基于风险控制的软件项目过程优化*)

徐如志　钱乐秋　薛云皎　赵文耘
(复旦大学计算机系软件工程实验室　上海200433)

摘　要　本文提出一种基于风险控制的软件项目过程评估和优化方法，提出以软件项目风险大小作为评估当前软件项目过程优劣的依据，并从优化软件项目风险控制的视角对软件过程进行优化。给出一个基于风险传递的软件风险优化控制模型和一个动态规划的软件风险控制离散优化算法，以及使用上述方法解决问题的一个示例。本文给出的基于风险控制对软件过程进行事先评估和优化的方法，变以往对软件项目的事后被动控制为事先的积极有效预防，从而可显著提高软件项目的成功率。

关键词　软件过程，项目管理，软件风险，风险控制

Optimizing Software Project Process Based on Risk Control

XU Ru-Zhi　QIAN Le-Qiu　XUE Yun-Jiao　ZHAO Wen-Yun
(Laboratory of Software Engineering, Fudan University, Shanghai 200433)

Abstract　This paper presents a new approach for assessing and optimizing software project process based on software risk control, which evaluates and optimizes software project process from the view of controlling the software project risks. A model for optimizing software risk control is given, a discrete optimization algorithm based on dynamic programming is proposed and an example of using above method to solve a problem is also included in this paper. By improving the old passive post-project control into an active effective pre-action, this new method can greatly promote the possibility of success of software projects.

Keywords　Software process, Project management, Software risk, Risk control

1　引言

选择正确的软件过程以及对软件过程进行有效的过程控制是保证软件项目成功的关键[1]。然而在现实的软件工程实践中，过程评价发生在项目部分或全部完成之后，基于已完成项目的实际度量数据进行[2]。因此过程的评价及优化结果只能用于以后软件项目过程控制的依据。显然，这种"亡羊补牢"式的项目过程管理方式存在明显的缺陷。

作为软件工程中相对较新的研究领域[3]，软件风险管理目前已成为软件工程领域的一个研究热点。其核心思想是管理者在软件项目的计划阶段就对可能导致软件项目失败的风险因素进行识别和分析，并采取积极有效的预防和风险控制措施，主动化解软件项目可能产生的重大损失。因此风险管理被认为是管理软件项目特别是管理大型软件项目的最佳实践中最重要的方面[4]。随着CMM(Capability Maturity Model)在软件组织的推广应用，软件过程能力的不断提高，越来越多的软件过程数据及风险管理数据被收集和存放在过程数据库中[5]，这些以往软件项目的历史过程数据和知识为评估和优化控制当前的软件项目过程奠定了基础。

本文以软件项目风险大小作为评估软件项目过程优劣的判据，从而将软件项目的过程评估与优化转变为对软件项目风险的计算和优化控制问题。本文首先简要介绍软件风险管理的相关概念；之后引入软件风险传递算法，给出软件风险优化控制的模型，然后设计一个动态规划的软件风险控制优化算法。

2　软件风险与软件过程评估

2.1　软件风险与软件风险管理

软件风险是指软件开发过程中存在大量的需求、技术、人员、过程、组织等方面的不确定性，可能导致软件产品/服务的功能不能满足要求、费用超出预算、进度延迟或项目被迫取消等所不期望的后果。软件风险管理则是对可能导致上述不利影响的风险因素进行评估和控制，将风险降至管理者可以接受的范围内[6,7]。

软件风险评估包括风险识别和风险分析，主要是识别并记录可能对项目造成不利影响的因素，评估每一个已识别风险发生的概率与后果。在此基础上按风险影响的大小确定风险的优先级，最终形成一个软件风险列表作为软件风险控制的基础。软件风险控制包括风险计划、风险跟踪和风险处理，主要是制定风险控制的目标、策略、方法以及应对每个重要风险的方案，然后根据风险计划跟踪已识别风险的变化情况，及时调整风险应对计划或采取必要的风险处理措施，将风险控制在管理者可以接受的范围内。风险管理过程[7]如图1所示。

当前的软件风险管理主要基于三个基本的方法框架。一是基于COCOMO模型的风险评估；二是SEI的基于CMM/CMMI的风险评估和控制框架；三是根据实践经验而来的Top10风险分析控制方法[8]。第一种方法着重于对软件风险要素的评估，第三种方法则完全基于管理者的经验。这两种方法都是依赖于管理者的主观判断，缺乏客观的数据支持。当前基于CMM的软件风险管理被越来越多的软件企业应用。其

*)基金项目：本研究得到国家高技术研究发展计划(863计划)项目(No. 2002AA114010)；上海市科学技术委员会研究项目(No. 035115026)的资助。徐如志　博士生，高级工程师，主要研究领域为软件复用技术，项目管理。钱乐秋　教授，博士生导师，主要研究领域为基于构件的软件工程，软件过程工程等。

突出特点是风险管理与软件过程工程、过程管理整合在同一个框架中。在CMMI(Capability Maturity Model Integration)中,风险管理作为一个独立的PA(Process Area)得到进一步强化,强调风险管理的持续性,强调对风险管理过程数据的收集和利用,而且过程数据库(含风险数据库)在风险管理中扮演重要角色[5,7,8]。

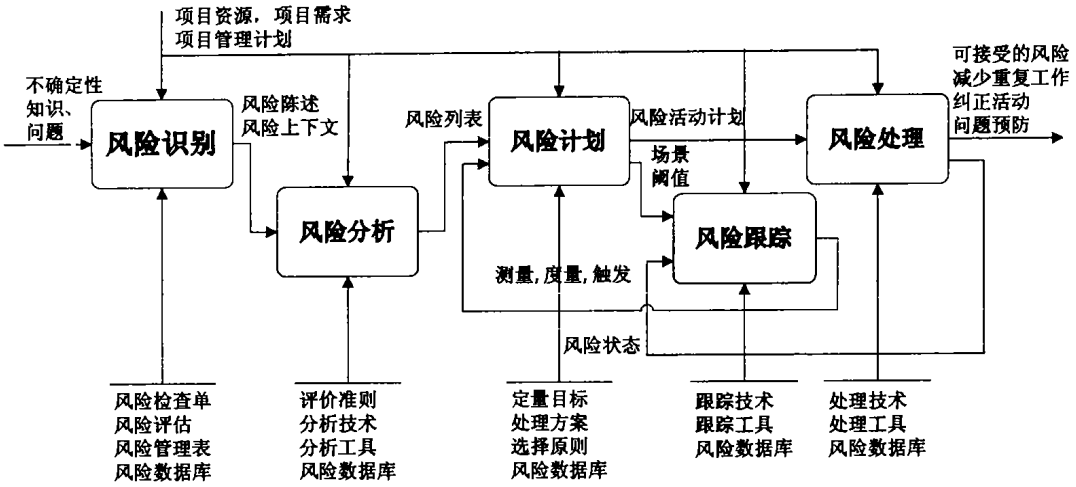

图1 软件风险管理过程

2.2 软件过程评估与软件过程优化

对于一个软件项目,通常在计划阶段利用WBS(Work Breakdown Structure)工具,将其分解为若干个相对对立的任务,然后参照所选择的开发模型按照各个任务之间的时序关系构建一个具有相互串行和并行关系的任务网络图。

在项目资源投入总量确定的前提下,在各个任务之间不同的资源分配方案必然会对项目目标的实现产生影响。显然使项目风险最小的软件过程资源配置方案最为合理,亦即软件项目的过程最优;反之则软件项目的过程较差。因此在项目给定资源投入确定的前提下,评估软件过程的优劣转化为计算和比较不同过程资源分配使用方案所导致的项目风险大小;相应地,对软件过程的优化也转化为对软件风险的优化控制问题。这种基于风险评估和风险控制的软件项目过程评估和过程优化可以在项目实施之前进行,变以往对软件项目过程的被动控制为事先的有效预防和主动控制,显然对即将实施的软件项目管理更有意义。

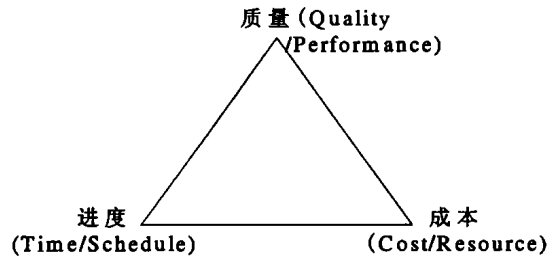

图2 项目三要素

尽管引起软件项目风险的原因多种多样(如技术、过程、管理等因素),但对于一个软件项目,它的过程状态主要体现在与项目进度、成本(资源投入)和质量三个基本要素相关的参数[11],如图2所示。相应地,软件项目的风险也最终表现为与项目的进度、成本和质量相关的风险。本文重点研究在满足质量要求的前提下,如何通过合理地在项目执行的各个任务之间调配给定的资源投入,最大限度地降低整个项目的进度(工期)风险,以保证项目在合理的成本范围内按期交付。

3 软件风险传递算法

与一般项目相比,软件项目具有更大的不确定性,在项目的实施中,各任务的执行都存在与计划偏离的风险,这些风险可以在软件项目各小组之间进行传递和累积,并最终影响整个项目的风险水平。下面以项目进度(工期)风险为例,介绍本文使用的软件风险传递算法。首先给出任务风险和项目风险的定义。

定义1(任务风险) 任务风险 r_i 为项目的第 i 个任务(又称任务 i)在计划条件下实际工期大于计划工期的概率,即该任务不能按计划完成的可能性。$r_i = p(t_i > t_i')$。

其中:t_i 和 t_i' 分别表示在计划条件下(未实施风险控制时)任务 i 的实际工期和计划工期。

定义2(项目风险) 项目风险 R 为在计划条件下该项目实际工期大于计划工期的概率,即该项目不能在计划内完成的可能性。$R = P(T > T')$。

T 和 T' 表示在计划条件下整个项目的实际工期和计划工期。

对于一个软件项目,从系统的分析、设计、到编码实现、测试及交付的各个任务之间的关系都可以归结为串行、并行两种最基本的传递关系,因此对软件项目风险传递的计算分为串行、并行和串行并行的混合共三种情况。

1) 若项目有 n 个串行任务,各任务的计划工期和工期风险分别为 t_i' 和 r_i。这时,

n 个串行任务的总工期风险为:$R = 1 - \prod_{i}^{n}(1 - r_i)$ (1)

整个项目的计划工期为:$T' = \sum_{i=1}^{n} t_i'$

2) 若项目有 n 个并行任务,各任务的计划工期和工期风险分别为 t_i' 和 r_i,设 $t_{max}' = \max(t_1', t_2', \cdots, t_n')$。在计划条件下,设各个并行任务的实际工期大于 t_{max}' 的概率为 $a_i = p(t_i > t_{max}')$,其中($i = 1, 2, \cdots, n$)。因为在并行情况下,只有当某个任务的实际工期超过了 t_{max}' 时,才会使总工期拖延。这时,

n 个并行任务的总工期风险为:$R = 1 - \prod_{i}^{n}(1 - a_i)$ (2)

整个项目的计划工期为:$T' = t_{max}'$

3) 对于项目的多个任务既有串行又有并行的混合情况,则可以把多个并行的任务看作是一个任务,先根据多个任务并行的总工期风险公式(2)得出"一个任务"的工期风险,再根据多个任务串行的总工期风险公式(1)得出整个项目的工期风险。

4 软件风险优化控制

一旦发现项目的风险超过可以接受的范围时，为了把项目的风险降至可以接受的范围内，管理者通常会首先采取在项目组内调整资源分配，或采取启用项目风险预备金的方式将项目的风险降至可以接受的范围内。但相同数量的资源投入在各个任务之间的不同分配方案会导致不同的软件项目风险。下面给出在资源投入总额确定的前提下对软件风险进行优化控制的模型和算法。

4.1 优化模型

通常，一个任务的风险与资源投入之间具有一定的负相关关系，即较多的投入意味着较小任务风险，即对某一任务增加投入或减少资源投入会导致该任务风险的增加或降低。假定任务 k ($1\leqslant k\leqslant n$)在原计划基础上增加投入 u_k(u_k 为负值意味着减少投入)前后的风险分别是 $r_k(0)$和 $r_k(u_k)$，将 $r_k(0)$和 $r_k(u_k)$分别简记为 r'_k 和 r_k，这时 r_k 和 r'_k 及 u_k 之间的关系用式(3)表示。其中 g_k 取决于任务 k 本身的特性和组织的过程能力

$$r_k=g_k(r'_k,u_k) \tag{3}$$

由于任务 k 的风险发生了变化，整个项目的风险将随之发生变化，这时整个项目风险 R 为：

$$R=1-(1-r'_1)(1-r'_2)\cdots(1-r_k)\cdots(1-r'_n)=1-\prod_{i}^{n}(1-r'_i)\times(1-r_k)/(1-r'_k) \tag{4}$$

根据3，可以通过增加项目中某个或某几个任务的投入来调整整个项目的风险。但是增加的投入与风险的下降之间呈非线性关系。对单个任务而言，其风险降到一定程度后，继续增加投入时风险一般不再明显下降。而且软件项目具有特殊性，不同阶段增加投入导致风险降低的效果是不一样的。因此仅靠对单个任务增加投入，可能难以将整个项目的风险降至管理者可以接受的范围。下面分析基于整个项目的多个任务风险优化控制问题。

假定项目在计划条件下追加一定数量的投入用于控制项目的风险(追加的这部分投入以下简称“风险控制投入”)，追加的风险控制投入总量为 U_{max}，并在风险传递路径上有 n 个任务，若 U_{max}在 n 个任务之间共有 m 种不同的分配方案，对于其中的第 k 种分配方案，各个任务增加的风险控制投入分别为 $u_{1k},u_{2k},\cdots,u_{ik},\cdots u_{nk}$，相应地各个任务单元的工期风险分别变为 $r_{1k},r_{2k},\cdots,r_{ik},\cdots r_{ik}$，这时，求解整个项目的风险优化控制问题用下式表示：

$$\begin{aligned}&\mathrm{Min}R_k\\&R_k=1-\prod_{i=1}^{n}(1-r_{ik})\\&r_{ik}=g_i(r'_{ik},u_{ik})\end{aligned} \tag{5}$$

where $0\leqslant\sum_{i=1}^{n}u_{ik}\leqslant U_{max}$，$1\leqslant k\leqslant m$，$1\leqslant i\leqslant n$，$\{u_{ik}\mid 1\leqslant i\leqslant n\}\in X$，$\{r_{ik}\mid 1\leqslant i\leqslant n\}\in Y$

R 代表项目的风险，r_{ik}代表 k 中方案任务 i 的风险。X 和 Y 分别表示管理者可以选用的风险控制方案集合及相应的风险状态集合。$U_{max}=0$意味着项目的风险控制投入为0，即在原计划投入条件下，项目各个任务间进行资源调配和优化。

4.2 算法设计

4.1中式(5)属于离散优化问题，目前离散优化的求解算法主要有穷尽搜索(exhausted -search method)法和离散控制优化算法(discrete control optimization method)。穷尽搜索的算法效率太低[9]，只适用于很小的软件项目；使用后一种算法，则要求风险控制投入与风险状态之间具有精确的对应关系[10]，只适用于软件过程非常稳定且完整定义的软件项目。因此这两种算法目前都不能很好地适用于求解软件风险的优化控制问题。

下面我们设计用动态规划算法来求解上述软件风险优化控制问题。假定通过风险分析已知项目当前各个任务的风险状态，且风险控制投入总量为 U_{max}。对于一个软件开发过程中的某个任务，当风险控制投入即式(5)中的 u_i 已知时，根据当时的风险状态(r'_i)，可从过程数据库中得到相应的风险数据 r_i，或由风险管理专家根据过程数据库中类似项目的风险数据给出 r_i。

我们连续地列出函数 $f_0(h_0),f_1(h_1),\cdots,f_j(h_j),\cdots,f_n(h_n)$。这里：

h_0：项目的所有任务都没有风险控制投入($h_0=0$)，这时项目的风险为项目初始状态下的风险。为统一其见，这时项目的风险用 $f_0(h_0)$表示。

h_1：项目的风险控制投入只分配给任务1且总投入为 h_1($0\leqslant h_1=u_1\leqslant U_{max}$)。除任务1的风险发生变化外，其他任务由于没有风险控制投入而保持原风险不变。这种情况下整个项目的风险用 $f_1(h_1)$表示。

h_j：项目的风险控制投入只分配给任务1到 j 且总投入为 $h_j(0\leqslant h_j=\sum_{i=1}^{j}u_i\leqslant U_{max})$。1到 j 的各个任务的风险都发生了变化；j 之后的任务由于没有风险控制投入而保持原风险不变。这种情况下项目的风险用 $f_j(h_j)$表示。

h_n：项目的风险控制投入分配给所有任务(1到 n)且总金额数为 $h_n(0\leqslant h_n=\sum_{i=1}^{n}u_i=U_{max})$。此时整个项目的所有任务的风险都发生了变化，这种情况下整个项目的风险用 $f_n(h_n)$表示。

$f_j(h_j)$值通过以下的动态规划计算公式求出(公式推导过程此略)：

$f_0(h_0)=f_0(0)=1-\prod_{i=1}^{n}(1-r_i(u_0))=1-\prod_{i=1}^{n}(1-r'_i)$，这里 $h_0=0,u_0=0,r'_i=r_i(u_0)$

$f_1(h_1)=\min_{u_1\leqslant h_1}\{1-(1-f_0(h_1-u_1))\times\beta_1(u_1)\}$，这里 $\beta_1(u_1)=(1-r_1(u_1)/(1-r_1(u_0))$

$f_j(h_j)=\min_{u_j\leqslant h_j}\{1-(1-f_{j-1}(h_j-u_j))\times\beta_j(u_j)\}$，这里 $\beta_j(u_j)=(1-r_j(u_j)/(1-r_j(u_0))$，$j=2,3\cdots,n$

5 应用示例

假设项目共有三个串行的任务，根据过程数据库中的历史数据得出这3个将要执行的任务不能按进度计划完成的风险分别是30%、20%和10%。项目的风险预备金 U_{max}为4千元。假定项目风险小于20%为管理者可以接受的风险范围。管理者可供选择的风险控制方案及相应的风险数据见表1(风险控制投入为0代表没有在原计划条件下增加或减少投入，即初始状态)。其中：风险控制投入单位为千元，负数代表减少投入，N/A(not available)代表不可用。

根据表1，在项目资源投入总量不变即项目风险控制投入总量为0时，项目的过程资源分配共有三种情况：

表1 管理者可以采取的风险控制方案

风险控制投入(u_1) \ 任务序号 No.	1	2	3
0	30%	20%	10%
−1	N/A	N/A	12%
1	15%	10%	5%
2	5%	5%	N/A
3	N/A	1%	N/A

(1)任务1、任务2和任务3的风险控制投入皆为0,即项目的初始状态,这时三个任务的风险分别为30%、20%和10%。整个项目的风险 R 为:

$$R=1-\prod_{i}^{3}(1-r_i)=1-[(1-30\%)(1-20\%)(1-10\%)]=50\%。$$

(2)任务1、任务2和任务3的风险控制投入分别为1、0和−1,即在任务1和任务3之间调配资源,但资源投入总量不变,这时三个任务的风险分别为15%、20%和12%。整个项目的风险 R 为:

$$R=1-\prod_{i}^{3}(1-r_i)\ 1-[(1-15\%)(1-20\%)(1-12\%)]=40\%。$$

(3)任务1、任务2和任务3的风险控制投入分别为0、1、−1,即在任务2和任务3之间调配资源,这时三个任务的风险分别为30%、20%和12%。整个项目的风险 R 为:即项目的初始状态,这时三个任务的风险分别为30%、10%和12%。整个项目的风险 R 为:

$$R=1-\prod_{i}^{3}(1-r_i)=\ 1-[(1-30\%)(1-10\%)(1-12\%)]=45\%。$$

显然,在项目没有增加风险控制投入($U_{max.}=0$)的上述三种情况中,第二种资源配置方案的项目风险最小,过程资源配置最为合理,即此种情况下的项目过程最优。图3为这三种情况下的项目风险分布情况。项目在实施前不确定的因素最多,风险最大,随着项目的实施即部分任务的完成,项目的风险将逐渐减少。

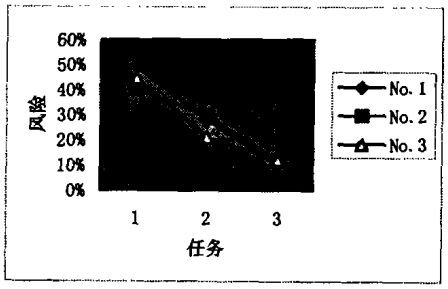

图3 $U_{max.}=0$时三种情况下的项目风险分布

尽管上述第2种情况下的项目风险最小,但仍超过管理者可以接受的范围,管理者可选择利用风险控制投入来降低整个项目的风险。图4为利用本文给出的优化模型和算法对给定的4千元对项目风险进行控制优化的结果。其中 No. 2为增加的4千元只投在任务1而其他任务投入不变的情况下经过风险优化控制后的项目风险分布;No. 3为增加的4千元投在任务1和任务2而任务3的投入不变的情况下经过风险优化控制后的项目风险分布;No. 4为增加的4千元投在三个任务的经过风险优化控制后的项目风险分布。为了比较和说明问题,图4中还增加了 No. 1和 No. 5两种情况。其中 No. 1代表项目的原始状态下(没有风险控制投入)项目的风险分布,No. 5为增加的4千元投入随意分配在三个任务即未经风险优化控制的项目风险分布。

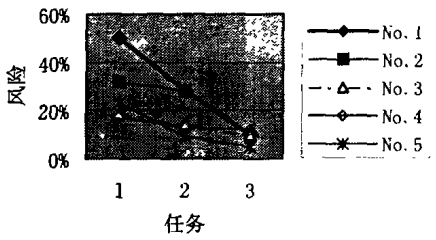

图4 $U_{max.}=4$千元经过优化控制的项目风险分布

显然在增加风险控制投入4千元时,No. 4的过程资源配置方案使得项目的风险最小(17%),降至管理者可以承受的风险范围20%以下,因而在原计划基础上对任务1、任务2分别增加2千元、3千元同时任务3减少1千元的资源配置方案最为合理,因而此种情况下软件项目的过程最优。相反,未使用本文给出的优化方法随意分配风险控制投入如 No. 5,虽然同样增加4千元风险投入但项目的风险依然较高(23%),这种情况下软件过程显然不是最佳。

总结 本文提出以软件项目风险大小作为评估当前软件项目过程优劣的判据,并从优化软件项目风险控制的角度对软件过程进行优化。给出一个基于风险传递的软件风险优化控制模型和一个动态规划的离散优化控制算法。

本文给出的基于风险控制对软件过程进行事先评估和优化的方法,变以往对软件项目"亡羊补牢"式的事后被动控制为事先的积极有效预防,为管理者在项目计划阶段实施过程优化和制定有效的风险防范决策奠定了基础。随着软件组织过程能力成熟度的提高,以及软件过程数据库的不断充实和完善,上述方法将具有更广泛的应用价值。

致谢 感谢美国 Saint Louis 大学计算机系的助理教授 Jacob Sukhodolsky 博士对本文所作的贡献和提出的建议。

参考文献

1 Chang C K, Christensen M. A net practice for software project management [J]. IEEE Software, 1999, 16(6): 80～88
2 徐如志,钱乐秋,等. 基于度量的软件项目过程优化控制研究. 电子学报(已录用待发表)
3 Boehm B. A Spiral Model of Software Development and Enhancement, Tutorial: Software Engineering Project Management. IEEE Computer Society Press, Washington, DC, 1987. 28～142
4 Charette R N. Large-scale project management is risk management [J]. IEEE Software, 1996. 110～117
5 Jalote P. CMM in Practice: Process for Executing Software Projects at Infosys [M]. Addison-Wesley, 2000
6 Risk Management Overview. CMMI-SE/SW Version 1.1. Copyright 2002 by Carnegie Mellon University, Jan. 2002. www.sei.cmu.edu/cmmi
7 Elaine M. Managing Risk: Method for software systems development. Addison-Wesley, 1998
8 Xu Ru-zhi, Qian Le-qiu. CMM-based Software Risk Control Optimization [C]. In: Proc. of the 2003 IEEE Intl. Conf. on Information Reuse and Integration (IRI-2003), Las Vegas, 2003. 499～503
9 Chang C K, Christensen M. A net practice for software project management [J]. IEEE Software, 1999, 16(6): 80～88
10 Sukhodolsky J. Object State Monitoring Optimization. In International Journal of Applied Science and Computations, 1997, 3 (3): 239～250
11 Simmons D B, Ellis N C, et al. Software Measurement. New Jersey, Prentice Hall, 1998