DOI:10.11992/tis.201603009
网络出版地址:http://www.cnki.net/kcms/detail/23.1538.tp.20170220.1009.006.html

一种基于知识树和约束的柔性活动动态细化方法

蒋艳荣,李卫华,杨劲涛
(广东工业大学 计算机学院, 广东 广州 510006)

摘 要:柔性工作流在应对业务建模过程中的动态不确定因素、提高工作流系统的柔性具有巨大的优势,然而,柔性活动的动态细化一直是柔性工作流建模和应用的一个难点。因此,提出一种基于知识树和约束的柔性活动动态细化方法。该方法以知识树的包含和泛化关系作为启发信息,以活动选取约束和时序约束作为指导和校验,实现柔性活动的动态细化。在介绍了知识树及其蕴含关系以及活动选取约束和时序约束规则的基础上,给出了柔性活动的动态细化算法,描述了活动选取校验和时序约束校验算法。最后给出了算法的实现和实例分析,其结果表明了所提方法的有效性,能够很好地解决柔性活动的细化问题。

关键词:柔性工作流; 动态细化; 时序约束; 工作流生成; 过程建模
中图分类号: TP391 **文献标志码**:A **文章编号**:1673-4785(2017)02-0158-08

中文引用格式:蒋艳荣,李卫华,杨劲涛. 一种基于知识树和约束的柔性活动动态细化方法[J]. 智能系统学报, 2017, 12(2): 158-165.
英文引用格式:JIANG Yanrong, LI Weihua, YANG Jintao. A dynamic refinement approach for flexible activity based on knowledge tree and constraints[J]. CAAI transactions on intelligent systems, 2017, 12(2): 158-165.

A dynamic refinement approach for flexible activity based on knowledge tree and constraints

JIANG Yanrong, LI Weihua, YANG Jintao
(School of Computer, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: Flexible workflow systems offer huge advantages in addressing dynamic uncertain factors during the modeling period; however, the dynamic refinement of flexible activities remains a challenge in the modeling and application of flexible workflows. Therefore, we propose a dynamic refinement approach of flexible activities based on a knowledge tree and various constraints. More specifically, we used the inclusion and generalization relationships of knowledge trees as heuristic information; further, we used activity selection constraints and temporal constraints to guide our verification processes. Given the introduction of a knowledge tree and its containing relationships, as well as activity selection constraint and temporal constraint rules, we provide a dynamic refinement algorithm and an algorithm for activity selection checkout and temporal constraint checkup. Finally, we provide a realization of our algorithms and offer case analyses. Our results show that our proposed algorithms are effective and can solve the problem of dynamic refinement of flexible activities quite well.

Keywords: flexible workflow; dynamic refinement; temporal constraints; workflow generation; process modeling

近年来,随着人工智能、自动控制等技术的发展,工作流在很多领域获得了广泛的应用。然而,传统工作流建模和实施的刚硬性限制了工作流应用的成功,其静态、无柔性的流程描述无法适应动态、复杂的业务过程。随着工作流应用领域的不断拓宽,这种复杂性不断加大,表现在业务过程中存在大量

收稿日期:2016-03-06. **网络出版日期**:2017-02-20.
基金项目:国家自然科学基金项目(61142012); 广东省科技计划项目(2015B010128005, 2013B021800115).
通信作者:蒋艳荣. E-mail:yrjiang@gdut.edu.cn.

的不确定、模糊和不完整的流程信息,但是在工作流的建模阶段,却无法涵盖所有的不确定因素。这对工作流的建模和应用带来巨大的困难。因此,如何针对业务过程具有的动态性和模糊性,提高工作流的柔性,成为工作流研究领域的一个热点问题[1-2]。

动态细化是提高工作流柔性、应对不确定性的一种有效方法[1]。其基本思想是:在流程实例运行之前,先对业务流程的已知部分进行建模,建立流程模型的整体框架,而不需要精确、完整地定义流程模型的每一个细节。对不清楚的局部细节可以用"黑箱"活动来描述[3-5],然后随着业务工作的展开和对具体问题认识的不断深入,在过程运行中动态细化流程模型,以提高流程模型的柔性。因此,动态细化成为提高业务过程柔性的关键,其本质是将过程中不确定的、模糊的区域进行逐步细化,用一种更明确、更细尺度的活动或子流程去进行代替。该不确定的区域在很多文献中被封装为柔性活动或占位符等。例如,S. Sadiq(2001)等[3]采用柔性区(pocket of flexiblity)来提高工作流的柔性,并为柔性区提供实例模版,支持多种控制结构。邓水光等[6]将过程中的不确定因素用"flexible activity"表示,用 ECA 规则表示活动之间的逻辑关系。

现有的研究工作在动态细化方面取得了较多的进展。文献[7]采用柔性活动对不确定因素进行封装,然后对柔性活动进行纵向分解,生成不同抽象水平的活动,从而提出一种支持柔性活动树状分解和增量细化的柔性工作流建模方法。Joonsoo Bae[8]提出了一种树形的过程模型的表示,采用 ECA 规则表示活动关系,并用存储过程实现了 ECA 规则。其缺点是,为每个 ECA 规则创建一个对应的存储过程并管理活动的状态,是一个耗时和成本高昂的事情。Y. Gil [9]采用工作流来实现计算实验的设计过程,将其功能实现逐步细化为组件选择、数据集选择和参数选择 3 个步骤,其缺点在于其建模过程的柔性不够,其建模和细化方法难以借鉴到其他应用领域,同时需要提供完备的组件库。S.A.Chun 等[10]在实现跨组织的工作流子流程自动生成的过程中,采用政府组织的本体树来帮助生成子流程,其缺点是生成的子流程控制结构比较简单,难以适应复杂流程结构的生成。同时适应面较窄,本体的建立和合理利用也比较困难。Shepelev[11]针对计算机辅助 VLSI 的设计领域,提出一种基于任务计划的图形逻辑表达的工作流自动生成方法。

上述研究工作,虽然在提高工作流建模的柔性和柔性活动细化上取得了较多的进展,但是存在如下不足:1) 在工作流建模和细化时,未能充分考虑到工作流的应用背景,如动态因素和不确定因素多,需要满足一定的约束条件等特点,而这对于柔性建模和活动细化非常重要;2) 一些工作侧重于某个特定领域的工作流自动生成,在通用性方面存在不足;或者过于强调灵活性,试图从头开始创建工作流,这非常依赖于模型设计者的技能和特殊知识,而没有充分利用领域知识来指导实现动态细化;3)现有的柔性活动细化的研究还不够,特别是考虑到约束条件等因素影响的细化研究还比较缺乏。

因此,本文提出一种基于知识树和约束的柔性活动细化方法,知识树可以为柔性活动的细化过程提供指导,而相关的约束规则可以为细化过程提供活动选择、活动时序关系的校验。

1　柔性工作流的建模

定义 1　柔性活动是一种特殊的活动类型,在建模阶段由于存在模糊和不确定的因素,而无法事先给出完整和明确的活动定义,需要获取更多和更完全的信息以将柔性元素逐步明确化。柔性活动可用一个多元组表示:FA = (Id, name, Ctx, FAP, FAR, FAC, Role, Rdata, Attr),其中 Id 为柔性活动的唯一标识;name 为柔性活动的名称;Ctx 为柔性活动的应用上下文,包括该柔性活动需要达成的目标和完成的功能,以及应用场景等; FAP = $A_{atom} \cup A_{comp} \cup A_{flex}$为柔性活动的活动池,分别由原子活动集、复合活动集合柔性活动集组成。也就是说,柔性活动可以由原子活动、复合活动和其他柔性活动构成。Role 是活动执行者对应的角色; Rdata 是与此活动相关的工作流相关数据,包括输入数据和输出数据。Attr 为活动具有的属性集。

FAR = FR ∪ CR 为描述活动池中活动关系的规则集,可以为空。其中 FR 为柔性规则集,CR 为普通规则集,采用 ECA 规则形式表示。FAC = $C \cup M$ 为柔性活动的约束规则集,其目的是用于柔性活动细化过程中的活动选择、活动时序关系的校验和生成子流图的正确性验证,以保证细化结果的正确。采用一元或二元谓词、预设的操作集和规则进行表达,在柔性活动的细化过程中,生成的子流程必须满足该约束条件的规定。其中,C 为约束规则集,用于约束活动的选取、活动的组合、活动之间的时序关系、活动的属性等;M 为修正规则集,用于对生成的流程进行修改。

定义 2　柔性工作流。柔性工作流是一个四元组,FWF = (ID, D, A, E),其中 ID 为柔性工作流

的唯一标识符,D 为柔性工作流的一般描述信息,如工作流的版本号,流程的创建日期等,$D=$(Version, Date, Desciption, …); A 为流程中的活动集合, $A=A_{atom}\cup A_{comp}\cup A_{flex}$; E 表示工作流中规则的集合,用 ECA 规则表示,用来描述活动之间的时序关系。

该模型采用柔性活动封装流程中的不确定的、模糊的因素,采用原子活动和复合活动描述具有不同抽象度的流程元素,可以快速建立流程模型,并随着领域知识的后续获取和分析的逐步深入,对柔性活动进行明确化和细化,以减低复杂领域的工作流建模难度,提高模型的稳健性和易用性。

2 细化算法

2.1 知识树及与流程模板的影射

在领域知识表示中,可以用不同粒度大小的知识点来表示不同的知识单元。而知识点不是孤立存在的,它们之间存在着各种关系,考察这些关系对于工作流的生成具有促进作用。

定义 3 知识点。知识点可以表示为一个多元组: $K=$(ID, name, des, KWS), 其中 ID 为知识点的唯一编号,name 为知识点的名称,des 为知识点的描述,包括该知识点的定义、阐述等,来源于领域知识;KWS 为知识的关键词集,是对该知识点的高度概括。

定义 4 包含关系。表示知识点 K_j 与 K_i 之间存在整体与部分的关系,记为 $a(K_i, K_j)$,表示 K_j 包含 K_i。知识点可以被分解为几个满足包含关系的知识点,依次类推,一直到不能分解为此。

定义 5 泛化关系。泛化关系可以定义为一般与特殊的关系,记为 $g(K_i, K_j)$, 表示知识点 K_i 是 K_j 的特殊情况, K_j 与 K_i 满足一般与特殊的关系。例如,面向对象设计、基于结构化的设计和面向过程的设计等是软件设计的特殊情况,是软件设计在面对不同领域的具体应用。

定义 6 父子关系。若对任意的两个知识点 x, y,满足 $a(x, y)$ 或 $g(x,y)$ 成立,则称知识点 y 和 x 满足父子关系,x 为 y 的子知识点,记为 $s(x, y)$。显然,这种关系具有单反性、传递性。

定义 7 知识树。对一个知识点 A 自顶向下地进行逐步分解,则最终可得到一棵根节点为 A 的树,我们称其为知识树,记为 $T(A)$。树中的孩子节点是双亲节点的子知识点,满足包含或泛化关系。当树中所有的知识点都不能分解时,则分解过程结束。

显然,可以用知识树表示不同的领域知识。这种知识表达具有直观、易于理解和易于建立的特点,可以由领域专家进行构建,而不需要建模和开发的专业知识。图 1 为一棵关于软件开发过程的知识树,其中包含了包含关系和泛化关系。

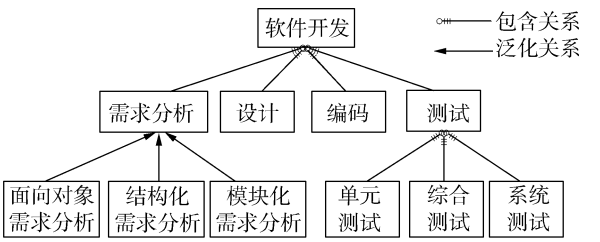

图 1 软件开发的知识树

Fig.1 Knowledge tree for software development

针对某一个领域的知识树数目比较多,则构成了一个知识树森林。那么在建立了领域知识的知识树之后,为了用于指导流程建模和柔性工作流细化,需要将流程模板与知识树的知识点之间进行关联,即为流程模板与知识树的映射。流程模板库的建立需要全体设计人员在设计过程中共同构建。需要注意的是,流程模板可以全部由定义完全的子活动或复合活动构成,也可以在流程中包含子柔性活动。

映射的步骤可描述为:1) 根据知识点的名称,在流程模板库中进行搜索,得到属性相同的流程模板集合;2) 考察集合中的每个流程模板的适用条件,得到各模板的适用性并进行排序;3) 若集合只有一个流程模板,则返回该模板,否则返回适用性最高的流程模板。

2.2 约束规则

柔性工作流的细化过程,其实质为柔性活动逐步细化和确定化的过程。在此过程中,需要将一些模糊的、抽象的需求进行具体化,将粗粒度的元素转换为细粒度的元素。显然,该细化过程需要遵循某些特定的约束和规则,满足该活动定义的初始意图。例如,在工作流的设计阶段,用 Ctx 参数规定了柔性活动需要达成的目标和完成的功能,以及应用的条件等。那么在确定化该柔性活动时,需要满足所设目标或功能的约束要求。在细节上,需要根据选取规则从系统提供的活动库中选取合适的活动,并用时序规则约束活动之间的时序关系。具体讨论如下。

1) 活动选取规则

用于约束活动的选取。用谓词 select( ) 表示活动的选取, 可以用逻辑运算符进行连接,分如下情况: 1) select(a) 表示选取活动 a; 2) NOT select(a) 表示不能选取活动 a; 3) select(a) AND select(b) AND…表示同时选取活动 $a,b,\cdots$; 4) select(a) OR select(b) OR…表示选择活动 $a,b,\cdots$ 中的任意活

动。在这里,逻辑运算符的优先性满足:NOT>AND>OR。

用产生式规则对活动选取的逻辑约束进行规定,形式如下:

IF〈context(1)〉AND〈 context(2)〉…

AND〈 context(n)〉THEN〈selections〉　(1)

式中:context 为与工作流相关的上下文环境,可以是当前的上下文事实,如"the calculus type is stag-horn",也可以是对某个活动的选取或未选取情况。selections 为一个或多个活动的选取逻辑,如 select(a) AND NOT select(b)。以上表达可以简写为 $C_1 \wedge C_2 \wedge \cdots C_n \rightarrow S$, where C_i means 〈context(i)〉and S means selections。规则的前题和结论可以为多个 item 的合取范式。对于多个 context 组成的析取范式,如$(C_{11} \wedge C_{12} \wedge \cdots) \vee (C_{21} \wedge C_{22} \wedge \cdots) \vee \cdots$则拆分为由简单的合取范式前提组成的多个规则如下:

$$\begin{aligned}(C_{11} \wedge C_{12} \wedge \ldots) &\rightarrow S \\ (C_{21} \wedge C_{22} \wedge \ldots) &\rightarrow S\end{aligned} \qquad (2)$$

2) 活动的时序约束规则

用于约束活动之间的时序关系。① 用 Previous(x,y)表示活动 x 在活动 y 之前执行(中间可以插入其他活动);② ImtPrevious(x, y)表示活动 x 在 y 之前执行,且 x 是 y 的直接前驱(中间没有其他活动);③ AndSyn (x, y)表示活动 x 与 y 为并行执行关系(AND-split/AND-join),必须要在 x 与 y 两者执行完之后才能执行下一活动;④ OrSyn(x, y) 表示活动 x 与 y 为选择执行关系(OR-split/OR-join),在执行时,根据上下文在 x 与 y 两者选择其一执行, x 与 y 任意一个执行完可以触发后续活动的运行;⑤ NotLimit(x, y) 表示活动 x 与 y 的执行次序随意,活动 x 可以在 y 之前执行,也可以在 y 之后执行。

可以用规则描述活动之间复杂的时序约束,规则的形式如下: IF selectOp(x_1) AND selectOp(x_2) AND selectOp (x_i) AND …。AND selectOp (x_n) THEN temporalCons1 AND temporalCons$_2$ AND…temporalCons$_n$。这里,selectOp 表示 select( ) 或 NOT select( ), temporalCons 表示这 5 个时序约束。

以下为一个时序约束的规则实例:

IF select(a) AND select(b) THEN Previous(a, b), 如果选择了活动 a 和 b,则活动 a 必须要在 b 之前执行。

一般情况下,规则的前件都是对涉及到的活动的选取情况进行判断,即很少有 NOT 的情况,此时前件可以忽略,则规则可以转换为时序约束集:{Previous(a,b), AndSyn(c, d), …}。

3)属性约束规则

包括时间约束规则、地点约束规则等,用于对活动属性的约束。如对活动的时间属性进行约束:"活动 a 的持续时间不能大于 3 天"、"活动 b 在活动 a 结束后的 3 hour 内必须开始"等。

显然,逻辑运算符的引入可以实现复杂的约束逻辑,以适应复杂多变的建模需求。那么通过对模糊信息的封装,在较大粒度的层面快速建立起流程模型之后,通过对柔性活动约束条件的设定,可以作为启发信息用于柔性活动的细化和确保创建的工作流的正确性。

2.3　柔性活动的动态细化算法

柔性活动的细化算法 FlexActRefining( ) 如图 2 所示。

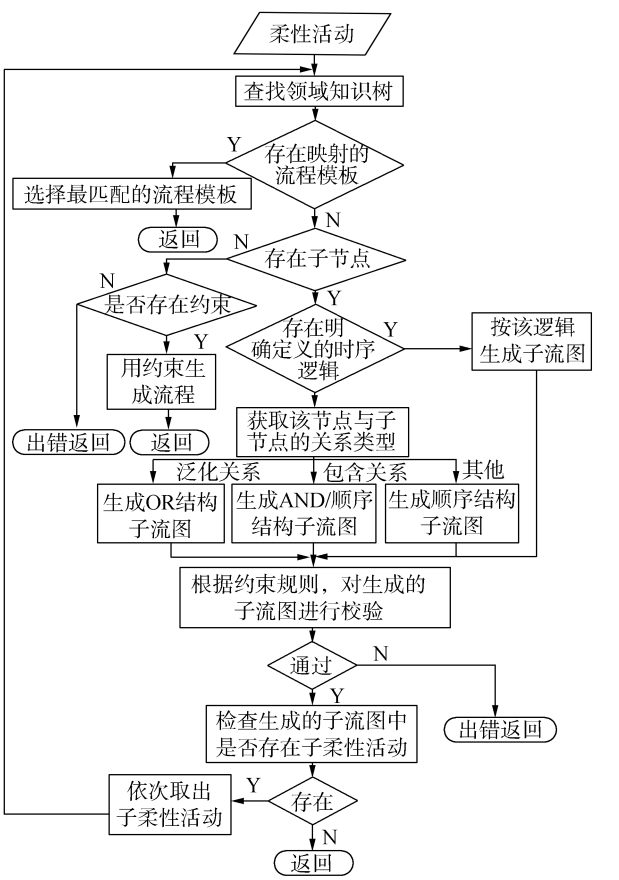

图 2　柔性活动细化算法 FlexActRefining

Fig.2　Refinement algorithm for flexible activity

在细化过程中,利用知识树中知识点之间的关系作为启发信息用于辅助生成子流图,用柔性活动的约束规则(包括活动选取约束、时序关系约束等)对细化后的子流程进行验证,其主要步骤如下:

1) 根据柔性活动的目标,在知识树中查找匹配的节点。对该节点映射的流程模板进行分析:若存在完善的流程模板,则选择匹配最佳的流程模板返

回;否则转 2)。

2)分析该节点与子节点的关系。若不存在子节点,则如果存在对应的约束关系,用约束关系生成流程并返回,否则出错返回。如果存在子节点,则根据如下规则将父节点的实现转换为子节点的实现,生成细化的子流图。①若存在明确的时序关系规定,则按照时序关系的要求去生成细化的子流图;②若该节点与子节点的关系为包含关系,则按照 AND 或顺序结构生成子流图;③ 若该节点与子节点的关系为泛化关系,则按照 OR 结构生成子流图;④ 其他情况则按照顺序结构生成子流图。

3)依据约束规则,对生成的子流图进行校验。分别调用函数 ActSelValid( )对活动选取进行校验,调用 TemporalCons( )对活动的时序约束进行校验。若没有通过校验,则出错返回。

4) 检查生成的子流图是否存在子柔性活动。若不存在,则返回生成的子流图,算法结束。否则,对每一个子柔性活动,转算法的步骤 1)。

3 校验算法

对生成的子流程通常存在着合法性检验问题。一个子流图只有符合柔性活动的设计目标和功能,满足其预先定义的约束条件,才能被认为是合法的和有效的。另外,子流图在流程结构上必须要定义良好,例如不存在孤立不可到达的活动、不存在死循环等,对此可以采用拓扑排序进行检查。下面从以上几个方面对柔性活动细化后生成的子流图的合法性进行检验,其中, ActSelValid ( )是活动选取规则检验算法, TemporalCons( )是活动时序约束检验算法。

设选取的活动组成的集合为 A, $A=\{a_1, a_2, \cdots, a_n\}$, 选择约束 sc 可以分解为左部 Lpart 和右部 Rpart, Lpart 由上下文事实和活动选择的操作组成,Rpart 为操作的逻辑表达式。如 Lpart = con_1 AND con_2 OR⋯AND con_n, $con_i = ctx_i \mid op_i$, ctx 为上下文事实,如“calculus。diameter≤20 mm”,op = select( ) | NOT select( )。在算法 ActSelValid( )中,E_i代表 true or false。

算法 1 ActSelValid ( )

输入 activity set A, activity selection constraints SC.

输出 return true if valid, false if not valid

```
{
for each selection constraint sc_i in SC{
  decompose sc_i into Lpart and Rpart;
  for each con_i in Lpart {
    if con_i is "NOT select(a)"
      if a ∈A then con_i = false;
      else con_i = true;
    if con_i is "select(a)"
      if a ∈A then con_i = true;
      else con_i = false;
    if con_i is ctx
      check if there exist fact fac_i and con_i = fac_i
    con_i = true;
      else con_i = false;
  }
  while Lpart is not a single "true" or "false"{
    if there exist "E1 AND⋯AND En" in Lpart{
  if there exist a "false" in sub-expression
    replace sub-expression with "false";
  else replace it with "true";
      }
      if there exist "E1 OR⋯OR En" in Lpart{
        if there exist a "true" in sub-expression
      replace the sub-expression with "true";
  else replace it with "false";
      }
  } // end while
  if Lpart is true then{// rule sc_i is triggered.
    for each op_i in Rpart {
      if op_i is "NOT select(a)"
    if a ∈A then op_i = false;
    else op_i = true;
      if op_i is "select(a)"
    if a ∈A then op_i = true;
    else op_i = false;
    }//end for
    while Rpart is not a single "true" or "false"{
      if there exist "E_1 AND⋯E_n" in Rpart{
    if there exist a "false" in sub-expression
    replace sub-expression with "false";
    else replace it with "true";
        }
        if there exist "E_1 OR⋯E_n" in Rpart{
    if there exist a "true" in sub-expression
    replace sub-expression with "true";
    else replace it with "false";
        }
  } // end while
      if Rpart is "false" then{
  report the information of error to user;
  return false; //quit
      }
  } // end of the deal of Rpart
  }// end of the deal of sc_i
  return true;
```

TemporalCons( )是活动时序约束检验算法,其

中 A 为活动集,TC 为时序约束集。算法首先检查时序约束 tc 涉及的活动是否存在于活动集 A 中,如果不存在,则返回 false. 时序约束规则 tc 可以分解出由时序约束组成的操作集 Rpart,然后对其中每个约束进行检查,对不满足约束的情况进行出错报告,并返回 false.

算法 2　TemporalCons( )

输入　activity set A, activity temporal constraints TC.

输出　return true if valid, false if not valid

```
{
for each temporal constraint tc_i in TC {
  decompose tc_i into Lpart and Rpart;
  if activities referred by Lpart are all in set A{
    for each op_i in Rpart{
  if op_i is "Previous(a, b)"{
    if there is no path from a to b{
  mark the error place;
  return false;
}
  }
  if op_i is "ImtPrevious (a, b)"{
if there is not any edge 〈a, b〉 then{
  mark the error place;
  return false;
}
}
  if op_i is "AndSyn (a, b)"{
    if there exists a common predecessor A_s
          and successor A_e for a and b then{
if satisfy Previous(A_s, a) and
            Previous(A_s, b) and Previous(a, A_e)
          and Previous(b, A_e) and there exist no
          path from a to b or b to a then{
return true;
  }
}
return false;
  }
  if op_i is "OrSyn (a, b)"{
    if there exists a common predecessor A_s
        and successor A_e for a and b then{
  if satisfy either (Previous(A_s, a) and
          Previous(a, A_e)) or (Previous(A_s, b)
          and Previous(b, A_e)) and a and b
          can not run at the same time then{
return true;
}
  }
  return false;
}
if op_i is NotLimit (a, b){
if there is no path from a to b or from b to a{
mark the error place;
return false;
  }
  } //end of the deal for each op_i
} //end if
} //end of the deal of each tc_i
return true;
```

4　实现及实例分析

为了验证所提方法的有效性,我们在一台高性能 PC 机上实现以上算法.开发的硬件环境:Intel E5200 CPU, 2G DDR, 250G HD,开发平台:JDK1.7 +MyEclipse2014, 工作流引擎采用 Activiti 5.18.0,数据库采用 MySQL 5.1.37。采用 Hibernate 实现领域知识树的知识查询与管理。

下面我们以图 1 所示的知识树为例,来验证本文所提方法的有效性和正确性。某软件开发过程的主流程如图 3 所示,其中的"需求分析"和"测试"为未定义的柔性活动,用 Activiti 的 call activity 类型的活动来表示。当工作流执行到活动"项目人员确定"时,由于后续的柔性活动"需求分析"未定义,因此柔性活动细化算法 FlexActRefining( ) 被激活。该算法首先在领域知识树中进行查找,通过对活动目标的匹配,找到对应的知识点"需求分析",获取其 3 个满足泛化关系的子知识点,即"面向对象需求分析"、"结构化需求分析"和"模块化需求分析",并生成一个 OR 结构。根据上下文及约束条件,"面向对象需求分析"分支被选择执行。因此,其对应的流程模板被实例化后通过 Activiti 工作流引擎部署到其流程定义数据表和部署表中,并开始执行该流程。当工作流执行完之后,返回主流程执行剩下的活动。

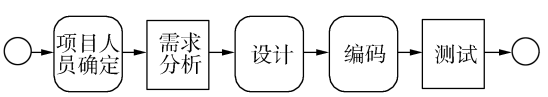

图 3　软件开发过程的主流程

Fig.3　Main flowchart of a software development process

同样的,在工作流执行到"编码"活动时,算法 FlexActRefining( ) 被激活以细化后续的柔性活动"测试"。查询知识树可得知,其存在对应的流程测试模板,并根据"测试"活动的 3 个满足包含关系的子知识"单元测试"、"综合测试"和"系统测试",细化生成流程模板中的"各类型测试"活动。依据其约束关系,采用顺序结构生成其时序关系,得到子流程:"单元测试"→"综合测试"→"系统测试",并部署到工作流引擎进行执行。整个过程可以用图 4 表

示,细化之后的流程图见图 5 所示。

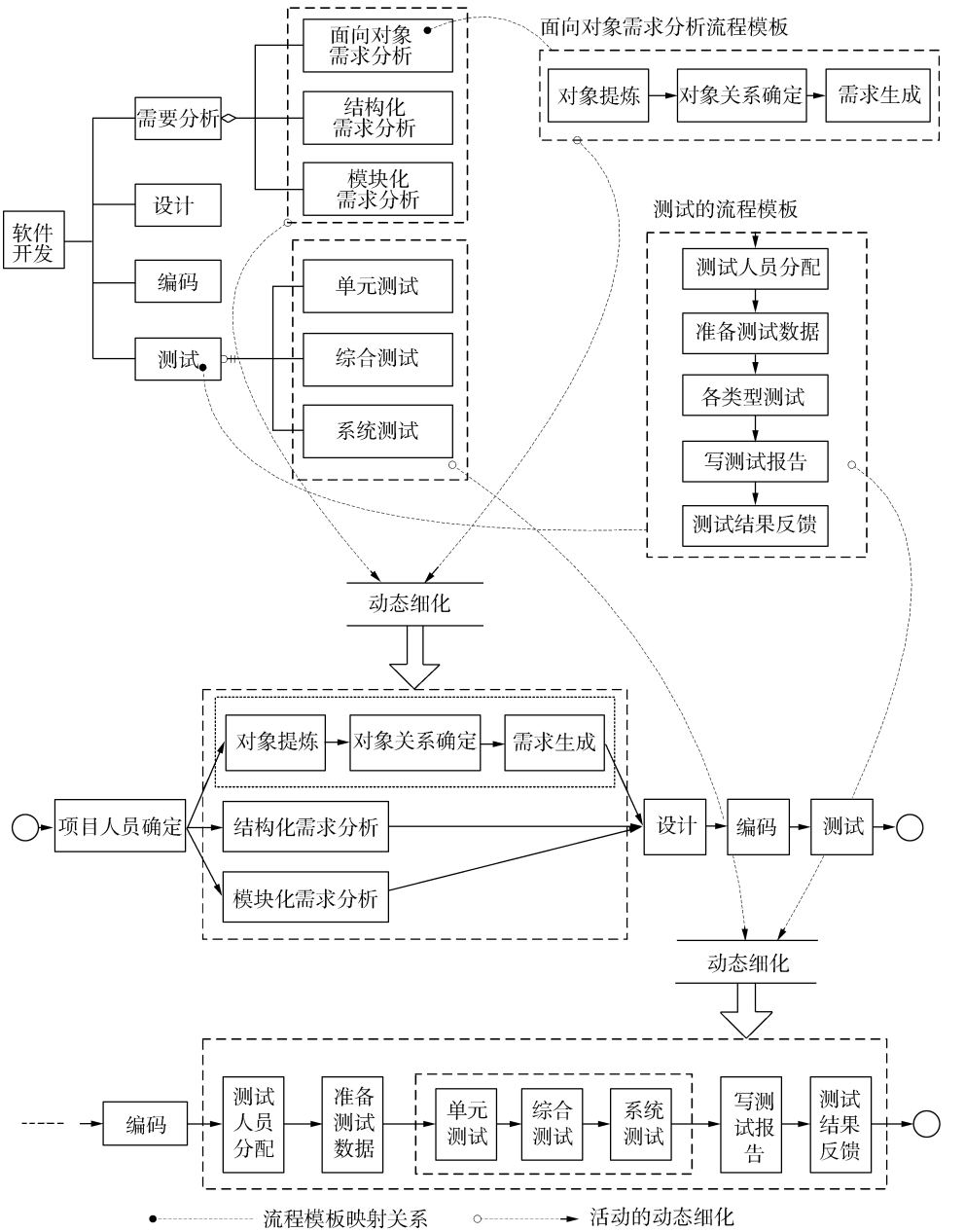

图 4 软件开发过程的动态细化方法示意图

Fig.4 Illustration of dynamic refinement for a software development process

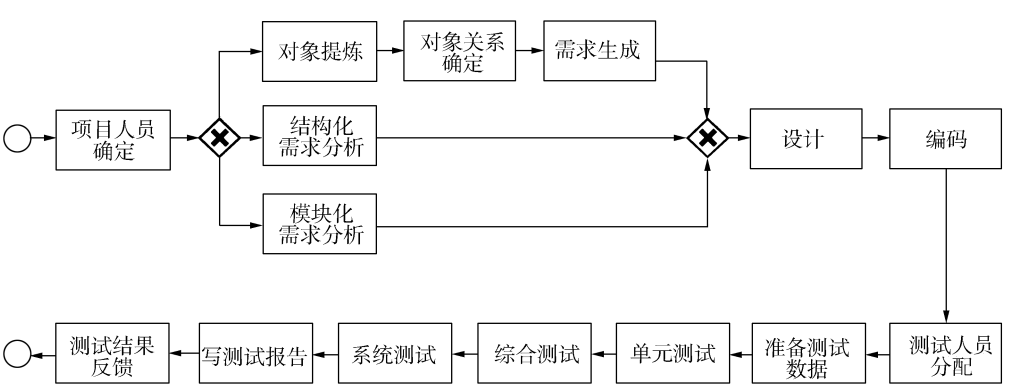

图 5 细化之后的流程图

Fig.5 Process model after refinement

5　结束语

柔性活动的动态细化一直是柔性工作流建模的一个关键问题和难点。其细化过程需要遵循纵向的知识关联关系和活动选取、时序关系的约束。因此,本文提出一种基于知识树和约束的柔性活动动态细化方法,给出了知识树和约束规则的定义,详细讨论了柔性活动细化的步骤,描述了活动选取校验的算法,以及根据时序约束对生成的子流图进行校验的算法细节。

由于 Activiti 支持对流程模型的动态生成和修改,因此在 Activiti 工作流环境中对以上算法进行了实现和实例分析。其结果表明,该方法能够根据知识树和约束动态地生成对应的子流程,并部署到 Activiti 的工作流数据库中进行执行。所生成的 bpmn 流程文件可以作为该柔性活动的细化结果,在主流程模型中对其进行替换。实验验证了本文方法的有效性。下一步工作将考虑结合上下文计算提高工作流的柔性和过程感知性,进一步完善柔性活动的细化方法。

参考文献:

[1]李竞杰, 王维平, 杨峰. 柔性工作流理论方法综述[J]. 计算机集成制造系统, 2010, 16(8): 1569-1577.
LI Jingjie, WANG Weiping, YANG Feng. Review on approaches of flexible workflow[J]. Computer integrated manufacturing systems, 2010, 16(8): 1569-1577.

[2]周建涛, 史美林, 叶新铭. 柔性工作流技术研究的现状与趋势[J]. 计算机集成制造系统, 2005, 11(11): 1501-1510.
ZHOU Jiantao, SHI Meilin, YE Xinming. State of arts & trends on flexible workflow technology[J]. Computer integrated manufacturing systems, 2005, 11(11): 1501-1510.

[3]SADIQ S, SADIQ W, ORLOWSKA M. Pockets of flexibility in workflow specification[C]//Proceedings of the 20th International Conference on Conceptual Modeling. Berlin, Germany: Springer, 2001: 513-526.

[4]VAN ELST L, ASCHOFF F R, BERNARDI A, et al. Weakly-structured workflows for knowledge-intensive tasks: an experimental evaluation[C]//Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. Linz, Austria: IEEE, 2003: 340-345.

[5]ADAMS M, TER HOFSTEDE A H M, EDMOND D, et al. Worklets: a service-oriented implementation of dynamic flexibility in workflows[M]//MEERSMAN R, TARI Z. On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE. Berlin Heidelberg: Springer, 2006: 291-308.

[6]邓水光, 吴朝晖, 俞镇. 支持动态建模的工作流管理系统的研究与设计[J]. 计算机辅助设计与图形学学报, 2004, 16(5): 712-718.
DENG Shuiguang, WU Chaohui, YU Zhen. Flexible workflow model based on ECA joined with activity[J]. Journal of computer-aided design & computer graphics, 2004, 16(5): 712-718.

[7]JIANG Yanrong, LI Weihua, YANG Jingtao. An incremental approach to modeling flexible workflows using activity decomposition and gradual refinement[J]. Journal of software, 2014, 9(10): 2628-2637.

[8]BAE J, BAE H, KANG S H, et al. Automatic control of workflow processes using ECA rules[J]. IEEE transactions on knowledge and data engineering, 2004, 16(8): 1010-1023.

[9]GIL Y, RATNAKAR V, KIM J, et al. Wings: intelligent workflow-based design of computational experiments[J]. IEEE intelligent systems, 2011, 26(1): 62-72.

[10]CHUN S A, ATLURI V, ADAM N R. Domain knowledge-based automatic workflow generation[C]//Proceedings of the 13th International Conference on Database and Expert Systems Applications. Berlin, Germany: Springer-Verlag, 2002: 81-93.

[11]SHEPELEV V A, DIRECTOR S W. Automatic workflow generation[C]//Proceedings of the EURO-DAC '96, European Design Automation Conference. Geneva, Switzerland: IEEE, 1996: 104-109.

作者简介:

蒋艳荣,男,1976 年生,讲师,博士,主要研究方向为机器智能、上下文感知计算、智能交互。发表学术论文 20 余篇,其中被 SCI 检索 8 篇、EI/ISTP 检索 10 余篇。

李卫华,女,1957 年生,教授,博士,主要研究方向为智能软件、网络信息系统、面向 Agent 计算。发表学术论文 40 余篇,出版著作多部。

杨劲涛,男,1971 年生,博士,主要研究方向为 Web 服务计算、模式识别、医学图像处理技术,发表学术论文 20 余篇。