分布系统软件开发技术文集

SDM-DS 2003年会议录

SDM-DS 1

Documents of SDM-DS 2003

Proceedings of the First International Workshop on
Software Development Methodologies for Distributed Systems

Edited by
Zhou Zhiying 周之英
Scott Tilley （加）斯高德

清华大学出版社
Tsinghua University Press

Series : SDM-DS 1

Documents of SDM-DS 2003

Proceedings of the First International Workshop
on Software Development Methodologies for Distributed Systems

分布系统软件开发技术文集

SDM-DS 2003年会议录

Edited by
Zhou Zhiying 周之英
(加) Scott Tilley 斯高德

清华大学出版社
Tsinghua University Press
北京

Series SDM-DS 1(2005-1)
ISSN 1673-1239
ISBN 7-302-11186-3

图书在版编目(CIP)数据

分布系统软件开发技术文集＝Documents of SDM-DS 2003/周之英，(加)斯高德(Scott Tilley)编. —北京：清华大学出版社，2005.6
ISBN 7-302-11186-3

Ⅰ.分…　Ⅱ.①周…②斯…　Ⅲ.方法论－分布－系统－软件开发－国际会议－英文　Ⅳ.TP31

中国版本图书馆 CIP 数据核字(2005)第 006255 号

Editorial organization: The Editorial Committee of Series SDM-DS
Chief Editor: Zhou Zhiying
Sponsored by: Tsinghua University
Visit our Web page on http://www.tsinghua.edu.cn/docsn/wb/interconf/softwaredev.htm
Published by: Tsinghua University Press

出版者：清华大学出版社　　地址：北京清华大学学研大厦
http://www.tup.com.cn　　邮编：100084
社总机：8610-62770175　　客户服务：8610-62776969
责任编辑：张　龙　张兆琪　　封面设计：常雪影　周之英
印刷者：北京市世界知识印刷厂　　装订者：三河市金元印装有限公司
发行者：新华书店北京发行所；北京市邮局；中国图书进出口总公司
开　本：185×260　印张：11.75　彩插：4　字数：342 千字
版　次：2005 年 6 月第 1 版　2005 年 6 月第 1 次印刷
印　数：1～1000
定价 Price：36.00 元(In China)/35 USD(Other Regions, Including customs duty)

序

分布系统的软件开发技术

《分布系统软件开发技术文集》收集了在阿姆斯特丹和北京召开的专业国际研讨会SDM-DS 2003论文(涉及基础概念,体系结构,部件和Web服务技术,测试和验证,实验性软件工程等)和以评审过程内容为主的文件,以便读者深入了解IT前沿的一些研究问题和国际交流状况。

随着经济全球化和网络计算发展,分布系统的应用越来越广泛。计算机网络支持的软件系统(实质就是分布系统)成为应用的主流;新增应用领域日新月异。无论是人们的日常生活,还是工业、商业、交通运输业、医疗卫生行业、教育科技、政治、外交、国防等,和分布系统已息息相关。若考虑可预见未来的分布系统,视频和无线通信支持下的沉浸式计算,智能微粒和微型传感器所组成的看不见的网络,几乎可造成无处不在的系统。分布系统有别于独立系统的特点十分显著:复杂性极大,模糊的边界,不确定因素的影响……目前大部分工作,仍沿用基于传统概念的独立系统的开发方法和研究方法。实际分布系统软件开发中,开发难度极大,不易控制和管理开发队伍,难于控制质量和工期。各种新老技术往往仅有非常有限的使用范围,推广难度也大。甚至,有些刚提出的新技术也会迅速受到客观事实的质疑。分布系统软件开发技术具有极大挑战性。

清华大学首先发起的分布系统的软件开发技术国际研讨会,作为2003年9月在荷兰阿姆斯特丹举行的软件技术和工程实践国际研讨会STEP 2003的一部分,针对当前IT研究发展的热点和重点,开展国际交流。SDM-DS主题是分布计算时代的软件开发技术,和建立理论与实践间合作发展。在阿姆斯特丹举行的STEP 2003约有100多人参加;但大部分中国代表没有参会。SDM-DS 2003的两次研讨会,共接受论文28篇,约50多人参与交流。北京会议安排了论文宣读、提问回答、小组活动、头脑风暴讨论、软件公司参观活动、实验室(清华大学、北京大学和中科院软件所)学术演示交流和校园参观活动、中关村软件园区的参观访问和中国文化欣赏等。

SDM-DS 2003的学术论文交流丰富多彩。英国专家介绍了在开发大型软件系统方面的实际经验和困难。加拿大专家总结了软件开发技术的发展。美国德克萨斯大学学者交流了在研究设计模式集成方面的成果。此外,在数据文档整理的方式和意义方面,交流的研究工作也很有启发。显然,在分布系统中,数据作用长期存在,不同场合下的不同可能表示,将使这方面研究成为一项重大任务。清华大学学者指出分布系统软件开发的复杂性和不确定性;提出以"完美球"取代开发中目标点的概念(基于分布系统中存在的不确定性)和更新传统测试概念的测试驱动方法,对传统的基础概念产生一定冲击。美国佛罗里达技术研究院、加州大学、北京航空航天大学、南开大学等在实验软件工程方面进行实验数据收集、分析,进而发现问题;虽然还欠成熟,但代表了当前在软件工程研究领域的一种新倾向。许多主流技术继续发挥作用:无论工业界,还是研究单位,均乐于接受UML进行软件开发设计;以软件服务方式提供功能;对性能等问题进行数据分析的处理模式。由于分布系统的复杂性,测试主题也是重要关注点。

常见的国际会议，往往集中关注论文的正规宣读，中国科技人员参与实质交流较少。SDM-DS 2003 吸取 STEP 的优点，不过分注重形式。来自学校、研究单位，软件公司的参会科技专家，工作的专业领域或关注点不一定相同；有些人虽在相近领域工作，却因研究需要和习惯不同，存在巨大认识差异。工业界关心直接可使用的技术，学术界关心是否有所创新，有所发现。目标有所不同；不同领域有不同的发展历史；使用不同概念和技术路线；使得关心和研究同一类问题的学者们，常互不了解、互不理解。为提高参会专业人员的视野和水平，在会前、会上和会后对论文充分评审、探讨共同兴趣的问题，通过研讨会期内的密切接触和自由切磋，加深了对各自研究工作的认识。

SDM-DS 对待不同观点的措施，稍有别于 STEP 及其他学术会议；以一种更自由的方式，扩大视野和启发新思路。读者可以注意到：在迅速发展的技术支持下，广泛的群体积极参与已经成为时代特征。学术论文的主流评审技术（方法）依靠客观指标和主观指标，有合理性。问题是，一旦考虑分布系统，客观指标表现为数量多、变化快。主观指标方式一般根据（具有一定权威性的）专家匿名和回避评审制度；但匿名副作用可凸现为回避责任，强化评审者与被评审者的不对等断裂关系，对创新成果容易误判；加强透明性的负面作用可能表现为增加评审选择成本。SDM-DS 2003 国际研讨会面对飞速发展的技术难题，涉及新兴领域，多学科结合、边缘科学等发展中的研究，强调评审者与论文作者较为平等的交流关系；用透明、公开、允许发表和暂时共存不同观点来处理不同学术观点分歧。中外评审专家志愿为会议服务，认真、充分地提出许多重要见解，使许多作者得益匪浅。

作为创新研究领域的高层次学术交流平台，文集专设一栏，公布论文的评审意见和作者所持立场；用以建设促进科技人员、特别是青年科技人员成熟的平台。欢迎 SDM-DS 领域从事研究和实践的专业人员和研究生们，加强学习、深入研究、实践和思考，发展建设性交流和批评。从更广泛意义来看，科研发展根本政策问题是评价问题。希望这种做法有利于科技创新，并逐步规范化。

预期在分布计算时代，从基础概念、体系结构、实现技术、测试验证到教育等方面进行深入探索和交流，突破一些传统观念的壁垒，将有益于适应分布系统体系的基础工作。

周之英

2004 年 9 月于清华大学

Co-Organizers

Zhou Zhiying (周之英) is professor, dept. of Comp. Sci. and Tech., Tsinghua University, China. She was invited as a conjunct associate researcher of Courant institute of math sci., New York University from the end of 1979 for 2 years; as a fellow of Computer division of EE, UC Berkeley in 1981 winter. She lectured software engineering in Tsinghua, and was responsible to that scopes in several large complex software system developments, which awarded nationally and internationally. Her publications include more than 60 papers and 20 books in Chinese and English. She, as Co-Organizer and Chair of international workshops on software development methodologies in distributed systems, works for promoting academic and technologies exchange between the east and west. Her current research interests include theories and practices based on distributed systems, software development methodology, ancient oriental philosophy, science history and cognitive science.

Scott Tilley (斯高德) is an Associate Professor in the Department of Computer Sciences at the Florida Institute of Technology, and Principal of S. R. Tilley & Associates, an information technology consultancy based on Florida Space Coast. Hehas a Ph. D. from the University of Victoria. His research interests include software evolution, program redocumentation, and technology adoption. He co-organized and co-chaired several International Workshops on Software Development Methodologies for Distributed Systems, and also on the Steering Committee of the SDM-DS series of workshops. He is Chair of the Steering Committee for the IEEE Web Site Evolution (WSE) series of events, and the current President of the Association for Computing Machinery Special Interest Group on Design of Communication (ACM SIGDOC).

Contributors on Organizing and Reviewing

Liu Chao, Beijing University of Aeronautics and Astronautics, China
Huang Tao, Institute of software, Chinese Academy of Sciences, China
Jin Beihong, Institute of software, Chinese Academy of Sciences, China
Shengyuan Wang, Tsinghua University, China
WenPin Jiao, Peking University, China
Sean Wong, Entena, USA
Shen Beijun, East China University of Science and Technology, China
Lv Jian, Nanjing University, China
Yushun Fan, CIMS Center, Tsinghua University, China
Jing Dong, Canada; University of Texas at Dallas, USA
Jingde Cheng, Saitama University, Japan
Rao Talasila, USA; iGATE Global Systems, Bangalore (India) and Wuxi(China)
Shihong Huang, China; University of California at Riverside, USA
Zhang Leilei, Tsinghua University, China

Preface

SDM-DS 2003

Distributed systems are notoriously challenging to engineer. Heterogeneous computing platforms, multiple programming languages, and increasing complexity all characterize the development of modern distributed software-intensive systems. An over-arching theme of the engineering of such systems is the need for a mastery of multiple topic areas. New technologies such as Web services offer innovative solutions to long-standing problems of coordination, integration, and deployment. At the same time, new risks inherent in the adoption of such technologies, such as security and reliability, must be addressed.

Software development methodology for distributed system has become a hot area of interest for information technology professions. Academics and researchers interested in distributed systems focus on fundamental areas from a theoretical point of view. For example, software engineering processes, software architecture, and net-centric computing. Industry is concerned with quality attributes that can affect the realization of a distributed system, such as Internet standards, quality of service issues, and usable tools and techniques from a practical point of view. Educators need to provide students with multiple perspectives for professional success. But gaps among the different groups of stakeholders do exist.

The goal of the two workshops on "Software Development Methodologies for Distributed Systems" that took place in 2003 was to bring together members of the distributed systems, software engineering, and information technology communities to discuss innovative methods for designing, constructing, and managing large-scale distributed systems, as well as to explore the new ground for the hard nuts of the distributed systems. Modern distributed systems would be benefit from transiting results from theory to practice.

The original workshop was scheduled to take place in July, 2003 in Orlando, Florida, USA, as part of SCI 2003 conference. However, 2003 was an unusual year for China as well as the rest of the world. For example, SARS delayed the workshop's participants from traveling to the event. The cooperative work with STEP 2003 successfully turned SDM-DS into a truly distributed workshop, both in time and in space: SDM-DS 2003 (A), in Amsterdam, The Netherlands on Sept. 20, and SDM-DS 2003 (B) in Beijing, China on Dec. 16-18. In the end, the result was perhaps more satisfying than the single day in Orlando would have been.

SDM-DS 2003 was a special event that featured the studying, understanding, exchanging the ancient Eastern philosophies and modern Western ideas in science and technology to develop the innovative methodology for distributed systems by learning each other from the strong points to offset the weakness. Then, the engaging discussion sessions, such as paper presentations, group work, brainstorming, technical demonstrations, and a visit to the Zhong-Guan-Cun Software Park, worked up the fruitful gains of exciting new research and empirical results from different positions.

The organizer's philosophy for SDM-DS 2003 is to provide a forum for enhancing the international exchange of information, and trying to connect the different viewpoints within a workshop for mutual benefits. The organizing work showed that there are a lot of different viewpoints in different positions, as the difference between the academic research and industry practice. Besides the workshop meetings, the publication of proceedings also is needed to reflect the reality. Here we provide comments to the paper with an opportunity to express author position.

The success of the workshop results from the contributions of all authors, participants, and volunteers. We are all beneficiaries of the fresh ideas, efforts, and enthusiastic scientific spirit to the final proceedings. SDM-DS 2003 provided a wonderful beginning to what hopefully will be a bright future to the cooperative participants.

Zhou Zhiying

Co-Organizer

Tsinghua University, China

Scott Tilley

Co-Organizer

Florida Institute of Technology, USA

Contents

Proceedings of the First International Workshop on Software Development Methodologies for Distributed Systems

Regular Papers
Fundamental

Regular Papers
Architecture, Components, Web Services Technologies

Regular Papers
Testing and Validation

Regular Papers
Experimental Works

Presentation Slides

Short Notes

Comments and Author Positions for Regular Papers

Regular Papers
Fundamental

The Perfect Ball in Software Development

Zhou Zhiying
Department of Computer Science and Technology,
Tsinghua University, Beijing, China

Abstract: *Many implications in the software practices motivated the work on perfect ball. It reflects the uncontrollable and the unknown parts in the real project, and offends the foundation of the most of existed methodologies, such as consistency. The concepts of perfect ball are resulted from exploring the interesting linkages among advance west sciences/technologies and current software reality. It absorbs the ideas of Heisenberg's uncertainty relation. Perfect ball aims to substitute for the pre-fixed goals of projects in the traditional software development, and for the dynamic goals of projects in the modern agile technology. The software developments under the perfect ball paradise reflect both subject and object issues in the software developments. Both subject and object grow up together. It means that the vivid learning behaviors in the software developing substitute for only mechanical behaviors in the traditional software developing.*

Keywords: *software development methods, perfect, goals, uncertainty relation*

1. Introduction

The computer system is the machine embedded human thoughts as automatic software system. Today the influences of the software systems in the society and the daily life are growing up rapidly. There are many stories about the serious damages from one bit fault within millions lines of codes of software system. Software development had to pursue the perfectly high quality in their development processes. Usually, the perfect means the qualified artifacts of the project. Zero error, such as the clean-room software development, dreamed to be the best. It might be a real target in the traditional stable environments, but it is not always good in current software development. Now many successful projects show the alternative trends for better solutions.

The next paragraphs analyses and distinguishes the premises of project requirements of the traditional software development methods from one of the modern correspondents. The facts about the unavoidable imperfect demonstrate why we need to push up the new ideas, and how to consider the influences from the unknown in the development cycle.

Then, a new idea, named as "perfect ball" for the software development in the very complex situation, is emerged. Heisenberg's uncertainty relation describes the observation relation between object and subject, and promotes the concepts of perfect ball.

The summary mentions the simple examples of perspective applications for the test-bed of the concepts of perfect ball, such as a case study on test driven method.

2. Premises of project requirements in traditional software developments

The traditional software developments assume that the development tasks work under the ideal and "perfect" conditions, as if no fault. There are many implications in the software practices. The following phenomena about "perfect" are the examples of implying understanding in the most of existed technologies.

a) It is common practices for developers that there are no written items about the faults in the platforms. Developers have to recognize the functionality of the platform as "perfect" as announced.

b) The goals or requirements of software project are generally to demand to improve until perfectly

complete and exact consistency. It also is the tasks of the technologies, tools and managements.

c) The software development organization could control and improve the development processes up to the perfectly automatic and efficient as CMM level 5.

d) Tools announced to be perfectly helpful and useful, endlessly.

e) The developers are qualified, or selected and trained to be the perfect as demanded.

f) Testing is the last line of quality, but it bears the great pressures and extreme difficulty for reaching the planned goals if the system is quite complex.

Generally, software development methods are relied on the project plans. The reality of the project plan is under the implications of above "perfect", and it is imperfect in facts for the very complex systems.

To reduce or avoid the development complexity, decomposition technologies try to control and reduce the development complexity. Then developers only consider several small components with hopeful less complexity and their relationships. The finite goals of the project become the main concerns for project success because only limited resources exist in the organizations.

As the system complexity increased by increased, the volume of details of components and relationships in the traditional technologies [1, 2, 3] had to be grown up greatly. The correspond methodologies and development processes in main technology stream have to process the great volume of works. They face the new challenges from huge workloads and complexity never seen in the past, since the economic globalize and network computing.

3. Features of the modern development technologies

The impacts of the economic globalization bring additional connections and the shift changes into the software developments. There were the real situations, whatever the best teams and the best managements worked properly out the best output artifacts, but the output artifacts could not satisfy the end-user's needs if requirements had been changed essentially or quickly. Then, the expected best became the unqualified in such situations ever if software project was not very complex.

The modern software development technologies reflect the shift changeable and adaptive. IT professionals look for the agile software development methods, which aim the dynamic goals of development project with the development plan in a short period. New methods focus to speed the development cycle, reduce the workloads, and adopt tools, which could ease IT skills of developing software applications for the great number of man forces qualified to tasks of projects. It is easy to explain why the agile software development methods become popular and gain better solutions in different situations [4, 5] if we think about a simple metric of the relationship between the change speed and developing time.

In formal terms, the traditional software development corresponds to the deductive processes from premises of the project requirements; the modern one corresponds to the set of simpler deductive processes from the sets of premises with the quite limited set size, for timely reaction to the changes. However, many implications in the software practices reflect the uncontrollable and the unknown parts in the real project, which offends the planning and controlling spirit of the existed methodologies. In fact, the modern methodologies, as the agile, also face many difficulties if they attempt to keep stable successes in the different cases or extensive applications.

4. Unavoidable imperfect in the software development

The community of software development looks for the innovation methods. As the first step, we must properly explain the phenomenon in the reality from the different viewpoints.

a) Platform

Now it is too complex to characterize the true situations of the platform, such as impacts of power, temperature, electromagnetic induction, radiation, integration structure, compatible (but not same)

components (or parts) from different vendors or different manufacturing series of same vendor, quickly upgrading parts/systems, manufacture qualities, design assumptions, wire impacts and etc. Even if we do not consider the errors or faults, people could image how serious incidents caused from the platforms, which composed by the compatible software components based on compatible hardware and compatible drive programs. The incidents could be emerged in repeatable or unrepeatable ways, which caused by accumulating (or directly) many or less, and big or little tiny differences. The software patches in our daily are also the double-edged swords, which resolve current issues as well as insert potential issues. Activities of system maintenance would also bring serious problems of unreliable and unstable. We conclude that the development platform cannot be perfect to announced behaviors.

b) Working goal and related messages:

The goal of the project should conform to the user requirements according to the software development methods. In the reality, it is not realistic if the goal of the project only conforms to user satisfaction without the (money, mental, etc) interests of the development companies. Usually, goals of the project reflect the multi-viewpoints related. Some researchers indicate that the number of the viewpoints would be ever unaccountable.

We know that the most important thing is to define goals of project. Usually, "XX goals" or the finite goal trees are one with fuzzy, abstract, and un-concrete style of different understanding among the people evolved. Brief descriptions are benefit to the imagination space of creative works. The side effects of brief descriptions lead to misunderstanding and inconsistence among people with different backgrounds. Extensive descriptions are benefit to the more accurate definitions for reducing the misunderstanding and inconsistence from one side, but with the worse side effects resulted from larger volumes. The most of the technology focuses on the visible parts without considering the implications in documents, agreements, communication and artifacts of the project. All above mean that project goals and related messages are imperfect for developers.

c) Organization and people

Software development processes demand the visibility for management control and execution. Visibility is the special representation of the known. Strategy and tactics consideration of commercial interests, the protection of intelligent right, and lack of time/cost/capability are obstacles and limitations to acquire necessary knowledge. For example, the openness and transparency drives the demands of interoperability of the distributed systems, but we cannot dream to implement completely on the current common foundation, such as patents.

People directly work for the objective goals and use the existed materials and self/group-talents to guide software project processes. It is ideal that the capability of developers should be good at their tasks, and results from training courses and experiences, which imply the repeatable and directly helpful/useful knowledge by training and self-experiencing. However, the difficulty is how directly evaluate the past developing processes and products. Usually the measurements always delay. Organizations and developers are very difficult to get the concrete, direct, simple, and feasible working guides for the new complex systems. They have to select, amend, relearn, adapt, and ever create new approaches. The implicit and not timely guides become the imperfect guidance to the tasks.

d) Testing reaction and debugging:

According to the traditional methods, the test plan designed is directly after the completion of requirement specification. It represents the known about the system.

How about the "non-known", except the known?

Let us analyze the system and circumstance complexities. There are many kind of the hided or unknown information. Such as the complete knowledge about the project, circumstance, customers, organization, and platform; the inherent and potential faults of the platforms; some reactions of intelligent components; the abnormal events caused by virus/ hackers or faults; intelligent copyrights; commercial interests; the actual effects linked with openness/ transparency advertisements; different culture backgrounds and development errors interweave into the extreme complexity. The features related to the unknown cannot be included in the test plan. It is out of control to development team and

testers.

Software testing is determined by comparing actual output and expected output under certain input, which make the decision if the "tested" is reaching the pre-fixed (planned) goal. In addition, the traditional testing theories say that it is undeterminable if system is acceptable when passing test, but how to determine if system is acceptable under the condition of some unknown existed, when not passing test. This testing picture is imperfect. It is difficult or unfair to make decision only by passing test or not, and how to handle the testing, and make proper reaction when testing failure?

e) Methodology

Every software development methods seemed OK at its beginning, but followed with the endless improvements and increasing volumes during their executing and practices.

The ideal process of improvements consists with identifying faults during their executing; developing new improvements for resolving specific faults, and integrating them into the original. "Integrating" may act as a patch in the current software. The classical test theory declares the picture: fixing a bug might bring more bugs into the system. The patches could fix the bugs, and may bring new or potential bugs, especially if patches not passed by the complete testing. We know that the complete testing is impossible. The patch style affects the positive results only if it is within the view of testers or within very narrow domains. The negative effects of patches also existed. It would bring new issues in the broader domains, and be out of the view of maintainers and testers. The further patches would make the system with the continue proliferation commonly. Similarly, the aided tools and development environments become complex and larger until almost uncontrollable. This phenomenon is not what we want.

f) The psychology factors:

Pursuing success, person and group interests, preemption, marketing advertisement effects, the finite and available resources, and the weakness of human being increases the inflated styles in software development, which greatly enlarge the complexity and hardness of the system.

5. Concepts of Perfect ball with uncertainty relation

In the view of the traditional software engineering, system developments and improvements are relied on the objective facts. It introduced the volume workloads useless and not timely. Now people are conscious that the cooperative works, the consciousness, the understanding and the creative spirits of subject facts play the more important role in the complex software development. The concepts of innovative methods must reflect the difference from the concepts adopted in the simpler system. Now it is time to turn from only pursuing the fine and accurate objective into the reality consisted of both subject and object.

Perfect ball is motivated and enhanced from two great thoughts: Heisenberg's uncertainty relation and the ancient oriental philosophies.

Heisenberg's uncertainty relation [6] recalls the duality, which explores the observation relation between object and subject. In microcosms, the duality of the wave-particle reflects the mutual relation or interference of subject, as the observer of the wave-particle, and object, as the observed of the particle-wave. Heisenberg's observer cannot determine the position and the momentum of the observed one simultaneously, as the formula of $\Delta x \cdot \Delta p \geqslant h$.

The duality in the software developments means the explicitly described objective facts companying the implication of subject facts and hidden information related to the project. It is the uncontrollable as the uncertainty of the software development. The named perfect ball substitutes for a perfect point as the planned goals of project. It is the almost unreachable goal in the concepts of perfect ball for the reality of complex or distributed systems, refer the paper titled on goals and codes in the distributed systems [7].

How we can reach the success of the project if the development target is the uncertainty as perfect ball? Let's recall the relationship between object and subject in software development.

From 40's to now, object in the software development scope is the software system (artifacts) being developed, which is looked to be the nonliving in the traditional viewpoint of software development. Subject is developer, and it is independent to object.

Actually, both "object and subject" should grow up together. This is a very distinctive concept from the traditional software development methodologies, which fix the capability of developers and organizations and overlook the knowledge exchanging process between inside and outside of developers and organizations during the life cycle of the project.

According to the most of existed technologies, the fixed capability of software developer, during the development cycle, should be a root of the difficulty for planning and implementing of the development mission under the very complex situation.

The capability of good developer should not be pre-fixed one during the life cycle, which increased by absorbing the fresh knowledge outside. The capability of individuals and organizations in the software process will be improved and enhanced for conquering the difficulty. The understanding of perfect ball development paradise declares that the development process includes the study process as naturally breathe process, and not only a deductive process of the mechanical style. It promotes to resolve difficulties by learning and absorbing the new knowledge from the failures or pressures of the unexpected.

The creative process of perfect ball itself was an example of learning and absorbing the knowledge from outside software development scopes. The test driven method, with new testing concept, is to realize the practice of perfect ball. The graduate student project [8] on web-based tabular information integration acted as a test-bed of the concept of perfect ball, which improved the project progress obviously. The case study also gained to initiate the innovative architecture issues of the system. It demoed that studying and analyzing failures from software testing implied the new findings and fresh knowledge for the next progress. Developers understood that the failure of testing not be the project failure if they worked properly. Failure and success like the sunshine and raining days alternatively in the daily life. It helps to conquer the psychology barrier of overanxious for quick results, and turn the development efforts towards efficiently directions. The organizations and developers can be confident to treat the failures, pressures from the mission under very complex situation with encourage and learning spirits.

6. Summary

This work concludes the following:

(1) Imperfect is resulted from the uncontrollable and the unknown in software development, but implicitly. It is unavoidable for the complex environments in the most methodologies.

(2) The commonality and linkage of the unrelated areas of Heisenberg's study and software development is uncovered. The concepts of perfect ball are resulted from exploring the ideas of advance west sciences and technologies and current software reality. It is benefit to the future innovations on many topics of software engineering as well as other scientific arena by the mutual study, complementation, cooperation and coordination.

(3) Concepts of perfect ball clarify that the planning issues in the existed development methods are resulted from the concepts of the deductive processes. It is important and necessary to include the studying process into the development instead of deductive process only during complex software development. It promotes the vivid learning behaviors, explicitly entering subject facts, instead of the mechanical behaviors in software developing. The innovative system structures and math tools will be the next breakthrough.

Now "perfect ball" is just an initiation of the new concepts. The concepts of perfect ball would help to rethink many concepts in the software development, such as software architectures, development methodologies, testing standards. It might bring some transformation into IT industries. More researches on the fundamental issues about different philosophies, the metrics of perfect ball and its applications needed.

However, the modern science (such as math, physics, biology, etc) would become the powerful intelligent resources for the challenges from software development we faced.