Coordinate Metrics and Process Model to Manage Software Project Risk

Guoping Jiang, Yingwu Chen
School of Information Engineering and Management, National University of Defense Technology, China

Abstract—**Purpose of this paper is to analyze and evaluate risks in software development process quantitatively; evaluate risks influence upon project's object, and assist management decision based on risks. The primary content of this paper is to discuss the software metrics space which support software process modeling and risk quantitative analysis, research modeling method for software process and discuss risk management method under the risk management framework of coordinating software metrics and process model. The authors bring forward software project risk management framework coordinating software metrics and process model, present a new software metrics hypercube space, construct software process two level hybrid model based on general stochastic Petri net, and suggest risk management technique coordinating software metrics and software process model.**

Keywords—**Software metrics, Software process model, Software project risk, Petri net**

I. INTRODUCTION

The main risk components listed by Janne Ropponen ([1]) involves scheduling and timing risk, system functionality risk, subcontracting risk, requirement management risk, resource usage and performance risk, and personnel management risk. Exclude subcontracting risk, the left four are all related with the whole software process or process elements. Boehm ([2]) also listed top five risks as personnel shortfalls, unrealistic schedules and budget, developing the wrong software functions, developing the wrong user interface and "gold plating". Exclude the second item, others are all process related. Risks that are related to the overall process, process elements and their interaction are called process-related risk ([3]). Adapting process analysis technique to identify and analysis process-related risks is referred as process-based risk assessment. Based on the results of risk identification and analysis, simulating software process model to plan risk management, track and control risks is referred as process-based risk management.

Software process model is the basis of process-based risk management. Software process model can provide the necessary key information for risk assessment, and possesses more powerful expression capability than traditional risk analysis methods. But process model provides only information about process flow. As to the risk management, detail information of process artifacts and activities is greatly necessary. Software metrics is just the activity satisfying this demand. So coordinating software metrics and process model is an applicable method for risk management.

In the next section, software risk management framework coordinating process model and software metrics will be put forward. In section 3, a new software metrics space which support risk management and process modeling will be introduced and the difference among software organizations which is at different CMM maturity levels will be discussed. In section 4, software process two-level hybrid simulation model will be introduced. In section 5, software risk management technique coordinating software process and software metrics will be suggested. In the last section, a brief conclusion of this paper and the future work in this research domain will be show.

II. METHODOLOGY FRAMEWORK OF SOFTWARE PROJECT RISK MANAGEMENT COORDINATING PROCESS MODEL AND METRICS

The framework of software project risk management coordinating software metrics and software process model includes factors as follows:

- Data source: software metrics
- Models: software process simulation model, estimation model, quality assessment model, reliability growth model, risk estimation model, and etc
- Methods: risk identification method based on software process model analysis, risk assessment method based on software process simulation, risk decision method based on software process simulation, and etc.

The factors and the relationship among them are depicted in Fig.1. Based on the real software process and software organization, collect associated software metrics, then applying estimation model to predict time and cost for the entire project and every activity in process. As to the probability of activity's complication in predicted time, is estimated by integrating causal relationship among risk factors in the process to construct Bayesian network. Based on the work breakdown structure of software project and above estimation results construct process simulation model. Identify potential risk scenarios by qualitative analysis of the process model, and assess risk influence by simulating the process model, also carry out project alternative selecting and decision support.

0-7803-8519-5/04/$20.00 © 2004 IEEE

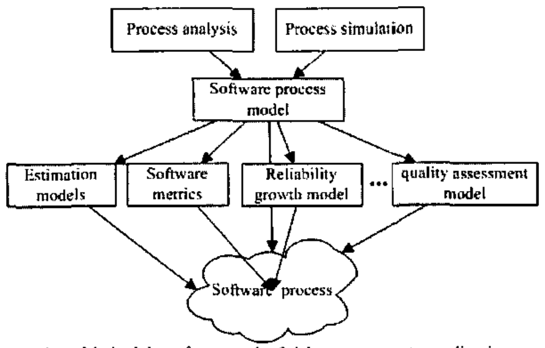

Fig.1. Methodology framework of risk management coordinating software metrics and software process model

III. SOFTWRE METRICS PARADIGM SUPPORTING RISK MANAGEMENT

Software process refers to a set of partially ordered process steps, with a set of related artifacts, human and computerized resources, organizational structures and constraints, intended to produce and maintain the requested software deliverables ([4]). Among software process, system analysis and definition, software development and system operation, and maintenance are experienced. And software development process can be partitioned as requirement analysis, outline design, detailed design, coding and unit test, assembling system test and delivering such six phases.

We treat software process as software process units flow united in series, parallel or arbitrary combinational order. Software process unit is depicted in Fig. 2. There are three entities in software process, as artifact, activity (or process) and execution agent. Artifact is the input and output of activity, activity's necessary or result, transition between artifacts is activity.

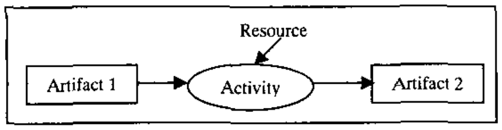

Fig. 2. Software process unit

Software artifact refers to the semi finished product in software process and the software system that delivered at the end. Software artifact exhibits various form. They may be documents of software process, training data, coding, or product specifications. Software metrics are used to describe software artifacts' attributes. For example, the relative attributes of requirement specification are volatility, integrality, veracity, consistency and etc. Software metrics can support risk quantitative analysis and management decision, and artifacts' attributes also decide the continued process even the whole project in some degree.

Based on the above description of software process, we bring forward software hypercube metrics space. Except three dimensions characterizing the three entities in software process, there is another dimension named as project dimension to characterize global information about project, organization, custom and environment. So there are four dimensions in the software metrics space, as depicted in Fig.3: project dimension, activity dimension, resource dimension and artifact dimension. Activity dimension includes milestone, scheduling, cost, fault introduced, fault eliminated and much more information about activities' attributes of time, cost, quality and etc. Resource dimension describe resource attributes, constraints, and resource collocation. As to software project, resource mainly refers to personnel resource. Artifact dimension is the metrics aimed at every artifact in the process including documentation, design specification, code, test plan, and etc.

So artifact dimension characterizes process artifacts' attributes. Activity connects artifacts, so activity dimension metrics depict attributes of process activities. Resource is consumed in process, so resource dimension metrics describe resource attributes and constraints. Project dimension is the common attributes of the whole project, and restrict all the artifacts and the whole process.

The hypercube software metrics space has the advantages of strong operational capability, supporting risk holistic and dynamic management, supporting risk-based management decision, integrating personnel factor and supporting software process discrete event modeling.

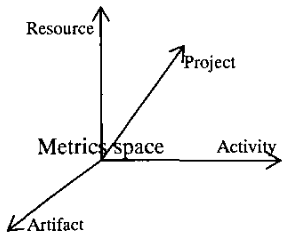

Fig. 3. Hypercube metrics space

Software organization can only measure what they can observer. As to different maturity level, securable metrics is different. The CMM is composed of five maturity levels: initial, repeatable, defined, managed and optimizing ([5]).

At the initial level, the organization typically does not provide a stable environment for developing and maintaining software. During a crisis, projects typically abandon planned procedures and revert to coding and testing. Schedules, budgets, functionality, and product quality are generally unpredictable. In a word, the software process capability of Level 1 organizations is unpredictable. So software metrics in those organizations have only two dimensions: project dimension and resource dimension.

At the repeatable level, the summarized character of the organization is to obey the regulations. Project is typically developed based on the performance of previous projects or experience. So they can not measure the activity metrics, and in artifact dimension, they can only measure the delivering product not the semi finished products. Software metrics in CMM 2 organization have project dimension, resource dimension and part of artifact dimension.

At the defined level, software process capability of the organization is standard and consistent. Within established product lines, cost, schedule, and functionality are under control, and software quality is tracked. Software engineering and management activities are stable and repeatable. So the organizations manage the whole software process effectively. They can measure all the four dimensions metrics.

At the managed level, the summarized character of the organization is predictable. Software metrics repertory is built in the organization and metrics data is collected and analyzed. There are well defined and consistent metrics for software process. So besides the four dimensions metrics, those organizations pay more attention to relationship measurement among four dimensions.

At the optimizing level, the entire organization is focused on continuous process improvement. The most outstanding character of software process capability is continued improvement. So organizations in the optimized level consider information feedback and management activities' improvement and innovations besides those metrics that organization at the managed level do.

The main difference of software metrics space between organizations in different maturity level is showed mainly in the activity dimension. Higher maturity level of the organization, more detailed division of software process, and measurement for each activity and artifact is more detailed; moreover, information feedback is measured to improve management.

IV. HYBRID SOFTWARE PROCESS MODEL

Software process modeling is very complicated for the process system is uncertain, stochastic and dynamic, and feedback mechanism existed. So software process model must have the capability to exhibit uncertainty and stochastic property, deal with dynamic behavior and provide feedback mechanism.

Nowadays, there are three main types of software process model: discrete event simulation model, system dynamic model and state-based model. The discrete model can describe process activities, event queue and process offsets based on entities' attributes, and provides information about time and product in the product line, thus is very favorable to describe software process. System dynamic model can only depict problem on project level, and is inapplicable to software risk management. We choose discrete event model to describe software process. At the same time, we consider of constructing hybrid simulation model to make up discrete model's limitations that it can not record changed factors' influence and can not provide feedback mechanism. At the high level, discrete model is used to depict software process flow; while at the low level, some continuous models, analytic models or experience functions existed to provide detail information about process entities. The low level models can reflect change factors' influence, and embody historic experience to make up the discrete model's shortcoming of feedback mechanism.

In our study, at the high level of software process model, hierarchy general stochastic Petri net is adopted to describe software process flow, for Petri nets' inherent characters. The graphical representation for Petri nets makes them simple, clear and easily understood. There are rigorous mathematic foundations for Petri nets. And there are many mature mathematic analysis methods for Petri nets and Petri nets simulation is more simple and direct than others. The hybrid software process model is depicted in Fig.4.

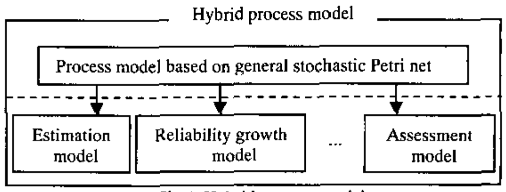

Fig.4. Hybrid process model

There are several Petri net based software process models in software engineering literature ([6], [7] and [8]). They are most based on high Petri nets and their extensions such as Predicate/translate nets (Pr/t nets) and Environment/Relationship nets (ER nets). Every model has some advantages and behaves well in certain application. The common limitations of those models are that they can not depict process uncertainty and can not express software metrics' assistant function for software management decision, thus they can not support risk analysis and assessment well.

Software process is a very complicated system, basic Petri net can only describe software process flow, and can not exhibit artifacts' and activities' attributes and uncertainty, so a metrics table is attached to each artifact to show the associated attributes. There are human participations in software process, so activities in software process are not always occurred once qualified. Besides duration time of the activity, there may be time delay for start-up and also activity's occurrence or accomplishing may be probabilistic. In other word, there are three types of transitions: prompt transition, fixed delay transition and probabilistic transition. So we model software process based on general stochastic Petri net. General stochastic Petri net can also embody scheduling and cost uncertainty. So, in our study, general stochastic Petri net is used to describe software process, artifacts and resource in project process are spaces in Petri nets and activities are transitions. Modeling such a complicated system as software process, the idea of clearly depicting the entire model in one level is hard and obscure, so based on the work breakdown structure, modeling sub-processes as transitions embodying sub-nets. Summarize the above statement, we propose a Petri-net software process model, namely software process net (SPNet), that is based on hierarchy general stochastic Petri net with metrics table attached to every space.

Definition 1: Software Process Net (SPNet)

$(P,T;F,B,M_0)$ is a Software Process Net (SPNet), if and only if:

1. $(P,T;F)$, a basic Petri net
2. P is a set of artifacts and resource in project process
3. T is a set of activities in the process. $T = Ta + Td + Tr + Ts$, Ta is the set of prompt activities, Td is the set of fixed delay activities, Tr is the set of stochastic activities, and Ts is the set of activities that represent subnet
4. F is a flow relation $F \subseteq (P \times T) \cup (T \times P)$ for the set of arcs
5. B is a set of metrics tables attached to each space
6. M_0 is a set of initial mark of the net system.

Low level models in hybrid software process model include estimation models, reliability growth models, assessment models, and etc. They provide detailed information for project benchmark, artifacts and activities.

V. Risk Managenet Coordinating Metrics and Process Model

Barry Boehm considered ([2]) that the general procedure of software project risk management is composed of two main steps, and each divided into three sub-steps:

- risk assessment, consisting of
  1) Identification of those risks likely to cause problems
  2) Analysis to determine the loss probability and loss magnitude for each risk and to develop a compound risk
  3) Prioritization to rank the identified risk items according to their compound risks
- risk control, consisting of
  4) management planning to bring the risk items under control
  5) Resolution to eliminate or resolve the risk items
  6) Monitoring to track the project's risk reduction progress and to apply corrective action where necessary.

Coordinating software metrics and software process model can support the risk management entire procedure. First, we explain some terms which will be used in the following text. Predicted project results are the project's schedule, cost and quality planed at the beginning. Simulated project results are the schedule, cost and system quality obtained by simulating based on project's current state. The difference between these two does not mean that whether the project is out of control or not. Only combing stakeholders' or managers' risk tolerance then you can judge whether the project is under control or not.

Software metrics provide up-to-date data of software process, and process model depicts the entire procedure flow. Coordinating both, construct Petri net's reachability graph by generic analysis method of general stochastic Petri net to obtain all the reachable states or paths of the software process. Contrasting the reachability graph with risks list can identify potential risk scenarios.

There are two approaches for analyzing and evaluating risk coordinating process model and software metrics. One is to utilize models at the low level of software process hybrid model. Based on project actual state (behave as metrics) and causal relationship between risk and risk factors, construct risk Bayesian network, thus to estimate risk occurrence probability and most probable risk scenario by probability inference. Another is to carry out process model simulation to get risk occurrence probability and simulated project results. It is the manager who decides whether to take measures to control risk after comparing simulated results with predicated project results.

Risk plan is the precondition of risk track and control. To establish risk plan, apply every alternative to the process model, and perform simulation to get simulated project result, and then select the alternative which has optimum result as the risk plan.

Risk control is a multistage, iterative dynamic procedure. According to the risk plan, take measures in the process and assess the effect sequentially. Based on the difference between simulated project result and predicted result to judge whether the project is under control or out of control, then decided to re-plan or follow the fixed risk measures.

VI. Conclusion and Future Work

Software metrics can provide associated information about artifacts and activities in software process, while software process model provide key necessary information for risk assessment by process model analysis and simulation. So, Coordinating model software metrics and software process can carry through quantitative analysis and evaluation for software process related risks. The framework of coordinating software metrics and software process model to manage software project risks is presented in this paper. Hypercube software metrics space is put forward to support process modeling and risk management, and the difference among organizations at different maturity level is discussed. Hybrid process model based on discrete event simulation model and continued, analytical models is built, and SPNet based on general stochastic Petri net is defined. In the end, methods of risk analysis, evaluation, control and decision are discussed coordinating software metrics and software process simulation model.

The future work, we plan to study qualitative analysis and quantitative simulation methods for software process model SPNet; establish actual and feasible software process net for some software organization, then carry out risk management activities and investigate new problems emergent in practice.

REFERENCES

[1] Janne Ropponen and Halle Lyytinen, "Component of software development risk: how to address them? A project manager survey (Periodical style)," *IEEE Transactions on Software Engineering*. vol.26, no.2, February 2000.

[2] B. W. Boehm, "Software risk management: principle and practices (Periodical style)," *IEEE Software*. vol. 8, no. 1, pp. 32-41, Jan. 1991.

[3] Alfred Bröckers, "Process-based software risk assessment (Published Conference Proceedings style)," in *the Proceedings of the Fourth European Workshop on Software Process Technology*, Apr. 1995.

[4] Jaques Lonchamp, "A structured conceptual and terminological framework for software process engineering (Published Conference Proceedings style)," In *the Proceedings of the Second International Conference on the Software Process*, pp. 41–53.

[5] Mark C. Paulk, Charles V. Weber, Suzanne M. Garcia, Mary Beth Chrissis, and Marilyn Bush, "Key practices of the Capability Maturity Model^{SM}, Version 1.1(Report style)," Technical Report, CMU/SEI-93-TR-025, ESC-TR-93-178, Feb. 1993.

[6] Sang-Yoon Min and Doo-Hwan Bae, "MAM nets: A Petri-net based approach to software process Modeling, Analysis and Management (Published Conference Proceedings style)," In *the Proceedings of International Conference of Software Engineering and Knowledge Engineering (SEKE'97)*, pp. 76-87.

[7] Wolfgang Emmerich and Volker Gruhn, "FUNSOFT Nets: a Petri-Net based software process modeling language (Published Conference Proceedings style)," in *the proceedings of the sixth international workshop on software specification and design*, pp. 175- 184.

[8] Sergio Bandinelli and Alfonso Fuggetta, "Computational Reflection in Software Process: the SLANG Approach (Published Conference Proceedings style)," in *the proceedings of the 15th international conference on software engineering*, Baltimore, Maryland, 1993.

[9] Raffo D. M., W. Harrison, and J. Vandeville, "Coordinating models and metrics to manage software projects (Periodical style)," *International Journal of Software Process Improvement and Practice*, vol.5, pp. 159-168, 2000.

[10] Paolo Donzelli and Giuseppe Iazeolla, "A hybrid simulation modeling of the software process (Periodical style)," *Journal of Systems and Software*, vol.59, pp. 227-235, 2001.

[11] Girault and Valk, *Petri nets for system engineering- A guide to modeling, verification, and applications* (Book style). Springer-Verlag, 2003.