RUP估算过程模型

杜云梅[1] 李师贤[2]
(华南师范大学增城学院计算机系 广州 511363)[1]
(中山大学信息科学与技术学院计算机系 广州 510260)[2]

摘 要 软件行业估算追踪记录显示软件项目的失败率仍很高,估算问题是基本的原因之一。估算方法的创新没有出现期望的突破,而通过可控的过程,可以获得期望的结果。提出了一个过程模型,用于指导软件项目展开一系列估算相关的活动。该过程模型包括两部分,一是RUP估算过程,其详细描述了RUP开发模型里每个开发管理阶段应如何进行估算;二是用贝叶斯网络对RUP估算过程模型建立图形化推理模型,它能有效地用于估算分析、交流、权衡以及风险预测等。RUP估算过程解决了估算活动的定义问题,但不便于形成清晰的估算视图。软件估算的特点很适合用贝叶斯网络进行建模。贝叶斯工作量估算模型是RUP估算过程模型的抽象;ESFQ模型详细建模了软件项目关键因素之间的权衡关系。案例分析证明了该过程模型的适用性。

关键词 软件估算,估算过程,贝叶斯网络
中图法分类号 TP311 **文献标识码** A

Estimation Process Model for RUP Project

DU Yun-mei[1] LI Shi-xian[2]
(Department of Computer Science, Zengcheng College of SCNU, Guangzhou 511363, China)[1]
(Department of Computer Science, Sun Yat-sen University, Guangzhou 510260, China)[2]

Abstract The track record of software industry estimates shows that the failure rate of software projects is still high, and the estimate is one of the basic reasons. The innovation of estimation method does not expect a breakthrough. Through a controlled process, you can get the desired results. Proposed a process model to guide software project to launch a series of estimates related activities. The process model consists of two parts. The first is RUP estimation process model, which gives a detailed description of how to estimate for each development and management of stage. The second is to create a graphical step-by-step process model using Bayesian network inference model, which can be effectively used to estimate analysis, communication, balance, risk prediction. The step guide is to solve the problems of the definition of estimation activities, but it is not easy to form a clear view of the estimate. The features of software estimation are suitable for modeling with Bayesian network. BN workload estimation model is abstraction of the step guides. ESFQ model is the detailed model of the trade-off relationship between the critical factors of software projects. Case studies prove the applicability of the process model.

Keywords Software estimation, Estimation process, Bayesian network

1 引言

软件行业估算追踪记录显示:软件项目失败率仍很高,预算超支与进度失控仍是普遍现象。分析发现,成功的项目都具有"可控"的共同特点,不合理估算下的管理"失控"是项目失败的主要原因之一。

对于项目是否可控,估算显然是关键技术。一直以来,估算新方法被视为"银弹",人们试图提出一种全新的估算技术来解决那些令人头痛的估算问题。而从估算现状来看,这些方法可能有用,但没出现期望的突破。以往提出的估算方法都有各自的优缺点,适用于特定的应用领域或项目阶段。没有一种单一的方法能解决估算过程中的所有问题。

现在的软件开发过程与项目管理过程,都缺乏对估算活动的充分定义。而估算不是项目周期某个时刻发生的一次性步骤,而是贯穿开发与管理的一个过程,包括在项目初始阶段如何利用有限的信息辅助前期的重要决策,也包括随着项目进展不断精化估算,以支持项目计划与控制;包括开发周期中对项目数据的记录,提供执行活动追踪和计划调整参数。人们之前把估算等同于项目开始前对成本与进度的一次性预测,其实伴随性估算在项目资源配置、计划的及时调整和积极的跟踪控制方面,比开始的猜测更有意义。

软件的开发与管理缺乏良好的估算过程定义,就会产生估算问题继而导致项目管理问题与开发质量问题。估算和开发管理一样,是软件项目周期中展开的3个方面的活动,估算

到稿日期:2012-08-13 返修日期:2013-01-05 本文受广东省自然科学基金项目(S2011010003442)资助。
杜云梅 女,副教授,主要研究方向为软件工程、软件成本估算;李师贤 男,教授,博士生导师,主要研究方向为软件工程、本体、物联网。

伴随着开发管理过程，需展开一系列的活动。

而且估算的目的是为了更好地支持软件开发与管理，提高软件项目成功率，为此估算急需解决3个方面的问题：一是早期项目估算，二是支持项目管理决策的多因素之间的权衡分析，第三就是随着项目进展精化估算的过程。

所以，我们需要一个配合并支持项目开发与管理过程的软件估算过程，它能明确定义从项目起始到结束的每个开发与管理阶段应展开的估算活动，包括选用适当的估算方法，如何得到并提交估算结果，历史数据的采集与利用，当前估算支持的项目管理活动等。

通过估算过程的定义，可解决开发与管理中缺乏估算活动定义的问题，同时也为软件估算本身急需解决的3个关键问题提供可行方案，因为它解决了长期困扰估算者的历史数据问题、估算方法的比较与选择问题。

估算过程要指导实践，为项目的估算活动提供指南，就必须具体而详细，而这样，估算者又很容易陷入细节而失去清晰的估算整体视图。为解决此问题，我们先分析估算的特点。

估算不是业务目标和承诺，而是对项目持续时间或成本的预测。现实中估算的对象不是一成不变的，软件项目从起始到交付会发生显著变化，即估算对象存在很大的不确定性。

另外，软件项目的开发活动是人的创造性活动，所开发的软件产品也是一种逻辑产品，影响项目结果的因素广泛而复杂，而且很多因素难以度量或无法直接观测。

还有，估算不是求解问题的唯一解。软件项目的成本、进度、交付功能和质量之间存在着非线性的关系，可以根据约束条件或干系人的合理意愿，在这几者之间进行一定范围的权衡取舍。

所以估算是利用不完全的直接或间接的历史数据、广泛存在的专家经验对包含极大不确定性的多因素进行推理的过程。

而贝叶斯网络(Bayesian Network，BN)是描述因果关系的有向无环图，能对包含不确定性的因果关系建立推理模型，明确建模影响因子；可以从原因推理结果，也可以从结果推理原因；根据新证据推翻以前的信任；在不完全数据下进行预测；结合主观信任和客观数据；基于可视、可审计的推理得出结论。

贝叶斯网络的这些特点，很适合用于对软件估算进行建模，从而解决估算的不确定性推理、不完全的历史数据、与专家经验相结合等方面的问题。

本文提出了一个过程模型，用于指导软件项目展开一系列估算相关的活动。该过程模型包括两部分，一是RUP估算过程模型，其步骤式指南详细描述了每个开发管理阶段应如何进行估算；二是用贝叶斯网络对RUP估算过程模型建立图形化推理模型，它能有效地用于估算分析、交流、权衡以及风险预测等。

过程模型搭起了骨架，其中涉及多种方法如肌肉，而隐含在过程模型中的估算思想则如血液，它们一起形成一个有机整体，以可控过程的力量，帮助机构对软件项目估算做出预测和承诺。

2 软件开发、管理与估算

软件开发过程、项目管理过程和估算过程，是同一软件项目过程中展开的3个方面的工作，彼此协调，不能割裂。估算提供项目管理活动所需的参数，并让干系人达成共识。有效的项目管理为估算提供经验与数据，控制项目达到估算目标。估算与管理相互促进，一起推动开发工作顺利进行。开发过程产生经验与数据，反作用于开发与管理过程。

现代软件开发实践中依照顺序的瀑布模型进行开发会产生多方面的问题，原因在于完全顺序的执行方式不适合复杂的软件开发工作，现在的软件开发都采用了迭代的思想来组织开发过程。迭代开发可以降低风险，灵活处理不断变化的需求，实现持续集成，尽早得到有关项目信息与认识。

本文以RUP软件开发与管理过程为例，给出了RUP估算过程模型。

RUP(Rational Unified Process，统一软件开发过程)是一个面向对象且基于网络的程序开发方法论。它是用例驱动的、以架构为中心的迭代式增量开发过程。RUP项目生命周期在时间上被分解为4个顺序的阶段，分别是：初始阶段(Inception)、细化阶段(Elaboration)、构造阶段(Construction)和交付阶段(Transition)，项目经过4个阶段交付产品。而每个阶段又分解为多次迭代，每次迭代相当于一个袖珍项目，通过一系列的工作流(需求、分析、设计、实现、测试)产生一个增量结果，所有迭代产生的增量积累成最终产品版本。不同阶段迭代的侧重点有所不同。每个阶段都有主要里程碑，每次迭代也都有次要里程碑。达到里程碑的准则标志着可以进入下一阶段或迭代。

RUP定义清晰严明，实用性好，但缺少估算活动的定义。在其开发过程模型上整合估算活动，形成RUP估算过程。

在初始阶段项目需要时进行猜测性估算，支持初期决策；在初始阶段里程碑处进行第一次正式估算，这时仍然是比较粗略的，以制定项目初步计划；在细化阶段里程碑处进行详细估算，可制定详细的项目计划；而在构造阶段的每次迭代完成时，可以进一步精化估算，对详细估算中的某些参量进行调整。在构造阶段里程碑处进行产品质量的估算、维护估算等；而在交付阶段里程碑处汇总项目数据，评价估算过程。

估算提供项目计划参数，为项目配置合理资源提供依据；估算提供跟踪基线，以加强项目控制。而项目估算不能自已变成现实，只有在估算所假设的项目管理水平、人员能力经验级别等条件下正常地管理控制和运作项目，才能达到估算所指示的目标。

3 RUP估算过程

从时间流程上可以将RUP过程看成一系列“阶段-门槛”(Stage-Gate)过程，每个阶段都有一定的软件开发活动，这些活动可以组织成一到多次迭代来完成，如图1所示。门槛则是用于确定项目可否完成上一个阶段(或迭代)而进入下一个阶段(或迭代)的出口准则。

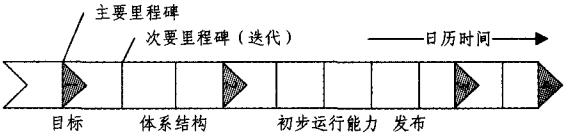

图1 RUP阶段-门槛过程

从估算的角度来看，应该在主要“门槛”1(目标)、“门槛”2(体系结构)、“门槛”3(初步运行能力)处对软件项目进行重估，因为到此时项目的不确定性有了质的改善，并以里程碑的形式正式呈现。在“门槛”4(发布)处对整个估算过程进行总

结与评价；而在次要里程碑（一次迭代完成）处对估算值进行周期性修正，因为一次迭代经历了全部核心工作流，产生了本项目新的生产率等经验数据。

根据RUP开发与管理过程，将估算过程划分为多个阶段（猜测性估算、粗略估算、详细估算、精化估算、修正估算），每个阶段包括3项任务：

A. 估算，根据阶段可用信息，确定可选用的估算方法、估算、提交估算结果的步骤；

B. 数据记录与整理，形成项目关键因素的连续记录；

C. 估算支持的项目管理活动。

其中估算任务包括以下几个步骤：

1. 分析软件需求并考虑需求的变化与增长速度；
2. 估算规模；
3. 识别机构与项目属性，估算软件开发生产率；
4. 估算工作量；
5. 估算进度和人员需求；
6. 估算开发成本；
7. 提交估算结果并使之被接受。

与开发管理阶段相适应的RUP估算过程如表1所列。过程中涉及多种估算方法、需采集的一定的项目数据，还有估算阶段能支持的项目管理活动，这些方法、数据和活动的详细定义参见相关文献。

表1 RUP估算过程

RUP阶段	任务	估算活动、数据收集活动、项目管理活动
初始阶段I		猜测性估算：一些目标系统信息、类似项目信息
	估算方法	在必要时采用以下估算方法得到估算N： ②“T恤衫”式估算，粗略的特性筛选 ③项目多维分类，规模的初步猜测 ④行业经验法则，行业平均人员、成本、进度 ⑤类比，查找类似项目数据进行推理 以0.25N～4N提交估算范围
	数据记录	①记录项目信息 保存估算记录
	项目管理	初期购买/开发决策、放弃/继续决策、项目范围决策 初步项目计划（高层管理）
初始阶段里程碑1		粗略估算：已批准的市场需求
	估算方法	⑥产品WBS自顶向下估算得到估算结果 ⑦活动WBS自顶向下估算得到估算结果 综合③+④，⑥，⑦，专家小组采用宽带Delphi过程来收敛得到单点估算值N 以0.5N～2.0N提交估算范围
	数据记录	人员用于参与项目时间 记录首次迭代完成的市场需求数、用例数/市场需求 每个市场需求开发/测试/文档编写上的平均工作小时数
	项目管理	生成活动WBS 形成初步软件开发计划、人员计划、预算 首次迭代计划 估算只用于批准完成产品设计所需的预算，不用于其他预算与承诺
细化阶段II	估算方法	②、⑤、COCOMO早期设计模型
	数据记录	人员工作时间分配，每个需求编写文档平均工作小时数 每市场需求转换为工程需求平均工作小时数
	项目管理	逐步形成详细的产品WBS，活动WBS 特性的实现优先级
细化阶段里程碑2		详细估算：详细需求（80%），稳定架构
	估算方法	⑧估算人员基于产品WBS自底向上估算 ⑨功能点方法估算规模 ⑩COCOMO模型估算工作量、成本、进度与人员（机构历史数据校准） 分析迭代上述估算，估算结果收敛于5%以内，平均值作为标称值N 0.8N～1.25N提交估算范围
	数据记录	人员工作时间分配、需求蔓延百分比、实现的用例数 每个用例开发/测试/文档编写上的平均工作小时数
	项目管理	详细的软件开发计划、人员计划、测试计划 项目跟踪与控制、PERT图、甘特图、网络图，利用COCOMO工具辅助决策 按照1.0N分配预算，0.25N分配紧急预算 只公布估算范围最大值1.25N，不用于对外承诺
构造阶段III		精化估算：构造阶段前期迭代完成
	估算方法	执行人员分别估算自己承担工作，根据产品WBS自底向上估算，可以采用⑪PERT估算法 比较分配预算与自己的估算，N=(2*较高估算+较低估算)/3 0.9N～1.1N提交估算范围，可公布1.1N并做出承诺
	数据记录	⑫人员工作时间分配、需求蔓延百分比、实现的用例数 发现缺陷数、修正的缺陷数 每个变更请求开发/测试/文档编写上的平均工作小时数 每个类开发/审查/测试所需的平均工作小时数 每个Web页（报告、对话框、数据库表等）平均整体工作小时数
	项目管理	⑬任务清单，跟踪控制，利用COCOMO工具权衡分析 ⑭公众的估算回顾 软件开发计划、人员计划、进度调整，网络图调整

RUP 阶段	任务	估算活动、数据收集活动、项目管理活动
构造阶段 III		修正估算：后期迭代
	估算方法	计算完成效率＝目前实际工作量/目前计划工作量 剩余工作量＝计划的剩余工作量/完成效率 添加遗漏任务的工作量得到 N 1.0N～1.1N 提交估算范围，可公布 1.0N 并做出承诺
	数据记录	⑫人员工作时间分配、需求蔓延百分比、实现的用例数 发现缺陷数、修正的缺陷数 修正前期迭代中的量 每个缺陷修正/回归测试平均所需的工作小时数 每个配置设置项的平均工作小时数
	项目管理	⑬跟踪控制，软件开发计划、人员计划、进度调整，网络图调整 利用 COCOMO 工具权衡分析，⑭公众的估算回顾
构造阶段里程碑 3	估算方法	产品质量估算，文档等后续工作估算，维护估算
	数据记录	人员工作时间分配、需求蔓延百分比、发现缺陷数、修正的缺陷数 每个缺陷修正/回归测试平均所需的工作小时数
	项目管理	发布计划，⑭公众的估算回顾
交付阶段 IV	估算方法	产品质量估算调整，维护估算 ⑮评价估算活动有效性
	数据记录	人员工作时间分配、需求蔓延百分比、发现缺陷数、修正的缺陷数 每个缺陷修正/回归测试平均所需的工作小时数 计算缺陷清除率
	项目管理	回顾整理项目数据，⑭公众的估算回顾
重大变化进行重估		在需求增加、主要需求定义发生变化、人员变更、进度目标改变时 在哪个阶段的增加或变更，就按哪个阶段的估算方法重估
项目完成		收集项目实际结果数据，包括代码行、工作量、进度、用例数、各阶段工作量、生产率、比重、错误清除率等，并进行整理汇总 ⑭公众的估算回顾，⑮评估每个估算值的准确度，⑯估算合理性检查 分析主要误差的来源 评估是否可以用更小的工作量获得同样的准确度 提出对估算过程的修改意见

4 贝叶斯网络推理模型

4.1 贝叶斯网络

贝叶斯网络是描述因果关系的有向无环图，包括结点和边，结点表示变量，边表示变量间的因果关系，由起因结点（父结点）指向结果结点（子结点）。没有父结点的结点称为根结点。每个结点有一个相关的结点概率表（NodeProbability Table，NPT），其给出了父结点输出组合下该结点每种可能输出发生的条件概率。

有两种方式可以确定 NPT 中的概率，一是根据客观统计数据，二是依靠专家主观概率。确定概率信息之后可以用贝叶斯网络进行多种分析，而在取得实际观测数据后，可以在 BN 中的任何位置输入任意多个观测结果（条件），进而根据 Bayes 定理更新未观测变量的结点概率表 NPT。

4.2 工作量估算模型

表 1 中详细描述了在 RUP 项目过程中如何得到估算结果、应展开的数据收集和估算能支持的项目管理工作。为了得到清晰的估算视图，进行了权衡与风险分析，用 BN 图形模式描述的 RUP 项目过程模型如图 2 所示。

图 2 是一个示意图，显示了表 1 估算过程会经历的 4 大阶段，每一阶段的估算采用多种估算方法得到多个估算结果（如 Effort_i1，Effort_i2 表示第 i 阶段的两个不同方法得到的估算结果），然后对每个估算结果赋予相应的权重，加权平均得到本阶段的工作量估算 Effort_i，上一阶段的估算结果作为下一阶段估算的一个参量。其中 Effort_i1 体现了本模型充分利用迭代项目早期产生的数据指导后期估算的思想。

Data_i－1 表示前一阶段观测到的项目数据。

Size_i 表示本阶段对软件项目规模的估算，以功能点（FP）为单位。

Productivity_i－1 表示由前一阶段迭代产生的数据而确定的最新项目生产率，以人月/功能（PM/FP）点为单位，即完成 1 功能点需要多少人月的工作量（注意这里的生产率是通常理解的 FP/PM 生产率的倒数，这样处理是为了方便建立贝叶斯网络模型）。

那么，Effort_i1＝Size_i＊Productivity_i－1，即 Effort_i1 是由第 i 阶段估算的软件规模乘以项目团队生产率得到的。

可以发现，此过程模型具有自相似的特点。4 个阶段的估算都具有如图 3 所示的基本结构。

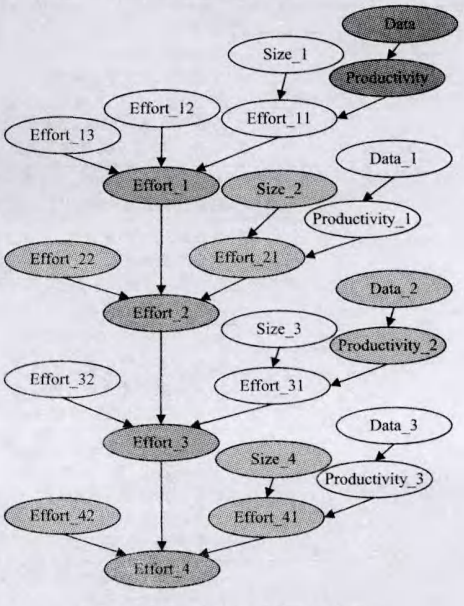

图 2 RUP_BN 总体估算过程模型

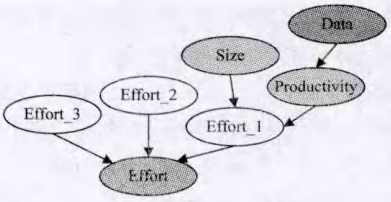

图 3　RUP 估算过程的基本结构

因而，只须对这一基本结构进行细化，确定每个结点的 NPT，也就是确定 BN 中结点间的因果关系，最终形成 BN 工作量估算模型，如图 4 所示。因为在此模型中，历史数据是以观测结果的形式在需要时输入到模型中的，所以图 4 中不需要专门设置 Data 结点。

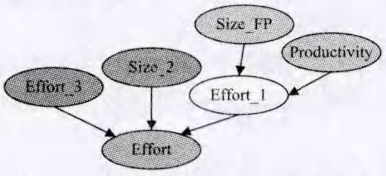

图 4　RUP_BN 工作量估算模型

同样，Effort_2 和 Effort_3 表示不同估算方法得到的工作量估算结果，Size_FP 是以功能点(FP)为单位的软件规模，Productivity 表示生产率(PM/FP)。这 4 个根结点代表的变量其取值都被定义为 5 个级别(Very Low，Low，Medium，High，Very High)，代表估算值与实际值之间的差距，即(估算值-实际值)/实际值，4 个根结点的 NPT 是一样的，如表 2 所列。

表 2　RUP_BN 工作量估算模型的 NPT

级别	概率	含义(估算值－实际值)/实际值
Very Low	0.25	<－50%
Low	0.3	－50%到－10%
Medium	0.25	±10%
High	0.1	10%到 50%
Very High	0.1	>50%

而 Effort_1 是由 Size_FP * Productivity 得到，所以其 NPT 也是由其父结点的概率表经该表达式计算得到。

Effort 是由 Effort_1、Effort_2 和 Effort_3 加权平均得到，其 NPT 由表达式 wmean(1.0，effort3，1.0，effort2，4.0，effort1)得到(wmean()是 BN 工具中加权平均函数)。Effort_1 被赋予较高的权重，即相对更相信由项目自身数据得到估算结果。

由专家经验确定贝叶斯网络模型(见图 4)中所有结点的结点概率表之后的 RUP_BN 工作量估算模型如图 5 所示。

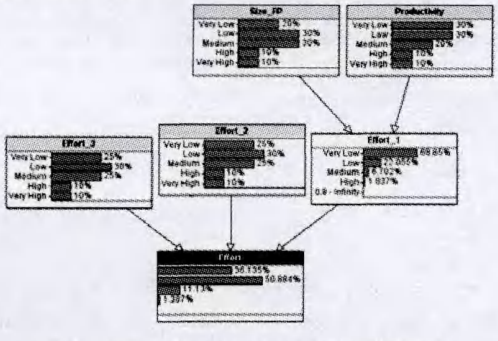

图 5　显示 NPT 的 RUP_BN 工作量估算模型

图 5 中显示了每个结点的输出概率分布情况。当有可用历史数据时，就将其作为观测结果输入到该模型中，它可调整其他结点 NPT 中的概率。

应用 RUP_BN 工作量估算模型可以进行很多有用的评价与分析工作，进而可以改进估算过程。假设项目结果显示某阶段估算的工作量与实际相比非常低(Very Low)，将此观测值输入到模型中 Effort 结点，重新计算后的贝叶斯网络模型如图 6 所示。

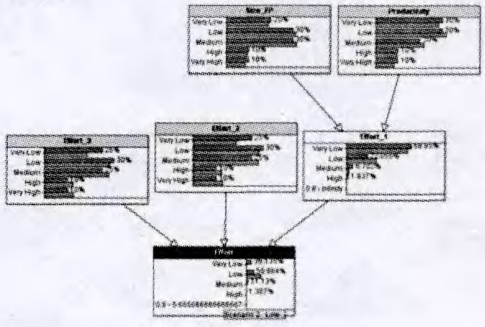

图 6　Effort 结点设为“Very Low”

可以看出其他各结点的输出概率均向 Low 和 Very Low 偏移。

进一步的对比发现，其他两种方法得到的估算结果(Effort_2 与 Effort_3)与实际结果比较接近(±10%以内)，再将此观测数据输入到模型中，重新计算后的结果如图 7 所示。

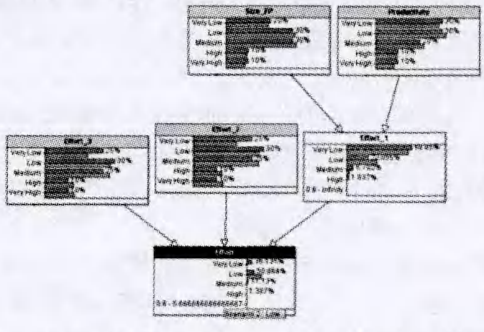

图 7　Effor_2 和 Effort_3 结点设为“Medium”

结果显示，Effotr_1 等级为“Very Low”的概率为 1，Size_FP 与结点的“Low”与“Very Low” 等级的概率都变得很大。可以推断很可能是本阶段的软件规模估算太低，或前一阶段因为某种原因使团队生产率(FP/PM)太高，导致 Productivity 的 PM/FP 值太小。

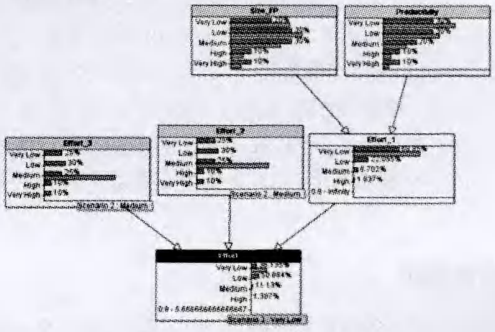

图 8　Size_FP 结点设为“Medium”

接着结合本阶段产生的代码行与完成的功能，进一步分析本阶段的规模估算，发现 Size_FP 的估算与实际比较接近

(±10%以内)。输入此观测结果重新计算后的模型如图 8 所示,可以看出这一阶段的生产率(FP/PM)明显低于上一阶段,从而导致本阶段的工作量估算"Very Low"。

4.3 ESFQ 模型:工作量、进度、交付功能与质量估算

估算是个复杂的问题,不仅涉及到很多的影响因子(有些因子还难于量化或直接观测),而且还有工作量、进度、交付功能与质量之间的权衡问题。RUP_BN 工作量估算模型还不足于支持这类的权衡分析,所以用 BN 对工作量、规模与生产率这部分进行了更详细的建模,如图 9 所示。

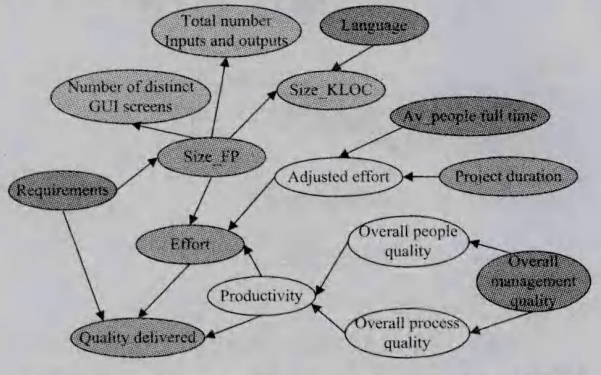

图 9 RUP_BN_ESFQ(工作量、进度、交付功能与质量)估算模型

其中 Size_FP 是以功能点为单位的软件规模,可由 GUI 屏幕数、输入输出数和源代码千行数(KLOC)来指示。KLOC 与 FP 之间的关系与具体的语言有关。生产率(Productivity)由管理质量、整体过程质量和整体人员质量因子决定。进度(Project Duration)、平均参与人数(Av_people full time)和工作量之间存在非线性的关系。而交付的质量(Quality delivered)由需求质量、生产率和项目所用的工作量决定。图 9 中结点比较多,每个结点的 NPT 涉及到 BN 工具的具体功能与表述方式,在此就不一一缀述。

应用此模型,可以进行多种类型的预测、风险分析,并能进行工作量、进度、交付功能与质量之间的权衡,详见第 5 节的案例分析。

之所以将得到估算结果的过程(见表 1)与进行分析权衡的贝叶斯网络模型(图 2,图 4 和图 9)独立开来,是因为如果要让贝叶斯网络模型进行具体值的估算,BN 结点的 NPT 必须包括所有可能的结果范围,如规模结点(Size_FP)可能需要 0～30000FP 的范围,这么宽广的范围反而增加了项目的不确定性。对于某个具体的项目而言,其规模的变化范围要小得多,可能只是 20～100FP 就足够了。由 RUP 估算过程得到规模、工作量和进度等估算结果,贝叶斯网络估算模型不讨论不同项目间工作量、进度、规模等参数的巨大差异,只考虑本项目参数相对的变化范围,从而避免了在贝叶斯网络模型中引入不必要的不确定性,又不会使模型局限于特定的项目范围。

5 案例分析

目前商业与政府的很多软件项目,都采用招投标的运作方式。对这类项目来说,投标前的估算与中标后的预算都相当重要,它们一定程度上决定了机构能否获得这个项目、获得项目后能否保证有适当利润并交付让客户满意的产品从而维护机构声誉。基于 BN 的估算过程模型能较好地用于这类项目投标前的估算与随后仍然重要的权衡分析。

招投标项目在估算时,需求基本已明确并且以正式文档的形式呈现。这对估算来说是相当有利的。表 1 的估算过程就可以从"详细估算"开始,即相当于进行到 RUP 的细化阶段,已完成了大部分需求。

应用图 4 的估算模型和表 1 中相应阶段的估算方法估算如下:

先对需求进行功能点分析,假设经过分析,某项目需求包括 3000 个功能点,平均开发生产率为每人月 10 功能点(即 0.1PM/FP),从而可简单计算 Effort_1 为 300PM。

应用估算的功能点数和 COCOMO 模型,输入影响因子等级,得到工作量估算 Effort_2 为 410PM。

再由估算人员对需求中详细的工作分解结构(WBS),结合行业经验,进行自底向上的估算,得到工作量估算 Effort_3 为 240PM。

应用图 4 的估算模型中设定的权重,得到工作量估算 Effort 为 310PM。如果上述 3 个工作量估算结果相差太远,可以先分析原因并重新进行估算,再计算 Effort。

此时就可以在 0.8～1.25Effort 的范围进行竞标工作。

一旦中标得到该项目,就要在标的约束条件下完成项目。此时,工作量、进度、交付质量与功能之间的权衡分析就非常重要。应用图 9 的工作量、进度、交付功能与质量估算模型,能很好地支持这几个因素之间的分析,从而辅助管理者做出正确的决策。

作为示例,创建"基本"与"权衡"两个场景,"基本"场景用于对比,除了软件规模(Size_FP)结点外,不输入其他观测值。"权衡"场景输入某些约束条件(观测值),从而对比在这些约束条件下会对其他结点产生什么样的影响。

首先,假设两种场景下都需交付 3000 功能点,但"权衡"场景下要求交付质量为"perfact",观测结果的输入如表 3 所列。图 10—图 13 分别显示了过程、人员、进度与工作量结点的变化。

表 3 权衡分析 1 输入的观测数据

	基本	权衡
Size_FP	3000	3000
Quality delivered		perfact

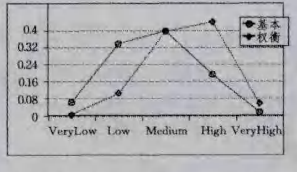

图 10 Overall peoople quality

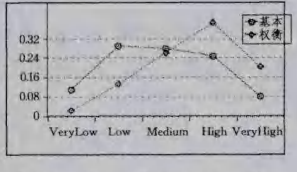

图 11 Overall process quality

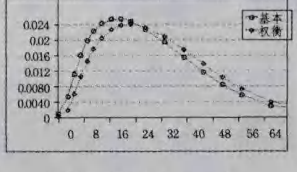

图 12 Project duration

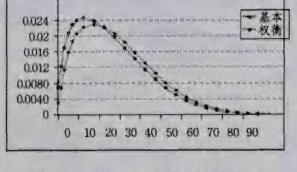

图 13 Av_people full time

从图中可以看出,提高交付质量要求,会使工作量与进度

稍有增长，过程与人的质量要求也更高。

现假设人员与过程质量只是“medium”，观测结果的输入如表 4 所列。图 14－图 17 分别显示了过程、人员、进度与工作量结点的变化。

表 4　权衡分析 2 输入的观测数据

	基本	权衡
Size_FP	3000	3000
Quality delivered		perfact
overall peoople quality		medium
overall process quality		medium

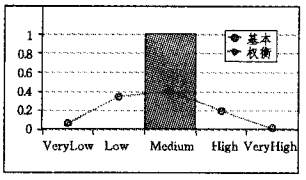

图 14　Overall peoople quality

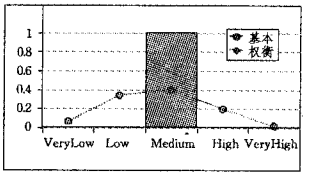

图 15　Overall process quality

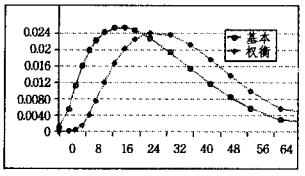

图 16　Project duration

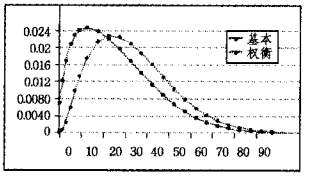

图 17　Av_people full time

从图中可以看出，提高交付质量要求，但人与过程质量只是中等，会使工作量与进度有明显增长。

再假设有强硬的进度压力，要求项目在 10 个月内完成，观测结果的输入如表 5 所列。图 18－图 20 分别显示了过程、人员与工作量结点的变化。

表 5　权衡分析 3 输入的观测数据

	基本	权衡
Size_FP	3000	3000
Quality delivered		perfact
project duration		10

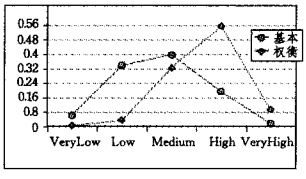

图 18　Overall peoople quality

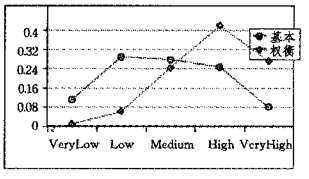

图 19　Overall process quality

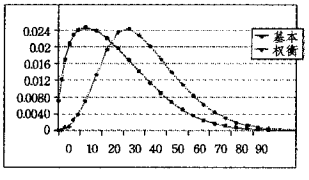

图 20　Av_people full time

从图中可以看出，提高交付质量要求且有强硬的最后期限要求，会使团队规模有明显增长，而且对人与过程的质量要求很高。

再假设现在只能投入 7 人全职参与项目，观测结果的输入如表 6 所列。图 21、图 22 分别显示了过程、人员质量结点的变化。

表 6　权衡分析 4 输入的观测数据

	基本	权衡
Size_FP	3000	3000
Quality delivered		perfact
project duration		10
av_people full time		7

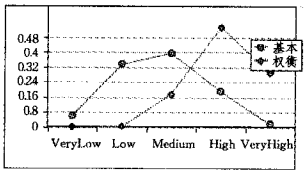

图 21　Overall peoople quality

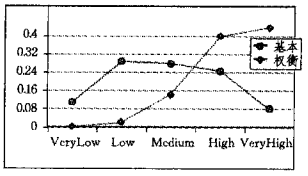

图 22　Overall process quality

从图中可以看出，若提高交付质量要求且有强硬的最后期限要求，并且只有 7 个全职人员可以参加，那么对人与过程的质量要求就非常高。

现假设机构无法做到让人员质量与过程质量达到非常高的级别，那么在只有 7 人可以参与项目并且必须在 10 个月内交付 3000 个功能点的软件，交付质量会怎么样呢？观测结果的输入如表 7 所列。图 23 显示了交付质量结点的变化。

表 7　权衡分析 5 输入的观测数据

	基本	权衡
Size_FP	3000	3000
Quality delivered		
project duration		10
av_people full time		7
overall peoople quality		medium
overall process quality		medium

从图中可以看出，在这些约束条件下交付的软件质量非常差。

最后假设，有 10 个月的强硬的进度要求，只能投入 7 人全职参与项目，而且人员与过程的质量只是“medium”，而且要求完成质量为“perfact”，现在能权衡的只有交付的功能。观测结果的输入如表 8 所列。图 24 显示了软件规模结点的变化。从图中可以看出，在这些约束条件下就只能交付 1/10 的功能。

表 8　权衡分析 6 输入的观测数据

	基本	权衡
Size_FP	3000	
Quality delivered		perfact
project duration		10
av_people full time		7
overall peoople quality		medium
overall process quality		medium

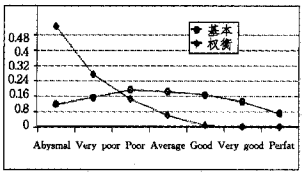

图 23　Quality delivered

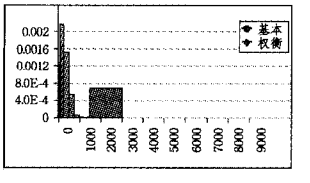

图 24　Size_FP

干系人通过上述图形化的分析方式，可以很直观地看到某种项目决策（如进度压缩）会对项目其他方面造成的影响，从而做出取舍。

结束语　普遍的软件项目预算超支和进度失控问题并不是单纯的估算方法问题。软件估算的特点及其与开发和管理

的关系，决定了估算必须以过程的方式来组织。估算和开发管理彼此协调，相互促进，一起形成了一个更加完整的有机整体，提高了软件项目成功率。

本文根据软件开发与管理过程对估算活动的需要，同时考虑到估算为更好地支持开发与管理要解决的关键问题，给出了估算过程模型，它包括两部分，一是 RUP 估算过程，二是应用贝叶斯网络对估算过程建立的推理模型。简言之，主要工作在于：

1. 估算和开发管理一样，是软件项目过程中展开的 3 个方面的活动，三者是密不可分、相互促进的。不能孤立地研究估算方法来解决估算相关的全部问题。通过估算过程模型方案，解决了开发与管理中缺乏估算活动定义的问题，同时也为估算本身急需解决的 3 个问题提供了可行途径。

2. 详细的步骤式估算过程，为干系人提供了清晰的估算线路图。

3. 贝叶斯网络推理模型以图形化的方式，结合历史数据、专家经验和项目本身的数据，进行分析、权衡和评价。

本文将得到估算结果的 RUP 估算过程模型与进行分析权衡的贝叶斯网络模型独立开来，一方面可获得详细的估算指南，另一方面可获得清晰的分析视图，在不失细节的情况下，保持了贝叶斯网络模型的简单性，避免了在贝叶斯网络模型中引入不必要的不确定性，但又不会使模型局限于特定的项目范围。

案例分析验证了本估算过程模型具有良好的适用性，对项目开发与管理有极大的推动作用。

参 考 文 献

[1] Brooks F P. The Mythical Man-Month[M]. Addison-Wesley Professional, 1995

[2] Boehm B W. Software Engineering Economics [M]. Prentice Hall PTR, 1981

[3] Boehm B W, Abts C, Brown A W, et al. Software Cost Estimation with COCOMO II[M]. Prentice Hall PTR, 2000

[4] Jones C, Engineering S. The state of the Art in 2005(Version 5) [M]. Software Productivity Research WhitePaper, February 2005

[5] Jrgensen M, Shepperd M. A systematic review of software development cost estimation studies[J]. IEEE Transactions on Software Engineering, 2007, 33(1): 33-53

[6] Stutzke R D. Estimating Software-Intensive Systems[M]. Upper Saddle River, NJ: Addison-Wesley, 2005

[7] Goldratt, Eliyahu M. Critical Chain[M]. MA: The North River Press, 1997

[8] Jones C. Software Assessments. Benchmarks, and Best Practices, Reading[M]. MA: Addison-Wesley, 2000

[9] Project Management Institute. A guide to the project management body of knowledge (PMBOK Guide) (Fourth Edition) [M]. ANSI/PMI 99-001-2008

[10] Shepperd M. Software project economics: a roadmap[C]// Future of Software Engineering(FOSE'07). IEEE, 2007

[11] Yang Da, Wang Qing, Li Ming-shu, et al. A Survey on Software Cost Estimation in the Chinese Software Industry[C]// ESEM'08. Kaiserslautern, Germany, October 2008

[12] Fraser S, Jrgensen M, Boehm B, et al. The role of judgment in software estimation[C]// 31st International Conference on Software Engineering. Companion Volume, ICSE 2009. 2009: 13-17

[13] Ahmed F, Bouktif S, Serhani A, et al. Integrating function point project information for improving the accuracy of effort estimation[C]// The 2nd International Conference on Advanced Engineering Computing and Applications in Sciences, ADVCOMP 2008. 2008: 193-198

[14] Wang Hao, Peng Fei, Zhang Chao, et al. Software project level estimation model framework based on Bayesian belief networks [C]// International Conference on Quality Software. 2006: 209-216

[15] Yang Y, Jesal B, Boehm B. Value-based processes for COTS-based applications[J]. IEEE Software, 2005, 22(4): 54-62

[16] Jones C. Patterns of Software Systems Failure and Success[M]. International Thomson Press, 1996

[17] AgenaRisk 5.0 User Manual[OL]. www.agenarisk.com

[18] Steve McConnell. 软件估算—"黑匣子"揭秘[M]. 北京：电子工业出版社，2007

[19] Jones C. 软件项目估计[M]. 北京：电子工业出版社，2008

[20] Jacobson I, Booch G, Rumbaugh J. 统一软件开发过程[M]. 北京：机械工业出版社，2002

[21] 李明树，何梅，杨达，等. 软件成本估算方法及应用[J]. 软件学报，2007, 18(4): 775-795

(上接第 15 页)

[61] Inkumsah K, Xie T. Improving Structural Testing of Object-Oriented Programs via Integrating Evolutionary Testing and Symbolic Execution [C]// Proceedings of the International Confe-rence on Automated Software Engineering. 2008: 297-306

[62] Baars A, Harman M, Hassoun Y, et al. Symbolic Search-based Testing [C]// Proceedings of the International Conference on Automated Software Engineering. 2011: 53-62

[63] Briand LC, Labiche Y, Buist K, et al. Automating Impact Analysis and Regression Test Selection based on UML Designs [C]// Proceedings of the International Conference on Software Maintenance. 2002: 252-261

[64] Briand LC, Labiche Y, He S. Automating Regression Test Selection based on UML Designs [J]. Journal of Information and Software Technology, 2009, 51(1): 16-30

[65] Wu Y, Offutt J. Maintaining Evolving Component-based Software with UML [C]// Proceedings of the European Conference on Software Maintenance and Reengineering. 2003: 133-142

[66] Yu Y, Jones J, Harrold M. An Empirical Study of the Effects of Test-suite Reduction on Fault Localization [C]// Proceedings of the International Conference on Software Engineering. 2008: 201-210

[67] Artzi S, Dolby J, Tip F, et al. Directed Test Generation for Effective Fault Localization [C]// Proceedings of the International Symposium on Software Testing and Analysis. 2010: 49-60

[68] Zhang X, Gu Q, Chen X, et al. A Study of Relative Redundancy in Test-suite Reduction while Retaining or Improving Fault Localization Effectiveness [C]// Proceedings of the Symposium on Applied Computing. 2010: 2229-2236