ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.12, December 2010, pp.3029−3041
doi: 10.3724/SP.J.1001.2010.03716

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

风险驱动的软件项目缓冲分配方法*

谢利子[1,2], 王 青[1+], 肖俊超[1]

[1](中国科学院 软件研究所 互联网软件技术实验室,北京 100190)

[2](中国科学院 研究生院,北京 100190)

Reducing Plan Change: A Risk Driven Software Project Buffer Allocation Method

XIE Li-Zi[1,2], WANG Qing[1+], XIAO Jun-Chao[1]

[1](Laboratory for Internet Software Technologies, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

2(Graduate University, The Chinese Academy of Sciences, Beijing 100190, China)

+ Corresponding author: E-mail: wq@itechs.iscas.ac.cn

Xie LZ, Wang Q, Xiao JC. Reducing plan change: A risk driven software project buffer allocation method. ***Journal of Software*, 2010,21(12):3029–3041.** http://www.jos.org.cn/1000- 9825/3716.htm

Abstract: This paper proposes a risk driven project buffer allocation method which will allocate project buffer according to risks, schedule constrains and resource constrains between tasks. The result of simulation based experiments shows that compared with Critical Chain, the proposed method can remarkably reduce the frequency of plan change and minimize the negative impact to project duration. The method and the tool can help project managers to decide appropriate project buffer length and make better use of the limited buffer. The stability and trustworthiness of project plan schedule can be enhanced.

Key words: software project management; plan change; risk; project buffer; process simulation; resource constrain

摘 要: 提出一种风险驱动的项目缓冲分配方法,并开发了相应的项目模拟执行工具对方法进行验证.方法旨在通过综合考虑软件项目的风险因素、任务之间的进度约束和资源约束对项目的可用缓冲进行合理分配.模拟实验的结果表明,在风险较高的软件开发项目中,该方法相对于关键链项目管理中尾部集中的项目缓冲分配方法,可确保在对项目平均执行工期产生较小影响的同时,显著降低项目执行时计划变更的发生频率.该缓冲分配方法与项目模拟工具可以帮助软件项目经理确定合适的项目缓冲时间长度以及缓冲分配方案,进而提高软件项目计划的可信性和执行的稳定性.

关键词: 软件项目管理;计划变更;风险;缓冲分配;过程模拟;人力资源约束

中图法分类号: TP311 **文献标识码**: A

随着信息技术的应用领域不断扩展,软件产品的规模不断增大,复杂度也越来越高.这导致软件项目管理的

* Supported by the National Natural Science Foundation of China under Grant No.90718042 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.2007AA010303, 2007AA01Z186 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2007CB310802 (国家重点基础研究发展计划(973))

Received 2009-04-08; Accepted 2009-08-12

难度也越来越增加.软件项目进度滞后、成本超支的情况依旧屡见不鲜.近年来,软件开发的主流模式已由小规模的手工作坊式开发转向诸如软件外包等形式的跨地域、跨组织的协作软件开发.软件产品与生俱来的高复杂性导致软件开发过程相对于传统工业生产过程具有更高的不确定性.频繁的计划变更将严重影响软件开发效率,增加额外的工作量和沟通成本.同时,计划变更所带来的大量沟通协调工作也会给项目带来新的风险.如何制定出符合项目特点的、稳定的进度安排是项目经理们的首要任务,也是确保软件项目成功的关键.

关键链项目管理方法[1]是约束理论在项目管理中的成功应用[2,3].关键链是指"相关任务所连成的耗时最长线路,相关性既可以由任务之间的逻辑性造成,也可以由资源约束造成"[4].在对具体的项目和任务进行进度安排时,从项目经理、任务负责人到具体的开发人员都会考虑可能存在的风险而为自己的进度安排争取一定的安全时间,该安全时间称为可用缓冲[1].关键链项目管理方法取消了里程碑概念,它将项目缓冲设定为项目关键链长度的一半,并将其集中放置在项目尾端,通过监视缓冲消耗情况来监督项目进展.近年来,关键链项目管理技术在世界范围内被传统的工业企业广泛采用.通过实施关键链项目管理可以显著缩短企业生产周期,加快资源流转.但是,关键链方法在缩短工期的同时带来了高频率的计划变更[5],并不适用于风险较高、不确定性较高的软件开发项目.如何减少计划变更、提高项目执行的稳定性是项目管理领域的热点问题.稳定的项目计划将极大地提高协作开发效率,降低开发成本.尤其对于软件外包等协作型软件开发公司而言,稳定的项目进度安排和执行能力将提高其行业竞争力.

本文提出一种风险驱动的项目缓冲分配方法.该方法通过综合考虑软件项目的风险因素、任务之间的工作产品约束和人力资源约束,将有限的可用项目缓冲分配到各个任务中.同时开发了风险驱动的项目执行模拟工具对方法进行了验证.实验结果表明,在风险较高的软件开发项目中,风险驱动的项目缓冲分配方法相对于当前流行的关键链项目管理中尾部集中式的缓冲分配方法,在对项目平均执行工期产生较小影响的同时,能够显著降低项目执行时计划变更的发生频率.方法相对关键链管理理论中尾部集中的缓冲分配而言,更适用于风险较高的软件开发项目.方法和工具可为软件项目经理确定合适的项目缓冲长度和缓冲分配方案提供有力的决策支持,提高了软件项目计划的可信性和执行的稳定性.

1 相关研究

风险是指"可以对项目目标造成负面或正面影响的不确定因素"[6],通常情况下的风险指负面因素.风险管理作为IT项目管理的 9 个重要领域之一[7],是确保软件项目成功的关键.由于软件开发项目不确定性较高,项目在执行过程中实际进度和成本经常偏离计划,此时对项目计划进行的调整称为计划变更[4].过于频繁的计划变更会对项目执行造成严重影响,会带来额外的沟通成本和风险.缓冲是指为了应对风险对进度造成的影响,在项目或任务的理想工期之外所预留的可用安全时间.通过设立缓冲来应对风险,在 4 个风险处理方式(规避、接受、转移、减缓)中属于风险接受的范畴,既接受风险发生,提前做好准备以降低风险发生时对项目进度、任务进度所造成的影响.

在项目管理领域,有一些针对关键链项目管理方法的研究和改进工作,如通过分析项目风险来对项目缓冲长度进行估算[8]和通过模糊逻辑估算项目缓冲长度[9]等.以上研究旨在获得更好的项目缓冲长度估计,并未涉及具体的任务缓冲分配方法,也没有考虑计划变更对项目的影响.文献[10]指出,项目经理应该根据项目特点在生产率和项目稳定性上做出均衡,而不能盲目地套用关键链管理方法.

项目管理的实践者们意识到有必要通过为具体的任务设立可用缓冲来提高项目进度的稳定性.比如微软公司内部的一些开发小组为应对软件开发过程的高风险和高不确定性,采用 50%缓冲方法[11]安排项目进度.该方法易于实施,只需统一地对每个任务的进度增加任务理想工期长度一半的缓冲时间,但其缺点是没有考虑任务之间的差别和联系,也没有对任务进行细致的风险分析.基于价值的软件工程[12]思想指出,软件工程中各因素不能等价视之,2/8 原则[13]同样适用于软件工程.因此,不同的任务根据其风险情况应该具备不同级别的不确定性,分配缓冲时应该对这种区别予以考虑.

关键链项目管理中,尾部集中的缓冲分配和管理方法会得到最佳的平均项目执行工期,但是项目执行过程

中的计划变更频率也是较高的[5].我们在软件项目管理、风险驱动的需求管理和项目计划制定等领域做了广泛而深入的研究[14−17],本文基于以上研究基础,试图通过风险驱动的缓冲分配方法在计划变更和项目平均完成工期之间寻求均衡,以提高软件项目执行的稳定性.提出如下研究假设:

在风险较高的软件开发项目中,根据任务风险的不同分布和风险的影响程度对项目缓冲进行合理分配,相对于关键链项目管理中尾部集中的缓冲分配方法,可以在对项目平均执行工期造成较小影响的同时有效降低计划变更发生的频率.

本文提出一种风险驱动的项目缓冲分配方法,并通过项目模拟对方法的效果进行验证.第 2 节介绍软件项目模型和缓冲分配方法.第 3 节通过模拟实验对缓冲分配方法进行实验分析.最后是研究工作的总结和对未来工作的展望.

2 风险驱动的项目缓冲分配和项目模拟方法

2.1 整体框架

本文研究工作的整体框架图如图 1 所示.框架分为应用层、工具层和模型层.在应用层,项目经理首先整理和分析项目信息和风险信息,然后通过工具获得在给定可用项目缓冲范围内的项目进度安排,并对项目进度安排进行模拟执行.最后,项目经理根据模拟执行的结果做出相关决策.在工具层,风险驱动的项目缓冲分配模块提供自动的项目缓冲分配和进度安排功能,风险驱动的项目执行模拟模块则提供项目模拟执行和数据统计分析功能.在底部的模型层中,软件项目模型对框架中各个模块的实现和具体运作提供底层的描述支持.

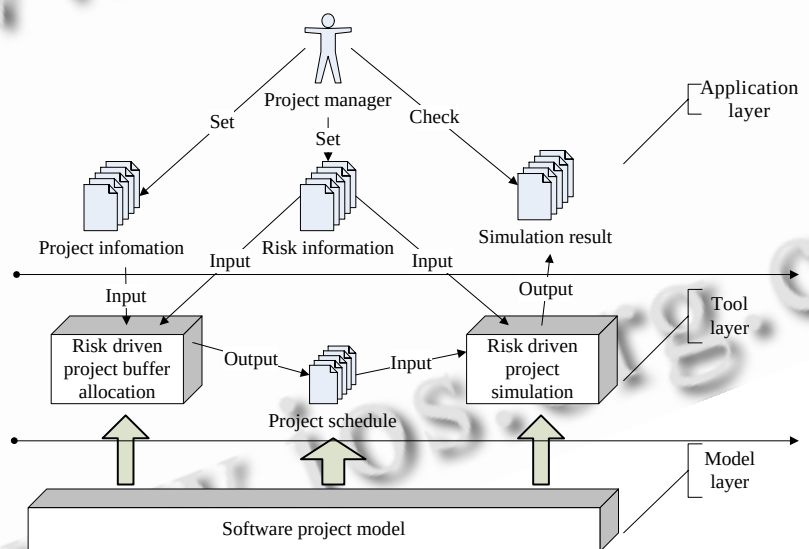


Fig.1 Main framework of the research

图 1 研究工作的整体框架图

2.2 软件项目模型

软件项目模型描述了软件开发项目中的要素(项目、任务、人力资源和风险)及要素之间的关系.该模型是实现缓冲分配和项目模拟执行的基础.

软件项目 *P* 定义为如下可扩展的五元组:

P(*project*)={*PN*,*PSD*,*PFD*,*PB*,*TS*,…}

-*PN*(project name):项目名称

-*PSD*(plan start date):项目计划开始日期

-*PFD*(plan finish date):项目计划结束日期

-*PB*(project buffer):项目可用缓冲,单位:天

-*TS*(task set):项目包含的任务集合.

$TS=\{T_1,T_2,\ldots,T_n\}$

--*T*(*task*)={*TN*,*PSD*,*PFD*,*TSB*,*IET*,*E*,*CP*,*HRS*,*PreTS*,*PostTS*,*RS*,…}:这里的任务是指 IEEE 相关标准中,工作分解结构(WBS)里的底层工作包[18,19],其成本、工作量、工期、工作产品和资源需求等属性是较为明确的.

---*TN*(task name):任务名称.

---*PSD*(plan start date):任务计划开始日期.

---*PFD*(plan finish date):任务计划结束日期.*T.PFD*=*T.PSD*+*T.IET*+*T.TSB*.

---*TSB*(task scheduled buffer):任务所分配的可用缓冲天数.

---*IET*(ideal execution time):理想情况下(不考虑风险)该任务需要的执行天数.

---*E*(executed):用来标识该任务是否被模拟执行过.取值为 true 表示该任务已被模拟执行过,取值为 false 表示该任务未被模拟执行过.

---*CC*(critical chain):用来标识该任务是否位于选定的项目关键链之上.取值为 true 表示该任务是关键任务,取值为 false 表示该任务不是关键任务.

---*HRS*(human resource set):分配给任务的人力资源集合

$$HRS=(HR_1,HR_2,HR_3,\ldots,HR_m).$$

----*HR*(human resource)={*HN*,…}:人力资源属性可以根据需要进行扩展,这里只列出人员姓名.

-----*HN*(human name):人力资源姓名.

---*PreTS*(pre-task set):前置任务集合,当前任务如果要开始,则其前置任务必须已经结束.

---*PostTS*(post-task set):后置任务集合,如果当前任务不结束就不能开始执行该集合中的任务.后置任务集合可以通过前置任务集合关系得到,这里为了后期缓冲分配和模拟执行的描述方便,单独列出.

---*RS*(risk set):对任务进度产生影响的风险集合

$$RS=\{R_1,R_2,R_3,\ldots,R_j\}.$$

----*R*(risk)={*RN*,*I*,*P*,…}.

-----*RN*(risk name):风险名称.

-----*I*(impact):风险对任务进度的影响程度.

-----*P*(probability):风险可能发生的概率.

在确定任务的前置后置任务集合时做如下考虑:

软件开发项目中,不同任务间存在前驱后继的进度约束关系,任务间的 4 种进度约束关系[14]为:完成-完成(finish-finish,简称FF),完成-开始(finish-start,简称FS),开始-开始(start-start,简称SS),开始-完成(start-finish,简称SF).在软件开发项目中,以FS型约束最为常见.该约束的具体表现是一个任务必须当其他某些任务完成后才可以开始.在软件开发过程中,造成这种进度约束关系的原因主要有两个:

(1) 工作产品约束

如果任务的执行需要其他任务提交相应的工作产品作为输入,那么该任务与其他任务之间存在 FS 型约束关系.比如系统设计任务和设计评审任务之间就存在这种约束关系,因为设计评审任务的执行需要系统设计任务完成并提交相关工作产品(设计文档).

(2) 资源约束

如果某任务的执行需要其他任务完成以释放必要的资源,那么该任务与这些任务之间也会存在 FS 型约束

关系.在软件开发项目中,该资源主要是指人力资源.

2.3 风险驱动的项目缓冲分配方法

本研究工作只考虑可能对任务进度造成影响的风险.我们对风险的特性进行了相应的简化,简化后的风险模型仍具普遍意义.

(1) 原子性:每个风险要么发生一次,要么不发生.对于可能发生 N 次的风险,我们将其表达为 N 个单独的风险.

(2) 独立性:只考虑独立的风险,风险之间的相关关系暂时不作考虑.

(3) 叠加性:发生在同一个任务上的多个风险对任务进度所产生的影响是叠加进行的.比如,任务T_1存在n个风险(R_1,R_2,R_3,…,R_n),则所有风险都发生情况下的任务工期为

$$T_1.IET \times \prod((T_1.R_n.I+1)) \tag{1}$$

(4) 无时效性:某个任务的风险在任务执行过程中任意时间发生,对任务进度所造成的影响是相同的.比如任务“模块 A 编码”存在风险“硬件故障”,该风险在任务执行的任何阶段对任务导致的延期天数是相同的.

为方便计划变更后项目进度的自动调整和项目执行过程中关键链迁移发生的几率,在对项目进行缓冲分配及后期进行项目模拟时,我们假定项目所有已分配的人力资源均只为该项目工作.风险驱动的项目缓冲分配方法流程如下(如图 2 所示).

Step 1. 信息输入

用户输入项目计划,包括项目信息、任务理想工期、人员信息、人员分配情况、任务之间的工作产品约束及风险信息.

Step 2. 约束关系设定

系统根据任务之间的工作产品约束和任务之间的人力资源约束确定任务之间的前驱后继关系.

Step 3. 理想进度安排

系统根据任务之间的约束关系和任务的理想执行时间,形成任务的理想进度安排.该安排的总工期称为理想项目工期.本步骤确定了在不考虑风险情况下所有任务的理想开始日期和结束日期.任务 T 的具体处理算法如下:

$$\begin{cases} \text{如果}T.PreTS \neq \varnothing,\text{则}T.PSD = \text{Max}\{s \mid s = t.PFD, t \in T.PreTS\} \\ \text{如果}T.PreTS = \varnothing,\text{则}T.PSD = P.PSD \end{cases} \tag{2}$$

$$T.PFD=T.PSD+T.IET \tag{3}$$

Step 4. 分析关键链

系统分析理想进度安排中的项目关键任务链.如果项目计划存在多条关键链,则随机选择其中之一进行处理.

Step 5. 缓冲分配

按照任务之间的约束关系,依次为全部的关键任务分配缓冲,详细流程分为如下 4 步:

(1) 计算任务风险缓冲权重

假定所选定的关键链上存在n个任务{T_1,T_2,…,T_n},记任务T_1的风险缓冲权重RW(risk weight)为(风险的影响与风险发生概率的乘积$R.I \times R.P$被称为风险的暴露度(risk exposure).该值在风险管理领域被用来衡量风险的重要性大小):

$$T_1.RW = T_1.IET \times \left\{\left[\prod(T_1.R_n.I \times T_1.R_n.P+1)\right]-1\right\} \tag{4}$$

(2) 计算任务的可分配缓冲天数

任务T_1的分配缓冲天数为

$$T_1.TSB = \delta\left(P.PB \times \left(\frac{T_1.RW}{\sum T_n.RW}\right)\right) \tag{5}$$

函数$\delta(x)$为对 x 取整运算.当 x 的小数部分大于等于 0.5 时,对 x 取上整;当 x 的小数部分小于 0.5 时,对 x 取下整.由上定义可知 $\sum T_n.TSB = P.PB$,即缓冲分配后,选定的关键链上各个任务的缓冲之和等于项目的总可用缓冲.

(3) 调整任务的计划结束日期

将任务T_1的计划结束日期更新为

$$T_1.PFD=T_1.PSD+T_1.IET+T_1.TSB \tag{6}$$

(4) 调整后续任务的计划工期

根据Step 3 中的方法递归调整任务T_1所有后续任务的计划开始日期和计划结束日期.

Step 6. 处理非关键链任务

按照任务之间的约束关系,循环处理所有的非关键链任务.调整非关键链任务的可用缓冲时间为其计划结束日期和其后继任务的最早开始日期之间的时间跨度.我们为软件项目确定非关键任务开始日期的方法有别于传统工业生产过程.传统工业过程中,为了避免库存开销以及帕金森症[20],采用及时生产(just in time,简称JIT)思想[21]安排最佳任务开始日期.软件开发有别于工业生产,软件生产的库存开销很小,可以不去考虑.而软件开发过程的不确定性较大,因此在我们的方法中,确定关键链任务进度时采取尽早开始策略,这样可以降低非关键链任务的进度出现异常时对关键链的冲击.

具体处理方法如下:

根据公式(2)重新计算非关键链任务的计划开始日期:

$$T.PFD=\text{Min}\{s|s=t.PSD,t\in T.PostTS\} \tag{7}$$

$$T.TSB=T.PFD-T.PSD-T.IET \tag{8}$$

Step 7. 输出缓冲分配结果

以甘特图的形式输出缓冲分配后的项目进度安排,供用户查看.缓冲分配前后,项目进度的关键任务集合保持不变.

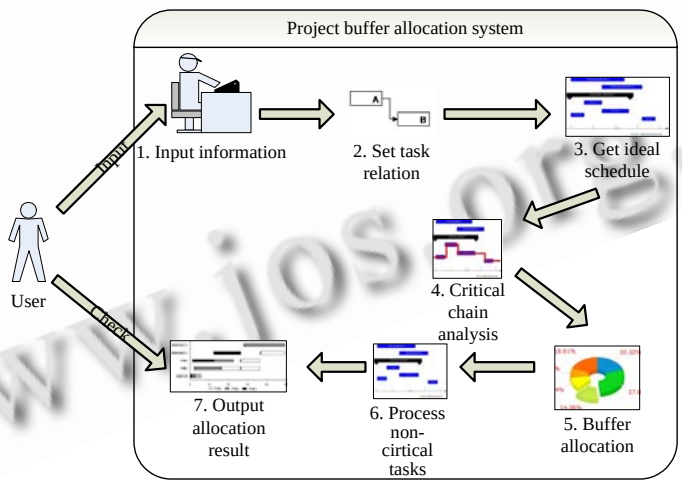


Fig.2 Risk driven project buffer allocation method

图 2 风险驱动的项目缓冲分配方法

2.4 风险驱动的项目执行模拟工具

PEST v1.0(project execution simulation tool)是我们开发的风险驱动的软件项目执行模拟工具(如图3所示).通过该工具,可以很方便地进行项目进度安排和项目风险分析.在用户输入项目信息后,系统可以自动进行进度调整、缓冲分配、项目模拟执行以及相应的数据采集和统计绘图.

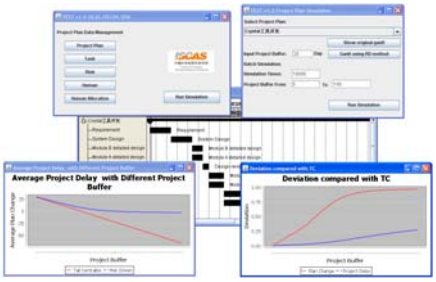


Fig.3 PEST v1.0

图 3 PEST v1.0 工具

工具可以在项目模拟执行过程中自动记录计划变更情况.我们对该工具中涉及的“计划变更点”做出如下定义:

定义(计划变更点). 如果某任务在模拟执行过程中,计划开始日期或计划结束日期发生变化,则记录一个计划变更点;如果计划开始日期和计划结束日期都发生变化,则记录两个计划变更点.

由以上定义可知,一次计划变更可能包括多个计划变更点.

3 实验分析

3.1 案例介绍

我们选取一个具有两个迭代开发周期的软件开发项目“Crystal工具开发”作为实验案例.该项目是为基于组织实体能力的软件过程建模方法(OEC-SPM)开发新版本的原型工具[22,23].项目的任务信息见表 1.任务名称后面的*I*2 表示属于第 2 个迭代开发周期的任务.系统中采用相对日期(相对于项目开始日期,即项目开始第*N*天)来表示任务的计划开始日期,并且在进度安排时不考虑节假日.任务的后置任务集合(post-tasks)可以根据前置任务关系获得.项目的理想工期长度为 130 天.

Table 1 Task information list

表 1 任务信息列表

ID	Task name	Plan start date	Ideal execution time	Critical chain	Pre-Tasks
T_1	Requirement	0	14	Y	—
T_2	System design	14	14	Y	T_1
T_3	Module *A* detailed design	28	7	Y	T_2
T_4	Module *B* detailed design	28	5	N	T_2
T_5	Design review	35	4	Y	T_3 T_4
T_6	Module *A* coding	39	10	Y	T_5
T_7	Module *B* coding	39	8	N	T_5
T_8	Module *A* test	49	14	Y	T_6
T_9	Module *B* test	49	10	N	T_7
T_{10}	Integration test	63	20	Y	T_8 T_9
T_{11}	Requirement *I*2	83	7	Y	T_{10}
T_{12}	System design *I*2	90	7	Y	T_{11}
T_{13}	Design review *I*2	97	2	Y	T_{12}
T_{14}	Module *A* coding *I*2	99	7	Y	T_{13}
T_{15}	Module *B* coding *I*2	99	5	N	T_{13}
T_{16}	Module *A* test *I*2	106	10	Y	T_{14} T_{15}
T_{17}	Module *B* test *I*2	106	7	N	T_{14} T_{15}
T_{18}	Integration test *I*2	116	14	Y	T_{16} T_{17}

该项目计划中被识别的风险信息见表 2,每条记录表示某风险在某任务执行阶段发生的概率 *P* 以及发生后会导致任务延期的百分比 *I*.从风险列表可以看出,该项目不确定性较高,且在项目早期潜在的风险数量和风险影响都较项目后期略高.

PEST 工具在每次进行项目模拟执行时,为了体现风险不确定性的特点,风险发生的概率和影响将取以[表 2 取值±(表 2 取值×10%)]为上下限范围内的随机值.

表 3 列出了任务所分配的人力资源情况.人力资源用其姓名拼音的缩写表示.通过表 3 可以发现,开发任务 14“Module *A* coding *I*2”,任务 15“Module *B* coding *I*2”和测试任务 16“Module *A* test *I*2”,任务 17“Module *B* test *I*2”,由于采用了交叉测试方式,导致任务 14 和任务 17 之间、任务 15 和任务 16 之间产生了由于资源约束而导致的 FS 型进度约束.

Table 2 Risk list

表 2 风险列表

Risk ID	Task ID	*P* (%)	*I* (%)	Risk ID	Task ID	*P* (%)	*I* (%)
1	T_1	80	60	11	T_9	20	20
2	T_1	60	50	12	T_{10}	20	10
3	T_2	60	50	13	T_{11}	80	30
4	T_3	50	50	14	T_{11}	60	20
5	T_4	50	50	15	T_{12}	30	20
6	T_5	50	50	16	T_{14}	40	50
7	T_6	10	40	17	T_{15}	20	40
8	T_6	50	60	18	T_{16}	20	20
9	T_7	40	40	19	T_{17}	20	20
10	T_8	30	30	20	T_{18}	20	10

Table 3 Human resource allocation result

表 3 人力资源分配列表

Task ID	Human resource (name)	Task ID	Human resource (name)
1	HH, QSY	10	HH, DPL
2	HH, LR	11	HH, QSY
3	SZ	12	HH, DPL
4	WSF	13	HH, DPL, SZ, JZ, QSY, WSF, LR
5	JZ, WSF, LR, SZ, DPL, HH, QSY	14	SZ
6	DPL, LR	15	DPL
7	WSF, HH	16	DPL
8	SZ, HH	17	SZ
9	DPL, LR	18	LR, JZ

这里以选取 40 天可用项目缓冲为例来对尾部集中的缓冲分配方法和风险驱动的缓冲分配方法做比较.采用尾部集中的缓冲分配方法形成的进度安排如图 4 所示,通过风险驱动的项目缓冲分配方法进行进度调整后的进度安排如图 5 所示.图中矩形表示任务的理想工期,三角形表示任务的可用缓冲.

由进度图示可以看出,两种缓冲分配方法的项目计划结束日期相同.区别在于尾部集中的缓冲分配方法是将所有可用缓冲集中放置到项目尾端,而风险驱动的项目缓冲分配方法则是依据风险和任务关系将可用项目缓冲分解后放入每个任务的尾部.

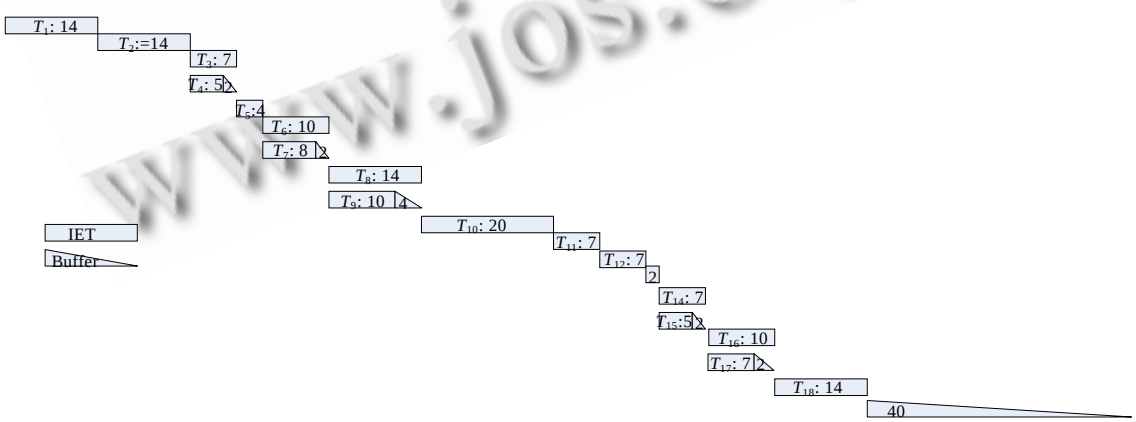


Fig.4 Result of tail centralized project buffer allocation method

图 4 尾部集中的缓冲分配结果

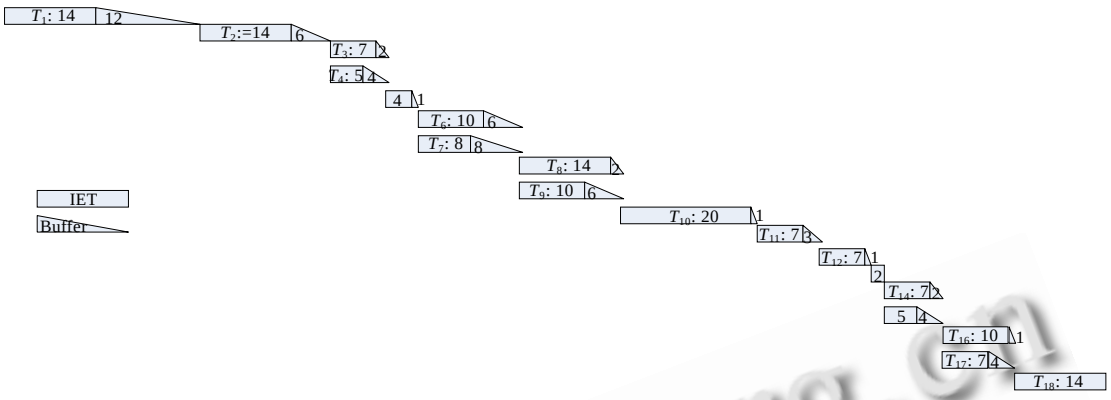


Fig.5 Result of risk driven project buffer allocation method

图 5 风险驱动的缓冲分配结果

3.2 实验结果分析

我们选择可用的项目缓冲范围为 0~100 天来进行模拟实验.在 101 个不同的可用项目缓冲长度下,对尾部集中的缓冲分配方法与风险驱动的项目缓冲分配方法所形成的两种项目进度安排方案分别模拟执行 10 万次,记录两种方案的平均计划变更点数和平均项目完成日期.两种缓冲分配方法简称如下:RD(risk driven,风险驱动的项目缓冲分配方法),TC(tail centralized,尾部集中的项目缓冲分配方法).我们的研究假设为“假设在风险较高的软件开发项目中,根据任务风险的不同分布和风险的影响程度对项目缓冲进行合理分配,相对于尾部集中的缓冲分配方法,可以在对项目的平均完成工期造成较小的影响的同时有效降低计划变更发生的频率”.因此,我们将从计划变更点数和项目平均完成工期两个方面对 RD 方法和 TC 方法进行对比分析.

首先对比分析两种缓冲分配方法在不同的可用缓冲天数下计划变更点数的均值情况.图 6 列出了计划变更点的数量随着可用缓冲的增加而变化的趋势.图中上方直线表示尾部集中的风险分配办法,下方曲线表示风险驱动的项目缓冲分配方法.横坐标为可用缓冲天数(范围 0~100 天),纵坐标为平均计划变更点数.由该图可以看出,当可用缓冲天数较少时,RD 方法较 TC 方法在计划变更点数量上差别不大.随着可用缓冲的增加,RD 方法所导致的计划变更点数明显减少.当可用缓冲达到某个极值时,项目模拟执行的计划变更点数恒定为 0.不过一般情况下,软件项目很难拥有这么多的可用缓冲.

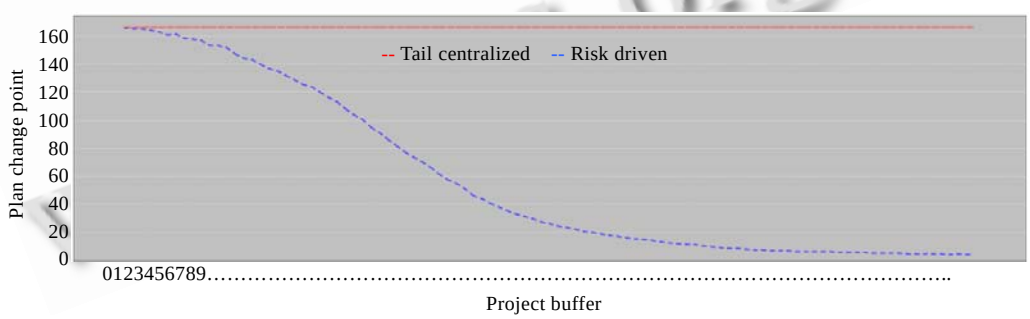


Fig.6 Comparison of average project plan change point

图 6 平均计划变更点数变化趋势示意图

接下来分析两种方法在不同可用项目缓冲下对项目平均完成日期所造成的影响.图 7 列出了两种方法模拟执行后项目的平均结束日期和计划结束日期之间的差别.图中上方曲线代表 RD 方法,下方斜线代表 TC 方法.横坐标为可用缓冲天数(范围 0~100 天),纵坐标为模拟执行项目平均延期天数.

模拟执行项目平均延期天数=模拟执行的平均项目结束日期–项目理想工期–
可用项目缓冲–项目实际开始日期.

由该图可以看出,在可用缓冲天数较少时,两种方法在该项目下导致的平均延期天数差别很小.随着可用缓冲的增加,TC 方法下项目的平均延期天数越来越小,负值表示项目提前完成(TC 方法由其尾部集中的缓冲分配特征决定了其平均执行日期不受可用缓冲长度影响).而 RD 方法下项目的平均延期日期随着可用缓冲的增加也会迅速地变小,最后会稳定在 0 以下的某一数值(该数值由项目最后一个任务的风险情况和该任务的可用缓冲决定).

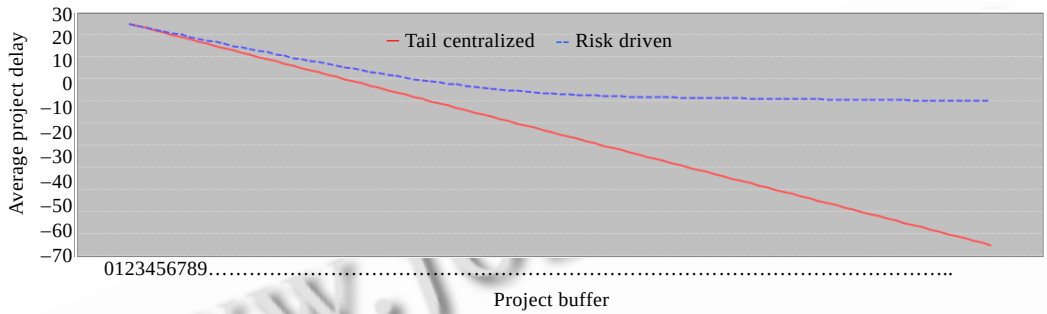


Fig.7 Comparison of average project delay

图 7 平均项目延期天数对比图

最后,将两种方法在计划变更点数和平均项目工期两方面同时进行对比分析.图 8 列出了在不同缓冲天数下,RD 方法在计划变更点数和项目结束日期上与 TC 方法的差距.横坐标是可用缓冲天数(范围 0~100 天),纵坐标是偏差比率.图中上方曲线表示计划变更点数偏差,下方曲线表示项目结束日期偏差.

计划变更点数偏差=(TC 方法的平均计划变更点数–RD 方法的平均计划变更点数)/
TC 方法的平均计划变更点数,

项目结束日期偏差=(RD 方法的平均项目完成日期–TC 方法的平均项目完成日期)/项目计划执行工期.

从图中可以看出,当可用缓冲较少时,RD 方法与 TC 方法在计划变更点数和项目平均执行工期上差别不大.随着可用缓冲的增多,RD 方法相对 TC 方法在减少计划变更点数上的效果愈加显著,同时带来的平均项目延期比率也会相应增高,但是其增长速度远远低于计划变更点数减少的速度.由此可以证明本文最初提出的研究假设:“在风险较高的软件开发项目中,根据任务风险的不同分布和风险的影响程度对项目缓冲进行合理配,相对于尾部集中的缓冲分配方法,可以在对项目的平均完成工期造成较小影响的同时有效降低计划变更发生的频率”.

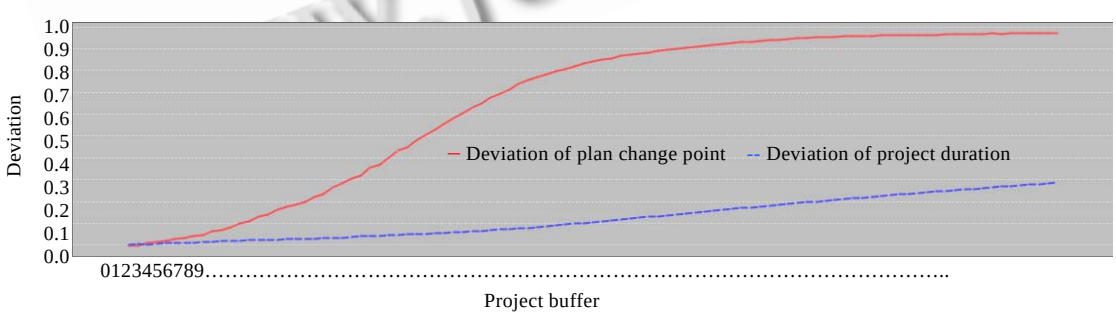


Fig.8 Comparison of project plan change point and project delay

图 8 计划变更点和项目延期情况对比图

RD 方法和模拟工具可为项目经理提供决策支持.如图 9 列出的是图 8 中统计结果的一段(可用缓冲从 40

天~50 天),从图中可以看出,在 40 天~50 天的可用缓冲范围,RD 方法将较 TC 方法平均减少约 78%的计划变更点数,同时,项目工期平均增长约 8%.图 10 列出的是可用缓冲从 20 天~30 天的模拟执行结果.RD 方法将较 TC 方法平均减少约 35%的计划变更点数,同时,项目工期平均增长约 3.7%.这些统计数据可以为项目经理做出相关决策提供信息支持,项目经理可以根据项目特点,综合考虑不同缓冲天数所能获得的计划变更收益和项目工期损耗来为项目选择合适的项目缓冲数目和具体的分配方案.在该案例中,项目经理最终选择 40 天的项目缓冲.

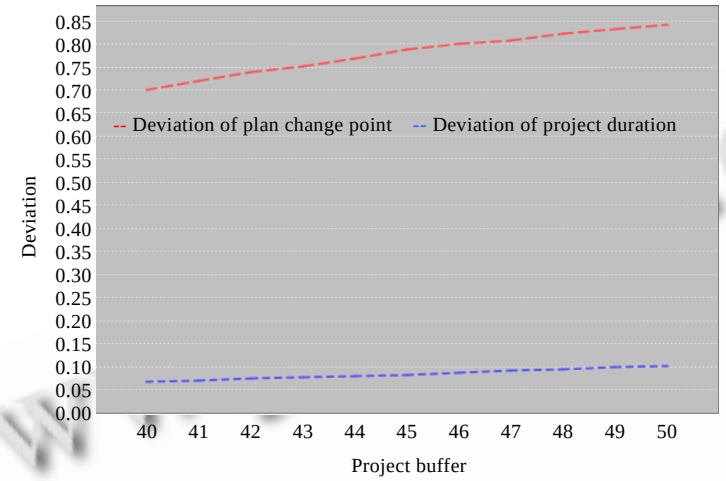


Fig.9 Simulation result when the available project buffer is 40~50 days

图 9 可用缓冲 40 天~50 天的对比结果

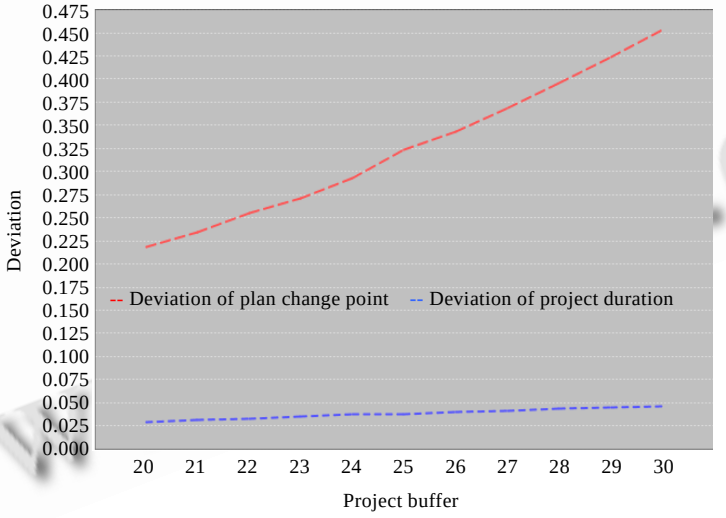


Fig.10 Simulation result when the available project buffer is 20~30 days

图 10 可用缓冲 20 天~30 天的对比结果

通过对以上项目模拟数据的分析可以看出,相对于关键链项目管理中尾部集中的项目缓冲分配方法,风险驱动的项目缓冲分配方法能够在对项目的平均执行工期造成较小影响的同时,显著降低计划变更发生频率.因此,该方法相对传统的关键链缓冲管理方法更适用于高风险、高不确定性的软件开发项目.方法可以帮助企业提高项目计划执行的稳定性,降低计划变更成本.而通过软件项目模拟方法可以有效地帮助项目经理获得较佳项目缓冲长度以及项目缓冲分配方案,进而为项目经理制定项目计划提供有力的决策支持.

4 结论和下一步工作

本文提出了一种综合考虑风险因素和任务约束的项目缓冲分配方法,并通过项目模拟执行对方法的效果进行了验证.实验表明,在不确定性较高的软件项目中,该方法相对于业界流行的关键链项目管理中尾部集中的项目缓冲分配方法,能够在对项目执行工期影响较小的情况下有效降低计划变更的发生频率.该方法除对一般软件项目管理具有辅助意义以外,尤其适用于不确定性较高、风险较高、同时对项目进度执行稳定性要求较高的软件开发项目.方法和工具可以协助项目经理进行合理的项目缓冲估计和优化项目缓冲分配,进而提高此类项目的计划可信性和执行稳定性.

下一步的工作将对方法进行适应性扩展,如对方法适用的项目风险特征进行量化,逐步引入风险之间的关系和风险的时效性,以及在进度安排和项目执行模拟时加入对成本变更因素的考虑.

References:

[1] Goldratt EM. Critical Chain. The North River Press, 1997.

[2] Rand GK, Critical chain: The theory of constraints applied to project management. Int'l Journal of Project Management, 2000,18(3): 173−177.

[3] Van de Vonder S, Demeulemeester E, Herroelen W, Leus R. The use of buffers in project management: The trade-off between stability and makespan in resource-constrained project scheduling. Int'l Journal of Production Economics, 2005,97(2):227−240.

[4] Kerzner H. Project Management: A System Approach to Planning, Scheduling, and Controlling. 9th ed., Beijing: Publishing House of Electronics Industry, 2006.

[5] Willy H. Critical chain project scheduling—Do not oversimplify. Project Management Journal, 2002,33(4):46−60.

[6] Schwalbe K. Information Technology Project Management. 4th ed., Beijing: China Machine Press, 2006.

[7] PMI. A Guide to the Project Management Body of Knowledge: (Pmbok Guide). 3rd ed., Newtown: Project Management Institute, 2008.

[8] Ashtiani B, Jalali GR, Aryanezhad MB, Makui A. A new approach for buffer sizing in critical chain scheduling. In: Helander M, Xie M, Jiao R, Tan KC, eds. Proc. of the 2007 IEEE Int'l Conf. on Industrial Engineering and Engineering Management. Los Alamitos: IEEE Computer Society Press, 2007. 1037−1041.

[9] Li K, Chen YX. Applying critical chain in project scheduling and estimating buffer size based on fuzzy technique. In: Helander M, Xie M, Jiao R, Tan KC, eds. Proc. of the 2007 IEEE Int'l Conf. on Industrial Engineering and Engineering Management. Los Alamitos: IEEE Computer Society Press, 2007. 1068−1072.

[10] Steyn H. Project management applications of the theory of constraints beyond critical chain scheduling. Int'l Journal of Project Management, 2002,20(1):75−80.

[11] Cusumano MA. Microsoft Secrets: How the World's Most Powerful Software Company Create Technology, Shapes Markets and Manages People. New York: Free Press, 1995.

[12] Boehm B, Jain A. A value-based software process framework. In: Wang Q, Pfahl D, Raffo DM, Wernick P, eds. Proc. of the Int'l Software Process Workshop and Int'l Workshop on Software Process Simulation and Modeling 2006. LNCS 3966, Berlin, Heidelberg: Springer-Verlag, 2006. 1−10.

[13] Koch R. The 80/20 Principle: The Secret to Success by Achieving More with Less. New York: Doubleday Business, 1999.

[14] Xie LZ. A project scheduling method based on human resource availability. In: Knowledge Systems Institute Graduate School, ed. Proc. of the 20th Int'l Conf. on Software Engineering and Knowledge Engineering. San Francisco: Knowledge Systems Institute, 2008. 161−166.

[15] Wang Q, Li MS. Software process management: Practices in China. In: Li MS, Boehm B, Osterweil LJ, eds. Proc. of the Int'l Software Process Workshop 2005. LNCS 3840, Berlin, Herdelberg: Springer-Verlag, 2005. 317−331.

[16] Li MS, Huang M, Shu FD, Li J. A risk-driven method for eXtreme progreamming release planning. In: Osterweil LJ, Rombach HD, Soffa ML, eds. Proc. of the 28th Int'l Conf. on Software Engineering. ACM Press, 2006.

[17] Huang M, Shu FD, Li MS. A risk-driven method for prioritizing requirements in iteration development. Journal of Software, 2006, 17(12):2450−2460 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/2450.htm [doi: 10.1360/jos172450]

[18] IEEE Standard for Software Project Management Plans. IEEE Std 1058-1998, 1998.

[19] IEEE/EIA guide for information technology-software life cycle processes-implementation considerations. IEEE/EIA 12207.1-1997, 1997

[20] Parkinson N. Parkinson's Law: And Other Studies in Administration. Columbia: Ballantine Books, 1964.

[21] Sugimori Y, Kusunoki K, Cho F, Uchikawwa S. Toyota production system and kanban system materialization of just-in-time and respect-for-human system. Int'l Journal of Production Research, 1977,15(6):553−564.

[22] Zhang L, Wang Q, Xiao JC, Ruan L, Xie LZ, Li MS. A tool to create process-agents for OEC-SPM from historical project data .In: Wang Q, Phahl D, Raffo DM, eds. Proc. of the Int'l Conf. on Software Process 2007. LNCS 4470, Berlin, Herdelberg: Springer-Verlag, 2007. 84−95.

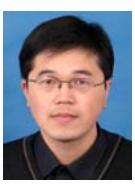
[23] Xiao JC, Wang Q, Li MS, Zhang L, Liu DP. An organization-entity capability based software process modeling method. Journal of Software, 2008,19(3):533−544 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/533.htm [doi: 10.3724/SP.J.1001.2008.00533]

附中文参考文献:

[17] 黄蒙,舒风笛,李明树.一种风险驱动的迭代开发需求优先级排序方法.软件学报,2006,17(12):2450−2460. http://www.jos.org.cn/1000-9825/17/2450.htm [doi: 10.1360/jos172450]

[23] 肖俊超,王青,李明树,张蕾,刘大鹏.一种基于组织实体能力的软件过程建模方法.软件学报,2008,19(3):533−544. http://www.jos.org.cn/1000-9825/19/533.htm [doi: 10.3724/SP.J.1001.2008.00533]

谢利子(1982－),男,安徽灵壁人,博士生,主要研究领域为软件过程技术,过程管理.



肖俊超(1978－),男,博士,助理研究员,主要研究领域为软件过程技术,过程管理.

王青(1964－),女,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为软件过程技术与质量管理.