基于规划图的过程活动流实时规划

柴啸龙[1,2] 姜云飞[1] 陈蔼祥[2] 毛明志[1]
(中山大学软件研究所 广州 510275)[1] (广东商学院数学与计算科学学院 广州 510320)[2]

摘 要 软件过程的自动化计算技术是近年来出现的新兴研究方向,对软件过程的研究有着重要的影响。基于规划图方法设计了一种过程活动实时规划系统,能对过程活动的安排做出实时自动规划,能在实际执行中对过程活动流中每一过程活动所出现的各种可预见的执行效果做出实时反应,使软件开发始终处于有序的过程控制中,并最终取得既定目标。另外实际项目中软件的过程活动常会出现重复执行或者多次执行,已有的一些相关软件过程规划都没谈及该问题,所提出的实时规划算法较好地解决了该问题。

关键词 软件过程,智能规划,规划图,过程活动流规划,实时反应
中图法分类号 TP311,TP18 **文献标识码** A

Planning of Process Action Flow with Real-time Reaction Based on Planning Graph

CHAI Xiao-long[1,2] JIANG Yun-fei[1] CHEN Ai-xiang[2] MAO Ming-zhi[1]
(Institute of Software, Sun Yat-Sen University, Guangzhou 510275, China)[1]
(School of Mathematics and Computing Science, Guangdong University of Business Studies, Guangzhou 510320, China)[2]

Abstract The software process control with intelligence also with its computing technology is a new research direction in recent years. And it is promising that it will exert an influence on the study of software process. PARPS (Process Action Real-Time Planning System) based on planning graph was presented. It can make a planning of process action arrangement automatically. Furthermore, the algorithm of process action planning has the ability of real-time reaction. It can make a real time reaction to all kinds of real effects after executing any ordered process action in the process action flow. Yet all the real effects should have been foreseen. Thus the software development will be under the control of ordered process actions all the time and will reach the established goal at last. In practice, the process actions of the software item are often being taken repetitiously. Yet this case hasn't been dealt with in the related works of software process planning. The real-time planning algorithm presented in this paper solved the problem.

Keywords Software process, Intelligence planning, Planning graph, Planning of process action flow, Real time reaction

1 引言

近年来软件过程的自动化计算技术得到迅速发展[1,2]。美国南加利福尼亚大学的 Barry Boehm 在 2005 年分析到关于软件过程技术的发展时,认为软件过程的自控制和充分计算将是软件过程管理的一个发展趋势[3],近年来软件过程的自动控制计算技术的迅速发展也印证了这一观点。

由于软件规模不断增长,软件过程活动迅速增多,如何对软件过程活动做出准确高效的规划已成为软件开发面临的重要课题。由于不同活动执行效果相互影响,互为条件,甚至相互制约,在开发过程中若能将众多过程活动的安排转换成完善的过程活动执行规划,将减少软件开发中过程活动安排的盲目性,并且每一活动的执行都将获得明确的全局意识。

基于规划图算法[4]给出了一种过程活动实时规划方法,设计了过程活动实时规划系统(Process Action Real-Time Planning System, PARPS),以对过程活动流在具体执行中出现的每种可预见的执行效果做出实时反应,使软件开发始终处于有序控制中,最终取得既定目标。

国内较早开展软件和软件过程自动化相关工作的有梅宏、吕建、李明树、王青等。他们在软件和软件过程自动化、软件的智能化开发环境、软件过程的主动度量等方面做了很多相关的工作[5-12]。

国外最早给出软件过程规划系统的可能是德国 Oldenburg 大学的 Heinrich Jasper,他在 1994 年提出在构建软件过程模型时可以结合主动知识库技术。他在系统中设计了一个事件检测器,可以检测到软件开发中的各种事件,并触发相应的软件过程活动[13]。加拿大 Calgary 大学的 Ahmed Emran 等给出了一种软件的重新规划算法[14],当给定的一些软件各项具体目标的计划,如果突然出现了如成本被低估等意外,能够自动规划出一个新的软件目标。但这一工作没有涉及软件

到稿日期:2008-07-10 返修日期:2009-02-03 本文受国家自然科学基金(60773201),广东省自然科学基金(06301003)资助。
柴啸龙 博士生,讲师,主要研究方向为智能规划、软件过程控制;姜云飞 教授,博士生导师,主要研究方向为自动推理、智能规划、基于模型的诊断;陈蔼祥 博士,讲师,主要研究方向为智能规划与诊断、演化计算;毛明志 博士,副教授,主要研究方向为软件工程技术。

目标在开发过程中如何实现的具体操作和步骤。而本文更侧重于计算具体的规划过程。另外美国 Texas 大学 Dallas 分校的 Manish Gupta 在 2004 年给出一种自动生成测试用例的规划算法[15]，但该工作以固定的测试目标为导向，不具备实时和应变规划能力。因此国内外同类的软件过程自动化工作与本文工作其实并没有太多可比性。

2 相关概念

定义 1(状态及其表示) 状态是软件过程活动执行之前和执行之后的软件开发情况，它由命题公式集描述。用 $P(s)$ 表示用来描述状态 s 的命题公式集。

定义 2(过程活动及其前件和后件) 过程活动是用来实现软件开发状态转变的操作。过程活动有前件和后件，过程活动 a 的前件是执行 a 所需要满足的前提条件，用 $Precond(a)$ 表示，a 的后件是执行 a 之后，所产生的效果，用 $Effect(a)$ 表示。$Precond(a)$ 和 $Effect(a)$ 都是命题公式集。引入一个特殊的过程活动，称为维持活动(*Maintenance Action*)，它是一个空活动，因此执行这样的过程活动后，原命题维持不变。

定义 3(过程活动互斥) 互斥的过程活动不能同时执行。称活动 a_1 和 a_2 是互斥的，若满足下列条件之一：

(1) $\exists p_1 \in Effect(a_1)$，$\exists p_2 \in Effect(a_2)$，使得 p_1 和 p_2 是相互否定的命题公式。

(2) $\exists p_1 \in Effect(a_1)$，$\exists p_2 \in Precond(a_2)$，使得 p_1 和 p_2 是相互否定的命题公式。

(3) $\exists p_1 \in Precond(a_1)$，$\exists p_2 \in Effect(a_2)$，使得 p_1 和 p_2 是相互否定的命题公式。

(4) $\exists p_1 \in Precond(a_1)$，$\exists p_2 \in Precond(a_2)$，使得 p_1 和 p_2 是相互否定的命题公式。

(5) $\exists p_1 \in Precond(a_1)$，$\exists p_2 \in Precond(a_2)$，且 p_1 和 p_2 分别是由互斥的过程活动 b_1 和 b_2 产生。

定义 4(过程活动和过程活动集的执行) 称过程活动 a 在状态 s 下是可执行的，如果满足 $Precond(a) \subseteq P(s)$。同样称过程活动集 $\pi=\{a_1,\cdots,a_k\}$ 在状态 s 下是可执行的，若满足以下两条件：

(1) π 中任意两个过程活动都不是互斥的。

(2) $\forall a_i \in \pi$，都有 $Precond(a_i) \subseteq P(s)$。

如果过程活动集 π 在状态 s 下是可以执行的，则定义 π 的前提条件为：$Precond(\pi)=\bigcup_{a\in\pi} Precond(a)$。定义 π 的执行效果为 $Effect(\pi)=\bigcup_{a\in\pi} Effect(a)$。

定义 5(有效过程活动流) 称 $\Pi=\langle\pi_1,\pi_2,\cdots,\pi_k\rangle$ 是从状态 s 出发的一个有效过程活动流，如果它满足以下 3 个条件：

(1) π_i 是一个过程活动集，且其中的任意两个过程活动都不是互斥的，$i=1,\cdots,k$。

(2) π_1 在状态 s 下是可执行的。

(3) $\text{Effect}(\pi_i) \supseteq Precond(\pi_{i+1})$，$i=1,\cdots,k-1$。

定义 6(可达) 称状态 s 是从初始状态 s_0 出发可达的，如果存在从 s_0 出发的一个有效过程活动流 $\Pi=\langle\pi_1,\pi_2,\cdots,\pi_k\rangle$，使得 $Effect(\pi_k) \supseteq P(s)$。称命题集 P 是从初始状态 s_0 出发可达的，如果满足：$\exists$ 状态 s'，s' 是从初始状态 s_0 出发可达的，且 $P \subseteq P(s')$。由于有维持活动 Maintenance Action 的存在，因此 $P(s_0)$ 也是从 s_0 出发可达的。

定义 7(命题层和活动层) 令 $P_0=P(s_0)$ 为第 0 层命题层，如果第 i 层命题层 P_i 已经得到了，则规定第 $i+1$ 层命题层 P_{i+1} 和第 $i+1$ 层过程活动层 A_{i+1} 如下：如果 $\exists a \in A$，$Precond(a) \subseteq P_i$，且在 P_i 中的 $Precond(a)$ 是从初始状态 s_0 出发可达的，则 $Effect(a) \subseteq P_{i+1}$。同时有：$a \in A_{i+1}$。

定义 8(不动点) 含有多个命题层的已扩展规划图 G。如果有 $P_m=P_{m+1}$ 成立，则称第 $m+1$ 层命题层 P_{m+1} 是规划图 G 的一个不动点层，将层数最小的不动点层定义为图 G 的不动点。

定义 9(实时情形集 Φ) 将过程活动集 A 中每一过程活动或并行的过程活动集在执行后，预计可能出现的各种实际执行效果定义为 A 的实时情形集，用 Φ_A 标记，表示为：$\Phi_A=\{\varphi_i \mid \varphi_i=\langle\pi_i:\diamondsuit_1=P_1,\cdots,\diamondsuit_{m(i)}=P_{m(i)}\rangle,\pi_i$ 为 A 中的过程活动或过程活动集，且 $m(i)>1\}$。其中 $P_1,\cdots,P_{m(i)}$ 分别表示执行 π_i 后，所可能出现的 $m(i)$ 种不同的状态命题集。

定义 10(过程活动流实时规划问题) 把过程活动流实时规划问题定义为五元组 (s_0,g,A,Σ,Φ_A)，其中 s_0 为初始状态，g 为目标状态所需实现的公式集，A 为有限过程活动集。Σ 包括过程活动集与状态命题集之间的前件关系、后件关系、活动之间以及命题之间的互斥约束关系。Φ_A 为 A 的实时情形集。

定义 11(完应对解) 完应对解是实时规划问题 (s_0,g,A,Σ,Φ_A) 的一个具备实时反应能力的解。它能对 Φ_A 中所预计到的过程活动的所有不同实际执行结果都做出实时反应，使得通过完应对解的控制，最后总能达到包含目标公式集 g 的目标状态。完应对解是 (s_0,g,A,Σ,Φ_A) 的一个规划解集合。对过程活动的每种实际执行结果，总能在完应对解中匹配到一个规划解，实时给出过程活动流控制，以取得既定目标。

定义 12(拟象过程活动) 过程活动 $a_i \in A$，a_i 的执行可能会出现 k 种不同的实际执行效果，分别记为：$Effect_1(a_i)$，$\cdots$，$Effect_k(a_i)$。定义一个 A 中并没有的新过程活动集 $\{a_{i1},\cdots,a_{ik}\}$，使得：

$Precond(a_{ij})=Precond(a_i)$，$Effect(a_{ij})=Effect_j(a_i)$，$j=1,2,\cdots,k$。

即新过程活动集：$\{a_{i1}\cdots a_{ik}\}$ 中的每个过程活动都只有 1 种执行效果。将 a_{ij} 称为 a_i 的一个拟象过程活动，$j=1,2,\cdots,k$。把 $\{a_{i1},\cdots,a_{ik}\}$ 称为 a_i 的拟象过程活动集，并记为 $Fact(a_i)$。如果 a_i 的执行只出现 1 种执行效果，也规定 $Fact(a_i)=a_i$。

定义 13(元象过程活动) 过程活动 $a_i \in A$，且 $Fact(a_i)=\{a_{i1},\cdots,a_{ik}\}$。则 $\forall a_{ij} \in \{a_{i1},\cdots,a_{ik}\}$，称 a_i 是 a_{ij} 的元象过程活动，并记为 $Orig(a_{ij})=a_i$。

定义 14(活动的确定性执行) 过程活动集 π 在状态 s 下是可执行的，且满足以下两条件：

(1) $\forall a_1,a_2 \in \pi$，$Orig(a_1) \neq Orig(a_2)$；

(2)过程活动集 π 在状态 s 下的执行结果是唯一的。

则把在状态 s 下过程活动集 π 的执行称为确定性执行，并把执行后的新状态用 $Run(s,\pi)$ 表示。

3 过程活动流的规划图方法

过程活动流的规划图方法主要有 3 个模块：(1)过程活动

规划图的扩展；(2)过程活动规划图上的过程活动流提取。(3)系统的集成与过程活动流的生成。下面分别介绍。

3.1 过程活动规划图的扩展

设 s_0 为初始状态，当执行一个过程活动之后，会得到一个新状态 s_1，然后再执行新的过程活动得到 s_2，依此类推可得到一个有序的状态序列$\langle s_0, s_1, \cdots, s_n\rangle$。如果 $g \subseteq P(s_n)$，即系统在状态 s_n 下满足了目标要求。设过程活动 a 的前提条件 $Precond(a)=\{Precond_1(a), \cdots, Precond_k(a)\}$都已存在或被满足，则 a 可被执行，产生效果命题 $Effect(a)=\{Effect_1(a), \cdots, Effect_m(a)\}$，如图 1 所示。

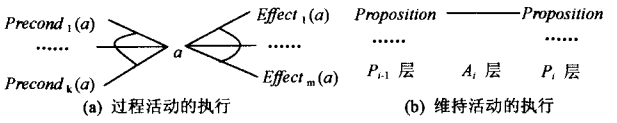

图 1 过程活动的执行带来命题层的扩展

若 $a \in A_{i+1}$，将 $Effect(a)$ 添加到 P_{i+1} 中后，再通过连接弧记录下 A_{i+1} 中的 a 与 P_{i+1} 中的 $Effect(a)$ 命题集的效果联接关系。再把所有这样能在 P_i 命题集环境下能得以执行的 A_{i+1} 中的 a 与其在 P_i 中的前提条件 $Precond(a)$ 里的命题通过相应的前提条件连接弧记录下其前件联接关系。另外由于存在作为空活动的维持活动，一个命题在执行维持活动后，原命题仍然保持存在。因此所有 P_i 中的命题，通过执行维持活动，都得以保留到 P_{i+1} 中。因此总有 $P_i \subseteq P_{i+1}$。将 P_i 中的命题用连接线越过过程活动层 A_{i+1}，直接连接到 P_{i+1} 中相应的相同命题，如图 1(b)所示。这样可以利用各个过程活动的前提条件和效果产生一个命题层和过程活动层交错出现的过程活动规划图。

3.2 规划图上的过程活动流提取

将过程活动规划图扩展到一定层数，并至少扩展到使最后一层命题层 P_k 满足 $P_k \supseteq g$。然后可以进行过程活动流的提取，并观察是否提取成功。首先在最后一层过程活动层 A_k 中选择过程活动子集 π_k，使：

(1) $\forall a_1, a_2 \in \pi_k$，$a_1$ 与 a_2 在 A_k 中不是互斥的过程活动对；

(2) $\bigcup_{a \in \pi_k} Effect(a) \supseteq g$；

(3) $\bigcup_{a \in \pi_k} Precond(a) \subseteq P_{k-1}$。

然后将 $\bigcup_{a \in \pi_k} Precond(a)$ 当成上面的 g，继续向前层做倒序提取，按照同样的方法选择 A_{k-1} 中的过程活动子集 π_{k-1}，若碰到提取失败，则回溯到上一步并重新做过程活动子集的选择，直至提取出一个完整的过程活动流 $\Pi=\langle \pi_1, \pi_2, \cdots, \pi_k\rangle$，$\Pi$ 是从初始状态 s_0 出发的有效过程活动流，且满足 $Effect(\pi_k) \supseteq g$。

3.3 系统集成与过程活动流的生成

将软件过程活动流的规划问题先转化为更适合处理的形式，再通过调用过程活动流的扩展模块与过程活动流的提取模块，实现对过程活动流规划问题的求解。

步骤 1 对过程活动集 A 进行重新设定。

(1)从状态 s_0 出发，到目标公式集 g 的实现过程中，将不可能相干的过程活动从 A 中删除。

(2)拆分或组合 A 中的过程活动。如果过程活动太大，在活动执行中间会影响或受影响于别的活动的执行，这样的过程活动需进行拆分。如果 a 与 b 是两个独立的过程活动，它们之间存在强连接先后关系(即执行完 a 后，接着执行 b，且 a 与 b 中间无别的过程活动阻隔，a 的执行也不直接影响 b 以外的过程活动)，则 PARPS 把过程活动 a 与 b 在计算上视为 1 个过程活动 c，且满足：$Precond(c)=Precond(a)$；$Effect(c)=Effect(b)$。

步骤 2 映射集 Σ 的生成。

对于新生成的 A，$\forall a \in A$，将$\langle Precond, a, Precond(a)\rangle$与$\langle Effect, a, Effect(a)\rangle$作为关系元素存入 Σ 中。令 $formula(a)=Precond(a) \cup Effect(a)$，将 $\bigcup_{a \in A} formula(a)$ 中的命题互斥对集合存入 Σ 中，根据定义 3 给定的判断方法，计算出过程活动互斥对的集合，将它存入 Σ 中。

步骤 3 调用过程活动规划图的扩展模块生成过程活动的扩展规划图 G，然后调用过程活动流的提取模块，在扩展图 G 上进行过程活动流的提取，若提取成功，返回一个过程活动流 Π，如果提取失败，则在已有扩展规划图 G 的基础上继续进行扩展，生成新的图 G，然后再提取，依此类推。但是 PARPS 不允许规划图无限制扩展，最多扩展到不动点为止，对有限过程活动的实时规划问题$(s_0, g, A, \Sigma, \Phi_A)$而言，不动点是一定存在的，它在有限层数内会出现。

4 过程活动流实时规划系统

过程活动集 A 中过程活动的执行结果可能是不确定的，每个过程活动都可能出现多种不同的实时执行效果，因此问题的解必须具备实时反应能力，对执行过程活动流出现的各种可预见实际效果都能做出实时反应，使最终仍能达到目标状态。

4.1 PARPS 的设计

为求解问题$(s_0, g, A, \Sigma, \Phi_A)$，过程活动流实时规划系统主要设计为 5 个步骤模块，按顺序描述如下：

步骤 1 根据过程活动集 A 和实时情形集 Φ_A 生成新的过程活动集 A'，然后生成相应于 A' 的新的映射与约束关系集 Σ'。同时建立过程活动替换索引表 TABLE。

令 $A' \leftarrow A, \Sigma' \leftarrow \Sigma$。$\forall a_i \in A'$，根据 Φ_A，取出 a_i 所有可能的不同执行结果，每一种执行结果对应 a_i 的一个拟象过程活动。设 a_i 有 k 种不同执行结果，若 $k=1$，不做任何改变，若 $k>1$，将 k 种不同执行结果分别记为：$Effect_1(a_i), \cdots, Effect_k(a_i)$。然后将 A' 中的 a_i 替换为 a_i 的拟象过程活动集 $\{a_{i1}, \cdots, a_{ik}\}$，且满足：

$Precond(a_{ij})=Precond(a_i)$，$Effect(a_{ij})=Effect_j(a_i)$，$j=1,2,\cdots,k$。

根据以上公式建立$\{a_{i1}, \cdots, a_{ik}\}$中每个拟象活动的前件和后件映射，加入 Σ' 中，依据定义 3 重新计算 A' 的互斥动作对，将互斥关系集存入 Σ' 中，同时将 Σ' 中含有 a_i 的前件和后件映射删去。在 A' 中加入 $a_{i1}, \cdots, a_{ik}$，并删去 A' 中的 a_i。与此同时，建立过程活动替换索引表 TABLE。$\forall a_i \in A'$，若将 a_i 替换为其拟象过程活动集$\{a_{i1}, \cdots, a_{ik}\}$，$(k>1)$。则将有序对$\langle a_i, \{a_{i1}, \cdots, a_{ik}\}\rangle$存入 TABLE 中。索引表建好后，通过它可查找到：

$\forall a_{ij} \in \{a_{i1}, \cdots, a_{ik}\}$，$Orig(a_{ij})=a_i$；$Fact(a_i)=\{a_{i1}, \cdots, a_{ik}\}$。

步骤 2 对问题$(s_0, g, A, \Sigma, \Phi_A)$，在 A' 和 Σ' 都生成完毕

后，基于规划图技术进行过程活动规划图 G 的扩展。扩展后的规划图 G 至少扩展到最后一层命题层包含所有目标命题，G 最多扩展到不动点出现。

步骤 3　在所得扩展规划图 G 上进行解提取。

当转入解提取时，设 G 最后一层命题层为 P_k，依据规划图算法提取此时 G 中所包含的所有的有效解，设一共有 L 个有效解，组成有效解集合：$\{\Pi_i \mid \Pi_i = \langle \pi_{i1}, \pi_{i2}, \cdots, \pi_{ik} \rangle, i=1, \cdots, L\}$，记为 Π_k^*。

步骤 4　根据 Π_k^*，判断原问题 $(s_0, g, A, \Sigma, \Phi_A)$ 是否存在完应对解，若存在则求出。

根据 Π_k^* 若能找到完应对解，则成功返回，若找不到完应对解，则返回步骤 2，在已扩展图 G 基础上继续扩展，然后再执行步骤 3，步骤 4，依此类推，G 最多扩展到不动点出现。下面给出根据 Π_k^* 计算 $(s_t, g, A, \Sigma, \Phi_A)$ 的完应对解方法。其中 s_t 是从 s_0 出发的任一可达状态。只需用 s_0 替换 s_t 即可得到原问题 $(s_0, g, A, \Sigma, \Phi_A)$ 的解。该算法是递归方法，以下为其描述。

设 $(s_t, g, A, \Sigma, \Phi_A)$ 有完应对解 $SOLVE_1$，则 $SOLVE_1$ 是 Π_k^* 的一个满足以下条件的最小子集（这里的最小是指：去掉其中任一元素后，条件都不再满足），所要求条件是：

$\forall \Pi \in SOLVE_1$，设 Π 的第 1 层过程活动集为 π_1，设 $\pi_1 = \{a_1, \cdots, a_k\}$。

令 $\pi_1^{*1} = \{\{a_1', \cdots, a_k'\} \{a_i' \in Fact(Orig(a_i)), i=1, \cdots, k\}$。

令 $\pi_1^{*2} = \{\pi_1 \mid \forall \Pi \in SOLVE_1, \pi_1$ 为 Π 的第 1 层过程活动集$\}$。

则有 $\pi_1^{*1} = \pi_1^{*2}$。且 $\forall \pi_s \in \pi_1^{*1}$，下列两项至少有一个成立：(1) $P(Run(s_t, \pi_s)) \supseteq g$；(2) 用 s^* 表示状态 $Run(s_t, \pi_s)$，则 $(s^*, g, A, \Sigma, \Phi_A)$ 有完应对解存在。

步骤 5　将所求得的完应对解转换为过程活动流的实时规划控制树 ControlTree。

如果得到 $(s_0, g, A, \Sigma, \Phi_A)$ 的一个完应对解 $SOLVE$，执行 $SOLVE$ 中的每一层过程活动或过程活动集后，对每种实际执行效果，$SOLVE$ 中都存在一相应规划解，使得刚好能够匹配上实时出现的这一执行结果，但 $SOLVE$ 的数据结构是规划解集合，因此每次实时反应都要在 $SOLVE$ 上进行搜索和匹配，这样会影响实时效率，可以将所得到的完应对解 $SOLVE$ 转换为过程活动流的实时反应控制树 $ControlTree$，转换后不再进行搜索即可完成过程活动流的实时反应。$ControlTree$ 的节点代表过程活动，节点引出的线代表过程活动的执行效果，由命题集表示，一个节点引出的多条分支代表过程活动产生的不同的执行效果。分支后面联接的节点表示在过程活动的实际执行效果出来后，所采取执行的下一步的过程活动。将 $SOLVE$ 转换为 $ControlTree$ 的过程一共有 4 部分，分别描述如下：

(1) 任取 $SOLVE$ 中一个解 Π，设 Π 的第 1 层过程活动集为 $\pi_1 = \{a_1, \cdots, a_m\}$。

令 $\pi_1^* = \{\{a_1', \cdots, a_m'\} \mid a_i' = Orig(a_i), i=1, \cdots m\}$，把 s_0 和过程活动集 π_1^* 联接在一起作为 $ControlTree$ 的树根，此时树根是 $ControlTree$ 中唯一节点，也是唯一叶节点（尚未扩展出分支的节点），把树根节点的标记添加到叶节点索引 $LeafIndex$ 之中。

(2) 任取 $LeafIndex$ 中的一个叶节点标记所对应的叶节点 $Leaf_i$，设 $Leaf_i$ 代表的过程活动集为 $\pi_i^* = \{a_{i1}, \cdots, a_{in}\}$。根据 π_i^* 在实时情形集 Φ_A 中所对应的命题集，设有 $m(i)$ 个，分别扩展得到 $Leaf_i$ 的 $m(i)$ 个新分支和每个新分支上的状态命题集。

(3) 对 $Leaf_i$ 所扩展的每一新分支 $Branch_j$，设该分支上所表示的命题集为 $Effect^*$。若 $g \subseteq Effect^*$，则该分支扩展结束，若 $g \not\subset Effect^*$，设从根节点单向线性地走到 $Leaf_i$ 一共有 r 个节点，所构成的过程活动流为 $\langle \pi_1^*, \cdots, \pi_r^* \rangle$。由于 SOLVE 是完应对解，$\exists \Pi' \in SOLVE, \Pi' = \{\pi_1', \cdots, \pi_k'\}$，$k > r$，满足条件：

$Orig(\pi_j') = \pi_j^*, j=1, \cdots, r$。$Effect(\pi_r') = Effect^*$

把 $Orig(\pi_{r+1}')$ 作为 $Leaf_i$ 通过分支 $Branch_j$ 联接到的新节点，同时把 $Orig(\pi_{r+1}')$ 记为 π_{r+1}^*，并将它的标记添加到叶节点索引 $LeafIndex$ 之中。如果 $Leaf_i$ 所扩展出的所有分支，要么联接到了新节点，要么由于分支上的命题集 $Effect^* \supseteq g$ 而停止扩展，则将 $Leaf_i$ 的标记从 $LeafIndex$ 中删去。

(4) 如果 $LeafIndex = \phi$，则 $ControlTree$ 已成功构建完毕，输出并返回，否则转到(2)。

4.2 PARPS 的性质

性质 1　PARPS 算法如果返回了非空解，则该解是 $(s_0, g, A, \Sigma, \Phi_A)$ 的有效解。

性质 2　$(s_0, g, A, \Sigma, \Phi_A)$ 所有的有效解都可以在 PARPS 所产生规划图上以某种提取方式得到。即 PARPS 具备解完备性。

性质 3　如果 PARPS 返回失败，则问题 $(s_0, g, A, \Sigma, \Phi_A)$ 不存在有效解。

性质 4　PARPS 在扩展过程活动流规划图 G 时，会在有限步内出现不动点。

性质 5　问题 $(s_0, g, A, \Sigma, \Phi_A)$ 如果存在解，则 PARPS 会在有限步内输出一个有效解，如果 $(s_0, g, A, \Sigma, \Phi_A)$ 不存在解，则 PARPS 也会在有限步内识别出并结束。

性质 6　$P_i = \bigcup_{s_i \in I} P(s_i)$，其中指标集 I 表示从初始状态 s_0 出发，到第 i 步时，所有可能出现的可达状态集。

性质 7　任一命题层 P_i，都有 $P_i \supseteq P(s_i)$，且当第 i 层及前面任一命题层出现命题互斥对，或第 i 层以及前面任一过程活动层出现过程活动互斥对的时候，不存在状态 s_j，使得 $P(s_j) = P_i$。

性质 8　过程活动流规划图 G，如果能在其上面提取得到有效解，则其最后一层命题层 P_k 满足：$P_k \supseteq g$。该条件是由 G 可以提取出有效规划解的必要条件，但非充分条件。

5　实例应用

给定过程活动流实时规划问题 $(s_0, g, A, \Sigma, \Phi_A)$，其中初始状态 s_0 包含的命题集是：{用户不清楚详细需求，开发者对需求尚不了解}，目标状态须包含命题集 g = {有需求分析规格书}。过程活动集 A 及其前件与后件在表 1 中列出。

表 1　过程活动集 A 及其前件、后件

Process Action	Precondition	Effect
确定系统的需求特点()，确定需求采集方式方法()	用户不清楚详细需求	实时
根据客户描述制定需求简本()	开发者对需求尚不了解	实时

框架内容与需求结构分析()	客户基本同意需求框架	获需求结构与内容蓝本
理论论证和模型推演()	需求逻辑性复杂策略性主观性明显	实时
详细规格的制定()	获需求结构与内容蓝本	需求分析规格书
搜集相关的实际数据(),由客观情况构建需求数据()	需求须满足很多客观规律和条件制约	实时
需求分析的制定()	内容明确,论证合理,逻辑相容	需求分析规格书
重新与客户商议需求()	客户否定框架	实时
原型具体化确定出需求规格()	构建出系统原型并获得认可	需求分析规格书
从大量需求中提取合理需求()	获得大量的直接需求数据	需求分析规格书
根据客观限制制定符合要求的需求()	获得很多对需求的限制和客观要求	需求分析规格书
在共识框架上扩展制定需求()	取得需求共识框架	需求分析规格书

实时情形集 $\Phi_A=\{\varphi_1,\cdots,\varphi_5\}$,其中:

$\varphi_1=\langle\{$确定系统的需求特点(),确定需求采集方式方法()$\}:\diamond_1=\{$系统需求的逻辑性复杂,系统策略性主观性明显$\},\diamond_2=\{$系统需求必须满足很多客观规律和现实条件制约$\}\rangle$

$\varphi_2=\langle\{$根据客户描述制定需求简本()$\}:\diamond_1=\{$客户基本同意需求框架$\},\diamond_2=\{$客户否定框架$\}\rangle$

$\varphi_3=\langle\{$理论论证和模型推演()$\}:\diamond_1=\{$内容明确,论证合理,逻辑相容$\},\diamond_2=\{$构建出系统原型并获得认可$\}\rangle$

$\varphi_4=\langle\{$搜集相关的实际数据(),由客观情况构建需求数据()$\}:\diamond_1=\{$获得大量的直接需求数据$\},\diamond_2=\{$获得很多对需求的限制和客观要求$\}\rangle$

$\varphi_5=\langle\{$重新与客户商议需求()$\}:\diamond_1=\{$取得需求共识框架$\},\diamond_2=\{$需求不合理或无法实现,商议失败合作终止$\}\rangle$

根据系统的执行,首先生成扩展过程活动规划图,整理得到过程活动流实时规划的数据结构,如图 2 所示。

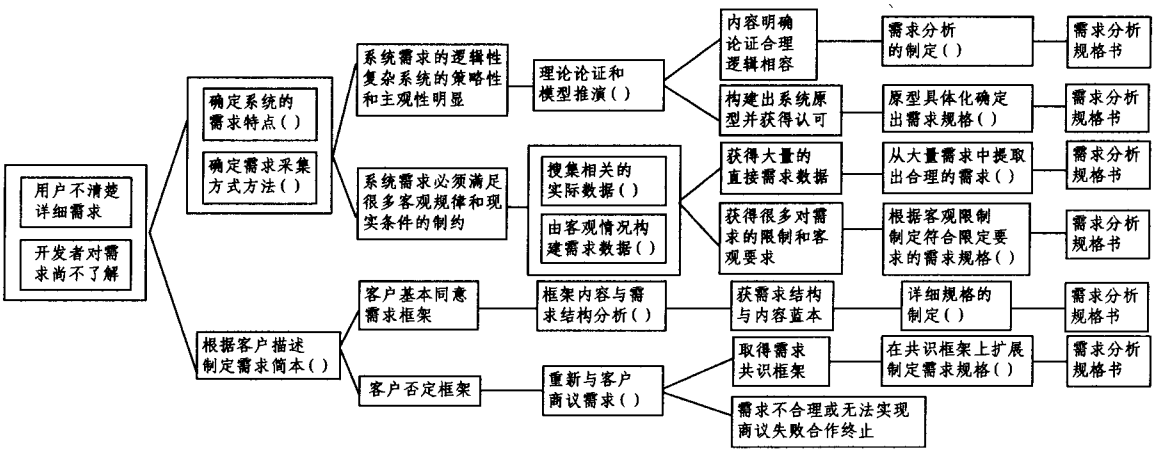

图 2 基于 PARPS 生成的过程活动流扩展规划图

系统提取得到 6 个有效解,分别是:

$\Pi_1=\langle\{$确定系统的需求特点(),确定需求采集方式方法()$\},\{$理论论证和模型推演()$\},\{$需求分析的制定()$\}\rangle$

$\Pi_2=\langle\{$确定系统的需求特点(),确定需求采集方式方法()$\},\{$理论论证和模型推演()$\},\{$原型具体化确定出需求规格()$\}\rangle$

$\Pi_3=\langle\{$确定系统的需求特点(),确定需求采集方式方法()$\},\{$搜集相关的实际数据(),由客观情况构建需求数据()$\},\{$从大量需求中提取出合理的需求()$\}\rangle$

$\Pi_4=\langle\{$确定系统的需求特点(),确定需求采集方式方法()$\},\{$搜集相关的实际数据(),由客观情况构建需求数据()$\},\{$根据客观限制制定出符合限定要求的需求规格()$\}\rangle$

$\Pi_5=\langle\{$根据客户描述制定需求简本()$\},\{$框架内容与需求结构分析()$\},\{$详细规格的制定()$\}\rangle$

$\Pi_6=\langle\{$根据客户描述制定需求简本()$\},\{$重新与客户商议需求()$\},\{$在共识框架上扩展制定需求规格()$\}\rangle$

这 6 个有效解都不是完应对解,因为它们都不具备根据实际执行结果进行实时反应的能力。系统判断出存在完应对解,计算后输出完应对解:$SOLVE=\{\Pi_1,\Pi_2,\Pi_3,\Pi_4\}$。然后将完应对解 *SOLVE* 自动转换为实时反应规划控制树 *ControlTree*,如图 3 所示。

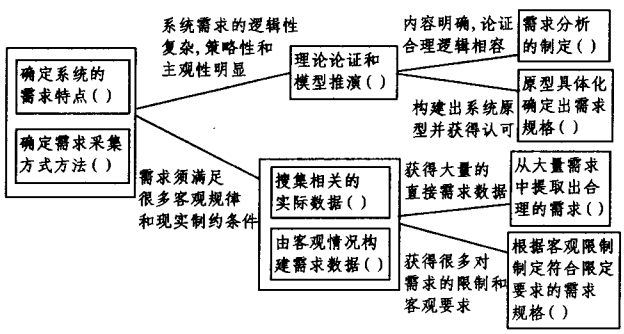

图 3 由完应对解转化得到的实时反应规划控制树 *ControlTree*

根据 *ControlTree*,在从初始状态下要执行的第 1 层过程活动集为:{确定系统的需求特点(),确定需求采集方式方法()}。然后观察执行结果,如果发现执行结果是:{系统需求的逻辑性复杂,策略性和主观性明显},则为了最终能实现既定目标,下一步需要执行的过程活动是:{理论论证和模型推演()}。如果执行完第 1 层过程活动集后所观察到的执行结果是:{需求须满足很多客观规律和现实制约条件},则系统做出的反应是:下一步需要执行的过程活动集为{搜集相关的实际

数据()，由客观情况构建需求数据()}，接下来再观察它的执行结果。接下来如果观察到实际执行结果为：{获得大量的直接需求数据}，则根据 ControlTree 知：再下一步需要执行的过程活动集为{从大量需求中提取出合理的需求()}，执行完这一活动集后即可实现目标要求。

结束语 本文提出了一种过程活动流实时规划算法，以对过程活动流的安排做出实时自动规划。可以对过程活动执行中出现的各种可预见的实际效果做出实时反应，使软件开发始终处于有序的过程活动控制之中，并取得既定目标。试验表明，所产生的规划有助于减少软件开发中过程活动安排的盲目性，提高执行每个过程活动时的全局意识。另外过程活动常常会出现重复执行或者多次执行，已有的一些相关规划工作[13-15]都没有谈及如何解决这个问题，本文提出的实时规划算法较好地解决了该问题，如果有可能再次执行某过程活动，则该过程活动的前提条件一定会被产生，将这些前提命题加入到能够实时地产生它们的那些相应动作的实时情形集中即可。

但本文也存在一些不足之处，项目的软件过程要考虑时间、成本和资源的约束，这些都是很重要的因素，本文暂时还没有加入这些因素。这些方面的解决可以考虑通过引入时序约束和条件约束的方法，这些方面将作进一步的深入研究。

参 考 文 献

[1] Li Mingshu, Boehm B W, Leon J. Osterweil. Unifying the Software Process Spectrum [J]. Journal of Software, 2006, 17(4): 649-657

[2] Fuggetta A. Software process: A roadmap. The Future of Software Engineering[C]// 22nd International Conference on Software Engineering. ACM Press, 2000: 25-34

[3] Boehm B. The Future of Software Processes. Keynote Address [C]// SPW2005. Beijing, May 25, 2005

[4] Blum A, Furst M. Fast planning through planning graph analysis[J]. Artificial Intelligence, 1997, 90: 281-300

[5] 赵欣培，李明树，陈振冲，等. 一种基于协商的软件过程协同方法[J]. 计算机研究与发展，2006，43(2)：314-320

[6] 李健，金茂忠. 有效改善软件过程方法研究[J]. 计算机研究与发展，2001，38(1)：26-35

[7] 黄罡，梅宏，杨芙清. 基于反射式软件中间件的运行时软件体系结构[J]. 中国科学 E 辑，2004，34(2)：121-138

[8] 吕建，陶先平，马晓星，等. 基于 Agent 的网构软件模型研究[J]. 中国科学 E 辑，2005，35(12)：1233-1253

[9] 吕建，徐家福. 软件自动化的智能化途径[J]. 科学通报，1993，38(2)：184-185

[10] 柳军飞，唐稚松. 软件过程建模语言研究[J]. 软件学报，1996，7(8)：449-457

[11] 王青，李明树，刘霞. 一种支持软件过程控制和改进的主动度量模型[J]. 软件学报，2005，16(3)：407-418

[12] 袁峰，李明树. 基于 MDA 的 TRISO-Model 模型管理方法及应用[J]. 软件学报，2007，18(7)：1612-1635

[13] Jasper H. Active databases for active repositories [C]// Proc. 10th International Conference on Data Engineering. Houstong. IEEE Computer Society Press, Feb. 1994: 375-384

[14] Al-Emran A, Pfahl D, Ruhe G. DynaReP: A Discrete Event Simulation Model for Re-planning of Software Releases [C]// ICSP2007. Minneapolis, USA: 246-258

[15] Gupta M, Bastani F, Khan L, et al. Automated test data generation using MEA-Graph Planning [C]// Proc. of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04). Boca Raton, Florida (USA), 2004: 174-182

(上接第 71 页)

效果，而且其时空复杂度仅为 $O(k)$，通信开销方面平均发包数低于 2 个。因此，它适于在资源受限的传感器节点上实现。

虽然基于特定消息的报警处理有利于区分两次不同的假冒行为，但在同一局部区域同时测定多个假冒行为甚至假冒报警行为仍然需要进一步研究，这将是我们今后需要解决的问题。

参 考 文 献

[1] Akyildiz I F, Weilian S, Sankarasubramaniam Y, et al. A Survey on Sensor Networks[J]. IEEE Communications, 2002, 40(8): 102-114

[2] 郎为民，杨宗凯，吴世忠，等. 无线传感器网络安全研究[J]. 计算机科学，2005，32(5)：54-58

[3] Karlof C, Wagner D. Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures[J]. Ad Hoc Networks, 2003, 1(1): 293-315

[4] 曹晓梅，何欣，陈贵海. 传感器节点定位系统攻防机制研究[J]. 计算机科学，2008，35(7)：36-41

[5] Bhuse V, Gupta A, Al-Fuqaha A. Detection of Masquerade Attacks on Wireless Sensor Networks[C]// Proceedings of the IEEE International Conference of Communications (ICC '07). June 2007: 1142-1147

[6] Krontiris I, Dimitriou T, Giannetsos T, et al. Intrusion Detection of Sinkhole Attacks in Wireless Sensor Networks[C]// ALGOSENSORS 2007, LNCS 4837. 2008: 150-161

[7] Clark B N, Colbourn C J, Johnson D S. Unit Disk Graphs[J]. Discrete Math., 1990, 86: 165-177

[8] 路纲，等. 无线网络邻近图综述[J]. 软件学报，2008，19(4)：888-911