A Risk Control Optimization Model for Software Project

Ping Cao, FuJi Chen
College of Public Administration, Fuzhou University
Fuzhou, China
caoping330@sina.com, chenfuji@fzu.edu.cn

***Abstract*—Risk Control is of great importance for software project management. However, the software risk control currently depends on the manager's subjective judgments, and lacks quantitative supporting tools. In this paper a software risk control optimization model is presented. By forming project strategy and increasing activities' costs, the objective of maximizing the expected value of project's capital investment is achieved. Moreover, because the model is a NLP, a particle swarm algorithm approach is proposed. Finally an example is utilized to verify the effectiveness of this method. The research provides strong supports for effectively managing and quantitatively controlling software risk, control activities And reduce the probability of failure project, maximize the project expected revenue.**

Keywords-software project risk; risk control; expected revenue; Optimization; particle swarm algorithm

I. INTRODUCTION

The management of software risk has become a heat point of the software engineering area at the moment. It first came to the forefront with Boehm's tutorial on risk management [1]. Software projects are high risk activities, generating variable performance outcomes [2]. Industry surveys suggest that only about a quarter of software projects succeed outright and billions of dollars are lost annually through project failures or projects that do not deliver promised benefits [3]. As in any other business, software development organizations try to maximize their profits and minimize their risks. The risks represent uncertain events and conditions that may prevent enterprises from attaining their goals. Failing to understand and manage software project risk can lead to a variety of problems including cost and schedule overruns, unmet user requirements, and the production of systems that are not used or do not deliver business value. Unfortunately, research also finds that risk management is not always well-applied in practice [4], the risk knowledge area had the lowest maturity of all knowledge areas in the IS industry, and the risk maturity level. Ropponen found that 75% of project managers did not follow any detailed risk management approach, and only vaguely understood the software risk concept and its managerial implications [5].

Choosing the valid process control to the software process is the key that the software project is successful. The kernel of risk management is to identify and analyze the software risk factors that might cause project failed at the planning stage, and then to take positive and effective risk prevention and control actions to dissolve great losses that the software project may result in. Risk management is considered first among the best practices for managing software projects, especially for managing large software projects [6]. In this paper, we present an improved software risk management model with a detailed description of its main components.

The paper is structured as follows: In the next section, the literature is reviewed and reassessed. Following this, the empirical study is described and the main findings are presented before conclusions are reached on the comparison of practices with the prescriptions in the research literature. Finally, limitations of the study and implications for research and practice are discussed and conclusions are drawn on directions for future research and practice.

II. RISK AND PROCESS OPTIMIZATION

Being contrasted against the ordinary project, the software projects have the greater uncertainty, As organizations continue to invest time and resources in strategically important software projects, managing the risk associated with them becomes a critical area of concern. Advocates of IS risk management argue that identifying and analyzing threats to success allows actions to be taken to reduce the chance of failure. Articles have stressed the importance of empirically categorizing the sources and types of risks associated with software development projects [7]. Unfortunately, despite these recommendations there are relatively few tools available to help project managers identify and categorize risk factors in order to develop effective strategies.

A. Software risk and risk management

Risk in software projects is the uncertainty that exists in the process of development of software such as great quantity demands, technique, personnel staff, process and organization and so on. And this may cause that production and service of software unable to meet the demand and expense go beyond that budget and rate of advance delay either the project is forced to call off and these things which are not expected [8]. Usually defined as the probability-weighted impact of an event on a project [9].software projects have long been recognized as high-risk ventures prone to failure [10].

Software project risk management is usually defined as a set of principles and practices aimed at identifying, analyzing and handling risk factors to improve the chances of achieving a successful project outcome and/or avoid project failure.

Project-related risk management has attracted a steady stream of interest in the academic literature, practice-based methods (e.g., CMMI and PRINCE2), and standards (e.g., PMBOK and AS/NZS).Now, the management of software risk

978-1-4244-4507-3/09/$25.00 ©2009 IEEE

of CMM is used by more and more software business companies. Its feature is that risk management and the software process exists in the same frame. But the development of risk and risk management in the research and practice literature lags the requirements of the threat phenomenon in practice, and the adoption of risk and risk management concepts and methods in practice lags the understanding and prescriptions found in the literature. Hence there is a need for better risk management in research and practice.

B. optimization of software process

The handle of the risk, chiefly is the target to draw up risk control, tactics, means along with ought to be to every one significant risk scheme, core idea is that administrator can identify and analyze the risk that probably causes the software project defeated, and takes valid action to avoid it, and initiatively dissolves the great loss that the solution software project was probably come into being . then on the basis of the risk plan to adjust the scheme and dominates the risk limit that administrator can adopt it.

Industry survey data suggests that while there has been some improvement in project success rates, software projects are still more likely to fail on some key performance criterion than succeed outright.

III. OPTIMIZATION OF SOFTWARE RISK CONTROL

A. Model of optimization

To a software project, from systematic analysis, design, coding, implementation to the system testing, serial and parallel two kinds of most basic transmission relationships, the relations between all tasks can be classified into three kinds of situations: serial, parallel and mixed of above two.

Assuming that different Allocation of funds will cause different software project risk, when the project risk exceeds the level of an acceptable risk, in order to drop the risk of the project to the range that can be accepted and Increase the returns of the project, the Project manager can append funds to guard against risks. Following we give the risk control optimization model under the given resources. This model is described in detail in the following sections.

Model assumptions are as follows:

1) The project possess n parallel missions, the scheme of every mission construction period is t_i and independent event.

2) With the input of mission i is increased, the success probability of mission i increased.

Risk often is interpreted as the possibility of defeat. First of all, give out the definitions of mission success and project success. Events are defined as follows:

Let us denote A as a project success, and mission success A_i is mission i in the project can be completed on time. Then $P\{A\}$ is the probability of the project success, $P\{\overline{A}\}$ is the probability that the project can't be completed on time. According the definition of risk, the following formula can be obtained: $A = \bigcap_{i=1}^{n} A_i$

A project including n missions, and these missions will not influence each other in the process of implementing, then the events A1, A2, ..., An is an independent incident group.

Theorem: if A1, A2, ..., An is an independent group, the probability of success of the project are:

$$P\{A\} = \prod_{i=1}^{n} P\{A_i\} \tag{1}$$

Proof: according to the independent events multiplication theorem $P\{A\} = P\{\bigcap_{i=1}^{n} A_i\} = \prod_{i=1}^{n} P\{A_i\}$, i = 1, ⋯ , n

Usually, the successful probability of mission i is proportional to the investment of the mission ,

$$P\{A\} = P\{x_i\} \tag{2}$$

Where $P\{x_i\}$ as the successful probability of mission i when applying risk control investment x_i , and it is non-decreasing function of x_i .

So Project manager can adjust the task schedule risk by reallocation of the investment on the task.

As the risk of mission *i* is changed, the risk of the whole project will be changed correspondingly. In this paper, the Project manager enhances the successful probability of project and by increasing investment on one mission. Although increasing the investment can reduce the risk of project failure, the returns will decline at the same time. The goal of Project manager is to maximize returns of the project and control the probability of project failure at the range of (0, R) by increasing investment.

Before software development, Software development contractor signed a contract with the customer, including the total price (P), delivery time (T), and the amount of compensation (V), i.e. if the project is successful, the customer will pay P to the contractor; Otherwise, the contractor will compensate the customer by V.

The expected return E(x) of the whole project will be expressed as following:

$$E(X) = P\{A\} * P - P\{\overline{A}\} * V - \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \beta(t_i - T) \tag{3}$$

According to (1) and (2)：

$$\begin{aligned} E(X) &= \prod_{i=1}^{n} P\{A_i\} * P - (1 - P\{A\}) * V - \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \beta(t_i - T) \\ &= \prod_{i=1}^{n} P\{x_i\} * P - V + \prod_{i=1}^{n} P\{x_i\} * V - \\ &\quad \sum_{i=1}^{n} x_i - \sum_{i=}^{n} \beta(t_i - T) \end{aligned}$$

$$= \prod_{i=1}^{n} P\{x_i\}(P+V) - V - \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \beta(t_i - T) \quad (4)$$

We assume the maximum amount of the total risk control investment on a project is U_{max}, B as The contractor's minimum rate of profit.

$$U_{\max} = P - PB - \sum_{i=1}^{n} x_i \quad (5)$$

a_i as the risk control investment :

$$\sum_{i=1}^{n} a_i \le U_{\max} = P - PB - \sum_{i=1}^{n} x_i$$

Therefore, the optimization of expected return of the project for Software development contractor will be expressed as following:

$$\max \quad E(X)$$

$$E(x) = \prod_{i=1}^{n} p(x_i + a_i)(P+V) - V - \sum_{i=1}^{n} (x_i + a_i) - \sum_{i=1}^{n} \beta(t_i - T) \quad (6)$$

$$s.t. \qquad \prod_{i=1}^{n} p(x_i + a_i) \ge (1 - \mathrm{R})$$

$$0 \le \sum_{i=1}^{n} (x_i + a_i) \le P(1 - B)$$

$$u_i^L \le x_i + a_i \le u_i^H, \mathrm{i} = 1,2,\cdots,\mathrm{n}$$

$P(1-B) - \sum_{i=1}^{n} x_i = 0$ means the allocation of the project resource under the original project planning condition, that is the whole project risk control investment is zero.

IV. Algorithm Design

The previously described model is a complicated nonlinear programming model; in this paper a Particle swarm optimization algorithm is proposed to solve it.

Particle swarm optimization algorithm is one of the latest evolutionary optimization methods have recently been successfully applied to several NP-hard combinatorial optimization problems, which is applicable to continuous systems. In PSO a population of solutions, called particles, is determined. A swarm of these particles moves through the search space to find an optimal position. The movement of a particle is influenced by the particle's personal search history and by the best positions that have been found by other neighbored particles in the swarm.

Let us denote x and v as a particle coordinates (position) and its corresponding flight speed (velocity) in a search space, respectively. The best previous position of a particle is recorded and represented as *pBest*. The index of the best particle among all the particles in the group is represented as *gBest*. To ensure convergence Of PSO, Eberhart indicates that use of a constriction function may be necessary [11]. At last, the modified velocity and position of each particle can be calculated as shown in (7) and (8):

$$v_{k+1} = c_o v_k + c_1(pbest_k - x_k) + c_2(gbest_k - x_k) \quad (7)$$

$$x_{k+1} = x_k + v_{k+1} \quad (8)$$

Here, c1 and c2 are two positive constants which are called learning factors or rates. C0 is a factor of inertia suggested by Shi and Eberhart. in order to control the impact which the histories of velocities has on the current velocity.

Firstly the inequality constraints as the penalty function o added to the objective function, that is:

$$\min V + \sum_{i=1}^{n} x_i' + \sum_{i=1}^{n} \beta(t_i - T) - \prod_{i=1}^{n} p(x_i')(P+V) + M\{[\min(0, \prod_{i=1}^{n} p(x_i') - 1 + \mathrm{R})]^2 + [\min(0, (\mathrm{p}(1-\mathrm{B}) - \sum_{i=1}^{n} (x_i'_i)))]^2 + [\min(0, \sum_{i=1}^{n} (x_i'_i))]^2 + [\min(0, u_i^H - \mathrm{x_i})]^2 + [\min(0, x_i - \mathrm{u_i^L})]^2\} \quad (9)$$

Where, $x_i' = x_i + a_i$

A schematic representation of the PSO algorithm is shown below:

1) Generate a family of *N* random solutions (particles).
2) Search for the best particle.
3) REPEAT the following block until the condition terminates:

LOOP *i* = 1, … , *N*:

- BEGIN

Calculate the value of the objective function for particle *i* .

IF particle *i* gives a better value for the objective function, let particle *i* be the best particle.

Calculate the new velocity for particle *i* using (7).

Calculate the new position for particle *i* using (8).

- END

RETURN the solution of the best particle

The termination condition for the algorithm is when either the maximum number of iterations is reached or the objective function assumes a preset value .

V. A Computation Example

From above we know $P(x_i)$ is non-decreasing function of x_i. We can assume

$$p(x_i) = 1 - \alpha_i e^{-\beta_i x_i}$$

Function curve as shown in fig. 1.

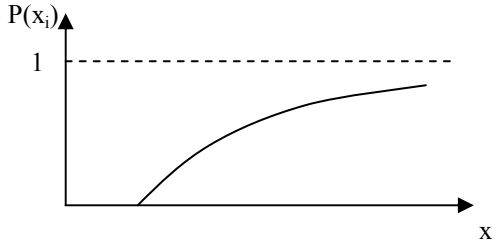

Figure 1 Non-decreasing function $p(x_i)$

Parameters as follows:

TABLE 1 PROJECT PARAMETERS

parameters	value
P	$80,000
V	$40,000
T	9month
R	0.15
B	0.2

TABLE 2 MISSION PARAMETERS

mission	t_i/Month	α_i	β_i	x_i($)	u_i^L ($)	u_i^H ($)
1	9.5	1.25	0.90	6000	6000	11,000
2	8.5	1.56	0.26	6000	6000	12,300
3	9	1.32	0.45	6000	6000	10,000
4	11	1.42	0.23	6000	6000	10,800
5	10	1.56	0.75	6000	6000	11,200
6	9	1.22	0.64	6000	6000	13,800

The mathematical models in this study are solved by programming using Matlab7.0. The basic parameters Settings as follows: initialize the number of particles n=6, the number of particle m=50, c_1 =2, c_2=2, c_0=1,v_{max}=0.5.Select different iterations N as comparison to test the algorithm.

In different iterations, test the algorithm 100 times; obtain the optimal solution X*= *(8.4305,11.4788,9.2685, 6.0012, 7.7758,6.9205*), the global-extremum E(x*)=$*18083*. The results are shown in fig. 2.

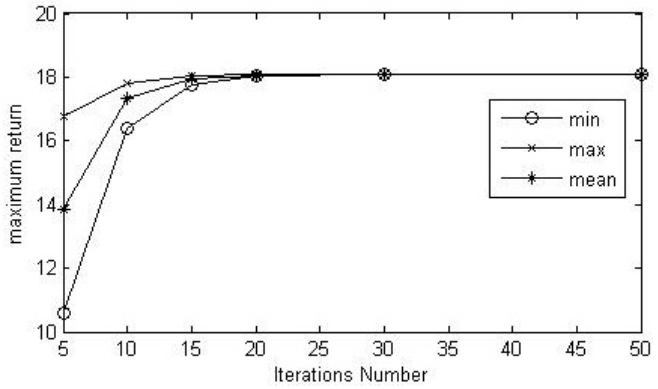

Figure 2 the results of Software project maximize returns

The simulation results show the algorithm has good convergence, and can effectively solve the NLP programming problem.

VI. CONCLUSIONS

Understanding software project risk can help in reducing the incidence of failure. Failing to understand and manage software project risk can lead to a variety of problems including cost and schedule overruns, unmet user requirements, and the production of systems that are not used or do not deliver business value. Being able to complete software projects successfully becomes even more important .This paper has presented a novel approach for optimizing software project process. It is based on project returns of the project. Also it takes the project returns as a criterion to assess the quality of a software project process. A model of optimizing software risk control is given and a optimization algorithm of PSO is proposed in this paper.

The new method of optimizing the software process based on project returns l before the implementation of the process can change the passive control into the active control. It provides managers an effective tool to make the risk control decisions and implement the process optimization at the project planning stage. this new method can greatly promote the possibility of success of software projects.

REFERENCES

[1] Boehm, BW. (1991). Software risk management: Principles and practices. IEEE Software, 8(1), 32–41.

[2] R. N. Charette. “Large-scale project management is risk management”. IEEE Software, pp.110-117, July 1996.

[3] Kwak, Y. H., & Stoddard, J. (2004). Project risk management: Lessons learned from software development environment. Technovation, 24(11), 915–920.

[4] Risk Management Overview. CMMI-SE/SW Version1.1. Copyright 2002 by Carnegie Mellon University,Jan. 2002.

[5] Shenhar, A.J., Dvir, D., 2007. Project management research: the challenge andopportunity. Project Management Journal 38 (2), 93–99.

[6] Ropponen, J., 1999. Risk assessment and management practices in software development. In: Willcocks, L.P., Lester, S. (Eds.), Beyond the IT Productivity Paradox. John Wiley & Sons, Chichester, pp. 247–266.

[7] Charette, R.N., 2005. Why software fails? IEEE Spectrum 42 (9), 42–49.

[8] Johnson, J., Boucher, K.D., Connors, Y., Robinson, J., 2001. Project management: the criteria for success. Software Magazine 21 (1), S3–S11.

[9] Charette R N. Large-scale project management is risk management [J]. IEEE Software, 1996.110-117.

[10] Elaine M. Managing Risk: Method for software systems development. Addisom-Wesley, 1998.

[11] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, Perth, Australia, IEEE Service Center, Piscataway, NJ, 1995.