完美球本质——论分布系统软件开发技术*

周之英+

(清华大学 计算机科学与技术系,北京 100084)

Perfect Ball in Nature—On Software Development Methodologies for Distributed Systems

ZHOU Zhi-Ying+

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

+ Corresponding author: Phn: +86-10-62773382, E-mail: zhouzhiy@tsinghua.edu.cn

http://www.tsinghua.edu.cn/docsn/wb/interconf/C-2005.htm

Received 2004-06-18; Accepted 2004-10-10

Zhou ZY. Perfect ball in nature—On software development methodologies for distributed systems. *Journal of Software*, 2005,16(12):2166–2171. DOI: 10.1360/jos162166

Abstract: Many implications and unavoidable imperfects in software practices, as the uncontrollable and the unknown parts, offend the foundation of most existed methodologies. Perfect ball is based on the duality and mappings. The uncertainty relation of distributed systems addresses that codes, as available products, and goals, as the announced features of the software products, cannot be determined simultaneously. A triangle relationship among address, thought and object is analyzed for clarifying the perfect point and non-zero area. It is intent to substitute the perfect ball for specific pre-fixed or dynamic modified goals and the step for reducing influences of the predestination and probability theories. The explanation of the linkage between software system and human body shows the concept-mapping between Qigong philosophy and software development. Software developments under perfect ball paradise include the vivid learning behaviors, rather than only the mechanical behaviors. Few applications of perfect ball are mentioned.

Key words: software development method; distributed system; perfect; goal; uncertainty relation; ancient oriental philosophies

摘 要: 软件开发实践中存在的隐含成分和无法回避的“不够完美”,就是现实软件开发项目中的无法控制或未知部分,这违反现存的大部分软件开发方法的基础,如一致性.完美球却是基于二重性,把微观世界的一些成分和复杂系统相映射.这是探索先进的西方科学技术和古代东方哲学的产物.分析由“陈述”,“思想”和“客观事实”组成的三角形,以说明完美点相区别的三角形.分布系统的测不准关系阐明:代码(代表可用产品)和目标(代表软件产品的性能)不能同时确定.在解释软件系统与人体间的联系中表明在气功哲学概念和软件开发概念之间可建立的某种映射关系.在完美球范畴下的软件开发强调生动活跃的学习行为,而不仅看重的自动机行为.用完美球替代特定目标点或动态可修改的目标点.系统复杂性日益增加,逐步减少概率论和预定计划影响.列举完美球

*ZHOU Zhi-Ying was born in 1939. She is a professor of Tsinghua University. Her current research interests include theories and practices on software development methodologies, distributed systems, ancient oriental philosophies, science history and cognitive science etc.

少许应用.
关键词: 软件开发方法;分布系统;完美;目标;测不准关系;古代东方哲学
中图法分类号: TP311 文献标识码: A

1 Introduction

Computer system is like a machine, and embedded human thoughts as an automatic software system. There are many stories about the serious damages from one bit fault within millions lines of codes. Software developments[1−4] have to pursue the perfectly high quality. The traditional software developments attempt to control development tasks based on the project plans. In the formal terms, the success of the project relates with the controlled finite goals. It corresponds to separable entities or controllable finite variables of the black box. The goal of the project is like the pre-fixed (planned) or dynamic points in the goal space of multi-dimensions, like the perfect point as a reachable goal. Many agile software development methods emerged with better solutions in different situations that focus on speeding development cycle, reducing workloads and building tools for easing IT skills. Agile methods still face many difficulties in different cases.

It is too complex to characterize the true situations of the platform, such as impacts of power, temperature, and electro-magnetic induction. Even if we do not consider errors, image how serious incidents caused from platforms, composed by compatible software components, hardware, drive programs, and by accumulating (or directly) many or less, and big or little tiny differences. The development platform cannot be perfect to announced behaviors.

Generally, the goals or requirements of software project are demanded to improve until perfect consistency. 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, which are one with fuzzy, abstract style of different understandings. Brief descriptions are beneficial to imagination space of creative works, but with misunderstanding and inconsistence among people of different backgrounds. Extensive descriptions are beneficial to more accurate definitions for reducing misunderstanding and inconsistence, but with the worse from larger volumes. The delay, missed, incomplete or inflated information become the unknown and wrong message. All mean imperfect.

The software development organization should control and improve the development processes up to the perfectly automatic and efficient level as CMM 5, which demands the visibility as 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 to acquire necessary knowledge. For example, the openness and the transparency are required from the interoperability of the distributed systems, but we cannot dream to implement it completely if patents exist. In addition, it is difficult to directly evaluate the past developing processes and products. Usually the measurements always delay. Organizations and developers are very difficult to get the concrete and feasible working guides. These become the imperfect guidance.

The designed test plan is direct after the completion of requirement specification. It represents the known about the system. The hided or unknown information cannot be included in the test plan. It is out of control to testers. The developers couldn’t be qualified, or selected and trained perfectly as the demands. Pursuing success, person/group interests, preemption, marketing advertisement effects, the finite available resources, and the weakness of human being increase the inflated style in software development, which enlarges the complexity and hardness.

The work based on that imperfect is unavoidable in the software development especially in distributed systems[5,6]. Next, we discuss “perfect ball” based on uncertainty in the distributed systems and understand the triangle dealing with social and nature sciences. The simple examples and summary includes conclusions and the further.

2 Perfect Ball: New Way Based on Science Achievements and Ancient Oriental Philosophies

Facts about the unavoidable imperfect push up the perfect ball, named with some kind of modern happiness style. It is enhanced from two types of great thoughts: the ancient oriental philosophies and the advanced science achievements in the 20th century, such as the theory of relativity, Quantum Mechanics and Chaos theories, which mainly resulted from the works in the west.

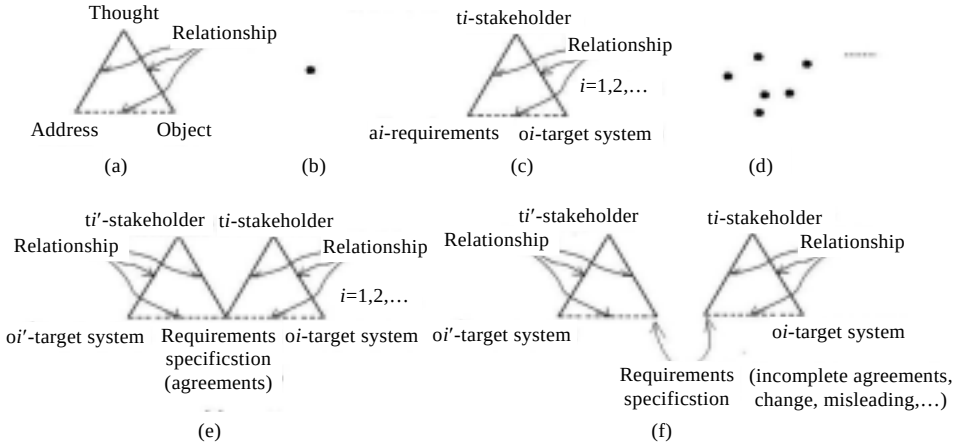
Firstly, perfect ball was initiated from Heisenberg's uncertainty relation. Imaging a mapping between the microcosms and the complex distributed systems. If codes (all source codes, executable codes, and documents are named as codes in this paper for simplicity) in the complex distributed project correspond to the role of the position of the particle in the microcosms, the goals or requirements of system behaviors correspond to the role of the momentum. Following Heisenberg' uncertainty relation, perfect ball explicitly addresses that the codes, as available products, and the goals, as the announced features of software products, cannot be determined simultaneously. In other words, the reality shows that it is almost impossible to define the complete features of the complex distributed system from the known codes, and it is very hard to produce codes of end product with the pre-defined features, especially if there are considerable changing influences from the outside environment.

The second mapping is also very interesting and helpful to understand the view of perfect ball. The particle-wave (also position-momentum) corresponds to visible-invisible in the microcosms. In the distributed system, the codes are the visible, which the texts located at the special positions in disc (paper, memory, tape, etc) of the software products, and the behavior or goals as the effects of code executing, are one with the global perspective, which is 'invisible' as the wave. Actually, the goals relate with the behaviors of the system, and the codes are the systems themselves. They are different. People usually regard codes of the product as equivalent to goals. It is a misunderstanding or fuzzy saying and might be validated only if validation processes and verification processes are ideally and completely successful. Developers direct the development efforts to codes, but not to goals of the project. Furthermore, the indirectness usually implies a dangerous misleading.

What will happen in distributed systems? Figure 1(a) shows thoughts, addresses and objects with their relations among the developers, artifacts and users. Usually it is expected that someone's address represents his/her thought, and someone's thought represents his/her understanding of object, which is "what object is". In the reality, there is a distance because of the limited time for the address, the way of presentation, the language difference, and so on. Similarly, there is also a distance between someone's thought as his/her understanding of the object, and what the object is, because of the limited knowledge, misunderstanding, and so on. The ideal situation of the triangle is similar to a perfect point with zero area. It is similar to the implications of traditional software engineering and the best practices announced ideally, refer to Fig.1(b)~(d) for the relations in ideal cases in the software development. Here goal is an object or target system; the mind of stakeholder corresponds to 'thought'. The codes of the project are transformed from specifications of the project as 'address' during software development. Generally, Fig.1(d) reflects the multi-perfect points in an incomplete agreement of the multi-people and the rather slowly stable changeable situations when each one is a perfect point.

In the reality, the area of this triangle is not zero. Ever though, how to measure the distance as the relationship will be the task for the innovation math. It is still possible to study what is the meaning[7] of non-zero area (not a point) in software development. One (as stakeholder-*i*) addresses incompletely (or some level of wrong) his/her thought because of the limited time or hided and unknown motivation. Similarly, thought represents the understanding of 'someone's address' and/or 'object/world', but the understanding might be incomplete (or some level of wrong), because of the hided and unknown, the misunderstanding by the fixation of self experiences, lack of knowledge, and so on. All of above brings the non-zero area of the triangle. It is very serious if there are many

subject and object evolved in the system. Figure 1(f) is a situation without the agreement. The reality in the distributed systems should be a kind of the mixture in Fig.1.



(a) Triangle among thought, address, object and their relations
(b, c, d) Ideal cases—in traditional view
(e, f) Real cases—the view of perfect ball

Fig.1 Thought, address, object and their relations in traditional view and the view of perfect ball

Software development engineering/re-engineering are processes from the goals/codes to the codes/goals. Both directions feature some non-inverse are against the traditional fundamentals of the formal methods. The appearance of perfect ball reflects the current change and evolution trends of software development methodologies from only pursuing ideal mechanics to the reality. It regards to steps beyond one from scratches to rigor rules.

In other hand, it is very interesting to the commonality between Heisenberg's uncertainty relation and ancient oriental philosophies, such as Chinese Qigong and India Yoga[8,9]. Both describe the relation between subject and object. It is another effort on the direction of digging the ancient oriental treasures[10]. Qigong exercisers activate themselves from the subjective consciousness step-by-step to improve or develop body potential capability to be a special unordinary functionality. Table 1 shows examples of Qigong sayings in the masses (Discuss with Zheng Zuojia, Shanghai, China, 2004). For the explanation of the linkage to software system, refer to Table 2.

Table 1 Examples of Qigong Sayings

Chinese (Pronunciation)	Explanation in terms of software engineering
气(Qi)	1 Potential capability (energy) and existed knowledge of the body, which might be transformed into functions 2. The process of building the body
外气(WaiQi)	Outside knowledge, pressures or forces
内气(NeiQi)	Knowledge and potential intelligence
功(Gong)	1. The functions of the body 2. The goal of building body
炼精化气(LiangJingHuaQi)	Analyzing and understanding the initial conditions of the body/system to transform into a process
炼气化神(LiangQiHuaSheng)	Analyze, understand and improve the processes for transforming the potential capability of the body into the functions
炼神入道(LiangShengRuDao)	Analyze, understand and improve functions to reach the goal of the target system
道(Dao)	Goal of the mission

According to the most existed technologies, the fixation of estimated capability of software developer during the development cycle should be a root of the difficulty for planning and implementing the development mission under the very complex situation. From the viewpoint of Qigong, the capability of a good developer should not be a pre-fixed one during Qigong practice, which increases 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. Understanding of perfect ball development paradise declares that development process includes the study process as breathe process of the metabolism, and not only a deductive process of mechanical style. Resolving difficulties by learning and absorbing 'WaiQi' corresponds to new knowledge from unexpected failures/pressures.

Table 2 Mapping between Qigong and software development

	Qigong	Software development
Activity	Qigong exercises	Tasks in different development stages according to the software development methodologies.
Mission to build	Human body	Software system
Function	Gong	Function, behaviors
Energies/Capability/Potential	Qi	Resources and the known / Knowledge (inside)
Subject	Qigong exerciser	Development team, developer
Object	Qigong exerciser	Software system

3 Application Issue—The Examples

Open source developments are the revolutionary new model for software development, especially for distributed systems[11]. OSD provides source codes to developers, who could aim at their own goals and focus on the codes firstly. It opens the potential of the codes for the flexible goals to the possibly unlimited number of developers with their own goals. It is different from the dynamically existed (pre-defined) set of goals. OSD developers could use the existed OSD codes to develop products, which satisfy OSD developers and extend the OSD scope if OSD regulations execute properly. As the whole view of OSD, it looks like the opposite direction of developing codes. In fact, the pre-defined goals for a special project, such as Linux, do exist without a forced goal for whole community. However, it is also the double-edged sword of reducing the cost of the development and possibly producing uncontrollable and harmful results. Of cause, the OSD communities develop the small kernel based on the pre-defined goals as the very limited parts of OSD from the viewpoints of time and spaces at the beginning. We conclude that open source developments allow flexible goals for the different developers and users, which are different from calculation-center, requirements-center (client-center, company-center) and user-satisfaction-center with specific pre-fixed or dynamic modified goals.

Another result from perfect ball is the test driven method with new testing concept. Reference [12] acts as the test-bed for test-driven method of perfect ball, which improves the project progress obviously. It indicates that studying and analyzing the failures from software testing may lead to the new findings and fresh knowledge for the next progress. Developers understand that the failure of testing is not the project failure if they work properly. Failure and success is 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 efficient directions.

Perfect ball provides better psychology factors, which help to reduce the unnecessary pressures of pursuing the unreachable absolute perfect. The unnecessary pressures would push people to work on unnecessary laborious tasks for engaging in the unreachable perfect. Pursuing the false perfect likes the commercial advertisements with some non-scientific characters. Software organizations and developers could remove their ideal perfect dreams and move back to the reality after knowing perfect ball.

At last, the creative development process of perfect ball itself is a good example of learning and absorbing the knowledge from the outside software development scopes.

4 Summary

"Perfect ball" is an initiation for better understanding the nature of the software development, and helps to

rethink many concepts in software development, which helps to have better solutions. There are many further studying: When the concepts of perfect ball are effective? What is the definition and measure of perfect ball formally? Reference [13] said that nature selection is strongly dependent on the environments without the plan, goal and direction. The parameters in Heisenberg uncertainty relation relate to the experiments in the microcosms with the velocity of light. It brings us to think the designing of good experiments. Rates of changes during developing should be considered. Actually, the nature of perfect ball is close to Chaos model for treating great complexity, which against the base of probability theories. The new math tools, such as the fractal dimension[14], might be helpful for infinitely increasing the number of the dimension of the traditional geometry. In addition, it is interesting introduce the social and living study, instead of only emphasizing the pure logic of formal methods of constructing the rigor rules as the traditional mechanics nature.

All those are the interesting approaches for our study. So far, Taiji Model has explored the facility of inserting the ancient Chinese philosophies as well as uncertain theory in the software technologies. However, the modern sciences[15] (such as math, physics, biology, etc.) with the refreshed ideas from the ancient oriental thinkers would become the powerful intelligent resources for the challenges from software development we face.

Acknowledgement The author would like to thank Dr. Scott Tilley, Dr. Rao Talasila, Dr. Jing Dong and Dr. Dong Yuan for their careful reading/comments.

References:

[1] Zhou ZY. Modern Software Engineering. Vol.1, Beijing: Science Press, 2002 (in Chinese).

[2] Zhou ZY. Modern Software Engineering. Vol.2, Beijing: Science Press, 2003 (in Chinese).

[3] Zhou ZY. Modern Software Engineering. Part III, Beijing: Science Press, 2000 (in Chinese).

[4] Graham I. Object-Oriented Methods—Principles & Practice. Addison-Wesley, 2001.

[5] Zhou ZY. The perfect ball in software development. In: Zhou ZY, Scott T, eds. Documents of SDM-DS 2003 with Proc. of the 1st Int'l Workshop on Software Development Methodologies for Distributed Systems. Beijing: Tsinghua University Press, 2005. 3–8.

[6] Zhou ZY. On goals and codes in the distributed system—An explanation of the concepts of perfect ball. In: Hausi M, *et al*., Proc. of the ACSE 2004 in ICSE 2004. Edinburgh: IEE Publication, 2004.

[7] Ogden CK, Ricards IA. The Meaning of Meaning. London: Routledge & Kegan Paul LTI., 1952.

[8] Li ZC. The Ancient Meaning of the Nature and the People—Scheme on the Science History of China. Zhengzhou: The Big Elephant Press, 1998 (in Chinese).

[9] Eliade M. Yoga: Essai sur Forigine de la mystique Indienne. Translated by Wu Xisheng. Beijing, China: China Zhigong Press, 2001.

[10] Zhou ZY. CMM in uncertainty environments. Communications of the ACM, 2003,46(8):115–119.

[11] Bayrak C, Davis C. The relationship between distributed system and open source development. Communications of the ACM, 2003,46(12):99–102.

[12] Shi L, Zhou ZY, Xiao HY, Gu TY. Case study: Prototype development and test-driven method. In: Zhou ZY, Tilley S, eds. Documents of SDM-DS 2003 with Proc. of the 1st Int'l Workshop on Software Development Methodologies for Distributed Systems. Beijing: Tsinghua University Press, 2005. 129–134.

[13] Nesse RM, Williams GC. Why We Get Sick—The New Science of Darwinian Medicine. Brockman, Inc. 1994.

[14] Mandelbrot BB. The Fractal Geometry of Nature. New York: W.H. Freeman and Company, 1983.

[15] Coveney P, Higfield R. The Arrow of Time. London: W.H. Allen, 1990.

附中文参考文献:

[1] 周之英.现代软件工程.第 1 册,北京:科学出版社,2002.

[2] 周之英.现代软件工程.第 2 册,北京:科学出版社,2003.

[3] 周之英.现代软件工程(下册).北京:科学出版社,2000.

[8] 李志超.天人古义——中国科学史论纲.郑州:大象出版社,1998.