文章编号:1001 - 9081(2005)01 - 0056 - 02

基于关键链的软件项目进度风险管理

蒋国萍,陈英武
(国防科技大学 信息系统与管理学院,湖南 长沙 410073)
(gpjiang1029@163.com)

摘 要:文中讨论基于关键链技术的软件项目进度风险管理方法。基于软件过程工作分解结构,预测各项工作在理想工作条件下的工期,考虑人力资源的约束与冲突,建立项目的关键链。通过对各项工作的风险分析,为关键链、非关键链分别设置项目缓冲、输入缓冲,通过对缓冲区的监控来进行风险的控制和管理。

关键词:软件项目;进度风险管理;关键链
中图分类号: TP315 **文献标识码**:A

Schedule risk management method of software project based on critical chain

JIANG Guo-ping, CHEN Ying-wu
(School of Information System and Management, National University of Defense Technology, Changsha Hunan 410073, China)

Abstract: Scheduling risk is one of the most critical risks of software project. The scheduling risk management based on task critical chain was discussed in this paper. Each task's duration in ideal working environment was estimated, personnel resource constraints were taken into account, and the project's critical chain was built. After the risk analysis for each task, the project buffer was established for the critical chain and feeding buffer was established for non-critical chain. The buffer area was treated as resort of risk control and management.

Key words: software project; scheduling risk management; critical chain

0 引言

风险管理被认为是减少软件项目失败的一种重要手段。所谓风险,是指未来可能发生的损失,主要有两个方面的属性:发生概率和发生后果。软件项目的风险主要是涉及阻碍软件项目的计划费用、在计划时间内达到计划系统功能等项目内外各方面的因素。Janne Ropponen[1]将软件开发风险划分为时间进度风险、功能风险、合同风险、需求管理风险、资源风险和人员管理风险等六大部分。本文讨论软件项目时间进度风险的管理。

传统的时间进度计划是基于工作分解结构之上,通过各工作的时间估计,构建计划网络,寻找时间关键路径,进行蒙特卡罗模拟等手段,获得工期的概率分布,以此来估计进度风险。而 Goldratt 提出的关键链[2]管理方法,用关键链代替 PERT/CPM 中的关键路径,不仅考虑了工作的执行时间和工作间的紧前关系约束,而且还考虑工作间的资源冲突,关键链是制约整个项目周期的一个工作序列。因此,本文将关键链技术用于软件项目风险管理。

1 关键链技术介绍

1997 年,Goldratt 出版了《关键链》一书,将约束集理论(Theory of Constraints, TOC)应用于项目管理领域,提出了项目管理的全新方法。Goldratt 定义关键链是既考虑工作间的依赖关系又考虑资源间依赖关系的最长的工作序列。Goldratt 认为在 PERT 中的工期估计中包含了大部分的安全时间,而安全时间并不能保证项目的按时完成。因此他将工作 50% 可能完成的时间作为工作工期的估计,并以此建立工作网络图。根据工作间的资源制约关系,修改网络图,确定关键链。然后通过为关键链和非关键链分别设置项目缓冲(Project Buffer)和输入缓冲(Feeding Buffer),来消除项目中不确定因素对项目执行计划的影响,保证整个项目按时完成。项目缓冲设置在关键链的末尾,以关键链上所有工作比 PERT 中少估计的工期和的 50% 为缓冲区的大小。输入缓冲设置在非关键链与关键链的汇合处,以非关键链上的所有工作节省工期之和的 50% 为缓冲区的大小。项目缓冲是为了保证项目在计划内完成。之所以设置输入缓冲,是为了保护关键链上的工作计划不会因为非关键链上工作的延迟而受到影响。

关键链技术的主要优点:

1) 既考虑了工作间的紧前关系约束,还考虑了工作间的资源冲突;

2) 标识了资源约束和资源瓶颈,有利于项目过程资源的配置,降低因资源而引起的进度风险;

3) 缓冲区的设置,为保证项目按时完成提供了有效的途径。

针对软件项目的特点和进度风险管理的任务,我们在本文中考虑软件项目中人力资源的约束。在风险分析的基础

收稿日期:2004 - 05 - 24;**修订日期**:2004 - 11 - 30 **基金项目**:国家自然科学基金资助项目(70272002)

作者简介:蒋国萍(1975 -),女,湖南人,博士研究生,主要研究方向:项目管理、系统建模与决策、风险管理; 陈英武(1963 -),男,湖南人,教授,博士,博士生导师,主要研究方向:系统理论、系统建模与决策、公共管理、项目管理、项目风险管理.

上,设置项目缓冲区和输入缓冲区,以应对项目过程中的不确定性因素,控制进度风险,确保项目整体的按时完工。文中提出了基于关键链技术的软件项目进度风险管理方法。首先对项目进行工作分解,估计理想工作条件下各工作的执行时间以及人力资源分配,建立工作节点网络图(Active on Node, AON);然后考虑人力资源的约束,确定工作节点网络图中的关键链;接着采用风险量 = 风险概率 × 风险时间这样的技术风险评估技术,对每项工作进行风险分析,在此基础上,为关键链配置项目缓冲,为非关键链配置输入缓冲;最后,在项目进行过程中,通过对缓冲区的监控,进行计划风险的管理。

2　关键链的确定

对项目进行工作分解之后,我们以工作在理想工作条件下的完成时间来估计该工作的执行时间。所谓理想工作条件是指既不考虑风险因素,也不考虑资源约束的"理想"状况。这样的理想工作条件实际是不存在的,就如同物理学研究中经常用到的理想气体一样。之所以采用理想工作条件下的完成时间(简称为理想工作时间),而不是 Goldratt 的 50% 完成的时间,是由于在 50% 的时间内肯定是不能完成工作的,太过紧张的计划时间会给工作执行人员造成不必要的压力,从而加大了项目的系统功能风险。而理想工作时间既不会因为有大量安全时间的存在而出现所谓学生综合症、帕金森症[3]等工作积压现象,又因为其存在按时完成的可能性而对工作执行人员起到激励的作用。

建立工作节点网络图。网络图中每个工作节点有一个三元组属性(a/b/c),其中 a 为理想工作条件下的工作执行时间估计,b 是该项工作需要的资源,c 是所需资源的数量。与 CPM 不同的是,关键链技术不是单纯以时间最长的路径为关键路径,而是在考虑了工作所需资源之后,根据资源约束,对网络图中工作的紧前关系进行必要的调整,然后再由工作时间,找出此时的关键路径,也就是关键链。

我们以一个简单的软件开发项目为例来说明方法的应用。该项目开发所需要的人力资源有:R1 系统设计人员,R2 程序开发人员,R3 数据库开发人员,R4 系统测试人员。工作节点网络图见图 1。其中工作时间 a 是考虑到不确定因素的非理想工作条件下的工作执行时间。图中粗线标识的路线是时间关键路径。

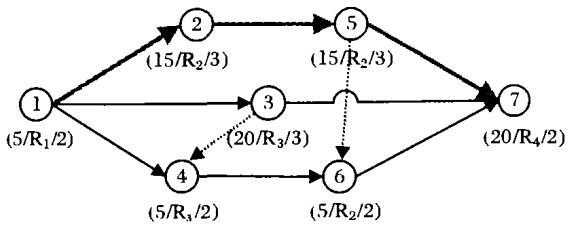

图 1　工作节点网络图

由于考虑到人力资源约束,从图 1 中可以看出,工作 3 和工作 4 资源冲突,工作 2、5 和工作 6 也存在资源冲突,我们将它们之间的并行执行关系转化为串行执行,如图 1 中虚线所示。同时重新按理想工作条件估计每项工作的执行时间,从而得到图 2。图 2 中的工作时间是理想工作时间,粗线标识的是考虑了人力资源约束之后的项目关键链。

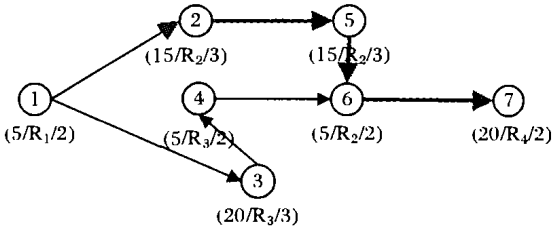

图 2　基于关键链的工作节点网络图

3　软件项目缓冲区的设置

为了保护关键链上的工作而不影响到整个项目的计划进度,关键链技术要求为关键链设置项目缓冲区;同时为了防止非关键链上的工作影响到关键链上工作的进度,在非关键链与关键链的汇合处设置输入缓冲。Goldratt 是以链上所有工作预测工作时间时节省下来的安全时间的 50% 作为缓冲区的大小。本文考虑以各项工作的风险量之和作为缓冲区的大小。

采用文献[6]中提出的技术风险分析方法为每项工作进行风险分析。在本文中,只关注时间进度风险暂时不考虑风险费用,因此风险量 = 风险概率 × 风险时间。

项目缓冲区的大小等于关键链上所有工作的时间进度风险量之和:

$$PB = \sum_{i \in CC} p_i \times rt_i \tag{1}$$

其中,CC 是关键链上的所有工作,p_i 是关键链上的工作的风险概率,rt_i 为风险时间。

输入缓冲区是非关键链上所有工作的时间进度风险量总和:

$$FB = \sum_{j \in NCC} p_j \times rt_j \tag{2}$$

其中,NCC 是非关键链上的所有工作。

继续第 2 节中的例子。在经过风险分析之后,得到各工作的风险量见表 1。

表 1

工作	风险量	工作	风险量	工作	风险量	工作	风险量
T1	1	T2	2	T3	4	T4	1
T5	3	T6	1	T7	3		

根据(1)式和(2)式分别设置项目缓冲(大小为 10)和输入缓冲(大小为 5),图示如下(图 3)。

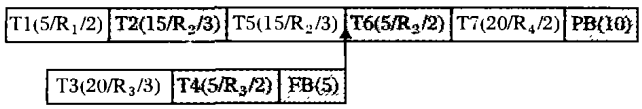

图 3　基于关键链的项目计划进度图

4　基于关键链的风险管理

缓冲区的设置是为了应对项目过程中可能出现的不确定因素,进行风险的监控和管理。Goldratt 对于缓冲区的管理是采用"三色"管理办法,将缓冲区三等分,每个部分分别以绿、黄、红三色表示。在项目进行过程中,检查缓冲区的占用情况。对缓冲区的占用处于绿色区时,认为情况良好;处于黄色区时,一般不采取特别的措施,而是进一步观察并制定风险计划;若缓冲区已被占用到红色区,说明项目已经存在相当严重的进度风险,必须采取相应的补救措施。(下转第 72 页)

2.4.4 内存使用的分析

空间方面,维护一个缓存并不一定会占用更多的内存。原因在于,Java 中释放无用对象的工作都是由垃圾收集器(Garbage Collector)来完成的,而文献[6]中的多拷贝方案非常依赖于这一机制;但是垃圾收集器并不是在对象引用数为0时马上就释放空间,大量还未来得及被释放的重复拷贝并不一定会比缓存占用的空间更少,更严重的是在一个很繁忙的系统中还会因为物理内存耗尽而引起换页操作。另一方面,即使垃圾处理器可以很快地释放空间,频繁地进行垃圾收集对 Java 虚拟机来说是有额外开销的,在高负载情况下必然引起显著的性能下降。缓存的引入,减少了给垃圾收集器造成的负担。当然在发生缓存淘汰时,还是需要垃圾收集器来回收被淘汰的对象的。

图6就是在系统相对稳定时的内存使用情况。有缓存时占用的内存发反而减少了 12% ~40%。另外,我们的缓存并没有超时淘汰的机制,在系统运行稳定后,无论负载如何,缓存占用的内存是不会变化的,负载升高引起的内存使用增加还是由于有更多的无用对象没有被及时释放造成的。从图6可以看出,缓存也使增幅有一定程度减小。最后我们尝试性地加入了缓存对象的超时淘汰,在低负载时确实减少了内存占用,但是在高负载时不仅对减少内存占用没有任何帮助,还造成了明显的性能损失。因此是否加入缓存超时淘汰机制以及如何高效地实现还有待进一步分析。

3 结语

该缓存机制的引入大大提高了系统的性能。在易用性方面,应用程序员也只需扩展 PersistentObject 类,遵循很简单的对象属性命名规则,提供相应的 Bean 方法就能构造使用缓存机制的永久对象类,并且也可以在自己的永久对象类中对缓存的使用方式进行配置。尽管该缓存机制是专为我们的远程支持系统所使用的轻型对象管理器设计的,但它可以被轻松地移植到其他类似的对象管理器中,同时,它的设计思想也能为其他的对象缓存机制的设计提供借鉴。

参考文献:

[1] TESCH T, VOLZ M. A Lightweight Object Manager for Group-Aware Applications[R]. GMD Report 47, 1999.

[2] KELLER W. Object/Relational Access Layers - a Roadmap, Missing Links and More Patterns[A]. Proceedings of the 3rd European Conference on Pattern Languages of Programming and Computing [C], 1998.

[3] KELLER W. Mapping Objects to Tables - a Pattern Language[A]. Proceedings of the EuroPLoP[C], 1997.

[4] AMBLER SW. The Design of a Robust Persistence Layer For Relational Databases. White paper[Z]. An AmbySoft Inc White Paper, October 1999.

[5] BERGLAS A. SimpleORM White Paper/Simple Java Object Relational Mapping[M/OL]. http://www.uq.net.qu/~zzaberg/simpleorm/whitepaper.html, September 2003.

[6] SONG J. Spine: A Light Persistent Object Manager[J]. Computer Science, 2004, (3).

[7] http://ehcache.sourceforge.net/documentation/[EB/OL].

[8] Patterns for Object / Relational Mapping and Access Layers[EB/OL]. http://www.objectarchitects.de/ObjectArchitects/orpatterns/index.htm.

(上接第57页)

基于关键链技术的软件项目风险管理通过对缓冲区的监控进行。关键链技术消除了每项工作的开始日期、完成日期,取而代之的是每条链的起止时间。但是我们是以每项工作的进度风险量之和设置缓冲区的大小,因此要避免各项工作的实际工作时间超出(估计时间 + 风险时间)。我们为缓冲区设置了安全底线,缓冲区的安全底线反映的是项目过程中各时刻缓冲区大小的最小值。在项目进行过程中,定时观测缓冲区的大小,若缓冲区处于安全底线以上,我们认为工作情况正常,低于安全底线,则有必要采取风险措施。

由表1,得到图4的项目缓冲区划分。如图中所示,若项目过程中观察到缓冲区处于安全底线以上的区域,则工作执行情况良好;若处于安全底线以下的区域,则有必要根据风险计划,采取相应的风险措施。

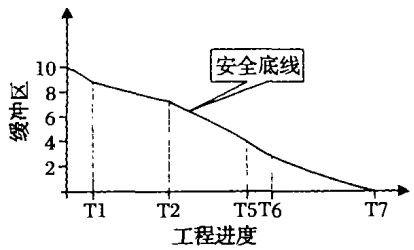

图4 缓冲区划分

5 结语

本文讨论了基于关键链的软件项目进度风险管理方法。关键链技术不仅考虑了工作间的紧前关系约束,还考虑了工作间的人力资源冲突。以理想工作条件下各个工作的执行时间建立工作节点网络图,考虑人力资源的冲突,确定关键链。在对各个工作进行风险分析的基础上,配置项目缓冲区和输入缓冲区,以消除不确定性,保证整个项目的按时完工。项目过程中,通过对缓冲区的监控和管理,实现对软件项目进度风险的管理。

但是,基于关键链的进度风险管理方法还存在一些问题。譬如,资源冲突时关键链的一般确定方法;存在多个资源约束时关键链的确定方法;基于缓冲区的进度风险的管理和监控等,这些都还有待作进一步研究。

参考文献:

[1] ROPPONEN J, LYYTINEN H. Component of Software Development Risk: How to Address Them? A Project Manager Survey[J]. IEEE Transactions on Software Engineering, 2000, 26(2).

[2] GOLDRATT EM. Critical Chain[M]. North River Press, Great Barrington, MA, 1997.

[3] PATRICK FS. Critical Chain and Risk Management - Protecting Project Value from Uncertainty[EB/OL]. http://www.focusedperformance.com/articles/ccrisk.pdf, 2001.

[4] 刘士新,宋健海,唐加福. 关键链———种项目计划与调度新方法[J]. 控制与决策,2003, 18(5).

[5] 李宁,吴之名. 网络计划技术的新发展——项目关键链管理(CCPM)[J]. 公路,2002,(10).

[6] MICHAELS JV. Technical Risk Management[M]. Prentice hall PTR, 1996.

[7] 徐哲,冯允成. 网络计划进度的风险管理[J]. 系统工程理论与实践,1998,(4).