New Discussion on Information Communication Model and Process Model in Software Organization

Xuejun Liu
Computer Science and Information College
Zhejiang Wanli University
8 Qianhu South Road, Ningbo, Zhejiang, 315100
China
http://www.zwu.edu.cn

Abstract: - A lot of process models for software development such as waterfall model and so on are almost based on description of the relationship between the scale and complex of software project and technology tasks and methods in every phase of development as the main content. They are suitable for the technique works of software development, but ignore the connotation of the management work in software process. Especially that they ignore the connotation of the management relevant with concrete business and for process management it is not enough to be only with technical standard. In this article, we propose an information communication model of software process and a software process model including concrete business and describe the basic tasks of team software process and relationship between the layers of software process. Starting from this, it can guide the process operation of software team, provide a new solution for theoretical research and description of the software engineering subject and provide a framework concept for the auxiliary system tools for the research of software engineering.

1 Introduction

1.1 The Shortcoming of Traditional Process Models

Because of the fact that a function can be implemented by various programming language with different algorithms, so in early times, people considered program design as a very personalized creation. At that time, since the program scale of most of the software was every small, accordingly the application and distribution area was also very limited. Moreover, most of the users of the software were developers (or the development personnel in the same mechanism), therefore, naturally these individuals or mechanisms continued to finish the work such as modifying and completing the software, as well as version upgrade by themselves. Because of this kind of personalized software environment, the process of "design" was implied process in the brain of people. There wasn't enough research and emphasis on the issue of using the method of working out software files to assist the work of development. In this situation, the process of software development was always the thing of one person or two persons and lots of the complexities of software development process were not prominent.

Later, with the decrease of computer price and popularization of application field, the need of developing large-scale computer application software grows day by day. Software users and development mechanisms started to separate, thus many "software workshops" appeared, which accepted orders on all kinds of computer application software and became the earliest "enterprises to develop software". Since then software development work formally became a project cooperated by various people to submit engineering that conforms to given function and performance objective to the users within the regulated time range.

Under the background of socialized application of computer technology, the separation of developers and users made fundamental change take place in the social nature of software development work, changing from the original work that can be dealt with from the aspect of "personal development" to a project that had to be finished in the form of "teamwork". It is a pity that at the beginning of the vigorous development of software industry, people didn't realize this point, but they gradually realized this through the occurrence of a series of general problems. When people started to realize this problem, it was by reason of they had just purchased the auxiliary equipments such as printer and so on, or because of the need of objective expansion and business improvement in the field. They proposed the

requirement of modifying the original software function, thereupon, the developers needed to read the source code again. At this time, people found that all kinds of resources such as man power, time and expense and so on expended in the software modification and maintenance were far more than the intuitive forecasting of people. Large quantity of development projects could not deliver the products on time. The vigorously developed software industry was questioned by people because of unguaranteed service quality. The trade union exclaimed "software crisis"!

By 1968, under this background, people applied the concept of "software engineering" for the first time, trying to draw ideas from the method of building construction and define normative design, coding and testing process for the software development process. In this period, people came to recognize many of the characteristics such as software life cycle and the universal applicability of the technical nature in each stage of software development. And further, models revealing the common law of software development process are summed up, such as the waterfall model, the evolution model, spiral model, the fountain model, iterative incremental model, and so on, with a good instruction on choosing the appropriate model based on the complexity and the scale of the project. Generally, these models are mainly stressing on the relation between the extent and complexity of a software project and the technical assignment and method of each stage of the development process.

With the accumulation of practical experience, people realized that there is a huge difference between Team Software Process and Personal Software Process, PSP. We can have a better understanding with following example: for a task for one person to transport bricks from the 5th floor to the 1st floor, there is hardly any process needing management; but if the same task is for five people to cooperate, there exist two plans: one is each person still work as they work alone, carrying bricks downstairs, the other is they work together with the "throw-and-catch" method. The most important thing for the "throw-and-catch" method is to define the regulation between two adjacent people to throw and catch coordinately, for otherwise it will cause safety problems. This example tells us that, if one is managing several people doing the same job, all he needs to care about is improving their ability to accomplish the task and examine their work; but when they are needed to follow a certain procedure and cooperate, he must define the corresponding standards for the procedure, and make the team member learn to follow these standards. He will find it quite challenging to make them adjust to the newly defined regulations, and will have to try his best to promote these standards and timely update them.

When people realized that writing a program in a team is no longer a personalized creation, they started to find a way to apply systematic, standardized and quantifiable method to software development, operation and maintenance.

Compared to PSP focusing on the technical abilities, the idea of "Systematic, standardized and quantifiable" approach to the TSP is essentially different: TSP models need to focus mainly on the management research, and to reveal relationship between the main task and the responsibility of each position of the team, thus being able to direct the project operation; the existing models cannot illustrate the software process from this viewpoint, but can only describe the relation between technical tasks and methods in each stage. We know that a model is "a pattern, plan, representation (especially in miniature), or description designed to show the main object or workings of an object, system, or concept." [1]. If the model could not accurately sum up the characteristics of a prototype, it will surely do harm to the research. As a result, re-examining the existing software process model, and setting up a new model to direct the operation of the team software project, is essential.

1.2 The Application of Basic CMMI Condition Is to Be Quested

Over the past decades, the development of the CMM theory has made the research content of people on software engineering go far beyond the widely-known software process model. Procedures including demand management, code review, quality tracking, project delivery, team culture building and so on, have become an important part of research in the field, providing the theoretical basis setting up a proper model for guiding the operation of a software project team.

Today, people think that "software is a kind of hierarchical technology [2]". Software engineering is the combination of a series of quality focus points and engineering process, engineering method and engineering tool. It takes the commitment of the organization to quality as the foundation, making the combination result of the focus of the organization on quality and engineering process, engineering method and engineering tool constitute the whole connotation of software engineering. People realize more and more clearly that software engineering is not only the necessary requirement on the technical

work, but also an engineering process that must be controlled and managed. That is to say, software engineering activity must be a process with planning and control, moreover, there is a necessity and possibility for the existence of normative plan and control in this industry.

Over the past decades, in CMMI document of more than 700 pages, people see the precision and maturity in this "metamodel". CMMI is a metamodel of software process...It defines the process feature that a software process shall pocess during the maturity of software establishment[3] . CMMI theoretical system has successfully made an effective distinction between process management and engineering method. Software engineering method provides a technical solution for the building of software, to solve the problem of how to do it. Generally the method includes communication, demand analysis, design modeling, programing, test and support. It relies on a group of basic principles, which cover all the technical field of software engineering, including modelling and other descriptive technologies, etc[4] ; the foundation of software engineering is process layer. It combines each technical layer so as to promptly, effectively and orderly implement engineering construction task. Therefore, "software process is a task sequence[5] " and it forms the foundation of software engineering management.

Some people say that "it is necessary for any software organization to draw ideas from CMMI concept", or we can say that "when the organization develops to a certain scale, it shall apply CMMI concept". However, the actual situation is that when most of the software organizations apply CMMI concept, they take some necessary "clipping", keeping the process specification suitable for their own team features and making some improvement on the foundation of it. It is rare to see the software organization that completely applies CMMI concept; More common situation is that after the application of CMMI, people discover that the cost of the organization increases while it is difficult to successfully digest and absorb CMMI concept in the application. To successfully apply CMMI concept, people are exploring for the basic point of CMMI application. Especially when people seek for the "interface" of application CMMI from the various software development process models such as the waterfall model, the evolution model, spiral model, the fountain model, iterative incremental model, and so on, they discover that it was originally "basic layer of engineering management---process layer" that "combined each technical layer together", but now it is rigidly dismembered: starting from the traditional process models, technicians ask the question in puzzle: I have already finished the task very well, if you don't believe it, you can have a test! Why do we have to waste our valuable time so as to make you read it or just understand it? Can"t you master some technology? Isn't it bureaucratic? It seems that normalized process standard hinders the technicians at each post to play an active role. In the eyes of these technicians, traditional software process models are transparent and visible, the reason why the management personnel can not understand or do not believe the progress and quality of the software is that they do not know the technology; Even if there is a quality problem, it is caused by bad management of some individuals. In the eyes of these technicians, traditional software process models have clearly explained the quality index of each technology in the development process and every one work according to these technology requirements, therefore, the management personnel shall assume the responsibility for the quality problem—they know that the managers have no idea about those Bug and the managers need them to solve the problem. They can not understand that for a task that is impossible to be free from defect, a necessary choice to ensure the quality reach a certain level is to rely on the process normalization and improve the process visibility. It is not enough to be only with technical standard, but it has to make the implementation process of the standard normalized and transparent. While the activity of normalizing process standard and improving process visibility itself is an important specification of technical work. Traditional process models only describe the technical task and natural from the aspect of technology, without giving a process about how to reach the technical standard in high quality; While the establishment of normalized software process is set up according to the quality of team members and cultural situation of the team, in which software engineering method or technical specification shall subject to software engineering process specification. Software process is the key link while the software technology is the mesh. Once the key link is grasped, everything falls into place. The relationship between the two can not be in disorder.

To sum up, we can see that the software team can not just regard CMMI as a kind of management form, but CMMI is closely in relation with many aspects of team construction. Mechanically and formalizedly introduction of CMMI without considering about the concrete situation of the team construction is the main reason why it encounters boycott. Therefore, we shall research in which situation does the software team possess the conditions of applying CMMI and

what is the most basic condition among these conditions.

The mechanical understanding and utilization of traditional process models of the people is not the only problem. In fact, not all the softwares need CMMI. For the small-scale software products, they rarely need a software process with strict definition. Besides, many facts also tell us that it is not that all the software engineerings without CMMI will fail. The existence of these problems propose an urgent requirement on the research of "what is the basic condition for the application of CMMI".

About the question of "what is the basic condition for the application of CMMI", if we answer this question from the two aspects of software scale or software team population, obviously there is no answer. It is because that there is no necessary connection with process concept of CMMI and software scale and software team population. The small-scale software products still can be produced under the process concept of CMMI; A software developed by a person can also apply the concept of CMMI; The application of CMMI in large-scale software engineerings with large population is even a matter-of-course.

As far as I am concerned, the basic condition of CMMI application is in relation with two factors (see Figure 1 below).

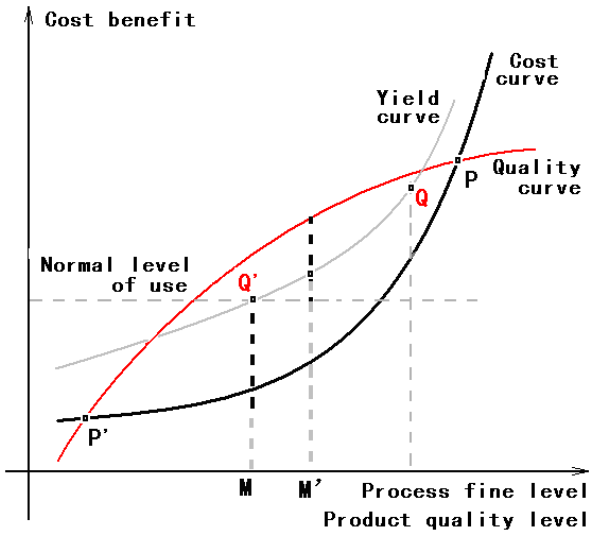

Fig.1 Relationship between software process and product quality and expense-cost

Firstly it is the market price factor of the software products.

For the software development organizations, they need to do their best to make the profit larger than cost, that is, try to make the profit curve above cost curve; However, nonlimitness theory of test tells us that after the product quality reaches a certain level, it is very difficult to improve. That is to say, after process fine level of product quality curve reaches a certain degree, it will be difficult to improve. Moreover, excessive implementation of fine process will rapidly increase the operation cost and reduce the benefit. The expenses needed to pay by the users will also rapidly exceed their anticipation range (Q'-Q point), that is, with the improvement of process fine degree, there is a crosspoint between cost curve and quality curve.

Regarding the development process from the view of the users, if they conclude development contract with software enterprises at different process levels, the expense they input and the software product quality they obtain will have huge difference. Those enterprises with low process level tend to be difficult to reach the normal level of use (Q'), therefore, they should try to merge with the enterprises M" whose process level is above M point. Moreover, when cooperating with the development organizations, they shall properly increase the input according to their own expense input ability and supervise the process level of software organization implementation during the cooperation to be above M point (such as M' point).

Therefore, we can see that when the market price of the software products reaches a certain level, it is necessary for the software organizations to apply high-quality process management (such as CMMI), through which they can effectively improve the market competitiveness while improving the product quality; On the other hand, from the aspect of the users, there is certainly a need to select a partner when inputting the capital to develop the products. However, when the products expenses reach a certain level, the work of selecting a partner and inspecting the software process level of the other party is very important, otherwise, the aftermath of concluding a development contract with an enterprise without certain process ability will be disastrous.

In fact, reasonable price of software products is decided by market mechanism, while the market quotation of the software organizations with strong process ability will become the market leader of reasonable price. It is without a doubt that the application of CMMI can improve the competitiveness of the enterprises, therefore, whether the software team shall apply CMMI is in accordance with its status in the market competition of like products. In long term, the application of CMMI will be a necessary trend.

The second factor is the organizational framework of software organization. This is another important index of CMMI application. Same as any social organization, a software project team is manually

constructed. They are structured, "... are structures built up with incidents ... established based on the infinite variety of different purposes" [6], in other words, people build an organizational structure based on possible incidents while realizing the goal and the need to deal with them. Variety of needs will result in variety of structures. The structure of a software project team reflects people's practical experience and theory, and is related to specific incidents. An efficient organizational structure is composed of its architecture and members, as well as the assignment procedure.

Traditional process models have provided very good technical task sequence for the establishment of software organizational structure. We can conduct the design of software organization structure like constructing an organizational structure in ordinary management work. We have clearly know what "incidents" are included in the technical sequence to accomplish the task, we also cleary know what situations are there to deal with the "relevant problem incidents" in the cooperation with the users, and we are also familiar with the productivity and market demand among the like products in the business field, therefore, we can select our objective and strategy based on these theoretical and practical experience, organically construct proper technology and service organizations, confer these organizations and departments with corresponding responsibility and authority, define jurisdiction and affiliation, organically construct these objectives and strategies with technical task sequence and organizations and departments, combine the factors such as team culture and so on from the aspects such as product market and its development angle and quality constitution of team members, and define the software process (and other rules and regulations).

From this we can see that the basic condition for CMMI application has direct relationship with the market price and competition environment of the products, however, it is just the external factor of the problem. Construction problem of software organization is the internal reason that influences the application of CMMI with a decisive influence on the successful application of CMMI.

2 Information Communication Model in Software Process

To correctly establish the organizational structure of software development, it shall face the organizational objective and strategy, take the management process, engineering process and technical process of the software engineering project as a foundation and adapt to it, especially that it shall organically mix the demands of software engineering process together, from which we can say that it faces the software engineering process. A software organization established in such a way is a product of the organic combination of organizational objective and strategy and the software engineering process.

Then, how is the basic structure of such an organization? What about the information communication in the organization?

2.1 Information Communication Model of Software Organization

As far as I am concerned, for software engineering field, however big the scale of the TSP is, and whatever the kind of the problem is, from the view of management, we may divide the structure into the decision level, the implementation level, the quality assurance level and the coding level (as shown in Fig 2), the division of this organizational structure is based on the management need on software process and further simplification of model levels will be unfavorable for the control on process control from different responsibility aspects. Therefore, it is a model that can not be simplified any more. This model defines basic roles in software process (which will be described in the text below) and they are the most fundamental roles in the organizations suitable for CMMI system.

For the development of software products of the same kind in the same application field, different software organizations will have different construction plans for organization structure, which is very common. In other words, the construction of software organizational structure is greatly influenced by the individuality and experience of the people.

Although different management models will always have different information communication models, in these different information communication models there is a generally applied information communication model in any software process, which is decided by the common factor in software development work. For example, any software development organization shall communicate the problem in the field with the users, solve the quality tracking problem in the processes such as software design, coding, test and maintenance and so on, and try to improve the visibility of the invisible information, to make it easy to track and communicate. Therefore, by filtering the "personalized attached processes" purely belonging to organizational coordination and management, it can discover the generally applied model in the

software process information communication in the software organization.

This generally applied software organization and information communication model in its software process has the following several features:

(1) It is a vital information communication process in dealing with all kinds of business and technical problems in the production process of software products;

(2) It is not influenced by the software organizational structure and scale and the information distribution and communication mode in the model is generally applied;

(3) It has nothing to do with the product variety, what kind of computer software application and development area it is, the information and communication process in the model is vital to this software structure.

(4) Individualized information communication model of any software organization can expand, subdivide and construct on the foundation of this model and become the information communication model in this software organization.

(5) The focus point and process area of each ability maturity degree in CMMI can find an affiliation of business nature in this model. In other words, when the process management level of software organizations continues to improve, it can naturally access CMMI process standard on the foundation of this model, to expand and complete the management process and provide natural foundation for effectively improving the process management level.

For example, in the "cyclone model" described in this article, after the expansion and evolution, "quality assurance activity" in the "basic tasks of quality review" provides a foundation for the key process area for the implementation of "software quality assurance" in "repeatable grade" in CMM system[7] ; after subdivision and evolution, the task of "plan and supervise the internal and external communication of the team" in the "basic tasks of problem decision" provides a foundation for the for the key process area for the implementation of "software engineering plan" in "repeatable grade" in CMM system.

Starting from the above-mentioned opinion, the writer proposes "Software Organization and Information Communication Model in a Software Process" as shown in Figure 2.

As shown in Figure 2, for the horizontal relationship in the same level, a healthy team culture encourages the interactive study and communication among team members; for the vertical relationship, team members are divided into basic mission groups or management levels according to the technical nature of the task; for external cooperating relationship, team members should actively keep in touch with the external environment throughout the project process, having the external information fully reviewed and reaching agreement inside the team. Especially the decision level, the implementation level and the quality assurance level have to be in such an agreement in all their works while planning and examining. Even the coding level needs a complete understanding of the clients" requirement in function and performance, thus optimizes the algorithms. In sum, the information communication in software engineering has three forms as described above, which thoroughly show that a TSP is never a personal art, but a project which needs coordinate cooperation among team members.

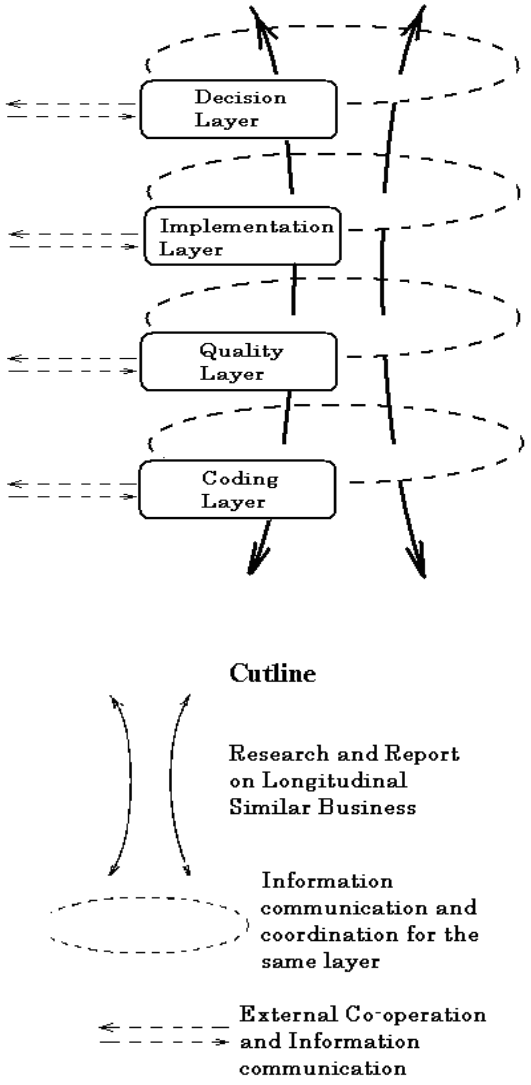

Fig. 2 Software Organization and Information Communication Model in a Software Process

2.2 Definition of management levels and responsibilities

As shown in Figure 2, the decision level is the topmost management level of a software project team, controlling the construction and operation of the software team. Their job is to organize the implementation procedure of the project, allocate the human resource and funds, as well as technical training. They are responsible for the whole team reaching the product goal and for the financial benefit of the team. Normally, this level refers to the managers who make the final decision for the goal, the tasks and the benefit of the project operation.

The implementation level refers to the managers who lead the technical groups to implement their given tasks. They are responsible for all matters concerning the group members to accomplish their tasks, and to report their work to the decision level. For the group members, they are the people in charge of the tasks, and they may break down a task into smaller tasks and assign them to smaller sub-groups; for different groups, they are the fully responsible coordinators for the tasks.

The main duty of the quality assurance level is to examine the quality of the works done by the team members. It gives estimation to the managers in the implementation level on conformity of the goal and the result. In fact, this work is also known to people as software testing and technical review, but I insist to call it the "quality assurance level" instead of the "testing level" or the "review level", in order to avoid the unnecessary misunderstanding for the managers who claim to "gradually decrease the formality of the technical review procedures", and to emphasize the indispensability of quality assurance jobs. Besides, in the view of the management levels, the quality assurance level is above the coding level, for the coding level must follow the quality specification formulated by the quality assurance level.

The coding level is formed from program designers skilled in programming techniques, whose main duty is do the coding job within the corresponding design plan.

Figure 2 does not show the level division of jobs responsible for requirement gathering, outline design and detail design, mainly because, in most conditions, every development or execution of a technical plan of a team (but not an individual) is realized under effective communication and coordination, and these abilities are the primary elements of management ability. People with only technical abilities are not capable for such jobs. So, as far as I am concerned, no matter how important the technical abilities are for a software project, they can only determine the information quality, while the critical factor for the quality of information communication is the management ability. Thus, the "information communication model" I proposed does not include these pure technical jobs, and I think these jobs can be the element in any of the decision level, the implementation level, the quality assurance level and the coding level.

In fact, members with excellent technical abilities as well as good management skills are the major candidates of positions in the decision level, the implementation level and the quality assurance level. Any software development team hopes the members in its management groups acquire not only good management skills but also good technical abilities. Management groups without technical abilities are undesirable: they cannot lead the professional members to accomplish their tasks. Assigning technical employer without any management skills to a management job usually causes bigger problems.

3 The definition of the cyclone model

In general situation, on the structural design of the information management system of an organization, we can clearly learn about the management process of this organization, where we can not see the static information of the organization, but also we can see the information flow of this organization, especially the business process (consisting of a series of tasks and is a sequence of tasks) together with the information flow process.

We can see that for the research of software process management, it is not enough to just rely on the information communication model of software organization. There is not any concrete sequence information in this model, that is, concrete process information. Therefore, it is difficult to guide the establishment of software process effectively and concretely, while individualized influence factor in the establishment of normalized software process can have effect very easily. Therefore, further establishment of a model with basic model process is completely necessary.

To obtain a vivid, understandable and descriptive framework, as an example of comparison, we have the "Hurricane Principle" summed up into Figure 3: a hurricane constantly absorb air flows from the surrounding environment and get them into its own spin. While rotating and delivering in the same level, they are also brought upward into other levels by the updraft. The center of rotation is the eye of the hurricane.

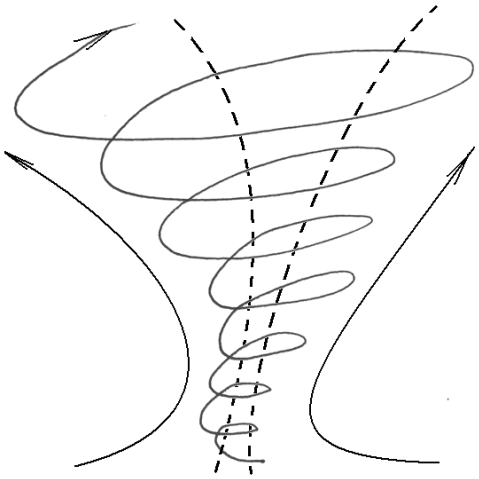

Fig. 3 The Sketch map for Cyclone

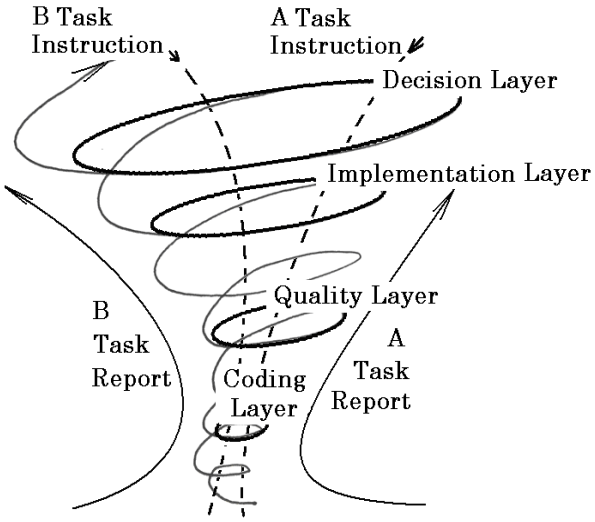

Fig. 4 Cyclone Volution Diagram

As a comparison, let's make an imitative imagination like this: each of the management level of a software team is arranged in a different height level of a hurricane, as shown in figure 3. Then we see the basic tasks of a software process (task A and task B in the figure) as the flows through each level of the hurricane. The instructions are delivered top-down to each level along the eye, while the reports are delivered bottom-up level by level. The instructions and the reports form a closed circuit of policy and execution.

This model describes the software process properly and vividly, and can be used to illustrate series of software process tasks, as well as the relationship between the procedure of decision and execution, and the management levels of the team. It is personally called the Cyclone model of a Team Software Process. The detailed description is as follows.

Normally, a TSP has six basic tasks:

3.1 Basic decision-making task

The major task of decision-making is to plan and supervise the internal and external communication of the team. It determines how to carry out other basic tasks. In particular, it formulates the "target granularity" of other basic tasks, such as the target range, content, clarity, accuracy and so on. The other five kinds of basic tasks are the natural continuation of the decision-making task, and it is the beginning and the end of each basic task of a new period, and a process circularly promote the progress of a project.The quality of software project management depends mainly on the decision-makers' control of the "target granularity". Especially when facing unclear requirement specification or changes in project circumstances, or dealing with the ubiquitous risks, proper "target granularity" is needed to control the process. Any technical means and methods should subject to the overall process of the team, rather than solitarily develop and achieve their own process and quality goals. For example, the demand-acquiring task, in many cases cannot be done directly. The process of the project is always accompanied by large or small changes of the objective. Many contracts are signed even when the objective of the project is no very clear. Such a concept of management to control the project process with target granularity is the most significant difference from other engineering projects.

3.2 The demand management task

In the past, people used to use the phrase "demand analysis" to refer to what is currently called "requirement engineering" or "requirement management". Although the objective and the meaning have hardly changed, the alternation of the terminology shows the growing emphasis on the fact that this work is not only the need of the technical nature, but also an engineering property requiring control and management. Literally, "requirement engineering" should include obtaining, analyzing, defining, verifying and managing of requirements, while "requirement management" stresses on the planning and controlling of everything concerning the requirement engineering. The word "management" is more suitable for describing all activities and regulations concerned, drawing people's attention on tracing requirement alternation

and keeping the consensus of the stakeholders and the project team.

Considering the regularity and technical nature, this task is undoubtedly one of the primary tasks of a software process. From the view of project management, another significance of this basic task is that, it timely captures the necessary terms for negotiation as soon as the project agreement changes: the reason, the content, how it changes and how the business cooperation clause will alter, etc.

3.3 The system verification basic task

The basic mission of the system verification task is the feasibility evaluation on the aspects of the plan, benefit, risk, operation and laws, etc, of the project, and providing necessary technical plans and expectations on management target for the "plan implementation task". It is raised for the targets needing detailed argumentation, after the progress of the requirement management task, and is especially important to projects with fewer experiences. Its result will become the issue of basic problem decision-making task again—whether and how to continue this project.

3.4 The basic implementation plan task

The plan implementation task is a procedure for the implementation level executing the development decisions made by the decision level. It draws up and implements the working plan according to the research result of the prior requirement management task and system verification task. It is mainly to promote, trace and control, and to concern the usage of resources and time, as well as the quality and risks, making full use of the good experiences from all aspects of the team (just as CMM encourages).

3.5 The basic quality evaluation task

The quality evaluation task mainly consists of the evaluation of the specification instructions, schematic design plan, testing plan and coding quality, as well as the supervision of the work consignment. Just as the CMM theory shows, whether and how the quality evaluation task launches is a symbol of the maturity of a software team. It defines the applicable quality standards for the team, and manages the quality assurance activities. Such a regular technical system should be included in a mature software team.

3.6 The project consignment task

The project consignment task is primarily about finishing the test on software installation and distribution, designing the setup package, modifying the setup package after the changes in the consigned software, arranging the personnel and their responsibilities for consigning the project, and implementing the consignment plan, making sure that the consignment is finished successfully.

3.7 The relationship of the 6 basic tasks

The relationship of the 6 basic tasks is shown in figure 4. The result of a task is directive and supportive to other tasks in the same "scope", and the results of other tasks are complementary and illustrative, or corrective under approval to this task.

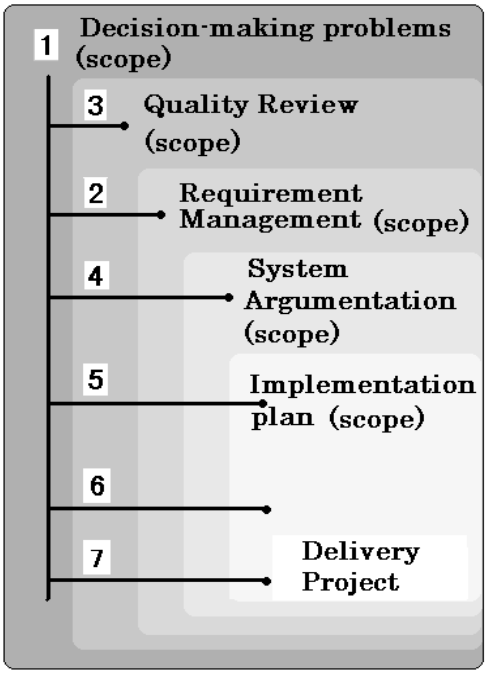

Fig. 5 The Relationships in 6 Basic Tasks

The starting order, the number of times and the project scale are related to the complexity, as well as the management standard and the overall quality of the team; the more basic tasks operate synchronously, the more complex the management tasks will be. For a normal project, a decision-making task (1) is raised in the decision level. Afterwards, the decision level of the project manages and controls the start and the termination of other basic tasks and, in a proper time, launches the requirement management task (2), the quality evaluation task (3) and the system verification task (4), which all report back to the decision level. After making the decision to implement the project and carrying out a plan, the decision level starts the plan implementation task (5), taking the project into

the development stage. In the plan implementation task, it is also necessary to launch sub-tasks (6) of the requirement management task, the quality evaluation task and the system verification task. But in this stage, the goal and the detail of each sub-task is focused on the sectional objective. Works are at first reported back to the implementation level, and then to the decision level. Received the report, the decision level decides how and in what range a new basic task needs to be launched. Thus cycling till the project ends. Finally, the decision level decides to start the project consignment task. (7)

An experienced team must have procedure regulations and quality regulations of requirement management, which should be consciously followed in each requirement management task. So, essentially, the quality evaluation task and the requirement management task are launched at the same time. It is just for clarity to remark them successively in the above illustration.

4 Conclusion

Through reviewing the generation and development of the traditional software process models, this article analyzed the features of the traditional software process models when facing the technical tasks. It thought that the process models didn't summarize the substitutive characteristics of "software process" as the prototype object. These models put the substitutive characteristics of software process as management nature; on the other hand, through the analysis on the general problems in the software enterprises in CMMI application, this article further proposed the question of "in which situation does the software team possess the conditions of applying CMMI", through the analysis on the two major factors influencing CMMI application in software organizations-market price factor of software products and organization and framework factor of software organization", it gave an information communication model (Fig. 2) of software organizational model and software process in software management problem from general aspect, moreover, the writer thinks that this model is a vital information communication process for dealing with and solving all kinds of business and technical problems in the production process of software products, having the 5 features of not being influenced of software organizational structure and scale, independent of products variety, can be taken as the foundation for individualized information communication models of different software organizations to expand, subdivide and establish and providing a natural foundation for the gradual application of CMMI to improve the process management foundation, etc.

On this foundation, this article extended the technical and management conception of software engineering. From a management point of view, it summarized the basic tasks of a software process, based on management ideas such as "management is coordination" and "coordination depends on effective communication". It compares the information communication model and the cyclone model, and brings forward a descriptive model for Team Software Process, including 4 management levels and 6 basic tasks—the writer called it as "cyclone model". It illustrates the series of tasks that form the software process and their relationship with team management levels, which may direct the organization structure and project operation of software team, bringing a new thinking to the theoretical research and elaboration of the software engineering subject and providing a new framework concept for the assistant tools for the research of software engineering.

References:

[1] Zhengnong Xia, *CIHAI*, Shanghai Cihai Publishing House, 2000.

[2][3][4] Roger S. Pressmam, *Software Engineering—Research Method of Practitioner* (translated by Renjie Zheng, Suxia Ma and Bai Xiaoying etc), China Machine Press, 2007.

[5] Joel Henry, *Software Project Management* (translated by Yuchi Liu and Wei Li etc.), China Electric Power Press, 2006.

[6] Shisen Zhang, *Harvard Business School the Theory of General Manager*, China Financial & Economic Publishing House, 2002.

[7] Sami Zahran, *Software Process Improvement* (translated by Xin Chen and Jinfeng Luo etc.), Beijing, China Machine Press, 2002.

[8] Freedman and Weinberg, *Walkthrough, Inspection and Technical Reviews: Evaluating programs*, Beijing , Tsinghua University Press, 2003.

[9] Marshall Brain and Craig C. Freudenrich, Secret of Hurricane, http://science.bowenwang.com.cn/hurricane.htm