文章编号: 1006- 5911(2009) 09- 1721- 10

基于过程代数的可变业务流程建模方法

姚 青, 洪余柯, 王海洋
(山东大学 计算机科学与技术学院, 山东 济南 250100)

摘 要: 为实现灵活的业务流程重组, 提出了一种面向流程变化调整的业务流程模型。该模型从业务逻辑中抽取出流程相关性语义, 用连接器的模式来实现一种可配置的流程属性。业务人员可以通过调整模型中的连接器对流程进行重新配置, 从而实现业务流程重组的软件调整。给出了模型的过程代数描述, 并介绍了建模步骤和流程调整方法。最后, 以机场塔台调度流程的建模和调整过程为例, 对相关问题进行了讨论。

关键词: 过程代数; 业务流程模型; 流程调整; 模型验证; 业务流程管理
中图分类号: TP311 文献标识码: A

Modeling of changeable business process based on process algebra

YAO Qing, HONG Yu-ke, WANG Hai-yang
(School of Computer Science & Technology, Shandong University, Jinan 250100, China)

Abstract: To achieve flexible business process reconstructuring, a new business process model oriented to process change adjustment was proposed. Relevant semantics were extracted from business logic to realize a configurable process property through a connector mode. The software model could be adjusted to work for the business process re-engineering by modulating the connectors. Definition of the business process model was provided by process algebra. The modeling steps and process adjustment methods were also introduced. An example of control tower scheduler was given to reveal process modeling and adjustment process. Related issues were also discussed.

Key words: process algebra; business process model; process change; model verification; business process management

0 引言

企业的业务流程要随着市场的快速变化进行动态的调整和优化, 实现灵活的流程重组, 相应的业务流程管理系统(Business Process Management System, BPMS) 应该能够支持这样的重组[1-4]。传统软件系统存在着以下局限: 没有在软件系统建模和实现之间凸显流程的特点, 从需求分析到设计过程, 主要关注的是典型业务流程的静态需求, 缺少对业务流程变化特征的定义和分析。当流程需求发生变化时, 必须重新分析、设计和实现 BPMS。

许多研究者探索了一些分离流程逻辑的方法, 如文献[2] 基于集成信息系统体系结构(Architecture of Integrated Information Systems, ARIS[5]), 提出了一个通用的业务流程体系结构。文献[6] 提出了一种流程驱动的体系结构模型。文献[7] 给出了一个采用面向业务域的系统体系结构的业务过程管理与集成系统, 该体系结构以企业业务需求的描述为基础, 通过对企业应用业务流程、活动功能和活动参与者的信息交换机制进行独立封装, 降低业务

收稿日期: 2009-02-03; 修订日期: 2009-03-16。Received 03 Feb. 2009; accepted 16 Mar. 2009.
基金项目: 国家自然科学基金资助项目(60673130)。**Foundation item:** Project supported by the National Natural Science Foundation, China (No. 60673130).
作者简介: 姚 青(1963-), 女, 山东济南人, 山东大学计算机科学与技术学院副教授, 主要从事软件理论与技术、流程管理等的研究。E-mail: yaoqing@sdu.edu.cn。

逻辑、业务数据和业务操作实体三者间的耦合,实现业务流程的柔性管理。随着企业管理从传统职能型向流程型的转变,以及流程技术与软件工程的研究相互结合,人们提出了面向业务流程的软件方法[8-9]。

从目前的研究成果来看,无论是基于工作流技术[10-13]还是 Agent 技术[14]、对等网络(Peer to Peer, P2P)[15]技术,均从各个方面探索了流程变化的控制方法,但没有针对由于数据依赖而产生的业务操作相关性语义进行抽取和建模,因而未能实现流程逻辑和应用逻辑的真正分离。如果能在建模时给出清晰明确的流程属性,将流程相关性语义从业务逻辑中抽取出来作为软件结构中的独立元素加以描述,不失为一个新的探索途径[5]。

本文提出了一种面向流程变化调整的业务流程软件模型(Process Change-Oriented Software Model, PCOSM),并给出其过程代数[16-17](Communicating Sequential Processes, CSP)形式化定义和建模方法。通过调整模型中的流程相关元素,利用已产业化通用的自动模型检验工具[18](Failures-Divergences Refinement checker, FDR)完成模型分析与检验,达到无需 IT 人员干预、实现 BPMS 调整的目的。

1 面向流程变化调整的业务流程软件模型及其过程代数描述

PCOSM 中,流程在整体结构上分为功能性的业务活动(功能活动)和连接器两部分。功能活动用来完成某个相对独立的业务操作;连接器具有复杂的逻辑运算功能,用来连接功能活动,以实现某种业务逻辑。

功能活动内部通过事件引发一系列行为来实现某个相对独立的业务目标,它由场景、行为、事件和消息槽组成,其内部逻辑结构由行为和连接器实现,如图 1 所示。

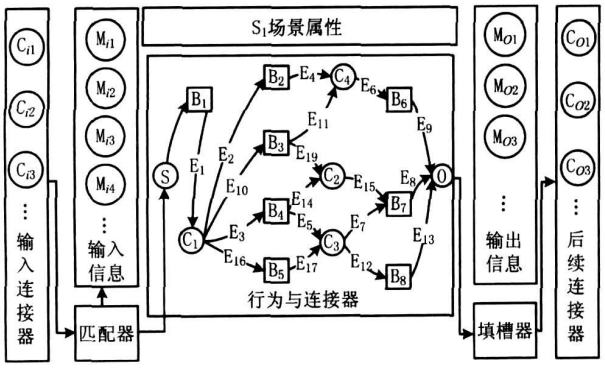

图1 功能活动的内部结构

连接器根据在模型中的位置分为两类,一类用来连接功能活动内部的行为,另一类用来连接功能活动,工作原理完全一样。其内部结构有前向输入逻辑和后继启动逻辑两部分,各自包括与(∧)、或(∨)、异或(⊕)三种基本逻辑操作符,前后两部分组合在一起实现某种连接逻辑,也可根据需要,级联组合出更多的复杂逻辑。如图 2 所示是一个行为连接器,左部是前向连接条件运算符,用来对输入事件进行集合运算,并对输出点连接行为的触发条件进行判定;右部是后继启动连接符,满足触发条件的行为将按照连接符的语义进行选择执行。

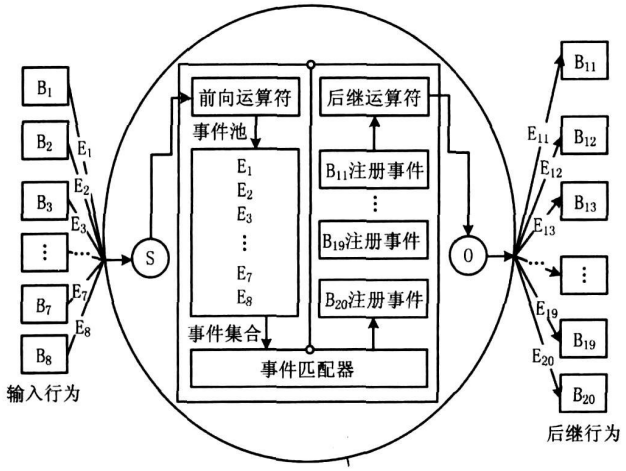

图2 行为连接器的内部结构

连接器的工作过程如下:

(1)输出点连接的后继行为,向连接器注册它被触发时所需要的事件条件。

(2)对输入点的输入事件,用前向连接符按照设定的运算规则进行集合运算,得到可以使用的事件集合。

(3)将后继行为的注册事件依照前向连接符对输入事件集合进行匹配操作。

(4)若前向连接条件运算符为"与",则表明注册的触发事件必须全部满足;若为"或",则触发事件必须至少有一个事件子集满足;若为"异或",则触发事件有且仅有一个满足。

(5)在所有可以被触发的行为中,根据后继启动连接符逻辑选择执行:①同时启动;②1 – n 个被启动;③唯一一个被启动。

把行为连接器中的事件换成消息、行为换成功能活动,则行为连接器变成功能活动连接器。功能活动连接器的前向输入部分用于实现业务流程逻辑结构,后继判定部分用于实现功能活动之间的依赖

关系, 如先后顺序依赖关系、并行依赖关系、互斥依赖关系、条件依赖关系等。目前, 按照前后不同的逻辑运算, 设计了九种组合的连接器, 这些连接组合足以表达出业务流程的各种结构。同时也为每种连接器规定了使用规则, 作为建模标准存入业务字典。

当一个业务流程被视为由功能活动和连接器两种元素组成以后, 流程的属性就被显示定义出来。行为连接器和功能活动器主要体现了流程调整的粒度, 使得流程变化的复杂性能控制在一定范围内。建模的关键是设计功能活动和连接器之间的连接关系, 而可变性研究的重点在于功能活动之间的消息交互规则和协议上, 即连接器与功能活动之间的连接方式。

1.1　PCOSM 定义

以下是 PCOSM 及其组成元素的巴科斯—诺尔范式(Bouks-Naur Form, BNF)定义:

Process::= <F_TypeID, ActivityList, ConnectorList>

Activity::= <A_TypeID, RequireMsgList, SendMsgList, Scenario, Context, ActualGoal>

Connector::= <C_TypeID, InputDotList, InputMsgSet, PreLogicConnective, ConnOutputDotList, ConnOutputObjectList, NextLogicConnective, Time>

ActualGoal::= <AG_TypeID, SupportGoalList, GoalParameterList>

SupportGoal::= <SG_TypeID, GoalParameterList>

Context::= <CT_TypeID, EventList, MsgList, DataList, Time, ResourceList, TimeSpent, ContextPreNext>

Message::= <M_TypeID, Source, DestinationList, ShareData, SendData>

Scenario::= <S_TypeID, BehaviorsList, ConnectorList, Infor>

Resource::= <R_TypeID, Rank, Attribute>

Event::= <E_TypeID, Time, Source, State, Rank>

Behavior::= <B_TypeID, Goal, InboundEventList, OutboundEventList, CallInterface, SupportGoal>

其中: Process 为业务流程; Activity 为组成 Process 的功能活动; Connector 为功能活动连接器; ActualGoal 和 SupportGoal 为实现目标和支持目标, 多个支持目标会完成一个实现目标; Context 为该功能活动的环境信息; Message 为功能活动之间传递的消息; Scenario 为存放一组行为的容器; Resource 为在某时段内的可用资源; Event 为事件, 可引发行为, 行为结束后又可产生新的事件; Behavior 为行为, 是完成某个支持目标的软件或人工动作。

另外, 所有的_TypeID 均是该元素的唯一标识符。RequireMsgList 和 SendMsgList 分别为表示功能活动被触发所需要的消息条件列表和执行完成后所发出的消息列表; InputDotList 表示输入点的集合, 可以为 0~ n 个输入点, 0 个表示该连接器为起始连接器, n 个表示该连接器为中间连接器。InputMsgSet 是该连接器输入的消息在 PreLogicConnective 运算符作用下生成的可用消息的集合; PreLogicConnective::= <AND | OR | XOR>; DataList 表示在一个时间段内的共享数据和传递数据; ContextPreNext 为该场景的上一个和下一个场景; ShareData 是共享数据, 是基本的数据项、数据表、组合数据结构, 用于在不同的功能活动之间共享; SendData 为不同的功能活动之间的传递数据; InboundEventList, OutboundEventList 表示该行为被触发时所需和结束后引发的事件; CallInterface 是行为调用具体的功能执行体的接口; Rank 表示资源在实现目标上的优先级; State 表示事件在行为开始执行时发出, 还是执行过程中或结束时发出。

上述定义描述了模型的组成内容, 它设有向业务流程建模者或系统建筑师提供一个形式化的理论基础, 通过这个基础, 对业务流程模型以一种严格的定量方式进行分析和验证。

1.2　PCOSM 的过程代数描述

过程代数是一种从通信角度描述并发系统、并能表达相关流程之间交互的计算模型。它有良好定义的数据传递序列过程语义, 并有严格的数学基础和精化、分析检测工具, 广泛用于高级语言建模和操作系统设计中。按照 CSP 的定义, 流程(Process, 或称为过程) 由一系列行为组成, 行为和一些事件关联。这些事件是原子的、并发的, 能够实现流程与环境的交互作用。一次通信过程被称作一个复合事件, 描述为 c. v, c 为完成通信的通道(通信处理流程) 名字, v 是通信所传递的值。以下用 BNF 范式给出 CSP 关于组合流程描述的简单语法:

P, Q ::= P| | Q| P| [A]| Q| P\A| P//Q| P △ Q| P ↖ Q| P □ Q| P ⊓ Q| P; Q| e → P| SKIP| STOP

e::= x| x . e。

式中: P, Q 各自是一个流程。两者之间的操作符表示它们的运行关系, 如并行、顺序、交叠等。A 是可引发流程执行的事件集。

CSP 有执行轨迹模型、稳定错误模型和不定错误模型三种语义模型用来描述流程的行为属性。它们对于流程模型的行为属性表达能力依次增强, 复杂性依次加大。目前, 较普遍采用第二种, 即 Failure(P)模型。

以下用 CSP 定义 PCOSM:

事件: Σ: {e| e∈αP}, αP 为某流程 P 的全部事件集合。

行为: B: $e \rightarrow P$, 其中: P 是一个不可再分的最小粒度的操作, e∈Σ, 它可以是多个事件通过逻辑运算符 LogicConnective 运算而成的复合事件。其轨迹模型为: traces($e \rightarrow P$) = {t| t= $\langle\rangle \vee (t_0 = e \wedge t' \in traces(P))$}。

连接器: 这里以行为之间的连接器为例(如图 3), 给出几个关键行为连接器的过程代数描述。

行为连接器: Cxx = AND/ /IN| [E] | LOGIN; MATCH; OUT, E 是进入该连接器的事件和后继行为注册事件的总和, xx∈{∧, ∨, ⊕}。

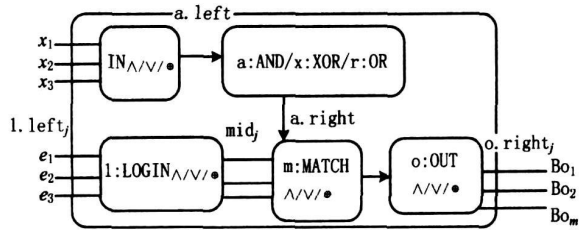

图3 〈与-与〉连接器结构

(1) 与- 与连接器 C∧∧

IN∧ = ||| 0< i< = n ini? $x_i \rightarrow$ l. left! $x_i \rightarrow$ IN

AND = a. left? $x_i \rightarrow$ a. right! ({x| x= x_j, x_j∈αBi1 ∪αBi2 ∪ … ∪αBin, j= 1…# αBi1 + # αBi2 + … + # αBin}) → overtime → SKIP

LOGIN∧ = ||| 0< j< = m l. leftj? $y_{jk} \rightarrow$ midj! ({y_j| y_j= y_{jk}, y_{jk}∈αBoj, k= 1…r}) → LOGIN

MATCH∧ = ||| 0< j< = m midj? {y_j| y_j= y_{jk}, k= 1…r} ||| a. right? ({x| x= x_j, x_j∈αBi1 ∪αBi2 ∪ … ∪αBin, j= 1…# αBi1 + # αBi2 + … + # αBin}) →

if {y_j| y_j= y_{jk}, k= 1…r} ⊆ {x| x= x_j, x_j∈αBi1 ∪αBi2 ∪ … ∪αBin, j= 1…# αBi1 + # αBi2 + … + # αBin}

then m. right! b_j

else MATCH∧

OUT∧ = m. right? b_j → {b_j} ∪P(bo) →

if P(bo) = bo then||| 0< j< = m o. rightj! bo → SKIP

else OUT∧

设前驱行为 Bi_j 有 n 个, j= 1, …, n; 后继行为 Bo_j 有 m 个, j= 1, …, m。x 为前驱行为的输入事件, 每个前驱行为的输入事件可能不止一个。y 为后继行为注册的启动事件, 每个后继行为可能有 r 个启动事件。b_j 事件表示在当前情形下, 行为 Bo_j 已获得自己的全部启动事件, 一个后继行为只会获得一个 b_j, 它是局限于连接器内部的事件, 仅表示行为 Bo_j 可以启动了。b_j∈bo, bo 为全部后继行为的启动事件集合, j = 1, …, m, 初始为空。$P(e)$ 是事件集合 e 的幂集: P(e) = {x| x⊆e}。overtime 是一个超时事件, 可以预先为每个连接器设置其有效等待时间, 它从第一个行为事件进入连接器后开始计时。

(2) 异或- 与连接器 C⊻∧

IN⊻ = □ 0< i< = n ini? $x_i \rightarrow$ l. left! $x_i \rightarrow$ IN⊻

XOR = x. left? $z_i \rightarrow$ a. right! ({z| z= z_i, z_i∈αBi1 or αBi2 or … αBin, i= 1…# αBij}) → SKIP//该过程只传递第一个进入连接器行为的全部事件集合。

MATCH⊻ = ||| 0< j< = m midj? {y_j| y_j= y_{jk}, k= 1…r} ||| a. right? ({z| z= z_j, z_j∈αBi1 or αBi2 or … αBin, j= 1…# αBij}) →

if # ({y_j| y_j= y_{jk}, k= 1…r} ∩ {z| z= z_j, z_j∈αBi1 or αBi2 or … αBin, j= 1…# αBij}) = 1

then m. right! b_j

else MATCH⊻

LOGIN 和 OUT 同 C∧∧。

(3) 或- 与连接器 C∨∧

IN∨ = Π 0< i< = n ini? $x_i \rightarrow$ l. left! $x_i \rightarrow$ IN∨

OR 同 AND。

MATCH∨ = ||| 0< j< = m midj? {y_j| y_j= y_{jk}, k= 1…r} ||| a. right? ({x| x= x_j, x_j∈αBi1 ∪αBi2 ∪ … ∪αBin,

0< j< = (# αBi1 + # αBi2 + … + # αBin)}) →

if {y_j| y_j= y_{jk}, k= 1…r} ∩ {x| x= x_j, x_j∈αBi1 ∪αBi2 ∪ … ∪αBin, 0< j< = (# αBi1 + # αBi2 + … + # αBin)} ≠ ≅

then m. right! b_j

else MATCH∨

LOGIN 和 OUT 同 C∧∧。

(4) 与- 或连接器 C∧∨

IN 同 C∧∧, MATCH 同 MATCH∨

OUT∨ = m. right? b_j → {b_j} ∪P(bo) →

if # P(bo) = MAX then||| 0< j< = MAX o. rightj! bo → SKIP

else OUT∨

(5) 与- 异或连接器 C∧⊻

IN 同 C∧∧, MATCH 同 MATCH⊻

OUT⊻ = m. right? b_j → o. rightj! bo → SKIP

功能活动: ACT(Mi, Cons, E, Bavs, Coups, Context, Mo)。其中: Mi 为输入消息集合, Mo 为输出消息集合, Cons 为连接器集合, E 为内部事件集合, Bavs 为行为集合, Coups 为行为- 连接器对集合。行为连接器对有两种形式: ③多个输入行为连接一个连接器 COUP(B- C); ④一个连接器连接多个输出行为 COUP(C- B)。功能活动由一系列 COUP 组合而成。

Cons = {C∧∧, C∧⊻, C∧∨, C∧∨, C∨∨, C∨⊻, C⊻∧, C⊻∨, C⊻⊻}, Coups ⊆ (Cons × P(Bavs)) ∪ (P(Bavs) × Cons)。

设 P(b, X) = b: X • start. b → SKIP, 集合 X 中的某个行为被其启动事件引发后成功执行, 直到结束。

则三种 COUP(B- C) 表示如下:

COUP(B- C∧x) = COUP((B1, B2, …, Bn), C∧x) = |||b: B• P(b, B)|[{start. C∧x}]|C∧x, 其中 x∈{∧, ∨, ⊕}, start. C∧x 为连接器 C∧x 的输入事件集合, B⊆BAV, start. y∈e, y∈{Cxx, a, b}。

COUP(B- C∨x) = COUP((B1, B2, …, Bn), C∨x) = Πb: B• P(b, B)|[{start. C∨x}]|C∨x。

COUP(B- Cŷx) = COUP((B1, B2, …, Bn), Cŷx) = □b: B• P(b, B)|[{start. C∨x}]|Cŷx。

三种 COUP(C- B) 如下:

COUP(Cx∧- B) = COUP(Cx∧, (B1, B2, …, Bn)) = Cx∧|[{b: B• start. b}]|||b: B• P(b, B)。

COUP(Cx∨- B) = COUP(Cx∨, (B1, B2, … Bn)) = Cx∨|[{b: B• start. b}]|Πb: B• P(b, B)。

COUP(Cxŷ- B) = COUP(Cxŷ, (B1, B2, … Bn)) = let XP(b, B) = Πb: B• P(b, B)

within XP(b, B)|[{b: B• start. b}]|□b: B• P(b, B)。

得: ACT(Mi, Cons, E, Bavs, Coups, Context, Mo) = mi→|||(COUP(B- COUP(C- COUP(B- …)))) → mo→ SKIP。

这里 (COUP(B- COUP(C- COUP(B- …)))) 实现了每个 COUP(B- C) 中的 C 与下一个 COUP(C- B) 是同一个的连接模式。

流程:

Process(Acts, Cons, Coups, Mess) = start. m→|||((COUP(A- COUP(C- COUP(A- …))))

□μX• (COUP(A- COUP(C- COUP(A- …)))); X)

→ end. m→ SKIP。

其中: start. m, end. m∈Mess, Mess 为消息集合; Acts 为功能活动集合; Coups 为活动- 连接器对集合, 它与“行为- 连接器”对完全类似。□μX• (COUP(A- COUP(C- COUP(A- …)))); X 表示在一个或多个连接器- 活动对之间的有限循环, {√}∈αCOUP。消息比事件具有更多的环境信息和数据, 但从流程的建模过程中, 这些环境信息和数据的具体内容并不需要表达, 因此流程的 CSP 描述和功能活动是类似的。

1.3 PCOSM 的合理性分析与检测

新建的或者重新修改过的流程必须保证其满足业务需求规约, 同时还应保证不应有锁的存在, 这里给出流程合理性定义。

定义 1　一个流程是合理的, 当且仅当:

(1) 它所有的流程实例都能发出结束消息, 且当流程发出结束消息时, 流程中没有未到达结束点的功能活动, 形式化表示为:

∀start. m, end. m, mess, ∃tr = ⟨e1, e2, …, en⟩, Process(Acts, Cons, Coups, Mess) $\xrightarrow{tr}$ end. m。

(2) 流程中不存在接收不到启动消息的功能活动, 形式化表示为:

∀ACT(mi)∈Process, ∃tr = ⟨e1, e2, …, mi⟩, Process‘Process $\xrightarrow{tr}$ Process’

功能活动的合理性定义与流程相似, 只是把消息的概念换成活动内的事件, 活动换成行为。

第 2 章将指出, 流程调整的最小粒度定为功能活动。换句话说, 可以不必验证参与流程的各个功能活动本身的合理性(默认其内部是合理的)。而连接器是以标准构件方式设计的, 因此流程的合理性验证是要对连接器- 活动对进行渐进增量式验证, 进而验证整个流程的合理性。

PCOSM 用 CSP 描述的一个主要动机是可以利用 CSP 的精化概念, 使用 FDR 来分析验证流程模型的行为属性[19-20]。FDR 将两个 CSP 过程, 即规约过程(业务需求) 和设计过程(流程模型) 作为输入, 利用过程代数中的互等价定理进行子过程的等价替换, 逐步精化, 这其中是以保证流程的行为属性不变为原则的, 然后穷举过程的状态空间中的全部状态, 验证设计过程和规约过程的一致性。

当且仅当 Q 的失效轨迹全部存在于 P 的失效轨迹集合中, 则流程 Q 失效精化(failure- refines) 了流程 P。此时 Q 具有 P 的所有行为属性, P ô F Q ⇔ Failure[Q] ⊆ Failure[P]。

其中: Failures(P) = {(s, X| s∈traces(P) ∧X∈refusals(P/s)}, 为流程 P 的失效迹集合。意为当 P 执行至轨迹 s 时, 若环境给出事件 X, 则陷入死锁。(s, X) 称为一个失败对, refusals 称为 s 之后的事件拒绝集。

对流程的属性检验是通过式(1) 来实现的, 其中 T(P) 为流程 P 的轨迹集合。

∀tr: T(P)• (tr, E)∉Failures(P)。　　(1)

流程的合理性定义中的 a. 可通过在 FDR 中的检验断言(2) 来实现。

DF ô F Process。　　(2)

其中: DF = Πx: E• x→ DF 是一个典型的不确定性无死锁流程。

对于合理性定义中的 b,需将流程中所有的活动- 连接器对 COUP(A- C),用逐步增量的组合方式进行断言(3) 的逐步检验。

如: COUP(A- C∧x) i ô FProcess\(Σ\αCOUP (A- C∧x) i)。 (3)

其中 Σ为流程 Process 的全部事件集。

2 可变业务流程的建模及调整方法

本文的研究基于面向服务架构(Service Oriented Architecture, SOA),得益于它的松耦合和灵活组合特征[21-22]。在前面的定义中,功能活动单元是一个能完成某个业务目标的相对独立的软件模块(或人工活动),在此被视为流程调整的最小单位,对应一个 Web 服务,用业务字典的方式进行标准化管理。一旦将某个业务操作以功能活动单元表达后,它们就是不可变的了,流程的变化限于功能活动模块的增删和彼此间逻辑结构的变化。

研究表明,制约业务流程变化的关键因素是业务操作之间的数据依赖,即流程的调整会使有依赖数据关系的两个业务操作出现逻辑上的错误。因此,建模中的重要环节是进行功能活动之间的数据依赖性分析。将这些依赖模式进行分类、显式描述,然后作为业务规则存入数据字典,并在以后的流程变化调整时遵循。

同时,把功能活动接收和发出的消息分为两类,即关乎业务流程中固定环节执行逻辑的系统消息和业务专家自定义的消息。系统消息之间的传递逻辑关系是不可变的。而另一类业务专家自定义的消息,则可根据实际的业务要求,调整消息的字段内容,如 destinct, share_data, send_data。由此,功能活动也分为两个大类: ⑴ 流程中必不可少、不允许被调整的功能活动,这类功能活动之间的逻辑构成采用系统消息进行交互; ⑷可变功能活动,采用业务专家自定义的消息进行交互,以最大的灵活性加入到业务流程中。

2.1 建模过程

综上所述,建模过程有以下两方面的工作,它们结合在一起完成初始业务流程的建立。

2.1.1 功能活动分析

(1) 确定哪些业务操作是人工的,哪些是计算机处理的。

(2) 将分析出来的独立业务操作抽象为功能活动。

(3) 分析功能活动之间的消息传递关系,对这些输入输出消息单独建模,见 2.1.2 节。

(4) 将功能活动关联到具体的场景(分析其内部必须具有的行为和执行顺序,以便映射到服务),场景要保持单一性和相对独立,其内部行为调用外部服务的接口必须稳定和明确。

2.1.2 数据依赖性分析及业务规则的确定

把数据分为三类: ⑴ 在功能活动中需要传递、共享的数据,将这些数据建模为流程的全局数据,放入全局数据缓冲区中; ⑷功能活动内部数据,这些数据的作用范围局限在行为之间,功能活动有一个自己的数据缓冲池; ㈣消息所携带的数据,它们在功能活动之间传递,这类数据具有很强的独立性,基本上只和提供数据的功能活动有关,并没有全局的意义,功能活动之间的数据依赖就发生在这部分交互数据上。

数据之间的依赖又可分为数据项前后依赖、并行依赖、数据项的多选一依赖三种情形。第 1 章给出的连接器模型能够实现以上三类不同的数据依赖。连接器定义了 InputDot, OnputMsgSet, PreLogicConnective,这几个字段可以构成功能活动的启动条件。PreLogicConnective 运算符可以构成各种不同的功能活动触发条件,以此来支持功能活动之间的数据依赖性。将这些依赖类型以业务规则的形式写入数据字典,作为建模和后续流程调整的依据和原则。

2.2 流程的变化调整方法

连接器是流程可变性元素的重要体现。业务流程的变化最终落实到功能活动外部消息接口的变化、连接器输入输出消息的变化、连接器前向/后继连接符的变化上,调整方法则需针对以上各个方面来进行。

功能活动消息接口的变化调整主要有: 添加(删除) 接收/输出消息(可能伴有连接器的添加/删除); 合并(拆分) 接收/输出消息。连接器的变化调整主要有: 前向(后继) 消息的添加/删除(功能活动的添加/删除); 前向(后继) 启动消息的合并/拆分; 连接器前向/后继连接符的更换(更换连接器类型)。

据此,总共设计了 21 类调整算法,通过一个调整类型识别过程接口进行相应的调用。作为例子,以下给出添加功能活动输入消息的算法:

算法 1 功能活动输入消息的添加

输入: 调整前的业务流程 C,添加消息的功能活动 A,添加到 A 的消息列表 AMList

输出: 调整后的业务流程 C

```
while(AMList! = empty)
{
    TMS= 取 AMList 中的一个元素, 从 AMList 删除 TMS;
MS= TMS;
if(C 中存在 MS)
{    CA= A 的注册连接器;
    if(MS 属于 CA 的输入消息集合)
    A 可以添加 MS;
    else
    {  MSS= MS 的 source;
      查询业务规则, 检查 A 和 MSS 的数据依赖性和逻辑关系
      Case 1: 无数据依赖性或者 A 依赖于 MSS
          Case 1. 1: 无逻辑关系
            MSS 向连接器 CA 发送消息 MS, 加上连线 LMS;
          Case 1. 2: 有并行关系或者互斥关系
          A 不可以添加 MS;
      Case 2: MSS 依赖于 A
          A 不可以添加 MS;
      CMSS= MSS 的注册连接器;
      if(CA, CMSS 之间的连接符合连接器的连接规则)
        A 可以添加 MS;
      else
          撤销 MSS 发送给 CA 的 MS 和连线 LMS
    }
}
else
{  MS 的 source 填上业务人员指定的功能活动号;
    填充 MS 的信息;
    MS 加入 AMList;}
}
```

经调整后的流程, 需按照 1. 3 节的思想和方法进行合理性验证。

3　举例与讨论

现给出一个航空港塔台调度流程的建模和流程调整过程。它需要管理调度五个区域的功能活动: ¹停机坪(LEAVEFLIGHT): 管理离港航班; ④滑行道(SLIDE): 离岗航班接到命令准备起飞, 进港航班着陆结束; ㈣跑道(RUNWAY): 离岗航班起飞, 进港航班着陆; ¼ 进近净区(ACCESS): 进港航班接到命令离开等候区进入跑道; ½ 等候空区(WAITHOVER): 进港航班分层盘旋等候, 最底层的先着陆。图 4 所示是从各个区域角度描述的业务流程图。

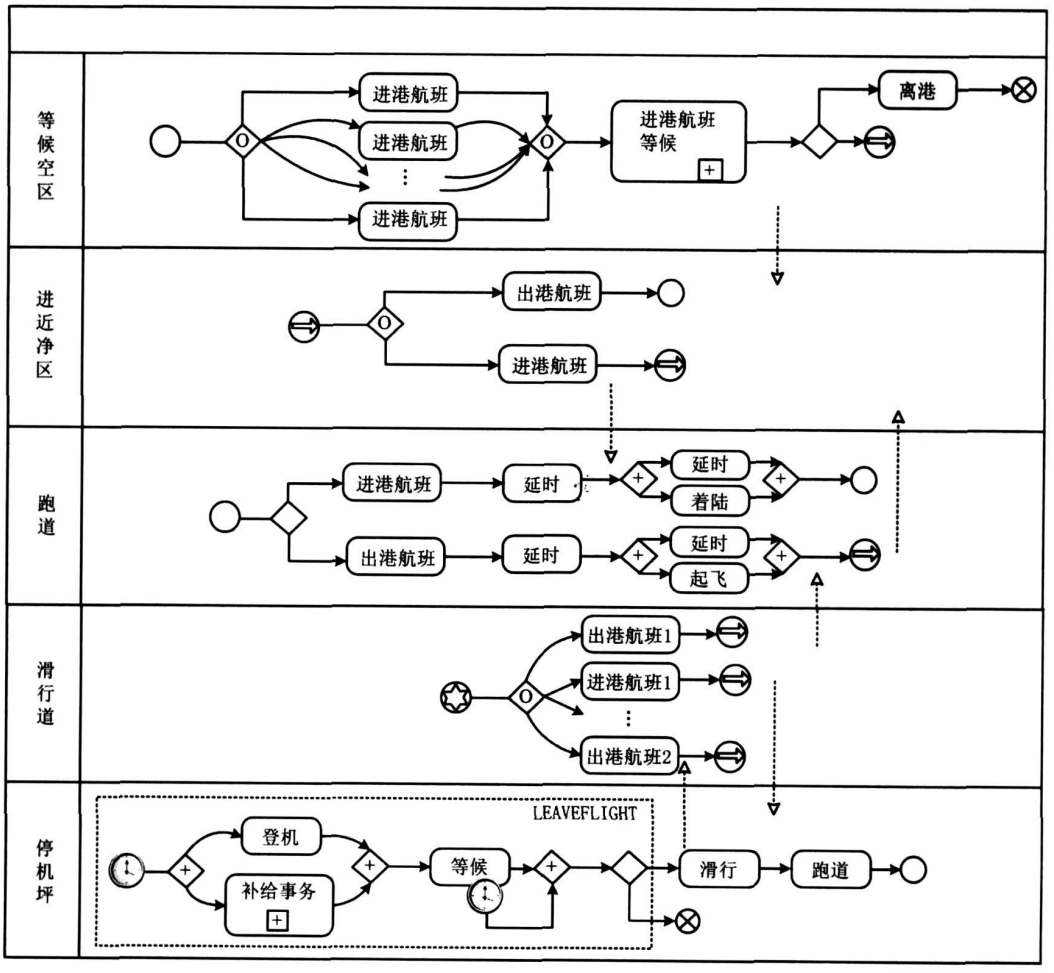

图4　机场塔台调度流程

用过程代数描述一个业务流程的步骤如下: ①为各个功能活动定义其事件集合; ④给出各个功能活动的行为描述; ㈣定义整个系统的合成行为。

下面为塔台调度流程(Control Tower Scheduler, CTS)设计流程模型。其中: Acts 表示 CTS 中的功能活动集合, αCTS 为 CTS 中的消息集合。

αCTS= {start, initflighting, initlanding, flightin, complete, beready, outport, inport, cancelout, cancelin, timeout, overtime, flyingoff, landfall, fail},

Acts = {LEAVEFLIGHT, SLIDE, RUNWAY, ACCESS, WAITHOVER, ENTERFIGHT, WAIT}。

步骤 **1** 给出各个功能活动的 CSP 描述。

LEAVEFLIGHT= initflighting → (BOARDING ||| SUPPLYING) |[{boardingend, supplyingend}]|

$C_{\wedge\wedge}$; WAIT → timeout → (beready □ cancelout) → SKIP

SLIDE: VAR= left? inport → $VAR_{outport}$

where

$VAR_{outport}$ = (left? outport → VAR_{inport} □ right! inport → $VAR_{outport}$)

RUNWAY= $|||_{0<i<=m}$ μX • (left$_i$? y → WAIT → right$_i$! y → overtime → X), αleft$_i$= αright$_i$, m 为机场跑道数, y∈{outport, inport}

ACCESS= μX • (left? y → right! y → X), αleft = αright, y∈{outport, inport}

WAITHOVER= W_{-}

where

W_{-}= left? x → W_{-x}

W_{-x-s}= (left? y → W_{-x-s-y} □ right! x → W_s), αleft= αright,

x, y∈{fight | fight= fight$_i$, i= 1 …Maxin}, Maxin 为航空港一天内进港航班最大总数。

步骤 **2** 逐个把以上功能活动和连接器进行组合

单个出港航班 i 的完整流程如下:

$COMPOUT_i$ = startflighting → COUP(COUP((LEAVEFLIGHT$_i$, SLIDE) − $C_{\curlyvee\curlyvee}$), ACCESS), $C_{\curlyvee\curlyvee}$), RUNWAY)) → flyingoff

一日全部出港航班的合成调度流程如下:

COMPOUT= start → ㊦$_{0<i<=Maxout}$ P_i

where if i= [a, b]//标号为 a~ b 的出港航班可连续离港, 此为进港航班稀疏区间

then P_i= μX • COUP((…((COUP(COMPOUT$_a$, $C_{\wedge\wedge}$), COMPOUT$_{a+1}$), $C_{\wedge\wedge}$), COMPOUT$_{a+m}$) …COMPOUT$_b$) → X

else if i= [u, v]//u, v 出港航班必须和进港航班交叉执行, 此为进港航班频繁区间

then P_i= μX • COUP(COUP((COUP((LEAVEFLIGHT$_u$, …, LEAVEFLIGHT$_v$, SLIDE) − $C_{\curlyvee\curlyvee}$), ACCESS), $C_{\curlyvee\curlyvee}$), RUNWAY) → X

else P_i = μX • (COUP((…((COUP(COMPOUT$_1$, $C_{\wedge\wedge}$), COMPOUT$_2$), $C_{\wedge\wedge}$), COMPOUT$_3$) …COMPOUT$_n$) → X)

□(COUP(COUP((COUP((LEAVEFLIGHT$_1$, …, LEAVEFLIGHT$_n$, SLIDE) − $C_{\curlyvee\curlyvee}$), ACCESS), $C_{\curlyvee\curlyvee}$), RUNWAY) → X)

其中, Maxout 为一日离岗航班总数。 ㊦$_{0<i<=}$ P_i = (αP$_0$, $\cup_{0<i}$ trcces(P_i)) 表示一个顺序执行的过程序列串。

进港航班调度流程如下:

COMPIN= initlanding → $|||_{0<j<=maxin}$ (COUP((COUP(ENTERFIGHT, $C_{\vee\vee}$); WAITHOVER), RUNWAY), $C_{\wedge\curlyvee}$) → RUNWAY → landfall) □ cancelin) → SKIP。

综上所述, 最后给出:

CTS= start → (COMPORT |[{inport, outport}]| COMPIN) → complete → SKIP。

为验证 CTS 的合理性, 这里给出 CTS 模型必须满足的一些行为约束规约(如式(4)~式(7))。式(4)意为 CTS 必须是无锁流程; 式(5)意为进近区的进港飞机必须进入跑道着陆, 其中 FALL = ACCESS; RUNWAY= μX • (inport → landfall → X)。

式(6)意为该业务流程或取消起飞(着陆)而结束或成功完成一次离港(进港), 其中: Sset= {cancelout, cancelin, overtime},

SS= (Πx: Sset • x → fail → SS) Πcomplete → SS。

式(7)意为每条跑道上只能同时有一架出(入)港航班, 其中: RR= (beready → RUNWAY → flyingoff → RR) □(inprot → RUNWAY → landfall → RR)

$$DF \hat{o}_F CTS, \tag{4}$$

$$FALL \hat{o}_F CTS \backslash (\sum \backslash \alpha FALL), \tag{5}$$

$$SS \hat{o}_F CTS \backslash (\sum \backslash \alpha SS), \tag{6}$$

$$RR \hat{o}_F CTS \backslash (\sum \backslash \alpha RR)。 \tag{7}$$

CTS 流程会经历的一种最常见的变化为: 增加航线(或某航线上的航班的增减) LEAVEFLIGHT$_x$。此时 CTS 的调整过程在 COMPOUT 中进行, 按照 2.2 节列举的调整算法, 它属于增减连接器的消息, 或者更换连接器类型: 如果该航班需要增加在进港航班稀疏时间区间, 则首先将 LEAVEFLIGHT$_x$ 构造成 $COMPOUT_i$, 然后采取以下两种调整方式: ①直接在 P_i 中按照该航班的时间顺序增加一个活动连接器对 COUP(COMPOUT$_x$, $C_{\wedge\wedge}$); ④并入某个已存在的活动连接器对, 但要将 $C_{\wedge\wedge}$ 改为 $C_{\curlyvee\curlyvee}$, 同时为其增加输入一个消息 flyingoff$_i$。如果该航班需要增加的时间属于进港航班频繁时间区间, 则调整方法是在(COUP((LEAVEFLIGHT$_u$, …, LEAVEFLIGHT$_v$, SLIDE)) − $C_{\curlyvee\curlyvee}$)中按照要

求的时间顺序增加 LEAVEFLIGHT$_x$, 并为 C$_{YY}$ 增加输入消息 bready$_i$。或者调整方式是上述两种情形的结合。然后将修改后的流程按照式(4)~式(7)进行合理性验证。

假如有航班需要调整起飞时间, 则应删除该航班与原来的连接器之间的消息, 在所需位置的连接器上, 增加该航班的输入消息, 并根据该航班与其他航班之间的逻辑关系, 更换连接器的类型。

以上建模过程中最为关键的工作是功能活动的划分和定义。针对不同的软件模型, 应有不同的分析方法和角度。PCOSM 模型定义业务流程的着眼点是日后流程的调整。因此, 从建模之初就要分析整个业务流程中的可变和不可变的业务操作环节, 加以准确划分并进行独立建模, 这是至关重要的。本例中, 出港航班的准备工作、跑道、等候空区等内部工作流程必须严格遵守, 因此将它们定义为独立的功能活动。而对于一个航空港的航班调度管理流程来说, 最为典型的变化是(出/入)航班、航线的增减, 因此把每一个航班独立建模成不同功能活动, 之间用连接器相连(这里大量地采用了 C$_{YY}$ 和 C$_{\wedge\wedge}$), 才能在模型的角度上体现出调整的可能, 且便于调整后的正确性验证。假如用传统的流程分析思想, 一个航班的业务操作是相同的, 显然它们应当被作为一个统一的标准模块出现在流程中。

4　结束语

基于 BPMS 对业务流程的调整灵活性需求, 提出了一种面向流程变化的业务流程模型, 并给出其过程代数的形式化描述, 以及验证分析方法。该模型主要由功能活动和连接器构成, 其中功能活动是完成某个特定业务操作的相对独立的软件模块, 连接器连接业务操作, 实现复杂的业务逻辑, 将功能活动按照需要的业务流程逻辑连接起来。连接器的作用是用来表达业务逻辑中的流程属性。当业务流程需要调整时, 主要是对连接器的调整。然后利用 FDR 验证工具, 分析验证流程的合理性, 以实现在无需 IT 人员干预下, 业务人员通过重新配置流程便可进行业务流程重组的软件调整。本文给出了模型的定义、CSP 描述、建模步骤和流程调整方法。最后以机场塔台调度流程的建模和调整过程为例进行了说明。

下一步工作包括以下几个方面: ①流程模型的实现。将功能活动和 Web 服务实现完整的映射, 研究针对该模型的 Web 服务搜索绑定技术, 并研究所实现的流程质量保证策略。④图形化流程设计工具的开发。

参考文献:

[1] VAN DER AALST W M P, TER HOFSTEDE A H M, WESKE M. Business process management: a survey[J]. Lecture Notes in Computer Science, 2003, 2678: 1-12.

[2] STRNADL C F. Aligning business and IT: the process-driven architecture model[J]. Information Systems Management, 2006, 23(4): 67-77.

[3] MCGOVERAN D. An introduction to BPM & BPMS[J]. Business Integration Journal, 2004(4): 2-10.

[4] GEORGE M, GIAGLIS A. Taxonomy of business process modelling and information systems modelling techniques[J]. International Journal of Flexible Manufacturing Systems, 2001, 13(2): 209-228.

[5] SCHEER A W, NUTTGENS M. ARIS architecture and reference models for business process management[J]. Lecture Notes in Computer Science, 2000, 1806: 301-304.

[6] ZHANG Haifan. Introduction to software engineering[M]. Beijing: Tsinghua University Press, 1994(in Chinese). [张海藩. 软件工程导论[M]. 北京: 清华大学出版社, 1994.]

[7] DONG Yunwei, LOU Wenxiao, HAO Kegang. The software architecture of a business process management and integration system SynchroFLOW[J]. Computer Science, 2004, 31(6): 138-140(in Chinese). [董云卫, 楼文晓, 郝克刚. 业务流程管理与集成系统 SynchroFlow 的软件体系结构[J]. 计算机科学, 2004, 31(6): 138-140.]

[8] HRUBY P. Mapping business processes to software design artifacts[J]. Lecture Notes in Computer Science, 1998, 1543: 234-236.

[9] DOWNS D, LUNN K. Analysis and design for process support systems using goal-oriented business process modelling [EB/OL]. (2000-11-18) [2008-07-04]. http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-109/paper6.pdf.

[10] KING S F, JOHNSON O A. VBP: an approach to modelling process variety and best practice[J]. Information and Software Technology, 2006, 48(11): 1104-1114.

[11] VAN DER AALST W M P, JABLONSKI S. Dealing with workflow change: identification of issues and solutions[J]. International Journal of Computer Systems, Science, and Engineering, 2000, 15(5): 267-276.

[12] SMARI W W, DONEPUDI S, SEUNG K, et al. Efficient handling of changes in dynamic workflow systems[C]//Proceedings of International Symposium on Collaborative Technologies and Systems 2006. Piscataway, N. J., USA: IEEE

Press, 2006: 440-449.

[13] CICHOCKI A, ANSARI H A, RUSINKIEWICZ M, et al. Workflow and process automation-concepts and technology [M]. Berlin, Germany: Springer, 1998.

[14] JENG J J, SCHIEFER J, CHANG H. An Agent-based architecture for analyzing business processes of real-time enterprises[C]//Proceedings of the 7th International Enterprise Distributed Object Computing Conference. Piscataway, N. J., USA: IEEE Press, 2003: 86-97.

[15] FAKAS G J, KARAKOSTAS B. A peer to peer(P2P) architecture for dynamic workflow management[J]. Information and Software Technology, 2004, 46(6): 423-431.

[16] HOARE C A R. Communicating sequential processes[J]. Communications of the ACM, 1978, 21(8): 666-677.

[17] BAETEN J C M. A brief history of process algebra[J]. Theoretical Computer Science, 2005, 335(2/3): 131-146.

[18] Formal Systems (Europe) Ltd. Failures-divergences refinement[EB/OL]. (2007-07-23) [2007-08-12]. http://www.fsel.com/documentation/fdr2/html/index.html.

[19] WONG P Y H, GIBBONS J. A process-algebraic approach to workflow specification and refinement[J]. Lecture Notes in Computer Science, 2007, 4829: 51-65.

[20] WONG P Y H. Towards a unified model for workflow orchestration and choreography[D]. Oxford, UK: Oxford University, 2006.

[21] PAPAZOGLOU M P, HEUVEL W V D. Service-oriented design and development methodology[J]. International Journal of Web Engineering and Technology (IJWET), 2006, 2(4): 412-442.

[22] ANDREA D, MARCHESE M. Towards a service-oriented development methodology[J]. Journal of Integrated Design & Process Science, 2005, 9(3): 53-62.

《计算机集成制造系统》投稿程序

《计算机集成制造系统》期刊已经启用远程网络投稿系统,请您登录 cims.diamt.net.cn 网站,进行投稿或查询稿件。具体步骤为:

1、进入作者投稿窗口,输入作者的用户名和口令(如果是第一次投稿,请先注册),按照提示上传 Word 格式文章。编辑部每日收取一次稿件,收到后将对稿件进行初审,并发 E-mail 回复作者;

2、作者收到稿件通过初审的回复后请汇审稿费 150 元至“北京 2413 信箱 34 分箱 CIMS 编辑部(100089)”。汇款时务请注明稿件编号或第一作者姓名,并写明开发票时的付款单位;

3、编辑部收到审稿费后,将稿件送同行专家评审,3~6 个月有评审结果(可以通过登录网站进行查询)。经过主编审定,或者退稿,或者退修(编辑部会将修改意见 E-mail 返回给作者);

4、编辑部收到作者返回的修改稿后,将根据修改情况发出通知,告知作者稿件是否录用以及交版面费的具体办法。编辑部收到版面费后,将寄出正式录用通知和发票。

请牢记您的用户名和口令,以便查询。