quality Measurement (PQM) model. The PQM model architecture is shown as Figure 5.

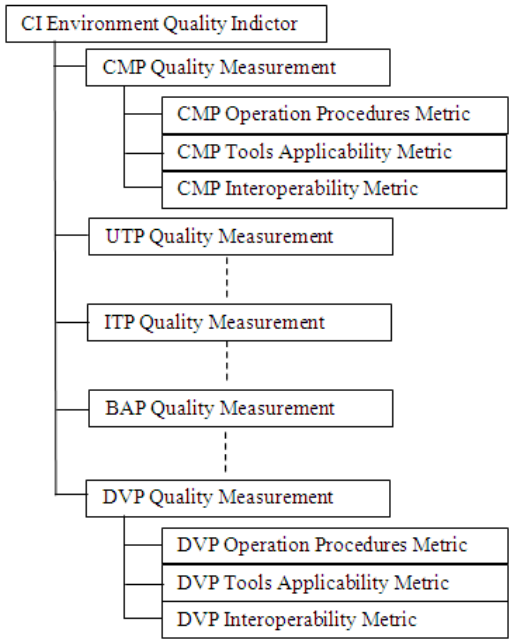

Figure 5. The architecture of process quality measurement model

CIEQI is a relative judgment mechanism and the basis to identify quality defects of CI environment. In CIEQM model, basic quality factors are combined into high layer quality measurements. High layer quality measurements are combined into a CIEQI. Therefore, if the CIEQI does not satisfy quality criterion, it represents CI environment existed some process quality defects. According to the combination formulas, the process tasks problems and quality defects can be timely identified. For improving CI environment quality, the related activities of poor quality should be rigorously inspected to identify the problem or defect and propose the corrective action.

5. CI-BASED IID PROCESS AND BENEFIT EVALUATION

5.1 IID OPERATION PROCEDURE WITH CI ENVIRONMENT

Iterative processes of IID still use the waterfall development model. Therefore, IID should refer to the major development phases of waterfall model. However, in order to combine CI environment with IID, IID not only must deploy high quality CI environment but also need combine with CM system and CASE tools to enhance IID phase activities quality. Based on the PQIM, the paper defines and plans the CI-IID operation procedure. CI-IID operation procedure divided into six phases (shown as Figure 6). The completed documents of each phase need pass quality assurance activities and conform to the CM and version control system and submit the documents to the repository, to build the documents traceability. The detailed phase activity are describes as follows:

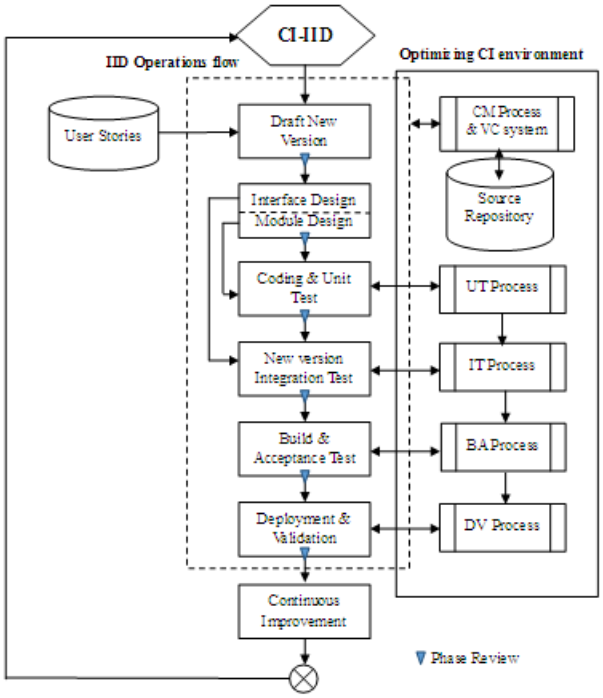

Figure 6. CI-IID operation procedure

(1) Requirement Analysis phase: Analysing user stories and generating a next version requirement, and based on the version requirement accomplish requirement specification assurance.
(2) Interface and detailed design phase: this phase decomposes into two sub-phases as follows:

- Integrated interface design: based on the repository requirement specification, design the integrated interface and pass review activities.
- Program unit design: based on the repository interface design and program unit specification, develop unit program test cases and module detailed design and pass review activities.

(3) Coding & unit testing: based on test cases and module detailed design, implement the module source code and pass automatic module unit testing. Passed unit testing, the test cases and source code should submit to the repository.
(4) Integrated testing: based on integrated interface design and test cases, handle automatic integrated test and pass integrated testing criteria.
(5) Build and acceptance testing: based on acceptance criteria, launch automatic system build and acceptance testing and pass the acceptance testing review activities.
(6) Deployment and validation: more times continuous integration, the development focuses on system deployment. Therefore, the user should be the important attendees of system verification and validation. User rapidly contacts new function and new version, progress of project and current situation can be fully controlled and understand.
(7) Continuous improvement: IID accomplished the new version requirement of user, and passed to system verification and validation activities. In this time, IID is necessary to work for next incremental development and more frequently. In IID process, for reducing the CI-IID operation problems and defects, data collection, problems identification and defects assertion must be persisted to revise the problems and defects of CI-IID operation procedure continuously.

5.2 CI-BASED IID BENEFIT EVALUATION

Software development model has tight relationship with user requirements. Waterfall development model clearly defines the phase mission and pays attention to the quality of phase documents. Phase documents cannot satisfy correctness, completeness and consistency, then process activities should be adjusted, phase documents need be revised and development flow cannot allow to enter next phase. In recently, the development model substantially adjusted the requirements generation methods. Based on IID, user requirements are unnecessary to confirm at one time, but can gradually submit new requirements. Agile model uses IID methodology. According to the environment and requirement change, user can dynamic submit new requirements or revise the requirement service items to effectively reduce the software development risk. Introducing into a new development model should collocate the suitable procedure. Lacking a suitable operation procedure often hard to effectively control process efficiency and quality. About process problems and quality defects cannot timely take adjustment operations and remedial measures. In this paper, applying CIEQM model optimize CI environment quality, and based on CI environment drafts IID operation procedure to norm the software development operations. Process efficiency and product quality can concretely control, and the operation procedure can be adjusted by continuous improvement phase. Making CI-based IID operation procedure has continuous improvement feature. Five advantages of CI-based IID operation procedure are evaluated as follows (shown as Table I):

(1) Process problems and quality defects identification: Automation tools and norm procedures are applied on each phase of CI-based IID. CI with continuous testing and integration makes process problems and quality defects to be identified in time and to be revised quickly. Process problems and quality defects can be avoided to affect and extend to the follow development phases.

(2) Intercommunication efficiency and developer morale: In agile software development, IID cares user involvement, sharable development documents and source code, frequently meeting, and using CM and version control procedure. IID and quality activities concretely increase the intercommunication efficiency and developer morale.

(3) Automation environment: In order to increase integration frequency, CI environment uses automatic tools to reduce human resource involvement. Automated unit test, functional module, subsystem and system quick automatic integrated, and assistance deployment and validation makes the development efficiency and system quality can be enhanced.

(4) Integration frequency: Traditional development is difficult to handle continuous integration. It is because of traditional development executes integration task only in the testing phase. It is too late to identify the process problems and quality defects and cannot be early identified and revised. Software project of traditional development existed high risk.

(5) Development quality and productivity: CI makes process problems and quality defects to be identified in time and to be revised quickly. Avoid problems and defects propagation and expansion that can effectively.

Table 1. Comparison between traditional and CI-based IID model

Development Models / Features	Traditional development model	CI-based IID model
Problems and defects identification	in development late stage	in development initial stage
Intercommunication efficiency and developer morale	Weak	strong
Automation environment	middle	strong
Integration frequency	less	more
Development quality and productivity	Weak	strong

6. CONCLUSION

In the information network age, everything requests the achievement of fast. Software project must overcome the changing requirements and to withstand environmental challenges that continued to introduce new technology. Therefore, development process should have the modification and adjustment ability to meet the diversified needs of the users. However, software project failure rate as high as over 60%, how to effectively reduce the risk of software development project, is a topic worthy of further exploration. In order to effectively reduce the requirement change to cause software development risks, IID has recognized as the very important and practical development methods in current stage. Many software development methods and models have been taken IID to reduce the risk of requirement changes.

The iterative integration testing can effectively identify the process problems and products quality defects. However, it need consume more time and human resources which is a critical challenge

of IID. Applying high quality CI environment can effectively overcome the challenge of IID. In this paper, the operation environment and advantages of CI are discussed. Analysis CI environment should have key processes and quality characteristics, high quality CI environment to make up the IID defects. Based on process quality measurement model, this paper proposes the Process Quality Improvement Mechanism (PQIM). Applying PQIM, in software development, the processes problems and the CI environment quality defects can be identified timely and indeed revised to reduce the risk of CI environment. Applying the high quality CI environment develops the CI-based IID operation procedure to assist automated unit and integration testing in software development. New version software can be effectively deployed, stakeholder intercommunication channels increase, and staff morale enhancement. CI-based IID software developers can achieve four results:

- Users can quickly use the new or updated workable version.
- Improving the major drawback of IID that version integration test need expend more manpower and resource.
- The process problems and software quality defects can be timely identified and quickly use the suitable improvement measures.
- In the frequent changes, effectively reduce 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] Eveleens J. L. and Verhoef, C. (2010) "The Rise and Fall of the Chaos Report Figures, IEEE Software," vol. 27, no. 1, pp30-36
[2] Boehm, B. W. (1991) "Software risk management: Principles and practices," IEEE Software, vol. 8, no. 1, pp32-41.
[3] Schach S. R. (2011) Object-Oriented and Classical Software Engineering, Eighth Edition, McGraw-Hill, New York.
[4] Larman C. and Basili, V. R. (2004) Iterative and Incremental Development: A Brief History, Computer, IEEE CS Press, pp47-56.
[5] Martin, R. C. (2002) Agile Software Development, Principles, Practices and Patterns, Prentice Hall,
[6] Martin Fowler, (2006) "Continuous Integration," martinfowler.com, http://www.martinfowler.com/articles/continuousIntegration.html
[7] Duvall, P.: Continuous Integration Servers and Tools, DZone Refcardz. (2015) https://dzone.com/refcardz/continuous-integration-servers#, (accessed Auguest 9, 2015)
[8] Duvall, P. and Matyas, S. and Glover, A. (2007) Continuous Integration: Improving Software Quality and Reducing Risk, Pearson Education, Inc.
[9] Saff, D. and Erns,M. D. (2003) Reducing Wasted Development Time via Continuous Testing, Proceeding of IEEE International Symposium on Software Reliability Engineering (ISSRE), pp281-292
[10] Hornstein, H. A. (2015) "The integration of project management and organizational change management is now a necessity," International Journal of Project Management, vol. 33, no. 2, pp291-298.
[11] STEWART J. (2015) Top 10 Reasons Why Projects Fail, http://project-management.com/top-10-reasons-why-projects-fail/.
[12] Symonds M. (2011) 15 CAUSES OF PROJECT FAILURE, https://www.projectsmart.co.uk/15-causes-of-project-failure.php.
[13] Szalvay, V. (2004) An Introduction to Agile Software Development," CollabNet, Inc.
[14] Hoda, Rashina, et al. (2017) "Systematic literature reviews in agile software development: A tertiary study," Information and Software Technology 85 pp60-70.

[15] Santos, M. A., et al. (2013) "Improving the Management of Cost and Scope in Software Projects Using Agile Practices," International Journal of Computer Science & Information Technology (IJCSIT) 5(1).
[16] Booch, G. (1994) Object-Oriented Analysis and Design with applications 2nd edition, Addison Wesley Longman.
[17] Beck, K. (2003) Test-Driven Development: By Example, Addison-Wesley.
[18] Beck, K. (2004) Extreme Programming Explained: Embrace Change, 2nd Edition (The XP Series) 2nd Edition
[19] Leon, A. (2015) Software Configuration Management Handbook, Third Edition, Artech House Inc.
[20] Fenton, N. E. (1991) Software Metrics - A Rigorous Approach, Chapman & Hall.
[21] Galin, D. (2004) Software Quality Assurance – From theory to implementation, Pearson Education Limited, England .
[22] Boehm, B.W. (1981) 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.