基于度量的软件项目过程优化控制研究

徐如志[1], 钱乐秋[1], 张敬周[1,2], 赵文耘[1]
(1. 复旦大学软件工程实验室, 上海 200433; 2. 上海计算机软件技术开发中心, 上海 200233)

摘 要: 本文给出一个软件过程度量的过程模型, 描述并分析了基于度量的软件过程跟踪和控制策略, 在此基础上, 结合实施 CMM 的软件工程实践, 提出一个软件项目过程控制优化模型, 并设计一个动态规划的软件项目过程控制优化算法. 当项目的实际过程偏离计划轨道时, 管理者利用本文给出的软件项目过程优化控制方法, 基于软件过程数据库中的历史数据, 迅速做出最佳的控制决策, 以确保项目目标的实现.

关键词: 软件过程; CMM; 过程数据库; 过程控制
中图分类号: TP311.5 文献标识码: A 文章编号: 0372-2112 (2003) 12A-2106-04

Research on Metrics-Based Software Process Control Optimization

XU Ru-zhi[1], QIAN Le-qiu[1], ZHANG Jing-zhou[1,2], ZHAO Wen-yun[1]
(1. *Laboratory of Software Engineering, Fudan University, Shanghai 200433, China;*
2. *Shanghai Development Center of Computer Software Technology, Shanghai 200233, China*)

Abstract: A process model of software process metrics is presented, and the strategies for tracking and controlling metrics-based software process are described and analyzed. Based on this and the practice of implementing CMM, a software process control optimization model and dynamic programming algorithm is developed and designed. When a schedule slip occurs, a project manager can immediately make an optimal process control decision to guarantee the success of the project by using the presented method together with the corresponding historical data stored in the process database.

Key words: software process; capability maturity model; process database; process control

1 引言

对软件项目实施有效的过程控制是保证软件项目成功的关键. 然而在现实情况中, 当软件项目的执行过程偏离计划轨道时, 管理者往往仅凭个人的经验和直觉做出控制决策, 具有很强的主观性和随意性, 需要一个有效的过程控制支持工具.

当前软件项目的过程控制主要是基于传统的项目管理网络技术(PMN)[1]对项目的进度和资源分配进行优化. 但由于软件过程的复杂性, 特别是软件项目的不确定性[2], 使用该方法通常只能优化软件项目的单个过程或阶段[3], 很难解决软件项目的全局优化问题. 其它的项目控制技术如基于挣得值(Earned value)的分析技术[4]、随机优化的方法[5], 以及基于 Gantt 图的软件过程控制优化方法[6], 也都没有被大多数软件项目管理者使用. 主要原因是这些工具没有充分考虑软件过程的特点, 而且这些工具的使用依赖于一些无法或不容易获得的数据.

CMM[7]提供了一个基于过程度量实施过程评价、过程控制和过程改进的框架. 经过完整定义的软件过程及其度量数据可作为一种软件资产被以后的软件项目复用[8,10], 从而为实施软件项目过程控制奠定了基础. 本文结合当前实施 CMM 的软件工程实践, 以过程评价和过程控制为目标, 给出一个软件过程度量的过程模型, 在此基础上描述并分析基于度量的软件项目跟踪和控制策略, 提出一个软件项目过程控制的优化模型和算法.

2 软件过程度量与过程控制

软件项目控制过程是一个持续的, 以四个阶段为一个生命周期不断循环的过程[7], 如图 1 所示.

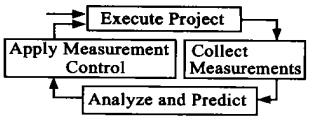

图 1 软件项目控制周期

第一阶段是项目的实施阶段, 如系统的分析、设计、编码实现及测试. 第二阶段是采集度量数据, 度量数据按照日、周、月连续采集. 在第三阶段中根据度量数据分析项目执行与计划偏离的程度, 判断项目是否能按计划完成. 第四阶段管理控制则是根据第三阶段的分析和预测结果, 来调整项目资源或计划.

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(人力资源投入等)和质量三个基本要素相关的参数.

收稿日期: 2003-09-30; 修回日期: 2003-12-28
基金项目: 国家高技术研究发展(863 计划)项目(No. 2002AA114010); 上海市科学技术委员会研究项目(No. 035115026)

在项目实际执行过程中,当其中的一个过程参数发生偏差时,相应地要求另外两个或一个参数发生变化或调整.

本文致力于在软件过程度量和分析的基础上对软件过程实施有效控制,研究当发生项目进度偏差时,管理者如何在满足质量要求的前提下,通过合理地在后续执行的各个任务之间优化给定的资源投入,最大限度地缩减后续执行任务的实际工期,确保软件项目在合理的成本范围内按期交付.

2.1　软件过程度量与跟踪

过程度量是过程跟踪与控制的基础,它与软项目的过程运作紧密相关,是一个由多个角色,在一定的条件和约束下,按计划进行的一系列活动.因此过程度量本身也是一个过程.图 2 为本文给出的一个软件度量的过程模型.其中的每个活动的输入都来源于上一个活动在当前时间之前的值以及相应的外部输入,使得当前时间得到的度量分析结果总能够反映截至当前时间为止过程的运作情况,进而达到过程评价、过程控制及过程改进的目的.其中度量活动包括数据采集、数据验证、数值转换、数据分析和度量决策五个活动.

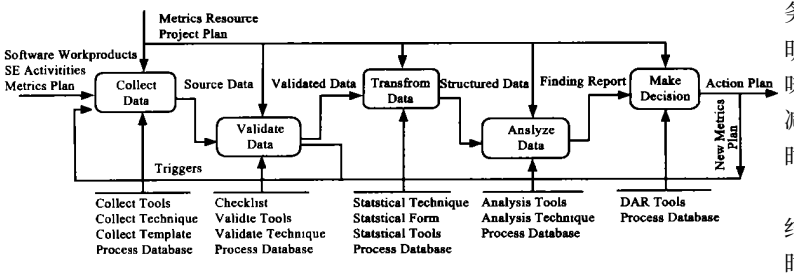

图 2　软件过程度量的过程

项目经理负责从项目组成员每天的活动、经历、软件参数中获取软件过程的实测数据,如工作量、进度以及发现的缺陷和缺陷的关闭情况,每周形成《项目状态报告》.在每个里程碑处,将所有《项目状态报告》的数据进行汇总,形成《项目状态汇总报告》,以集中反映项目在该阶段所有工作的完成情况,包括实际完成的工作、工作的进度、花费的工作量、需求变更数量、缺陷数据和有关的风险及处理情况等.表 1 所示为某个项目在其第三个里程碑结束时,计划和实际测量的与进度和资源有关的过程数据.

在项目结束阶段,项目经理负责将项目的所有实测数据进行汇总,并将项目最初的估算数据和软件过程的实测数据及分析结果存储在组织的过程数据库中.这些数据可作为今后类似项目进行估算、制定计划和计算组织过程能力基线的依据[7,9].

2.2　软件过程分析及控制策略

软件过程分析是将软件过程的实测数据与软件计划进行比较,分析和预测项目是否按计划执行和完成.下面我们只对关键路径上各个任务的执行情况和计划进行比较和分析.

假定项目的计划工期、实际工期和项目的计划与实际工期偏差分别表示为 D^p、D^A 和 $T(n)$.同时假定项目关键路径上任务 i 的计划工期、实际工期和计划与实际工期的偏差分别为 d_i^p、d_i^A 和 t_i, n 为项目关键路径上的任务数,则有:

$$D^p = \sum_{i=1}^{n} d_i^p;\ D^A = \sum_{i=1}^{n} d_i^A;\ t_i = d_i^A - d_i^p$$

$$T(n) = D^A - D^p = \sum_{i=1}^{n}(d_i^A - d_i^p) = \sum_{i=1}^{n} t_i$$

根据上面的公式,在项目关键路径上第 $k(1 \le k \le n)$ 个任务完成时,项目的实际工期与计划工期的偏差为: $T(k) = \sum_{i=1}^{k}(d_i^A - d_i^p) = \sum_{i=1}^{k} t_i$.如果 $T(k) \le 0$,则表示项目在执行完任务 k 时没有出现工期延误;如果 $T(k) > 0$,则表明项目到任务 k 完成时已经延期了 $T(k)$,这意味着如果不在随后实施的任务期间采取措施缩减工期确保 $T(n) \le 0$,项目将无法在计划规定的时间内完成.

通常情况下,依据上述软件过程度量分析的结果,一旦发现项目的实际过程落后于进度计划时,为了把项目余下的全部任务完成,管理者可以选择为项目组增加成员、推迟项目交付期限或缩减一部分项目需求.由于后面两种选择其实等于宣告了该项目的失败,管理者通常选择为项目组增加成员即增加项目控制投入(使用项目风险预备金)来保证项目按时交付.

如果使用增加的全部投入仍不能将项目控制在计划设定的目标范围,项目管理者应将上述情况及时通报给更高的决策层,由高层管理者决定是否继续追加控制资源或与用户协商变更项目目标.

3　软件过程优化控制

在增加的控制投入总数一定的情况下,不同的分配使用方案会产生不同的控制结果.下面给出软件项目过程优化控制模型和算法.

3.1　优化控制模型

通常,任务工期与投入之间具有负相关关系,即增加投入意味着缩短任务工期,且增加的投入越多,工期缩减数量越大.利用上述关系,决策者可以通过增加对某个任务单元的投入来缩减该任务的工期,从而减少整个项目的工期.假定某一个任务 $k(1 \le k \le n)$ 的工期为 d_k,在增加投入 u_k 后工期缩减的数量为 $t_k(u_k)$, $t_k(u_k)$ 和 u_k 之间的关系用式(1)表示.其中 g_i 取决于任务 i 本身的特性和

表 1　项目跟踪示例

Task No.	Planned schedule		Actual schedule		Schedule deviation (Days)	Effort (Person-days)		Effort Deviation (Person-Days)
	Start	Duration (Days)	Start	Duration (Days)		Planned	Actual	
3-1	01/03	15	01/03	19	+ 4	30	38	+ 8
3-2	16/03	14	20/03	21	+ 7	28	42	+ 14
3-3	30/03	15	11/04	18	+ 3	60	72	+ 12
3-4	15/04	13	29/04	13	0	39	39	0
3-5	28/04	16	12/05	19	+ 3	48	52	+ 4
Total		73		90	+ 17	205	243	+ 38

组织的过程能力.

$$t_k(u_k) = g_k(d_k, u_k) \quad (1)$$

根据上式,选择增加对项目中某个任务单元的投入来缩减项目的工期. 对单个任务而言,其工期降到一定程度后,继续增加投入时工期一般不再明显下降. 而且软件项目具有特殊性,在不同阶段增加投入缩短工期的效果不一样. 特别是在需求阶段,工期缩短的时间有限,过分压缩工期很可能导致软件质量的下降. 因此项目增加的控制投入通常是在后面实施的多个任务尤其是在关键路径上的多个任务之间进行分配.

假定整个项目的控制投入总量一定,最多不超过 U_{max},若关键路径上有 n 个任务,对各个任务增加的控制投入分别为 $u_1, u_2, \cdots, u_i, \cdots u_n$ 后,相应地各个任务单元的工期缩减数量分别为 $t_1, t_2, \cdots, t_i, \cdots t_n$. 由于同一个 U_{max},对 n 个任务可有多种不同的分配方案,不同的分配方案得到的项目工期缩减数量 T 值不同,这时对整个项目的工期控制优化问题可表示为求解 T 值最大的方案:

$$Max\ T \qquad T = \sum_{i=1}^{n} g_i(d_i, u_i) \qquad 0 \le \sum_{i=1}^{n} u_i \le U_{max}, \quad (2)$$

$$Where, 1 \le i \le n, \{u_i \mid 1 \le i \le n\} \in X, \{t_i \mid 1 \le i \le n\} \in Y$$

这里 t_i 代表单个任务 i 的工期缩减数量. X 和 Y 分别表示管理者可以选用的工期控制方案集合及相应的工期缩减数值集合. 由于项目的软件过程是基于组织的标准软件过程,经过适当裁减后确定的,项目的过程控制可以基于过程数据库中的过程数据和经验. 因此当管理者确定了可选的控制投入方案,即式(2)中的 u_i 已知时,相应的 t_i 的值可取自组织的过程数据库,或参照过程数据库中类似项目过程控制的历史数据得出.

3.2 优化控制算法及示例

3.1 中式(2)属于离散优化问题,目前可采用的求解算法主要有穷尽搜索法和离散控制优化算法. 穷尽搜索算法的效率太低[2],只适用于很小的软件项目;后一种算法则要求过程控制投入与过程状态之间具有精确的对应关系[9],只适用于软件过程非常稳定且完整定义的软件项目. 然而,在实践中管理者通常是在有限的几种控制方案中做出选择,因此上述两种算法都不能很好地适用于求解当前软件项目的过程控制优化问题.

为此,本文下面设计一个动态规划算法,来求解本文提出的软件项目过程控制优化问题. 下面我们连续地列出函数 $f_1(h_1), \cdots, f_j(h_j), \cdots, f_n(h_n)$. 这里:

h_1——控制投入只分配给任务 1 且总投入总额为 h_1($0 \le h_1 = u_1 \leqslant U_{max}$). 除任务 1 的工期发生了变化外,其他任务由于没有控制投入而保持原工期不变. 这种情况下整个项目的工期缩减数量用 $f_1(h_1)$ 表示.

h_j——控制投入只分配给任务 1 到 j 且投入总额为 h_j($0 \leqslant h_j = \sum_{i=1}^{j} \leqslant U_{max}$). 1 到 j 的各个任务的工期都发生了变化;j 之后的任务由于没有控制投入而保持原工期不变. 这种情况下整个项目的工期缩减数量用 $f_j(h_j)$ 表示.

表 2 管理者的可选方案

Control option	Task 1		Task 2		Task 3	
	Control cost(u_1)	Schedule slip reduction(t_1)	Control cost(u_2)	Schedule slip reduction($t2$)	Control cost(u_3)	Schedule slip reduction(t_3)
1	0	0	0	0	0	0
2	1	5	2	8	1	4
3	2	7	3	9	3	6
4	N/A	N/A	4	12	N/A	N/A

$f_j(h_j)$ 值通过以下的递归公式计算:

$$f_1(h_1) = \max_{u_1 \le h_1} \{t_1(u_1)\},$$
$$f_j(h_j) = \max_{u_j \le h_j} \{f_{j-1}(h_j - u_j) + t_j(u_j)\}, \quad j = 2, 3, \cdots, n \quad (3)$$

假定某项目在执行过程中发现比原计划延期了 17 天,如表 1 所示. 为保证该项目按计划完成,需要在随后实施的 3 个任务期间利用给定的项目控制投入 5 万元来缩减已发生的计划与实际的工期偏差. 表 2 为项目经理可选择的控制方案(投入为 0 表示不做任何投入),由于项目过程是基于组织的标准软件过程经过裁减而定义的,在列出对各个任务增加投入的可选方案后,相应地各个任务的工期缩减数量,可查询过程数据库或参照类似项目的历史数据得到.

图 3 为利用本文给出的动态规划算法通过求解式(2)得到的具有代表性的几种控制结果. 其中 No. 1 代表不追加投入的情况,No. 2 代表 5 万元只投入到随后的第一个任务而其它任务不增加控制投入的优化控制结果,No. 3 代表 5 万元只投入到前两个任务的优化控制结果,No. 4 代表 5 万元投入到所有 3 个任务的优化控制结果(提前 2 天完成). 而同样投入 5 万元,如增加的 5 万元投入在随后三个任务之间随机分配,则可能出现 No. 5(延期 3 天)甚至更糟糕的情况. 显然使用本文给出的优化控制方法,管理者可迅速做出正确的控制决策,即在增加控制投入不超过 5 万元的情况下,如希望项目能够提前完成,则优先选择 No. 4 方案;而如果希望项目在保证不延期的情况下,尽量减少受影响的任务数量,则选择 No. 3 方案.

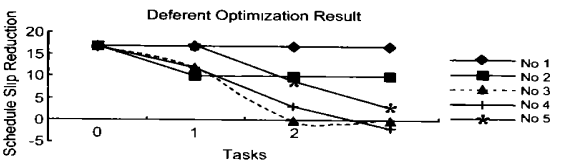

图 3 五种方案的不同控制结果

4 应用统计分析

将上述项目过程控制优化模型和算法集成于一家通过 CMM3 认证的大型软件公司的项目管理系统(简记 POMS),作为对软件项目进行计划、跟踪与控制的管理工具. 项目初期,项目经理根据软件估算制定项目计划,并将项目计划的主要信息,如项目目标及范围、重大里程碑和各项任务的进度、工作量、费用和人力资源等计划数据录入 POMS,并对项目及项目的各个阶段和任务进行编号.

项目执行过程中, 项目成员通过 POMS 填报项目工作日志. 每条日志记录包含项目编号、阶段编号、任务编号、工作任务描述、投入工作量和任务完成情况, 经项目经理审核后的数据写入项目数据库. 项目经理、高层经理可根据需要随时查阅项目的执行情况、按任务或阶段进行工作量统计, 并与计划进行比较. 当发现项目执行偏离计划时, 项目经理可调用 POMS 中的项目过程控制优化程序, 对可选的多个控制方案进行优化, 确定最佳的项目控制策略, 及时调整项目计划和采取有效的控制措施, 最大限度地缩减已发生的项目工期偏差.

本文对采用项目过程优化控制方法的 14 个软件项目, 与 1 年前同期未实施过程优化控制方法的 14 个软件项目的控制结果进行了对比. 如图 4 所示. 其中左图为项目的实际工期与计划工期比值, 右图为项目的实际投入与计划投入比值.

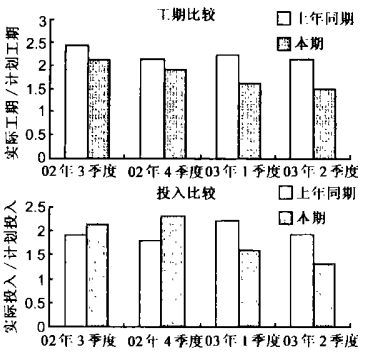

图 4　软件项目过程执行情况统计结果

从图 4 可以看出, 通过实施软件项目过程优化控制, 项目的平均实际工期与计划工期之比, 从过去的 2.1 降为 1.2; 实际投入与计划投入之比从过去的 1.9 降到 1.3. 这说明采用本文给出的项目过程优化控制方法后, 项目的进度控制能力提高了 41%, 项目的实际成本虽然开始时有所增加但全年的平均统计值不但没有增加反而下降了近 30%. 说明通过应用本文给出的过程控制优化方法, 既能够有效地增强对软件项目的过程控制能力, 同时也促进了对软件项目资源的动态合理调配, 使软件项目的资源利用效率有了明显提高, 从而显著地降低了项目的成本.

5　总结

本文给出一个软件过程度量的过程模型, 分析了基于度量的软件过程跟踪和控制策略; 基于 CMM 和过程控制理论, 结合软件过程工程的特点, 提出一个软件项目过程控制优化模型, 并设计了一个动态规划的软件项目过程控制优化算法.

当项目的实际过程偏离计划时, 应用本文给出的软件项目过程控制优化方法, 管理者可以在给定的投入条件下, 基于过程数据库或以往类似项目的过程数据, 对多个可选方案进行优化, 迅速做出最佳的控制决策.

上述方法集成于一家大型软件企业项目管理系统的应用实践表明, 本文给出的软件项目过程控制优化方法, 能显著地提升管理者对软件项目的过程控制能力, 促进资源的动态合理调配, 提高软件项目的成功率.

本文提出的软件项目过程优化控制方法的应用建立在业界广泛认可的 CMM 基础上, 所设计的算法针对实际、简单易用, 因此本文的研究成果可广泛应用于实施 CMM 的软件企业的项目管理和过程控制.

致谢

感谢上海计算机软件技术开发中心的朱三元研究员和美国 Saint Louis 大学计算机系的 Dr. Jacob Sukhodolsky 助理教授对本文所作的贡献和提出的建议.

参考文献:

[1] C K Chang, M Christensen. A net practice for software project management[J]. IEEE Software, 1999, 16(6): 80– 88.

[2] R N Charette. Large-scale project management is risk management[J]. IEEE Software, July, 1996: 110– 117.

[3] B Kanchana, et al. Software quality enhancement through software process optimization using Taguchi methods [A]. Proceeding of IEEE Conference on Engineering of Computer-Based Systems [C]. Nashville, Tennessee, March 1999. 188– 193.

[4] B Boehm. Software Engineering Economics[M]. Prenctice Hall, 1981.

[5] Barraza, B A, Back, W E, Mata, F. Probabilistic monitoring of project performance using SS-curves[J]. Journal of Construction Engineering & Management, 2000, 126(2): 142– 148.

[6] Jacob Sukhodlsky, W Smari. Software project control optimization[A]. Proceeding Of the 2000 IEEE International Conference on Information Reuse and Integration[C]. Honolulu, Hawaii, 2000. 42– 49.

[7] Pankaj Jalote. CMM in Practice: Process for Executing Software Projects at Infosys[M]. Addison-Wesley, 2000.

[8] Xu Ru-zhi, Qian Le-qiu. CMM-based software risk control optimization [A]. Proceeding of the 2003 IEEE International Conference on Information Reuse and Integration (IRI-2003)[C]. Las Vegas, 2003. 499– 503.

[9] Watts S Humphrey. 度量软件过程: 用于软件过程改进的统计过程控制[M]. 任爱华, 刘又诚, 译. 北京: 航空航天大学出版社, 2002.

[10] 朱三元, 钱乐秋, 宿为民. 软件工程技术概论[M]. 北京: 科学出版社. 2002.

作者简介:

徐如志　男, 1966 出生于山东泰安, 高级工程师, 现为复旦大学软件工程实验室博士研究生, 主要研究领域为软件过程工程, 项目管理, 软件复用.

钱乐秋　男, 1942 出生于江苏吴江, 教授, 博士生导师, 现为复旦大学软件工程实验室主任, 主要研究领域为基于构件的软件工程, 软件过程工程, 软件测试技术等.