A Novel Hurricane Model for Software Engineering

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 are almost based on description of the relationship between the scale, complexity of software project and technology tasks in every phases and methods as the main content. They can open out the characteristic of the technique works, but ignore the connotation of the management in software process. Based on such models, it will brings us lots of inconvenience and trouble for studying and opening out the characteristic of management in software development. In this paper, we propose a novel process model to Team Software Process. The model includes 6 basic tasks and 4 administrative levels, we call it hurricane model. The hurricane model describes relationship between basic tasks and administrative levels in a team, and can supervise all key aspects of the project operations for team software. It also provides a new solution for describing, theoretical research on the software engineering subject and the development of assistant tools.

Key-Words: Software Engineering, Hurricane Model

1 Introduction

Because of the fact that a function can be implemented by various programming language with different algorithms, so in early times, people consider program design as a very personalized creation. But the need of developing large-scale programs grows day by day, which forces people to work in teams to improve efficiency and quality, therefore, people come 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 realize that writing a program in a team is no longer a personalized creation, thus starting to find a way to apply systematic, standardized and quantifiable method to software development, operation and maintenance. It is observed 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.

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.

Over the past decades, the development of the CMM theory has gone 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.

2 The Hurricane Model

2.1 Analysis on the information communication model of a software process

2.1.1 Information communication model
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” [2], 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.

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. The following figure illustrates the information communication model of a software process.

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. 1 Information Communication Model in a Software Process

2.1.2 Definition of management levels
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 to do the coding job within the corresponding design plan.

Figure 1 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.

2.2 The definition of the Hurricane Model

As a comparison, we have the “Hurricane Principle” summed up into Figure 2: 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. 2 The hurricane Model

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 Hurricane Model of a Team.
Software Process. The detailed description is as follows.

![Cyclone Volution Diagram](image)

Fig. 3 Cyclone Volution Diagram

Normally, a TSP has six basic tasks:

2.2.1 The 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 be 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.

2.2.2 The requirement 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.

2.2.3 The system verification 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.

2.2.4 The plan implementation 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).

2.2.5 The 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.

2.2.6 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.

2.3 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. 4 The Relationships in 6 Basic Tasks](image)

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.

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.

3 Conclusion
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 hurricane model, and brings forward a descriptive model for Team Software Process, including 4 management levels and 6 basic tasks. It illustrates the series of tasks that form the software process and their relationship with team management levels, which may direct the operation of team software projects, bringing a new thinking to the theoretical research and the development of assistance tools.

References: