学校代码：10246
学号：12110240015

博士学位论文

业务流程建模与业务流程模型验证方法研究

Research on Methodology of Business Process Modeling and Business Process Model Verification

院　　系：　计算机科学技术学院
专　　业：　计算机软件与理论
姓　　名：　康国胜
指导教师：　张亮教授
完成日期：　2019 年 2 月 20 日

指导小组成员名单

顾　宁　　教授　博导

汪　卫　　教授　博导

李银胜　副教授　博导

目 录

图目录

表目录

摘 要

业务流程通过协调一系列的任务来实现特定的业务目标。在企业中，业务流程驱动着企业的运作，因此对流程类的业务信息系统来说业务流程的敏捷是组织敏捷的基础。为了实现业务目标，业务流程建模成为企业的一项常见的工作，然而如何有效地建立高质量的业务流程模型以快速地响应变化的业务需求却是一个挑战性的问题，与这个问题相关的具体问题涉及方方面面，比如：如何对异构的业务流程模型进行整合建模与业务监控、如何对新的业务需求场景复用已有的业务流程模型快速地进行建模以及如何验证所建立的业务流程模型的行为正确性问题等等，这些都属于业务流程建模的范畴。因此，本文针对业务流程建模与流程模型验证这一领域中尚未解决的部分问题展开研究并给出具体的解决方案。

企业的发展常通过合并、收购等方式进行扩大。同一个类别的业务流程在各分公司以自底向上的方式独立地建立，这种自底向上的发展方式常常导致企业内部业务流程的异构、冗余和不一致等问题，给企业的业务流程管理和决策带来了严重的挑战。为了了解业务进展的全局视图，并为全局的业务决策提供基础，需要对异构业务流程进行整合与监控。然而，已有的业务流程整合方法主要集中在流程模型的集成，其主要目标是业务流程的建模与管理。该方法在面对大规模异构业务流程时，得到的模型将过于复杂而难以理解和使用。由于各地执行过程的异构性，无法实现整体业务进展的统一监控。因此，现有方法不能用于大规模异构业务流程模型的整合问题。同时，为了促进业务的推广，需要在新的业务场景中建立新的符合业务需求的业务流程模型，这时就需要一种可配置的业务流程建模方法来支持业务流程模型的复用，提高业务流程建模的效率。然而，已有的业务流程可配置建模主要集中在传统以活动为中心的业务流程模型，将流程中的数据和业务规则掩盖于流程模型之下，无法适用于新型的 Artifact-centric 业务流程可配置建模。另外，对于新建立的 Artifact-centric 业务流程模型，Artifact 生命周期之间可能存在复杂的同步，在部署之前需要保证流程模型的正确性属性。然而，已有的 Artifact-centric 业务流程模型正确性验证方法主要考虑单个 Artifact 的行为，而忽略了 Artifact 之间的同步，因此对 Artifact-centric 业务流程模型同步的行为正确性进行有效的验证是亟待解决的问题。

本文针对传统以活动为中心的业务流程建模范式下异构业务流程环境中的流程整合与流程监控问题，以及新型 Artifact-centric 业务流程建模范式下流程的可配置建模与流程模型同步的行为正确性验证问题展开研究，具体内容如下：

（1）提出一种模式驱动的异构业务流程整合与监控方法，将业务流程模型化为抽象的业务模式的交互。首先，分析异构业务流程模型的结构，抽取频繁出现的业务行为，定义为业务模式。并对每个业务模式进行参数化描述，使其可以根据具体的应用场景进行配置为具体的业务模式。然后，为执行业务模型，将各业务模式转化为工作流模式或运作模式的描述，使得业务模型可以运行于目前大部分的工作流程引擎之上。为提供企业业务的全局视图，在业务模式的边界设置复杂事件处理监控点，这些监控点恰好与业务阶段对齐，因此可以监控业务的进展。最后，方法的实证研究验证了该方法的有效性。本文的流程整合与监控方法有助于提高业务流程建模的效率，简化业务流程的管理，并为业务决策提供依据。

（2）提出一种通过对流程中包含的所有 Artifact 生命周期模型的配置来实现 Artifact-centric 业务流程模型的整体配置的方法。首先，提出 Artifact 生命周期图的概念来对 Artifact 生命周期模型进行图形化描述；然后，提出 Artifact 生命周期图的合并方法；在合并的 Artifact 生命周期图中，加入配置点和设置配置选项，使其形成一个可配置的 Artifact 生命周期图；最后，对 Artifact 生命周期图进行配置，亦即 Artifact 生命周期模型的配置。在对所有 Artifact 生命周期模型对应的 Artifact 生命周期图进行配置后，可得到一个完整的 Artifact-centric 业务流程模型，其中包含 Artifact 数据模型和相关的服务、业务规则，说明了该方法的有效性。本文的可配置建模方法有助于提高业务流程模型的复用，降低业务流程建模的难度，并提高流程建模的效率。

（3）提出一种 Artifact-centric 业务流程形式化可配置建模方法框架，以细粒度的方式考虑流程模型中所有的组成元素。为获得多个 Artifact-centric 业务流程形式化模型变体的整合模型，提出 Artifact-centric 业务流程形式化模型的合并操作。根据模型元素的“可变”或“共同”特征，在整合模型中识别出可配置点，并为可配点设置相应的配置选项，最终得到可配置的 Artifact-centric 业务流程形式化模型。基于可配置流程模型的行为选择配置选项，可配置出新的 Artifact-centric 业务流程模型。为了辅助业务流程模型的配置过程，基于流程模型元素关系图的概念提出了配置指南。基于特征模型表达 Artifact-centric 业务流程形式化模型，对保障性住房申请审批流程的真实案例进行分析，表明了该方法的有效性。

（4）针对带复杂同步条件的 Artifact-centric 业务流程模型的行为正确性验证问题，提出一种基于转化为 Workflow Net 的行为正确性验证的解决方法。首先，将流程中涉及的每个 Artifact 生命周期分别映射为 Petri Net 的等价描述。然后，基于同步约束提出系列的合成规则将这些 Petri Net 合成描述为一个集中的

Workflow Net。基于得到的 Workflow Net 计算其可达图以获得原 Artifact-centric 业务流程模型描述的所有可能的服务（即活动）执行序列。最后，流程模型的行为正确性（即正常完成）的验证通过验证每个可能的服务执行序列是否在控制流和数据流两方面均正常完成来实现。一个真实的案例分析证实了该方法的有效性。本文的行为正确性验证方法有助于为 Artifact-centric 业务流程模型的修正提供基础，降低后续系统修改带来的维护成本。

关键词：业务流程整合、业务模式、Artifact-centric 业务流程、业务流程配置、Artifact 生命周期、配置指南、行为正确性、Workflow Net

中图分类号：TP3

Abstract

Business processes describe a set of tasks in coordination to achieve a particular business goal. In enterprises, business operations are driven by business processes, thus business process agility is the basis for organizational agility for business information systems in process style. To achieve business goals, business process modeling is a routine task within a company. However, how to build high-quality business process models effectively to respond to the dynamic business requirements is a great challenge. That involves a lot of research issues, like how to integrate heterogeneous business process models for consolidation and business monitoring, how to build business process models quickly for new business application scenarios by reusing the existing business process models, how to verify the behavioral soundness of process models, and so on. Consequently, this paper focuses on some unsolved research issues of business process modeling and verification, and proposes specific solutions.

Enterprises today usually grow through mergers, acquisitions. Processes of the same business in different subsidiaries are often built in a bottom-up manner independently, which frequently leads to problems of process heterogeneity, redundancies and inconsistencies. These situations cause serious challenges for business process management and operational decision. To derive the global business progress for supporting operational decision, business process consolidation and monitoring are needed for heterogeneous business processes. However, the existing business process consolidation approaches mainly focus on process integration, aiming at management of process models and process modeling. The resulted model would be too complex and hard to understand for a large number of heterogeneous business process models. Most importantly, they are not applicable for united monitoring due to the heterogeneous execution process in different scenarios. Therefore, the consolidation and monitoring is an unsolved research issue for a large number of heterogeneous processes. Meanwhile, for business promotion reason, enterprises usually have to build business process model for new business requirements. Under such a circumstance, a configurable modeling approach is needed for supporting model reuse, so that the efficiency of process modeling could

be improved. However, the existing configurable process modeling approaches mainly focus on the traditional activity-centric business processes, in which the data and business rules are covered under processes. Thus, these approaches are not suitable for configurable Artifact-centric business processes modeling. In addition, the correctness properties of Artifact-centric business process models should be guaranteed before deployment. However, the existing verification approach for Artifact-centric business process models focus on the behavior of single Artifact, while neglecting the complex synchronizations among Artifact lifecycles. Therefore, correctness verification for Artifact-centric business process models with complex synchronization is still a research issue.

This paper focuses on the following research issues: heterogeneous business processes consolidation and monitoring under the traditional Activity-centric business process modeling paradigm, configurable process modeling and behavioral soundness with complex synchronizations under the new kind of Artifact-centric business process modeling paradigm. The main contributions are as follows.

（1）This paper proposes a heterogeneous business process consolidation approach based on business patterns in which a business process is modeled as the interacting abstract business patterns. First, heterogeneous business process models are analyzed to extract the frequent business behavior structures which are defined as business pattern. Further, business patterns are described with parameters, so that each pattern can be further configured to meet a specific subsidiary's requirements. Moreover, considering the execution for business model, each pattern is transformed into a set of connected workflow patterns or operational patterns, thus the proposed approach can be realized on most workflow engines today. To provide the global view of business, we apply Complex Event Processing (CEP) on boundaries of business patterns that are aligned with business stages. Thus, the business progress can be monitored. A empirical study of the proposed approach in a large company reveals positive feedback with respect to process consolidation. The proposed approach is helpful to improve the efficiency of process modeling, manage the heterogeneous business process models, and provide foundation for business decision.

（2）This paper proposes an approach for configurable Artifact-centric process modeling, which is achieved by the configuration of Artifact lifecycle models for all the involved Artifact classes. First, we apply deterministic finite automaton to

formalize an Artifact Lifecycle Model (ALM). To derive configurable ALM, we propose merger operation for ALMs. Also an Artifact Lifecycle Graph (ALG) is proposed to describe ALM as a state diagram. Then, we propose configurable ALG (C-ALG) to describe the configurable Artifact lifecycle model, in which arcs and sets of business rules can be defined to be configurable points. Finally, by configuring C-ALGs for each Artifact class with individualization algorithm, structurally sound Artifact lifecycle models are derived (including data models of Artifacts, associated services and business rules), thus resulting a complete configured Artifact-centric process model, demonstrating the effectiveness of the proposed approach. The proposed approach is helpful to increase the reuse of process models, make process modeling easier and thus improve the efficiency of process modeling.

（3）This paper propose a configurable modeling framework for formal Artifact-centric business processes in this paper, which includes all the process elements in fine-grained manner. To derive the integrated model for multiple process model variants, we propose merger operation for Artifact-centric process model variants. We get a configurable Artifact-centric process model by identifying configurable points in the integrated model by *common* and *variable* characteristics. And the associated configuration alternatives are set accordingly for configurable points. New Artifact-centric process models can be derived by configuration based on the behavior of configurable model. To facilitate process configuration, guidelines are analyzed based on the notion of process element relation graph. A real world case study based on feature model representation is conducted to illustrate the effectiveness of our approach.

（4）To address the verification problem of behavioral soundness for an Artifact-centric process model with synchronizations, this paper proposes a verification approach by transforming an Artifact-centric process model to a workflow net, and verifying behavioral soundness of the workflow net with aspects of both control flow and data flow. First, each Artifact lifecycle involved is mapped to a Petri net representation. Then we propose rules to integrate the Petri nets into a workflow net based on synchronization constraints. With the workflow net, a reachability graph is calculated to derive all the implicitly specified service execution sequences. Finally, behavioral soundness (i.e., proper completion) is checked by verifying whether every specified service execution sequence can complete properly or not from its control flow and data flow respectively. A case study is presented to

demonstrate the effectiveness of our approach. The proposed approach is helpful to provide guidelines for correction of Artifact-centric process models, and reduce maintenance cost of business process systems.

Keywords: Business Process Consolidation, Business Pattern, Artifact-centric Business Process, Business Process Configuration, Configuration Guideline, Artifact Lifecycle, Behavioral Soundness, Workflow Net

Chinese Library Classification: TP3

第1章 绪论

在企业中，业务流程驱动着企业的运作，成为企业信息系统的重要部分。对流程类的业务信息系统来说，流程敏捷是组织敏捷的基础。在 Gartner 报告中，业务分析师将流程敏捷定义为：组织感知需求变化并及时响应变化而做出改变运作的能力[1]。为了实现企业的业务目标，业务流程建模是企业的一项常见的工作，然而如何有效地建立高质量的业务流程模型却是一个挑战性的问题，比如：如何对异构的业务流程模型进行整合建模与业务监控、如何对新的业务需求场景复用已有的业务流程模型进行快速建模以及如何验证所建立的业务流程模型的行为正确性问题等等，这些都属于业务流程建模的范畴。其中，异构的业务流程模型的整合建模属于流程建模的一个方面，目标是为了业务监控；而复用已有流程模型建立新的流程模型属于设计阶段的流程建模，目的是将流程推广到新的应用场景；流程模型的行为正确性验证是对建模结果的检验，为了确保流程模型在正式部署时可正常完成。因此，本文针对业务流程建模与业务流程模型验证这一主题展开研究，对目前这一领域中的三个相关问题给出解决方案。本章首先介绍本文的研究背景与动机，并讨论研究问题所面临的挑战；然后，介绍相关工作；最后，介绍本文的主要内容及章节安排。

1.1 研究背景与挑战

现今快速变化的业务环境使得公司的业务和相关信息系统处于不断的变化当中。业务信息系统的演化是处理利益相关者提出与目标的内容和行为相关的局部改变的应用，以对齐新的业务需求。一种处理应用行为变化的方法就是通过改变潜在的业务流程[2]。在动态的环境当中，寻求重用[3]和适应性[4]是成功的业务流程设计的标志。流程变化包括静态的（即设计时）和动态的（即运行时）[5]，本文研究主要针对流程模型级别的变化，因此属于适应业务需求变化的流程模型静态变化。流程模型的变化问题是 BPM（Business Process Management）领域的常见问题，尤其是当需要将一个业务流程模型部署到不同的应用场景时。由于不同的应用场景可能带来不同的业务需求，流程模型需要做出相应的改变以适应新的应用场景，因此高效率低成本地建立业务流程模型显得尤为重要。请注意：严格意义来讲，现实中没有流程这一概念的对应实体，因此在本文中当提到“流程”这一词时要么是指流程模型要么是指流程实例，具体语义请根据上下文确定。下面，针对本文的研究问题分别介绍其研究背景、动机及挑战。

1. 异构业务流程的整合建模与监控问题

随着企业的发展和业务需求的变化，一个流程的变体可能出现在组织内部或跨越不同的子公司。各分公司常以自底向上的方式各自建立业务流程模型及流程管理系统，导致企业内部出现同一业务拥有众多不同流程模型变体的情况。这种企业的规模扩大方式常带来大量异构、冗余和不一致的业务流程模型，给企业的流程管理和运作决策带来挑战。组织开发成百上千的业务流程模型这种现象是很正常的，文献[6]给出了对管理大量流程模型的技术和挑战。

以中国移动公司为例，该公司在全国拥有 31 家分公司，其办公系统的内部业务流程模型以自底向上的方式分别独立地建立，业务信息系统已经运行了 10 多年。这些业务流程在各分公司中独立地维护和演化，导致业务流程模型的异构性，导致中国移动 OA（Office Automation）系统中的业务流程模型数量高达 8000 多个[7,8]。为了统一地描述和管理异构的业务流程模型，需要进行业务流程模型的整合，以达到统一的业务目标，建立一个集中的 OA 系统，以支持异构业务流程的监控，为领导层提供业务进展的整体视图，以便为业务决策提供基础，这是本文的研究动机之一。

然而，因为企业的业务时刻都需要运转，业务流程管理系统 BPMS（Business Process Management System）必须时刻运行，至少不能长时间间断。因此，难以为各流程类别设计标准的业务流程模型，再统一部署。本文提出保持原来的业务流程模型不变，在业务抽象级别进行统一。在异构业务流程模型中识别出频繁使用的相似流程结构片段（即业务模式），每个业务模式形成一个业务阶段，在每个阶段的边界进行监控，从而达到在业务阶段级别进行监控。模式驱动的异构业务流程整合主要面临三大挑战：（1）如何发现业务模式。由于已有业务流程设计的复杂性和模糊性，使用的建模符号可能各不相同，分解的流程片段应与业务场景对应，同时流程片段应有适中的大小和较高的复用率，即应该存在于大量异构业务流程模型当中，这给自动化地分解有意义的流程片段带来困难。（2）如何为业务模式定义可配置的描述。由于业务流程的异构性，同一业务行为的细节在各分公司可能是不一样的，如何统一描述业务过程成为一大挑战。（3）集中化的异构业务流程监控。如何执行同样的业务模型来监控各异构的业务流程，这对于检查业务进展和制定正确的业务决策非常重要。

2. Artifact-centric 业务流程模型可配置建模问题

现今的企业处于一个动态的世界当中，常面临着新的机遇和挑战。只有能快速响应环境变化的组织才具有强大的竞争优势。经济学人上2009年曾有一则报道：调查显示在世界上接受采访的CEO中，有90%的人认为在当今快速变化的业务环

境中组织的敏捷（agility）是核心的砝码[9]。为解决这一问题，可配置流程模型为在参考流程模型中建模可变性提供了一个方法[10]。Gottschalk等人[11]从理论的角度给出了配置分析，为可配置流程模型提供了坚实的基础。一个可配置流程模型其实是一个通过变点的形式集中了同一个业务流程的多流程模型变体的一般模型，其中的变点称之为可配置的元素，可有多个设计选项。根据具体的业务需求为每个可配置元素选择一个配置选项，则可配置一个可配置的流程模型，得到一个具体的流程模型变体，而不需要额外的设计工作。

为说明可配置业务流程建模的迫切需求，文献[12]给出了几个领域的例子，比如医疗、银行、政府等领域，在这些领域中常存在同一个流程模型的许多变体。同样，为引出本文研究的动机，我们给出一个实际的应用例子。在杭州房管局的业务当中有很多关于房屋管理的业务流程，其信息系统是典型的流程类型的业务信息系统。该公司目前正向全国推广其业务，希望将房屋管理的信息系统部署到不同的城市。信息系统中一个典型的例子就是保障房申请审批流程（Public Housing Application，PHA），该流程的业务目标是处理特殊人群（比如：低收入者、自然灾害受害者等）对公共保障房的申请审批。由于不同城市的不同组织机构、执行能力及法律法规导致各自不同的需求，于是需要不同的流程模型变体，增加了业务流程建模的成本和时间开销，阻碍了该企业信息系统在不同城市的推广。因此，如何快速地复用已有的业务流程模型变体为新的业务场景建立新的业务流程模型是该企业面临的一个问题，这也是本文的研究动机之一。

受“设计重用”思想的启发，可配置流程模型作为一个可定制的模型可表达相似流程模型的“共同”和“可变”部分，其获得了业界极大的关注[13-15]。特别地，Artifact-centric业务流程建模建立在业务Artifact之上，结合了数据流和控制流两方面，为业务管理者提供了一个关于业务实体和操作的完整视图。目前的业务流程模型可配置建模方法主要集中在传统以活动为中心的业务流程模型，这些方法将数据和业务规则掩盖于流程之下，不能直接适用于新型的Artifact-centric业务流程可配置建模。因此，本文研究Artifact-centric业务流程模型的可配置建模问题。Artifact-centric业务流程模型的可配置建模主要面临三大挑战：（1）如何合并同一业务流程的不同Artifact-centric业务流程模型变体。Artifact-centric业务流程模型由多方面组成，涉及控制流和数据流两方面，而元素之间存在一定的依赖关系和层次关系，所以有些元素可能是不需要合并的。（2）如何选择配置点及设置配置选项。由于Artifact-centric业务流程模型的组成元素并不是都在图形化的表示中可以体现出来，所以需要考虑不同表示形式的配置点及配置过程。同时，需要一个在合并的模型中自动地识别可配置元素的算法。（3）配置算法。如何设计有效的配置算法辅助业务流程配置的建模过程，使得配置后

的模型后续处理可以自动完成，并保证配置的结果流程模型的结构化正确性[16]是配置算法需要应对的挑战。

3. Artifact-centric 业务流程模型同步的行为正确性验证问题

业务流程模型的正确性在业务流程管理中起到非常重要的作用，因为不正确的模型会导致流程的错误决策以及不符合要求的信息系统实现[17]。在一个 BPM 实践报告中显示在建立的流程模型库中有 20%是错误的[18]，所以需要在设计阶段对建立的业务流程模型的正确性进行验证，因为正确的业务流程模型是流程类型的信息系统可正常运行的前提。传统以活动为中心的业务流程建模范式建立一个集中式的流程，或者流程的编排，其中控制流在模型中显示地描述。Artifact-centric 业务流程建模是声明式的建模类型，控制流隐含在业务规则当中，也因此更加具备灵活性的特征。在 Artifact-centric 业务流程模型中可能涉及多个 Artifact，且它们的行为之间存在复杂的同步。因此，如何验证复杂同步情况下 Artifact-centric 业务流程模型的行为正确性是亟待解决一个研究问题，这是本文的研究动机之一。

Artifact-centric 业务流程建模范式将流程建模为 Artifact 生命周期的交互，它们之间的同步可以是复杂的，相比传统以活动为中心的业务流程模型可以表达各种各样的同步。Artifact-centric 业务流程模型结合了数据流和控制流两方面，因此要研究其行为正确性需要从这两方面着手。复杂同步情况下 Artifact-centric 业务流程模型的行为正确性验证主要面临如下三大挑战：（1）如何定义和描述带复杂同步约束的 Artifact-centric 业务流程模型以及其行为正确性。目前没有关于同步的 Artifact-centric 业务流程模型的形式化和显式的图形表示的研究。（2）如何识别出流程模型中描述的满足约束的行为。由于 Artifact-centric 业务流程模型控制流并没有显示的建模，因此流程执行的行为是难以发现的。（3）如何检测 Artifact-centric 业务流程模型所描述的行为在控制流和数据流两方面的行为正确性。具体地，控制流是否满足其行为正确性属性以及数据流在流程执行过程如何变化且是否满足其行为正确性属性。

1.2 相关工作

本文的主要研究内容如下：对异构的传统 Activity-centric 业务流程进行整合建模与监控、对新的业务需求场景复用已有的 Artifact-centric 业务流程模型进行可配置建模以及复杂同步情况下 Artifact-centric 业务流程模型的行为正确性验证，这些都属于业务流程建模的范畴。接下来，针对本文研究内容的相关工作分别进行综述。

1.2.1 异构业务流程的整合建模

为实现业务目标，企业常常首先需要进行业务流程建模，再部署到业务流程管理系统中执行业务过程，最后得到流程的产出物，完成业务目标。业务流程模型表达了业务的整体过程及目标，为业务中涉及的利益相关者之间的沟通提供了基础。目前，主要有两种主流的业务流程建模范式：以活动为中心和以数据为中心[19]。模式作为一个可以复用的结构，代表了以往经验的最佳实践。基于模式建立模型，可以减少建模的时间，同时提高模型的质量[6]。对于目前的两种常用的业务流程建模范式均有一些模式被提出来以供业务流程建模时复用。

对各种传统的以活动为中心的业务流程建模语言均带有基本控制流结构的模式[20,21]，而对于工作流建模语言还有其它的一些模式，比如资源模式[22]、数据模式[23]、异常模式[24]，当然这些模式也可以转化成其它的业务流程建模语言的描述。虽然在几乎所有的业务流程建模工具中均带有基本的控制流模式，但是这些工具不能支持用户正确地应用基本的控制流模式来建立业务流程。因此，Thomas 等人[25]提出了对业务流程建模工具的扩展，基于工作流模式的复用来辅助建立业务流程模型。Barbara 等人[26,27]提出了系列的控制流变化模式和变化支持特征，以提高流程的灵活性，适应变化的执行环境。La Rosa 等人[28]则考虑多个业务流程的共同子结构，通过合并的方式对业务流程进行整合。然而，该整合的流程并不是用于执行的，而是为了分析流程模型变体中的“共同”和“差异”部分。给定一个可配置的流程模型，分析者们可以得到单个具体的业务流程[13]。然而，这种方法当碰到大规模的异构业务流程时，得到的可配置流程可能会过于复杂，给流程的配置带来困难。Gao 等人[7]提出采用流程模型的片段化[29,30]、聚类[31,32]、合并[28]、检索[33,34]等技术手段来整合异构的业务流程模型，该项工作的主要目标是对异构业务流程模型的管理。这些对业务流程模型的整合与管理方法虽然有助于业务流程的建模，然而由于缺乏统一的执行过程难以对异构业务流程进行统一监控。

近十多年来，一种新的业务流程建模范式被提出来，即以数据为中心的方法，以 Artifact 为中心的建模是其中的一个典型代表。以 Artifact 为中心的建模方法将数据对象以及它们的生命周期视为一等公民[35-37]。一个 Artifact 的生命周期为一个具体的 Artifact 从创建到存档的处理过程。Rong Liu 等人[38]总结出了 9 中运作模式，可支持业务流程模型的建立。Liu 等人[39]研究了以 Artifact 为中心的业务流程模型的实现。后来，IBM 提出了一种名为 BizArtifact 的开源系统，以实现以 Artifact 为中心的业务流程建模和执行。虽然 Maroun 等人[40]提出对任务、仓库和流连接描述的异构 Artifact 生命周期进行集成，以便提供统一的视图

进行管理。然而，其中 Artifact 生命周期的合并中未考虑 Artifact 数据模型和业务规则的合并，且只涉及单个 Artifact 生命周期的整合而不是整个流程模型。

由于一个业务行为在不同的场景中可能以不同的方式被设计和执行以达到共同的业务目标，已有的两种建模方法均不能实现异构业务流程的整合。已有的业务流程整合主要集中在可配置模型的建立[28,41,42]，这种方法在面对大规模异构业务流程时，得到的可配置模型可能过于复杂而不能在实际中应用。最重要的是若已经部署的业务信息系统本身是独立建立和演化维护的，则要对同一业务的异构信息系统中执行的业务流程模型进行标准化是不现实的。因此，大规模异构业务流程模型的整合还有待研究，这正是本文要解决的问题之一。

1.2.2 Artifact-centric 业务流程可配置建模

参考模型通过提供通用的解决方案来简化特定模型的设计。一般商用的参考流程模型缺乏变点和配置决策的描述。该缺点可通过可配置的业务流程模型的概念来解决[13]。该概念对于（参考）流程模型的语义重用是一大进步。由于 EPC 在参考流程建模中被广泛应用[43]，因此一个扩展的参考流程建模语言，即可配置的 EPC，又称之为 C-EPC，被 Rosemann 等人[13]提出来。C-EPC 符号为可配置的功能和连接器添加配置选项。可配置的功能可以从模型中保留（included）或去除（excluded）；可配置连接器可以改变其类型，比如：从 OR 变到 AND，或者限制其输入/输出的分支。在图形上，可配置元素的图形边框用粗线表示[11]。然而，该可配置流程建模语言只集中在控制流方面，而没有考虑资源、数据和流程中参与的物理单据。为了解决该问题，La Rosa 等人[43]扩展了原始的 EPC 符号，对其中的功能（即活动）关联了角色和对象，因此一个集中的 EPC，即 iEPC 被提了出来。同时，提出了相应的可配置的 iEPC 建模语言，即 C-iEPC。进一步地，对 C-EPC 和 C-iEPC 分别提出了配置算法[13,43]，但没有给出概念证明的实现。因此，Jan Mending 等人[44]实现了一个自动化的配置算法通过最小化的创建从 C-EPC 产生语法正确的 EPC 模型，并且考虑了模型转化的语法与语义问题。

获得可配置的流程模型可通过流程模型变体的合并来完成，使得合并的流程模型可表达多个流程模型变体的行为。对导出的可配置流程模型一般有两个要求：（1）可配置的流程模型产生的每个实例是语法正确的流程模型；（2）可配置的流程模型是可逆的，即用于合并的原始流程模型应该是可从结果的可配置流程模型中产生的实例。流程模型的合并技术在已有的文献中已有一些研究。Gottschalk[45]详述了将流程模型合并为单个可配置的流程模型的方法。然而，为使对可配置转换产生语法正确的模型，还需要做事后处理。Mendling 等人[46]提出对流程模型的不同视图进行合并的方法，然而该方法不产生可配置的流程模

型。Shuang Sun 等人[47]针对块结构的流程模型进行合并。然而，这些方法从可配置的流程模型中均难以产生语法正确的流程模型。因此，Dennis 等人[48]通过 CoSeNet 描述流程的控制流，即一个可配置的树形流程模型表示，其流程模型的构建结果是语法正确的，且两个 CoSeNets 合并的可配置 CoSeNet 是可逆的。

另一种获得可配置的流程模型的方法是通过挖掘流程的事件日志。Chen Li 等人[49]提出通过从流程日志中挖掘的流程模型变体来创建新的参考流程模型的方法，然而可配置点的设置需要领域知识的指导来添加。Gottschalk 等人[50]提出两个大概的方案来获得可配置的流程模型：（1）对不同流程变体的日志分别挖掘出模型变体，再对模型变体进行合并得出可配置的模型；（2）对各流程模型变体的日志合集挖掘出可配置的模型。然而，这两种方案均需要事后再设置可配置点，也没有提出具体适合的挖掘算法。文献[51]通过扩展 ETM（Evolutionary Tree Miner）算法从事件日志的合集中挖掘可配置的流程模型，该模型描述了流程变体的家族而不是单个具体的流程模型。该文章在文献[50]的基础上，另外提出了两种方案：（1）对各流程模型变体的日志合集挖掘出共通的流程模型，然后在此基础上导出具体的流程模型，最后将它们合并起来。同样，事后再设置可配置点；（2）对各流程模型变体的日志合集挖掘出流程模型并设置可配置点。

然而，无论如何从流程模型变体家族中获得集中可配置的流程模型，得到的结果一般会比较复杂，含有大量的可配置元素，尤其是在模型变体数目众多的情况下[6]。一方面，需要自动化的方法辅助配置过程产生语法结构正确的流程；另一方面，需要提供领域知识来辅助配置决策过程。为解决第一方面的问题，Wil M.P. van der Aalst 等人[52]对 Workflow Net 提出了自动化的配置方法，使得产生的业务流程模型是结构和语义上正确的。尽管 Jan Mendling 等人[44]对 EPC 语言提出了自动化的配置算法，使得从可配置的 C-EPC 模型产生结构化正确的 EPC 模型。然而，这些方法缺乏在领域约束需求方面的考虑。为解决第二方面的问题，提出了一些工作来辅助获得业务需求和指导配置过程，比如：基于问卷的方法[53,54]、基于本体的方法[55]和基于模板与规则的方法[56]。还有使用非功能需求的方法来评估已配置流程模型的性能[57]。这些方法促进了对流程可变性的设计和配置，然而，一方面这些系统在很大程度上需要领域专家手动创建，这是耗时费力的工作；另一方面缺少对配置点之间关系的理解。为了从过去的流程配置经验中获得知识，Nour Assy 等人[58]提出自动地提取配置规则模型的方法，以便从业务流程模型仓库中获得具体的领域约束。之后，Nour Assy 等人[59]又提出使用关联规则的方法挖掘频繁关联的配置决策，并将这些配置作为规则来指导未来的配置过程，也可根据配置在已有模型变体中被采用的频度来优化可配置的流程模型，减少配置点的数量，提高建模效率。

目前，关于流程配置的研究工作基本上都是集中在传统以活动为中心的业务流程模型中。近十多年多，提出了一种 Artifact-centric 业务流程建模范式来建模业务流程，其中业务流程被建模为 Artifact 生命周期的交互[60]。业务 Artifact 的概念来源于文献[35]，其中业务运作模型构建为业务 Artifact 生命周期的集合及其交互。相比传统以活动为中心的业务流程建模，基于业务 Artifact 的运作模型具备一些优势，比如：描述的灵活性、分析变化的能力、应用管理的能力。进一步地，业务 Artifact 的概念在文献[37]中被采纳，并将业务流程模型构建为 4 个核心构件：Artifact、Artifact 生命周期、服务和关联（association）。为了实现这样一个 Artifact-centric 业务流程模型，该文提出一个三层的框架：业务运作模型（business operations model）、概念流（conceptual flow）和工作流（workflow）。Gerede 等人[61]提出针对中间层研究 Artifact-centric 业务流程模型中 Artifact 行为的描述和验证。Artifact-centric 业务流程模型的形式化分析进一步地在一些文献中也有研究[62,63]，也有一些对 Artifact-centric 流程建模范式进行扩展的工作，比如：[64,65]。因此，Artifact-centric 业务流程建模方法越来越受到关注，其带来的益处也记录在一些案例分析研究当中[66,67]。

然而，以上提到的业务流程可配置建模及模型合并的方法均集中在传统以活动为中心的业务流程模型，对新型的 Artifact-centric 业务流程模型的可配置建模问题还未见研究。声明式的建模方式涉及多方面的流程元素，使得 Artifact-centric 业务流程模型的可配置建模成为一大挑战。因此，解决 Artifact-centric 业务流程模型的可配置建模是本文的研究问题之一。

1.2.3 Artifact-centric 业务流程模型同步的行为正确性验证

在业务流程管理中，业务流程模型常描述了业务相关的不同方面，比如：控制流、数据对象流、角色指派等。控制流定义了任务可能的执行顺序，而数据流提供了任务之间交换信息的方式。通过从数据对象的读写，信息在任务间传递。在业务流程模型的验证研究领域，主要集中在控制流属性的正常完成。正确性的属性及其派生对流程模型的验证起到非常重要的作用[68]。综述的介绍可见文献[69]。这些属性源于对 Petri Net 描述的流程定义。基于 Petri Net 分析技术和相关工具实现，可对各种不同的流程建模语言（比如：workflow Net[68]、EPC[44]、YAWL[70]、BPMN[71]）建立的业务流程模型的正确性概念进行验证。在文献[72]中，在 UML 活动图中描述了控制流和数据流，并使用 CPN（Colored Petri Nets）对其数据流进行验证，然而数据被抽象为一个整体的业务对象（business object），而没有考虑数据模型的细节。文献[17]提出了一个综合的方法来验证集中的 EPC（integrated EPCs）语言描述的模型，其中描述了角色和对象。正确性的验证除

了考虑控制流方面，还考虑了对象的存在性和角色的可用性，并给出了具体的正确性属性定义以及验证方法。然而，这些验证工作主要集中在传统的以活动为中心的业务流程模型中，或者流程中的数据只是看成一个粗粒度的整体对象来考虑。

除了对传统 Activity-centric 业务流程模型在控制流和数据对象方面的验证，还有部分工作集中在对象生命周期的验证，和本文的研究工作也密切相关。对象生命周期用于识别当前数据对象的状态，以及从当前状态可到达的数据状态。业务流程操作的对象状态必须定义在参考对象模型中以确保其合规性，因为参考对象生命周期常表达了政策的信息，比如：对象处理的法律法规。合规性展示政策需求是满足的。Jochen 等人[73]提出将流程中对象生命周期进行组合，基于组合的对象生命周期产生业务流程模型，使得流程模型符合合规性。该文提出了对象生命周期合规性（Object Life Cycle Compliance）的定义。业务流程中使用的对象通常是标准化的参考对象模型。业务流程模型和对象生命周期是同一系统行为下提供的不同视图，因此它们之间要求保持数据对象处理行为的一致性（Consistency），即对象生命周期的描述也要在业务流程模型中有体现。为此，Ksenia 等人[74]提出了对象生命周期覆盖（Object Life Cycle Coverage）的定义。因此，业务流程模型和对象生命周期之间数据对象行为的一致性可以定义为对象生命周期合规性和覆盖同时成立。Andreas 等人[75]提出流程模型在数据对象方面的弱一致性（Weak Conformance）概念。该文定义了流程场景（Process Scenario）的概念，由流程模型和对象生命周期组成。若在流程模型描述中数据对象可以从 A 状态到达另 B 状态，则在对象生命周期中也可从 A 状态到达另 B 状态，而不验证其中间的过程，则称数据对象满足弱一致性。进一步地，Andreas 等人[76,77]提出带同步约束的生命周期与流程模型的弱一致概念，其中同步约束的验证简化抽象为一个真假函数判断，没有给出具体的验证方法。总之，这些方法虽然考虑了数据对象的生命周期，但本质上还是属于 Activity-centric 业务流程模型，没有考虑细粒度的数据模型。

对于 Artifact-centric 业务流程模型的验证，大部分工作集中在 Artifact 的属性。在 Artifact-centric 业务流程建模范式中，Artifact 生命周期通常用有限状态机描述。Rong Liu 等人对运作模型提出了 9 个运作模式，并基于 Petri Net 开发了一个计算模型对 Artifact 的 *Persistence*、*No split* 和 *Reachability* 属性做了形式化分析。当输入库所中包含 token（即 Artifact）时，则变迁变成使能状态，然而未考虑与服务输入相关的 Artifact 中具体数据属性的存在性。Gerede 等人[63]提出了针对 Artifact 属性的静态分析技术。同时，Bhattacharya 等人[62]对 Artifact-centric 业务流程提出了一个形式化模型，并研究了关于三个问题（即 *ability to complete an execution*，*existence of an execution "dead end"*和 *redundancy*）

的可判定性和复杂度结果。进一步地，Gerede 等人[61]基于计算树逻辑提出了一个逻辑语言来描述 Artifact-centric 业务流程模型中 Artifact 的行为，并提出几种不同问题下该语言的可判定性结果。文献[78,79]解决了 Artifact 系统的验证问题，静态地检查 Artifact 系统所有执行情况是否满足线性时态逻辑扩展中描述的属性。和之前关于业务 Artifact 大部分工作不同，GSM 以声明式的方式将流程建模为一个阶段的层次化结构[80,81]。目前，已有一些基于 GSM 的业务 Artifact 验证工作，其中文献[82]提出了一个方法通过将 GSM 描述的 Artifact 系统转化为标签转换系统以进行对声明式模型的模型检查。然而，对当前的实现不能应用于产生 GSM 模型。文献[83]提出了一个基于代理语义的 GSM，可验证参与者知识相关的信息理论属性。文献[84]为 GSM 建模范式的形式化验证提供进一步的基础。

除了形式化的验证，还有一些工作集中在合规性和一致性的检查。Niels[85]提出与其验证 Artifact-centric 流程模型的合规性，不如从合规的模型进行综合直接设计产生合规性的 Artifact-centric 业务流程模型。具体做法是：用 Petri Net 表示合并 Artifact 生命周期，同样政策和合规性规则也用 Petri Net 表示，然后合并和组合 Artifact 生命周期，最后对这些 Petri Net 进行合成，产生合规性的 Artifact-centric 业务流程模型。一致性是检查 Artifact-centric 流程模型是否描述了流程的实际执行，即模型描述与执行日志的一致性。Dirk 等人[86]基于 Proclet 系统的日志，检查流程的实际执行行为与 Artifact-centric 流程模型描述之间的一致性，其中多个 Artifact 交互行为的一致性问题分割为简单的单个 Artifact 的行为一致性问题，因此可用已有的技术进行验证。进一步地，Dirk 等人[87]提出交互流程中存在实例重叠的情况下 Artifact-centric 业务流程模型的行为一致性问题。同样的思路，将 Artifact-centric 业务流程模型的行为检查分解为 Artifact 的行为一致性问题，最后采用已有的一致性检查技术解决。

总之，已有的业务流程模型验证研究工作要么集中在传统以活动为中心的业务流程模型，要么集中在单个 Artifact 情况下的 Artifact-centric 业务流程模型的形式化分析或其它属性的验证。然而，已有的验证方法忽略了同步情况下的 Artifact-centric 业务流程模型的行为正确性问题，不能解决多个 Artifact 的同步验证问题，这正是本文要解决的问题之一。

1.3 本文主要内容及章节安排

1.3.1 本文主要工作

本文主要的研究内容如下：

- ✧ 针对流程描述、流程结构、系统实现各样异构特性的业务流程如何进行整合以便达到统一监控的目的是很多大型企业面临的一大挑战。本文提出了一种模式驱动的方法对异构的业务流程进行阶段性的抽象和监控而忽略其内部细节，解决异构流程的统一性问题；

- ✧ Artifact-centric 业务流程建模作为一种新型的业务流程建模范式，其建模方法给以往的建模人员带来了困难。因此，本文为了辅助建模人员更好地进行流程模型设计提出 Artifact-centric 业务流程（形式化）模型可配置建模方法，基于变点的发现为用户提供可选的操作项，为建模人员进行流程建模提供指南，简化建模过程，提高建模效率；

- ✧ Artifact-centric 业务流程模型的正确性是信息系统实现与正常运行的前提保证。因此本文针对 Artifact-centric 业务流程模型在控制流和数据两方面的行为正确性问题提出验证方法，尤其是对流程中的多个 Artifact 之间存在各种复杂同步交互的情况下给出验证方法，为流程模型的部署和执行提供正确性基础。

1.3.2 本文研究意义

本文研究内容的研究意义如下：

- ✧ 异构业务流程的整合与监控问题的研究有助于企业的高层领导对企业的整体业务进展有一个全面的了解，促进各分公司之间交流和协作，有利于企业各分部的统一管理和决策，增强企业在市场中的竞争力。

- ✧ Artifact-centric 业务流程可配置建模问题的研究有助于提高业务流程模型的复用，降低业务流程建模的难度，节省业务流程建模的成本，并提高业务流程建模的效率，更加快速地响应变化的市场需求。

- ✧ Artifact-centric 业务流程模型的正确性问题的研究有助于企业业务目标的正常实现，为业务流程模型的修正提供思路，为企业信息系统的实施提供正确的流程模型，降低后续系统修改带来的维护成本。

1.3.3 本文章节安排

本文章节的组织安排如图 1.1 所示，简要内容介绍如下：

- **第 1 章 绪论**

介绍异构业务流程环境下的整合与监控问题，以及新型 Artifact-centric 业务流程建模范式下流程可配置建模与流程模型行为正确性验证问题的相关背景、研究动机及所面临的挑战。然后，介绍本文研究内容的相关研究。最后，介绍本文

的主要研究工作内容、研究意义，和内容组织安排。

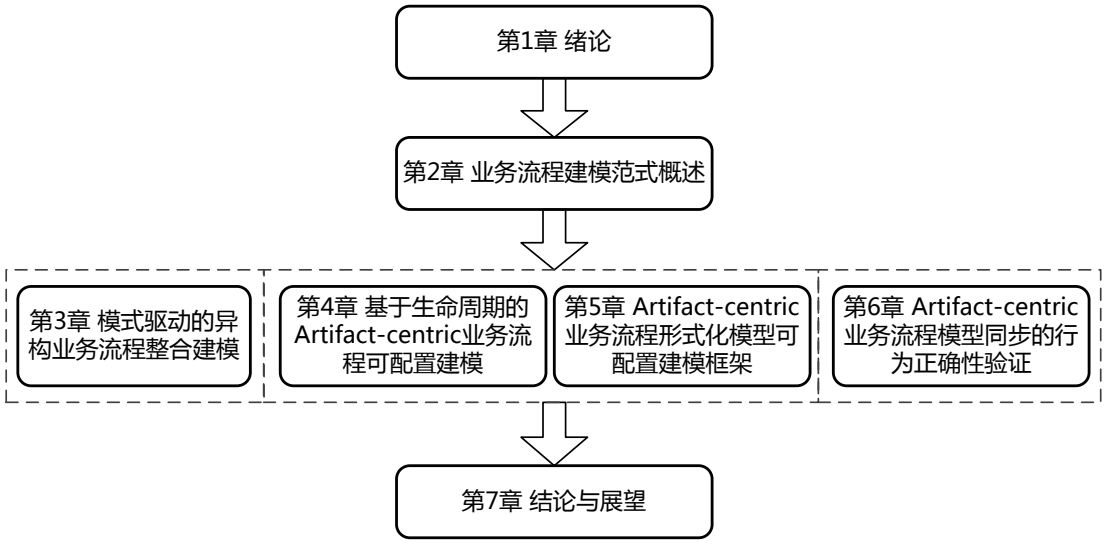


图 1.1 论文各章节组织安排

- **第 2 章 业务流程建模范式概述**

针对目前两种主要流行的业务流程建模范式展开介绍。首先，介绍传统的 Activity-centric 业务流程建模范式，主要强调活动的顺序（即控制流），需列举出所有的活动以及控制流。目前，已有很多相应的流程建模语言。在现今市场上占据着主流的地位。然后，介绍最近十来年由 IBM 提出的 Artifact-centric 业务流程建模范式，将数据流与控制流在模型中置于同等重要的地位，强调业务对象上数据的操作变化和状态的变迁。

- **第 3 章 模式驱动的异构业务流程整合建模**

针对企业内部业务流程模型的异构、冗余和不一致等问题，提出一种模式驱动的异构业务流程整合建模方法，将业务流程模型化为抽象业务模式的交互。每个业务模式是可配置的，可以根据具体的应用场景进行配置为具体的业务模式。为执行业务模型，将各业务模式转化为工作流模式或运作模式的描述，使得业务模型可以运行于目前大部分的工作流程引擎之上。在业务模式的边界设置复杂事件处理监控点，这些监控点恰好与业务阶段对齐，因此通过对监控点的检测可以监控业务的阶段性进展。

- **第 4 章 基于生命周期的 Artifact-centric 业务流程可配置建模**

针对新型的 Artifact-centric 业务流程建模范式，对具体业务的流程建模是一个困难的问题，一方面是业务本身的复杂性，另一方面是流程建模表示的困难。

本章提出可配置的 Artifact-centric 业务流程建模，为建模人员的流程建模提供基础。该可配置的业务流程模型是一个业务流程模型的多个变体的集中体现，它可以用来辅助设计新的业务流程模型，以适应新的业务场景。

- **第 5 章 Artifact-centric 业务流程形式化模型可配置建模框架**

针对新型的Artifact-centric业务流程建模范式，其形式化模型的可配置建模是一大难点。一方面形式化模型描述了流程元素组成的细节；另一方面，模型元素之间互相约束。本章提出一种Artifact-centric业务流程形式化模型的可配置建模框架，考虑流程模型中所有的组成元素。为了辅助业务流程模型的配置过程，基于流程模型元素关系图的概念提出配置指南。该可配置建模框架具有通用性，可促进Artifact-centric业务流程模型的复用，并使得Artifact-centric业务流程形式化建模更加快速和有效。

- **第 6 章 Artifact-centric 业务流程模型同步的行为正确性验证**

针对多个 Artifact 之间存在各种复杂同步交互的 Artifact-centric 业务流程模型，其流程的执行过程是复杂的，因此对其进行行为的正确性验证就变成了一大挑战。本章提出一种将同步的 Artifact-centric 业务流程模型向 Workflow Net 进行转化的方法，并基于转化后的 Workflow Net 对应的可达图对原同步交互的 Artifact-centric 业务流程模型进行其行为正确性验证。

- **第 7 章 结论与展望**

对本文的研究内容进行总结，并提出未来的研究工作。

1.4 本章小结

本章针对传统以活动为中心的业务流程建模范式下异构业务流程环境中的流程整合与流程监控问题，以及新型 Artifact-centric 业务流程建模范式下流程可配置建模与同步流程模型行为正确性验证问题的相关背景、研究动机及所面临的研究挑战进行了全面阐述。然后，从三方面介绍本文研究问题的相关工作。最后，介绍本文的主要研究内容、研究意义及内容组织安排。

第2章 业务流程建模范式概述

业务流程是由在组织和技术环境中协调执行的活动组成的集合，这些活动代表了业务要执行的工作单元。业务流程模型是业务流程的抽象表示，为了达成某个业务目标。BPM 是针对企业业务流程运作管理的领域，包括了支持业务流程设计、管理、配置、实施和分析的概念、方法和技术[69]。将业务流程逻辑从企业的应用逻辑中分离，使得业务流程建模可以从系统开发中剥离并独立修改，可提高企业应用的管理能力[88,89]。通过管理业务流程，BPM 实现提高企业生产力和性能的目标。BPM 是组织成功的关键因素，因它涉及业务流程的持续设计。组织为用户设计流程以执行和协调任务，从而保持竞争优势。而 IT 系统的目标就是要支持业务流程，使得业务流程具有敏捷性和适应性。

业务流程管理的生命周期主要包括如下几个方面：设计与分析、配置、实施和评估[69]。业务流程建模是 BPM 生命周期中的第一步，可见它在 BPM 研究领域中的重要性[90]。业务流程模型是为实现业务目标对业务流程所进行的抽象表示。业务流程建模是一个重要的工作，它描述了业务执行者需要完成的任务，以达到策略或操作目标。它主要关注业务方面，而非技术实现层面。同时，业务流程建模传达了业务的目标，使得企业管理人员及业务人员能交流和理解业务目标，有助于业务流程的管理、分析和 IT 实现[91]。基于本文的研究内容，我们将已有的业务流程建模方式从其强调的角度分为两类：传统以活动为中心的业务流程建模范式（即 Activity-centric 建模范式）[69]、以数据为中心的业务流程建模范式（即 Artifact-centric 建模范式）[35]。从语言学角度来看，以活动为中心的业务流程建模范式强调“做什么”，即谓语；以数据为中心的业务流程建模范式强调“在什么对象上进行了什么操作”，即宾语。本文后续第 3 章的内容是针对 Activity-centric 建模范式的业务流程，而第 4、5、6 章的内容是针对 Artifact-centric 建模范式的业务流程。本章将对两类业务流程建模范式分别进行介绍。

2.1 Activity-centric 建模范式

传统的工作流根据活动间的协作来实现业务目标的。传统的工作流作为一个典型的过程建模范式，描述了可能的活动集合，强调活动的执行顺序（即控制流）[92]，需列举出所有的活动以及控制流，被称之为以活动为中心的工作流建模范式。总之，非正式地，一个流程可形式化为一个二元组：$P = (SA, F)$，其中SA为一个有限的结构化的活动集合，F定义为SA上的控制流，即次序。

传统以活动为中心的业务流程建模以命令式的方式为主。命令式的建模将流程建模为任务（即活动）、网关、事件的集合，这些元素通过流（即变迁）相互连接。每个活动描述了一项工作单元，变迁描述了工作单元之间的次序。该方法严格描述了工作单元的执行次序。常见的流程建模符号有：BPMN（Business Process Model and Notation）[93]、BPEL（Business Process Execution Language）[94]、UML（Unified Modeling Language）活动图[95]、YAWL（Yet Another Workflow Language）[70]、WF-Net（WorkFlow Net）[96]、EPC（Event-driven Process Chains）[97]。目前，尽管有众多的流程建模语言，但 BPMN 已成为工业界的事实标准。

一个以活动为中心的业务流程模型可以看成一个有向图，其中结点有各自的标记。在这里，本章对不同的流程建模符号进行抽象，将流程模型表示为一个有向图，称之为**业务流程图**（Business Process Graph）[59]。该符号受文献[28]的启发，其中的元素来源于已有图形化流程建模符号的共同构件。因此，一个业务流程图可形式化定义如下：

定义 2.1：（Business Process Graph, BPG）业务流程图$P = (N, E, T, L)$是一个带标记的有向图，其中：

- N为结点的集合；
- $E \subseteq N \times$ N为连接结点的边集合；
- $T: N \to t$为一个函数，对每个结点$n \in N$赋予一个类型t，其中t依赖于每个标准符号的元素类型。如果是 EPC 符号，则$t \in \{event, function, connector\}$；
- $L: N \to label$为一个函数，对每个结点$n \in N$赋予一个标记。比如：对 EPC 符号，如果$T(n) = event \vee function$，则$L(n)$是一个名字（name），并且如果$T(n) = connector$，则$L(n) \in \{\vee, \wedge, \times\}$，其中$\vee = OR$，$\wedge = AND$，$\times = XOR$。

设$P = (N, E, T, L)$为一个业务流程图。我们定义连接器（即网关）$c \in N$的前序和后续集合为输入和输出边上的元素集合。

定义 2.2：（preset $\bullet c$, postset $c \bullet$）网关$c \in N$的前序集合表示为$\bullet c$，定义为：$\bullet c = \{n \in N: (n, c) \in E\}$。$c$的后序集合表示为$c \bullet$，定义为：$c \bullet = \{n \in N: (c, n) \in E\}$。

若$|\bullet c| > 1$，则网关c为 split 类型网关；若$|c \bullet| > 1$，则网关c为 join 类型网关。一个可配置的业务流程模型是带可配置元素的业务流程图，可配置元素可以是功能结点或网关结点。可配置的功能结点可从流程模型保留（即 ON）或去除（即

OFF）；若是 join/split 类型网关可配置的网关结点可通过改变其类型并保持其行为或限制其输入/输出分支来进行配置。

表 2.1 给出了网关类型配置的约束集合，其中可配置网关表示为$[label]^c$。表中每行对应可配置的网关，可配置为一个或多个列中网关，最后一列Seq对应简单的“顺序”。打钩的单元格表示可配置，空白的单元格表示不可配置。例如：可配置的"∨"可配置为任意的网关类型，而可配置的"∧"只能配置为"∧"。

表 2.1 可配置网关的配置约束

	∨	∧	×	Seq
$\vee^C$	√	√	√	√
$\wedge^C$		√		
$\times^C$			√	√

表 2.1 中的这些可配置约束可通过偏序关系$\preccurlyeq$进行形式化描述，其中具体的网关用于配置给定的可配置网关，如下定义所示。

定义 2.3：（partial order $\preccurlyeq$）设c^c为一个可配置的网关，c为一般的网关或一个“顺序”。$c \preccurlyeq c^c$，当且仅当$(L(c^c) = "\vee") \vee (L(c^c) = "\times" \wedge L(c) = "Seq") \vee (L(c^c) = L(c))$。

形式上，一个可配置元素的配置可定义如下：

定义 2.4：（Configuration Conf）一个可配置的结点n^c，其中$T(n^c) = "function" \vee "connector"$，它的配置为一个函数，定义为：

- 若$T(n^c) = "function"$，则$Conf(n^c) \in \{ON, OFF\}$；
- 若$T(n^c) = "connector"$，则当n^c为 join 类型网关时，$Conf(n^c) =< n, \bullet n >$；当$n^c$为 split 类型网关时，$Conf(n^c) =< n, n \bullet >$，其中：
  1. $n \preccurlyeq n^c$；
  2. 当n^c为 join 类型网关时，$\bullet n \subseteq \bullet n^c$；当$n^c$为 split 类型网关时，$n \bullet \subseteq n^c \bullet$。

除了可配置元素的配置选项，在可配置模型中还需要配置规则来约束配置过程，配置规则可定义如下：

定义 2.5：（Configuration rule）配置规则描述了在一个可配置流程模型中不同可配置元素的配置之间的关联，定义如下：

$$Conf_{h_1}, \cdots, Conf_{h_p} \rightarrow Conf_{b_1}, \cdots, Conf_{b_q}$$

其中，$Conf_{h_i}$：$1 \le i \le p$称之为规则头（head），$Conf_{b_j}$：$1 \le j \le q$称之为规则体（body）。规则头和规则体均表示可配置流程模型中不同可配置元素的配置。有了配置元素的约束和配置规则，可形式化定义可配置流程模型如下：

定义 2.6：（Configurable process model）可配置流程模型表示为$P^c = (N, E, T, L, B, Conf^c, CR^c)$，其中：

- N, E, T, L的描述如定义 2.1 一致；
- B：$N \to \{true, false\}$为一个布尔函数，对可配置结点返回真，否则返回假；
- $Conf^c$是满足定义 2.4 合法的配置集合；
- CR^c是配置规则的集合。

下面以事实标准的 BPMN 符号体系为例，给出一个业务流程模型的例子。图 2.1 展示了一个采用 BPMN 符号描述的业务流程模型。该流程为某城市的保障房申请审批流程，其中有 5 个角色参与了流程的执行，即：社区、单位、街道、区县和市局，分别在不同的泳道中表示。矩形框代表活动，其中“调查”和“协查”两个活动同时为两个子流程。

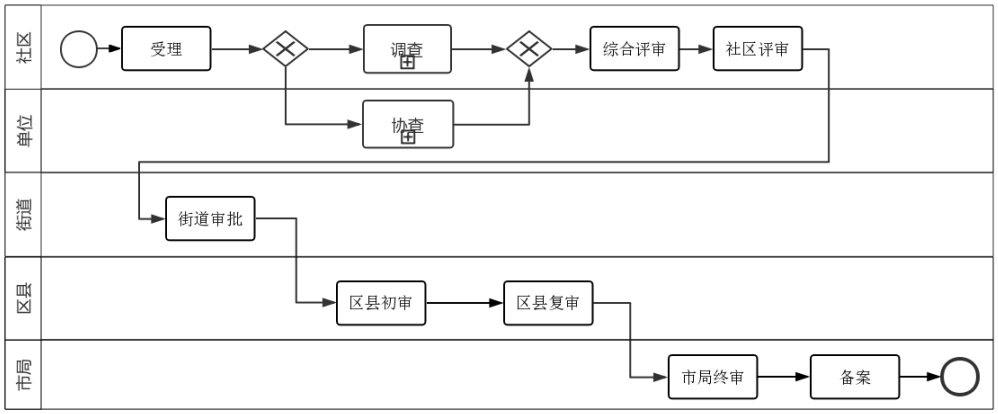


图 2.1 采用 BPMN 符号描述的业务流程模型

2.2 Artifact-centric 建模范式

传统以活动为中心的业务流程建模范式忽略了对业务信息的观察，没有业务信息的完整视图，数据掩盖于流程之下。业务执行者常严格依照流程模型的控制流，而不是依照业务规则决定“能做什么”，妨碍了业务运作的灵活性和创新[92]。当流程规模增大时，以活动为中心的业务流程模型的行为变得很难理解，不利于

深度的流程分析和动态条件下系统行为的监控、维护与预测。

对于任何一项业务，不管它的产出是服务还物理产品，业务的执行过程都需要业务记录，记录流程执行的状态和生产对象的细节信息，这些信息可用于业务目标的评价，而业务 Artifact 恰好是记录业务信息的对象[92]。Artifact 的概念由 IBM 于 2003 年提出，将其定义为具体、可标识、自描述的信息块，用于描述业务生产的细节[35,92]。根据业务活动对 Artifact 的处理，可描述业务的操作，因此可利用 Artifact 的生命周期对业务流程进行描述，这种建模范式称之为以 Artifact 中心的业务流程建模。以 Artifact 为中心的业务流程建模将流程建模为业务中关键 Artifact 生命周期的交互[60]，降低了流程的复杂度，使业务人员能够更好地理解业务操作，近年来在学术界和工业界均引起了诸多的关注[66,67,98]。

正如传统以活动为中心的业务流程建模范式有很多种流程建模符号语言一样，Artifact-centric 业务流程建模范式也有一些不同的建模方式来描述业务操作且有各自的特点。文献[99]综述了目前的几种 Artifact-centric 业务流程建模方法，包括 GSM[80]，ArtiNets[100]，AXML[101]，BPMN Extensions[102]和 ACP[64]。各种 Artifact-centric 业务流程建模方法的简要比较描述如表 2.2 所示，详细细节参见文献[99]。

表 2.2 各种 Artifact-centric 业务流程建模方法的比较

方法	信息模型	生命周期	服务	关联
GSM	编程数据类型	声明式	声明（事件）	声明式（ECA 规则）
ArtiNet	过程式	过程式	过程式（任务）	声明式（ECA 规则）
AXML	XML 元素	声明式	声明（功能调用）	声明式（ECA 规则）
BPMN	过程式	过程式	过程式（任务）	过程式（策略）
ACP	键值对符号	声明式	声明（动作）	声明式（条件动作规则）

本节借鉴文献[64]中提出的 ACP 模型来定义 Artifact-centric 业务流程形式化模型。一个 Artifact-centric 业务流程模型包含三类组成构件：Artifact、服务和业务规则。因此，Artifact-centric 业务流程模型可形式化地定义为如下：

定义 2.7：（Artifact-centric Process Model，APM）一个 Artifact-centric 业务流程模型是一个三元组$\Pi = (Z, V, R)$，其中Z是 Artifact 模式，V和R分别是Z之上的服务集合和业务规则集合。

一个 Artifact 在流程中是一个业务实体，每个 Artifact 包含数据属性和状态的集合。因此，定义 Artifact 类来描述同类 Artifact 实例的数据结构，如定义 2.8 所示。同时，一个 Artifact-centric 业务流程可能涉及多个 Artifact，Artifact 可以包含一个特殊的数据属性，用于存储另一个 Artifact 的标识，即关键数据属性的

值。这样，一个 Artifact 实例可以引用另一个 Artifact 实例的数据信息。从层次结构看，被引用的 Artifact 实例可视为引用 Artifact 实例的子 Artifact。因此，定义 Artifact 模式如定义 2.9 所示。

定义 2.8：（Artifact Class）一个 Artifact 类是一组 Artifact 的抽象，包含数据属性和状态。一个 Artifact 类C是一个二元组，即$C = (A, S)$，其中：

- ✧ $A = \{\alpha_1, \alpha_2, \cdots, \alpha_x\}$，$\alpha_i \in A(1 \le i \le x)$是标量类型的数据属性，比如：字符串、实数等，或者为一个未定义的值；
- ✧ $S = \{s_1, s_2, \cdots, s_y\} \cup \{s_{init}\} \cup S_{final}$，$s_i \in S(1 \le i \le y)$为一个状态，其中$s_{init}$为初始状态，$S_{final}$为终止状态的集合。

定义 2.9：（Artifact Schema）Artifact 模式Z包含了流程中所有 Artifact 类的集合，即$Z = \{C_1, C_2, \cdots, C_n\}$，其中$C_i \in Z(1 \le i \le n)$为一个 Artifact 类。

一个 Artifact-centric 业务流程模型可能涉及多个 Artifact 类，在流程执行过程中将产生多个不同类型的 Artifact 实例[103]，且 Artifact 生命周期之间可能存在同步。首先，第一个创建的 Artifact 称之为主 Artifact，该 Artifact 由外部事件触发而创建。之后，在主 Artifact 的执行过程中，其它辅助的 Artifact 可以被触发创建。同时，一个辅助 Artifact 在执行过程中也可以触发创建其它的辅助 Artifact。如图 2.2 所示，可以把 Artifact-centric 业务流程执行过程中 Artifact 实例的产生过程视为一棵树的生长，其中主干表示主 Artifact，枝干表示辅助 Artifact，枝干（与主干）之间的交点表示 Artifact 实例之间的交互。圆圈黑点表示 Artifact 实例的创建，对应 Artifact 的初始状态；方块黑点表示 Artifact 实例的执行完成，对应 Artifact 的结束状态。由图 1.1 可知，A_0为主 Artifact 实例，A_1、A_2、A_3、A_4为辅助 Artifact 实例。A_1、A_2、A_3在A_0的执行过程中被创建，A_4在A_2的执行过程中被创建。A_1与A_3在执行过程中有交互，A_0与A_4在执行过程中有交互。

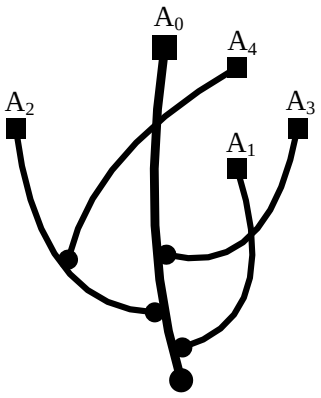


图 2.2 Artifact-centric 业务流程执行中的 Artifact 实例生成树

Artifact-centric 业务流程模型中的服务是用于对 Artifact 进行读写操作的任务。定义$V=\{v_1,v_2,\cdots,v_z\}$为内部或外部服务的集合，服务v可以读或更新一个或多个 Artifact 的数据属性，记为$v.rw=\{C_1,C_2,\cdots,C_k\}$，其中$C_1,C_2,\cdots,C_k$为被$v$读或更新的 Artifact。更具体地，定义$v$读写的具体数据属性集合，即$v.r=A^r_{C_1}\cup A^r_{C_2}\cup\cdots\cup A^r_{C_k}$，$v.w=A^w_{C_1}\cup A^w_{C_2}\cup\cdots\cup A^w_{C_k}$，其中$A^r_{C_i}\subseteq C_i.A$，$A^w_{C_i}\subseteq C_i.A$。$A^r_{C_i}$或$A^w_{C_i}$可能为一个空集，但它们中至少有一个为非空集合，否则C_i应该从集合$v.rw$中去除。

在 Artifact-centric 业务流程模型中，业务规则用于以 Condition-Action 的方式来关联服务和 Artifact[36]，如定义 2.10 所示。设$R=\{r_1,r_2,\cdots,r_m\}$为流程中涉及的业务规则集合，$R_{C_i}\subseteq R$为与 Artifact 类C_i关联的业务规则集合。

定义 2.10：(Business Rule) 业务规则管理哪个服务在什么前提条件下被调用。同时，定义了效果来限制服务调用后的后置条件。业务规则r定义为一个三元组$r=(\alpha,v,\beta)$，其中：

✧ α和β分别为前置和后置条件，均定义为无量词的一阶逻辑公式；

✧ $v\in V$为业务规则调用的服务。

前置和后置条件均可使用原子公式的合取或析取范式来表达。原子公式由标准的谓词和项组成。项包含变量（比如：Artifact、数据属性和状态）和常量（比如：数据属性和状态的值）。模式Z上的谓词包括两类：(1)状态谓词$instate()$；(2)属性谓词$defined()$或标量的比较操作[36]。具体地，如果属性$a\in C.A$有值，则$defined\ (C,a)$为真；如果状态$s\in C.S$处于激活状态，则$instate\ (C,s)$为真。流程开始时，$instate\ (C,s_{init})$为真，且对于$\forall a\in C.A$，$\neg defined\ (C,a)$为真。一个具体流程的业务规则集合隐含地描述了整个流程从开始到结束的控制流。为了维持业务规则r带来的合法状态变化，要求在前置和后置条件中有一对状态条件，即对$s_x,s_y\in C.S$，在$r.\alpha$中有$instate(C,s_x)$为真，且在$r.\beta$中有$instate(C,s_y)$为真。状态的变化使得 Artifact 从一个状态变迁到另一个状态或自身状态。设$T_C\subseteq C.S\times R_C\times C.S$为Artifact类$C$的三元组变迁关系。对于两个状态$s_s,s_t\in C.S$，变迁$t=(s_s,r_i,s_t)\in T_C$表示当前置条件$r_i.\alpha$成立时，调用服务$r_i.v$且 Artifact 从源状态$s_s$变迁到目标状态$s_t$。

每个 Artifact 除了数据模型，还有其生命周期[103]，定义了该 Artifact 类的状态变迁，即其行为，其中状态的变迁由在其上的服务调用完成来触发。本文采用确定性的有限状态机（Deterministic Finite Automaton，DFA）来形式化定义 Artifact 的生命周期模型，如定义 2.11 所示。

定义 2.11：（Artifact Lifecycle Model，ALM）设$\Pi = (Z, V, R)$为 Artifact-centric 业务流程模型，Artifact 类$C = (A, S)$，其中$C \in Z$，则C的生命周期模型可形式化定义为一个 DFA，$l = (S, s_i, S_F, T, \Sigma)$，其中$S$为状态的有限集合；$\Sigma$为输入符号的字母表，$\Sigma \subseteq (2^{R_C} - \emptyset)$；$s_i \in S$为初始状态；$S_F \subseteq S$为终止状态的集合；$T \subseteq (S \backslash S_F) \times \Sigma \times (S \backslash \{s_i\})$为 ALM 中的状态变迁关系。

字母表Σ中的每个元素是R_C的一个非空子集。因此，对$\theta \in \Sigma$且$s_0, s_1 \in S$，如果 ALM 中一个变迁为(s_0, θ, s_1)，则$|\theta| > 1$意味着当其中的任何一个业务规则$r_i \in \theta$的前置条件成立时，均可通过调用服务$r_i.v$来触发C从状态s_0到状态s_1发生变迁，即有变迁(s_0, r_i, s_1)成立。

例如一个典型的在线购物流程，该流程开始于一个用户提交订单信息，然后订单被送到制造商，对订单的商品进行打包，最后商品才被送到用户那里。采用 Artifact-centric 业务流程建模范式建立模型，基于该业务场景中涉及的业务单据，可以发现业务流程包含 3 个 Artifact：订单（*Order*）、运货单（*Shipment*）、发票单（*Invoice*），如图 2.3 所示。在识别出流程中所包含的 Artifact 之后，首先建立它们的数据模型，然后建立各 Artifact 的生命周期模型及其交互，如图 2.4 所示。从图中的 Artifact 生命周期模型，可以看到各 Artifact 可能经历的变迁和状态。

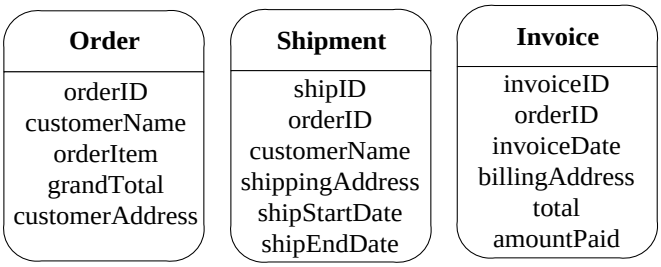


图 2.3 在线购物流程中所包含的 Artifact

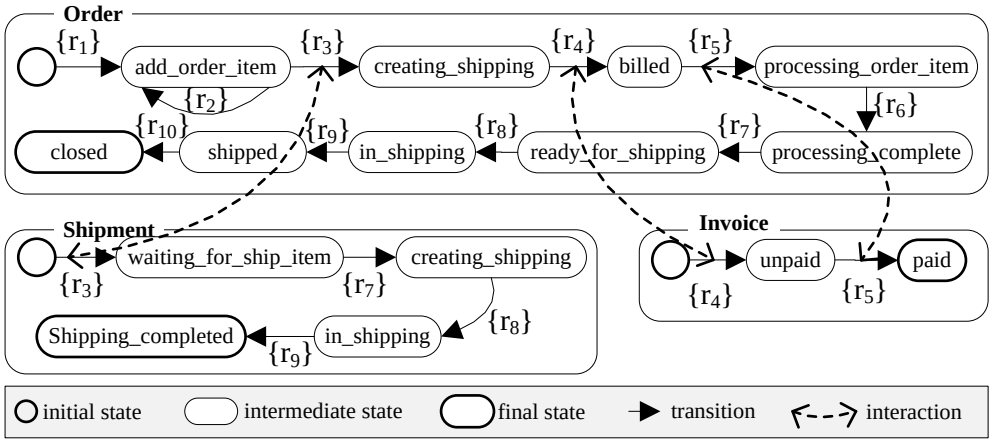


图 2.4 在线购物流程中所包含的 Artifact 的生命周期

状态的变迁通过业务规则的触发并调用相应的服务来实现，因此业务规则建立了 Artifact 与服务之间的联系。一个业务规则描述了在什么样的条件下需要调用哪个服务以及需要对哪些 Artifact 进行操作，并产生怎样的效果。为简单起见，图 2.4 中的例子中两个状态的变迁均只涉及了一个业务规则来实现。如表 2.3 所示，给出了两个业务规则的例子，采用 ECA 的规则进行描述。对于业务规则r_1，它的前提条件是 *Order* 处于 *init* 状态，并且数据属性$orderID$，$customerName$，$customerAddress$均未定义。如果该前提条件成立，则调用$createOrder$ 服务，并对 *Order* 进行更新。服务调用成功后，一个关于 Artifact 的后置条件需要满足，即 *Order* 从 *init* 状态变为 *add_order_item* 状态，并且数据属性$orderID$，$customerName$，$customerAddress$均已定义且赋予了具体的值。

表 2.3 业务规则举例

r_1: Customer requests to make an *Order* o	
Pre-condition	$instate\ (o, init) \wedge \neg defined\ (o, orderID)$ $\wedge \neg defined\ (o, customerName) \neg defined\ (o, customerAddress)$
Service	$createOrder\ (o)$
Post-condition	$instate\ (o, add\_order\_iten) \wedge defined\ (o, orderID)$ $\wedge defined\ (o, customerName) \wedge defined\ (o, customerAddress)$
r_5: Clear *Invoice* i for *Order* o	
Pre-condition	$instate\ (o, billed) \wedge instate\ (i, unpaid) \wedge defined\ (i, orderID)$ $\wedge$ o.$orderID = i.orderID \wedge$ o.$grandTotal = i.amountPaid$
Service	$payInvoice\ (o, i)$
Post-condition	$instate(o, processing\_order\_item) \wedge instate\ (i, paid)$

从以上的介绍可知，以 Artifact 为中心的建模范式结合了控制流与数据流两方面，将数据与流程放在同等重要的地位看待[103]，强调业务对象被操作的过程。基于关键业务数据在流程中的变化过程，对业务流程进行建模和分析[92,104]。以数据为中心的业务流程模型使用业务规则来确定业务流程中活动的流转，因此更具灵活性。关于 Artifact-centric 业务流程模型的形式化验证和挖掘，也已有一些研究工作，可参见文献[78,79,86,105-110]。同时，关于 Artifact-centric 业务流程模型的执行框架与平台已有一些研究工作[39,111-113]，IBM 也已发布可自动实现以数据为中心的业务流程模型的 MDBT Toolkit 工具，使得以数据为中心的业务流程模型易于 IT 实现。以数据为中心的业务流程已经被实践证明有诸多的好处[67]，比如：流程协作[114-116]、流程集成[117]、流程监控[118,119]、事务恢复[104]、流程的灵活变化[120,121]等。

2.3 本章小结

本章从业务流程建模范式的强调角度不同，分别介绍了两种不同的业务流程建模范式：Activity-centric 建模范式、Artifact-centric 建模范式，以及它们各自的特点和优缺点。总之，Activity-centric 建模范式忽略了对业务信息的观察，严格遵循控制流执行，缺乏灵活性。当流程规模增大时，业务流程模型的业务行为变得很难理解，不利于深度的流程分析和动态条件下系统行为的预测。Artifact-centric 建模范式通过使用业务规则引导控制流，具有更大灵活性。强调信息实体状态的变化，有利于实施监控与维护。尽管相比传统的 Activity-centric 建模范式，Artifact-centric 建模范式有一些优势，然而传统的 Activity-centric 建模范式在目前依然占据着工业界的主流。

第3章 模式驱动的异构业务流程整合建模

企业的发展常通过合并、收购的方式进行扩大来占领市场，这种发展方式常导致大量异构的业务流程。以中国移动为例，有 31 家分公司，该公司原来就是按照这种发展模式壮大的，办公系统内部的业务流程数量高达 8000 多个，严重阻碍了业务活动的运作和企业的统一决策。针对第 2 章中介绍的 Activity-centric 建模范式的业务流程，本章提出一种模式驱动的异构业务流程整合建模与监控方法，一个业务流程可以模型化为抽象业务模式的交互。每个业务模式又是可配置的，可以根据具体的应用场景进行配置为具体的业务模式。为执行业务模型，将各业务模式转化为工作流模式或运作模式的描述，使得业务模型可以运行于目前大部分的工作流程引擎之上。为提供企业业务的全局视图，在业务模式的边界上设置复杂事件处理监控点，这些监控点恰好与业务阶段对齐，因此可以监控业务的进展，为业务决策提供指导。最后，方法的实证研究验证了该方法的有效性。

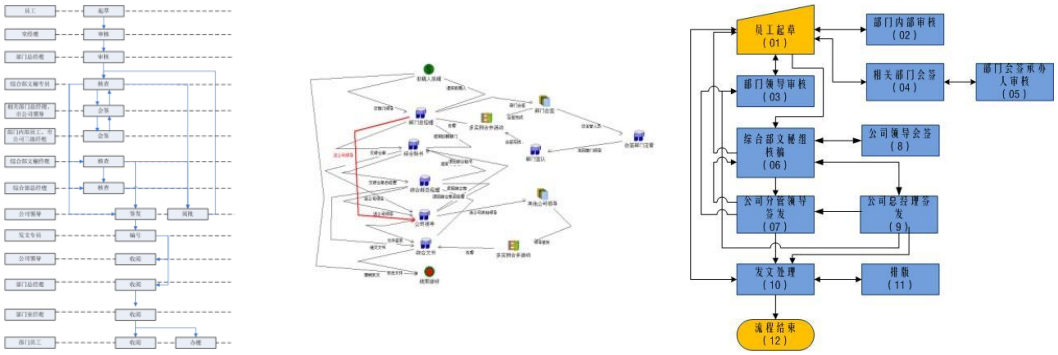
3.1 引言

目前，很多大企业在全国甚至在全世界都拥有很多分公司，各分公司可能是通过合并或采购的方式加入的，它们拥有各自的业务流程管理系统。尽管它们都拥有相似的业务，但是流程结构却各有差异。各分公司在加入之后，它们各自的流程系统由于历史遗留的各种原因依然继续各自演化和维护。同一业务流程在不同的组织中常以不同的方式实现。因此，对整个公司而言，就造成了大量异构的业务流程[8,122]，阻碍了总公司的统一管理和业务决策。为了整合异构的业务流程，需要对其进行某种程度或抽象的标准化工作，以便使得各分公司的业务流程拥有共同的业务阶段，并统计进展到各个阶段的业务流程实例的数量，从而可了解全局的业务进展，为总公司当前的业务调整或未来的业务决策提供依据。

下面以中国移动公司的例子来说明本章研究内容的背景、需求和挑战。中国移动公司现拥有 31 家分公司，各分公司办公系统内部业务流程分别独立地以自底向上的方式建立，系统已经运行了 10 余年。这些业务流程在各分公司中独立地维护和演化，导致业务流程的冗余和不一致。当总公司需要了解整体业务进展的情况时，无法对各分公司的情况进行汇总分析，阻碍了业务的统一规划和统一管理决策，也导致总公司的业务决策难以在各分公司开展。

图 3.1 展示了中国移动 OA 系统中云南、贵州、山西三省的发文流程，均分

别使用了不同的建模表示方法。**注意**：图 3.1 中只是给出了一个示例，主要是强调各分公司采用了不同的建模符号，且内部结构也各有其特点，任务的颗粒度也可能不一样，因此导致企业内部出现了大量异构的业务流程这一现象，读者无需了解具体的流程细节。目前，该公司正着手于业务流程的整合，旨在建立一个集中的 OA 系统，以支持异构业务流程的监控。



（a）流程 A　　（b）流程 B　　（c）流程 C

图 3.1 云南、贵州、山西三省的发文流程

实际上，跨组织的异构业务流程整合是一件费时费力的事。部分原因来自于企业的文化、组织结构、法律法规等，因此带来分公司个性化的业务需求，从而带来在建模符号的选择上、流程的设计上和具体的系统实现上的差异[38]。同时，这些差异的结果可能导致来自其它分公司的业务人员难以理解某分公司的业务流程，造成沟通和理解上的困难。然而，一个业务可以不同的方式执行而达到同一目标。正是由于这些特点导致在整个企业内部引发出大量的异构业务流程，使得跨组织的业务流程统一管理成为企业面临的一大挑战[8,122]。关于业务流程整合，已有的方法主要集中在可配置的业务流程模型上[41,42,123]。主要借鉴了软件产品线的思路，在单个业务流程模型中引入可变点，使得可配置的业务流程模型满足各具体应用场景的需要。因此，业务流程的建模可以直接根据具体应用场景对可配置的业务流程模型进行配置即可。然而，当一个可配置的业务流程模型要描述大规模的异构业务流程时，会导致复杂的可配置模型，给配置工作带来困难。这种方法的另一个重大的缺点是它们无法实现大规模异构业务流程的集中化监控，因此不能实现异构业务流程整合的最终目标，因其本质的目标主要还是流程建模而非业务统一的整合。

针对大量的异构业务流程整合问题，一般有两种可能的研究思路：一种是为各流程类别统一设计标准的业务流程模型，并在各子公司统一分别部署。然而，因为企业的业务时刻都在运转，业务流程管理系统（Business Process Management System，BPMS）必须时刻运行，至少不能长时间间断。一旦 BPMS 停止运行，

企业业务将无法开展，会给企业带来难以估计的损失。同时，面对大量异构的业务流程，首先需要对其进行充分的理解，然后再对其进行具体的标准化，这实在也是一项耗时费力的工作。因此，该方法表面上看似可行，实际无法实施。另一种方法是保持原来的业务流程模型不变，在业务抽象级别进行统一，即在业务的粗粒度级别进行抽象，从而达到业务流程标准化的目的。这种方法不需要打断当前的业务流程，只需要在业务阶段统一即可。实际调研表明，一般的领导阶层并不关心业务内部的具体细节进展，而是注重了解业务的阶段性进展。因此，我们相信第二种方法在实际当中是更为行之有效的。

模式驱动的异构业务流程整合主要有三大挑战：第一个是如何发现业务模式。由于已有业务流程设计的复杂性和模糊性，而分解的流程片段又应与业务场景对应，同时流程片段应有适中的大小和较高的复用率，即应该存在于大量的异构业务流程模型当中，这给自动化地分解有意义的流程片段带来困难。第二是如何为业务模式定义可配置的描述。由于业务流程的异构性，同一业务行为的细节在各分公司可能是不一样的。例如，可能包含不同数量的或者不同次序（即结构）的活动集合，或者有不同的业务规则。因此，发现的业务模式应该是可配置的。第三个挑战是集中化的异构业务流程监控。通过执行同样的业务模型来监控所有异构的业务流程，这对于检查业务进展和制定恰当的业务决策非常重要。

本章提出一个业务模式驱动的异构业务流程整合框架。通过人工的分析和讨论，总结出业务模式。利用业务模式，可以为每一业务类别建立统一的业务模型。采用参数来定义业务模式的描述，使得业务模式可以进行配置。为执行业务模型，将各业务模式转化为工作流模式或运作模式的描述，使得业务模型可以运行于目前大部分的工作流程引擎之上。为提供企业业务的全局视图，在业务模式的边界设置复杂事件处理监控点，这些监控点恰好与业务阶段对齐，因此可以监控业务的进展，形成企业业务的全局视图。

本章剩余部分的组织结构如下，第 2 节介绍基于业务模式的异构业务流程整合方法框架；第 3 节提出业务模式的发现和描述，以及业务模型的设计；第 4 节介绍业务模式的配置；第 5 节提出业务模型向工作流模式和运作模式的转化；第 6 节介绍异构业务流程的监控；第 7 节给出异构业务流程整合方法的评价；第 8 节总结本章的工作。

3.2 异构业务流程整合框架

基于业务模式的异构业务流程整合方法框架如图 3.2 所示，主要包含 4 个顺

序的步骤：业务模式的识别（Summarization）、业务模式的配置（Configuration）、业务模式的转化（Transformation）以及异构业务流程的监控（Intergrate Monitoring）[124]。

首先，根据已有的异构业务流程模型变体进行分析，用同一套流程建模符号来表示，找出具有相似业务理解的相似流程结构片段，作为业务模式的定义。之后，基于流程中抽取的业务模式，组合成业务模型。因此，业务流程可以模型化为业务模式的交互，可根据具体的业务场景各对业务模式进行配置操作，同时各业务模式可转化为工作流模式或运作模式的组合，因此业务模型可以运行于已有的工作流执行引擎（比如：YAWL[1]，Bizagi[2]等）或基于 Artifact 的业务流程执行引擎（比如：BizArtifact[3]等）之上。为实现统一的业务视图，在业务模式的边界设置复杂事件处理（Complex Event Processing，CEP）点。这样设置的复杂事件处理点是与业务阶段对齐的，因此业务人员或领导阶层可以统一地检查各异构业务流程的阶段性业务进展，形成统一的视图，为业务人员或领导阶层提供决策支持，或者为智能商业分析提供数据。

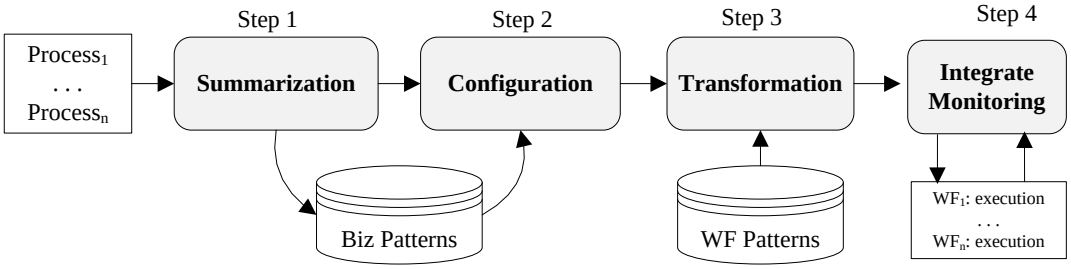


图 3.2 基于业务模式的异构业务流程整合方法框架

3.3 业务模式的识别与建模

本节首先基于具体的业务流程介绍总结的业务模式，包括其业务结构和业务规则。然后，根据业务模式的特征为其定义参数来描述，为业务模式的配置提供基础。最后，给出基于业务模式的业务模型的设计及其例子。

3.3.1 业务模式的识别

本章定义的业务模式是频繁发生在业务流程模型中可重复使用的业务行为。既具有相似的业务结构，又具有相似的业务理解和业务目标，可通过人工的探讨和总结得出。业务模式由一组业务相关的活动组成，属于粗粒度的业务描述。因

[1] http://www.yawlfoundation.org
[2] https://www.bizagi.com/en
[3] https://sourceforge.net/projects/bizartifact

此，容易被各分公司的业务人员理解，促进彼此的沟通，因为各分公司的业务在大体上是很相似的，只是在具体的执行细节上有不同的业务逻辑和业务判定规则。

需要注意的是：工作流模式[20]描述的是工作流的控制结构，即活动的次序。它作为一种模式，总结了流程中常用的控制结构，比如：并行结构、选择结构、循环结构等。它们都是一般的通用结构，与具体的业务无关。而运作模式[38]是与 Artifact 相关的运作结构，描述了 Artifact 被操作的模式，属于数据处理的模式，或者称之为 Artifact 的行为模式。而本章的业务模式，是对相似控制结构的总结与抽象，具有更高级的抽象粒度、更强的表达能力。因此，这三种模式是不一样的。业务模式与具体的业务场景息息相关，由业务人员对其进行解释。

针对中国移动公司的发文/收文/签报/会议纪要这几个业务流程进行分析，总结出一些业务模式，比如**会签**、**呈文**、**执行**、**抢先办**、**主控**等模式[8,122,124]。如表 3.1 所示，本章给出 5 个常见的业务模式作为例子进行介绍。这些业务模式在实际的业务流程中是频繁出现的，具有典型的业务特征，同类业务流程具有共同的业务模式集合[8,122,124]，形成统一的业务模型。由于业务模式的抽象度较高，因此针对同一业务得到的业务模式的数量相对很少，一般都在 3~5 个。这样，对于整个公司而言，得到的业务模型的数量也会相对较少，即一个业务对应一个，从而降低了管理的难度。

表 3.1 总结出的业务模式举例

ID	BP Name	Description
CS	Countersign	*n* parallel review tasks
DP	Document Preparation	draft-making first, then multi-level sequential review tasks from low level to high level
CO	Carry Out	multi-level task execution form high level to low level
CP	Competing	*n* independent tasks, but other *n-1* tasks are disposed
DG	Delegating	Task T is assigned to worker A first. When A is processing, T is aborted and assigned to worker B

3.3.2 业务模式的建模

任何一项业务，无论产出是一项服务还是物理产品，都依赖于业务数据[35]。业务数据记录了业务在产出过程中流程和目标的具体信息。近几年提出的业务 Artifact 正是这样一种业务数据的信息单元。Artifact 是由数据模型和行为模型（生命周期模型）组成，可用于理解和表示业务意图，从而使异构的业务流程在 Artifact 的业务上下文和业务行为上达成一致。因此，本节采用 Artifact 来描述在业务模式中涉及的一些相关信息。

业务行为的建模主要是描述在业务 Artifact 的生命周期。为了模型化业务模式，首先需要发现业务 Artifact，然后再根据 Artifact 的流转建立其生命周期。在以 Artifact 为中心的业务流程建模方法中，通常需要向业务人员咨询，通过与业务利益相关者进行深入的讨论和交流。在讨论过程中，一般会问到两个典型的问题[38]：（1）流程的结果是什么？（2）如何衡量你在做你想做的事情？这两个问题的答案有助于人们发现业务流程中涉及的信息实体。一般地，可以将业务中使用到的物理单据作为候选的业务 Artifact，比如：采购单、保险单等等。根据业务经验和业务 Artifact 对活动的影响，可以识别一个 Artifact 是否应该定义为关键 Artifact。另外，一个自动化的关键 Artifact 的识别方法是基于信息实体在业务流程上下文中的支配关系[60]。这种方法的前提是必须先有信息实体和业务流程模型。然而，基于中国移动公司的现状，这两项均不足具备。因此，本章主要采用人工方法讨论发现业务行为中可能涉及的业务 Artifact。由于一般的业务流程相对都比较简单，因此人工识别当中的业务 Artifact 相对较容易。

为了使业务模式可配置，本章提出参数化描述业务模式的方法。首先，介绍业务模式的统一描述。**业务模式**可定义化为 BP=(P, D)，其中*P*代表参数，*D*为描述，解释业务模式的业务过程[8,124]。进一步地，参数可形式化为 P=(*Role, Artifact, Activity, Rule, Map*)，其中

- *Role*：执行业务模式中活动的角色；
- *Artifact*：业务模式中操作的数据对象；
- *Activity*：业务模式中执行的活动；
- *Rule*：业务模式中业务规则；
- *Map*：为一个函数，为每个活动指派角色。

业务模式中参数除Map为函数外，其它每个组成元素都是一个集合。一个业务规则可进一步形式化为(R_{rule}, R_{desc})。R_{rule} 其实就是业务模式中描述业务规则的一组参数，具体含义根据具体模式而定，同时每个具体定义的业务模式需有对应的参数含义解析。在业务流程的具体执行过程中，可进行对 Artifact 信息的更新或创建新的 Artifact 实例，使得业务信息记录在 Artifact 当中。

业务模式的形式化描述是一个统一的模板，在具体的业务模式描述中将会有具体的参数。下面，我们以呈文和会签为例，给出它们的参数化描述。

会签模式：*n* 个部门分别签署同一份文件，它们的执行顺序任意，最后对 *n* 个签署意见进行汇总，交给一个决策活动进行最终决策[8,122,124]，参数的具体

描述如下：

- *Role*={r_1, r_2, …, r_n, r_d}；
- *Artifact*={csForm}，且 csForm={id int; subject str; $desc_1$ str; $suggestion_1$ str; …; $desc_n$ str; $suggestion_n$ str; finalDesc str; finalSuggestion str; state str}；
- *Activity*={$sign_1$, $sign_2$, …, $sign_n$, finalSign}；
- *Rule*: R_{rule}={*k*/*n*}，R_{desc}={当有多于 *k* 个签署意见为通过时，最终的签署意见为通过，否则为不通过}；
- *Map*: $sign_i$—> r_i，finalSign—> r_d。

呈文模式：拟稿后，逐步从低到高级别部门进行意见签署[8,124,125]，参数的具体描述如下：

- *Role*={r_0, <r_1, r_2>, <r_2, r_3>, …, <r_{m-1}, r_m>}，其中<r_i, r_j>的含义为：r_j 执行的活动紧跟在 r_i 执行的活动之后；
- *Artifact*={dForm}，且 dForm={id int; subject str; desc str; submitter str; accepter str; $desc_1$ str; $suggestion_1$ str; …; $desc_m$ str; $suggestion_m$ str; state str};
- *Activity*={creat_ manuscript, $sign_1$, $sign_2$, …, $sign_m$};
- *Rule*: R_{rule}={*m*}，R_{desc}={拟稿后，逐步由后续活动签署意见，一旦碰到意见不通过，则返回拟稿活动重新拟稿}；
- *Map*: $sign_i$—> r_i, creat_ manuscript—> r_0。

需要注意的是在业务模式的描述中给出的 Artifact 属性可能不是完整的，这里只列出一些必需的部分。在每个业务模式执行完成后，Artifact 的状态将会发生变化，从而使得业务级别的进展可以被监控。

3.3.3 业务模型的设计

通过对每一类业务流程进行业务模式的识别后，一个统一的业务模型将用来描述该类业务流程。根据不同的应用场景对业务模式进行配置后，一个业务模型可映射到不同的异构的同类业务流程中。如图 3.3 所示，为发文流程的业务模型及其配置。该业务模型包含两个业务模式：呈文模式和执行模式。可看出一个业务模型一般由很少的业务模式组成，从而可简化业务流程的建模，同时很容易被理解。一个配置好的呈文模式和执行模式可组合形成一个发文流程的条件是它们拥有共同的 Artifact。

基于统一的业务模型，可以在业务模式的边界设置复杂事件处理监控点，监控每个业务模式结点任务是否完成。内嵌的业务模式的进展可以忽略，主要关注最外层的业务模式的进展。在图 3.3 中有 3 个监控点，第一个监控点监控发文流程是否开始，第二个监控动点监控流程中的呈文模式结点任务是否完成，第三个监控点监控执行模式结点任务是否完成，即整个流程是否完成。

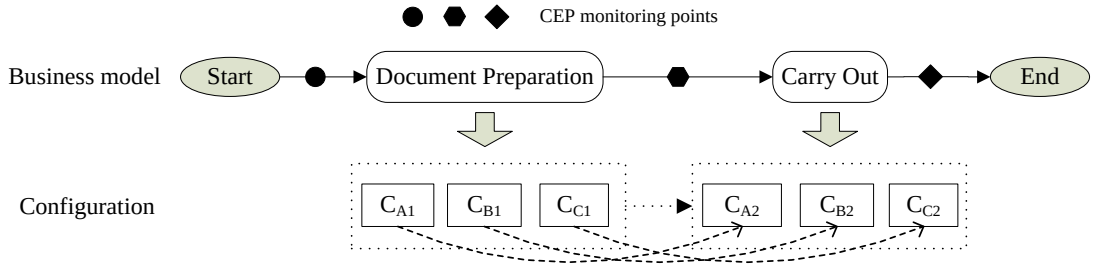


图 3.3 发文流程的业务模型及其配置

3.4 业务模式的配置

用参数将业务模式描述之后，则业务模式可以通过给参数初始化进行配置。这里，同样以会签和呈文模式为例说明业务模式的配置。如表 3.2 和表 3.3 所示，为云南的发文流程配置，包含了两个会签和一个呈文的配置。

业务规则中的参数根据不同的业务模式具有不同的含义，不同业务模式的参数需分别有相应的解析规则。表 3.3 中，在会签模式的配置 1 中，总共有 3 个部门主管进行意见签署，如果 3 个人中有 2 个人签署通过，则最后的会签决策就是通过，否则为不通过。具体地，在签署的意见过程中有 2 个不通过，因此最后的结果为不通过。在会签模式的配置 2 中，总共有 5 个公司职员进行意见签署，如果 5 个人中有 3 个人签署通过，则最后的会签决策就是通过，否则为不通过。具体地，在签署的意见过程中有 4 个通过，因此最后的结果为通过。

表 3.2 会签模式的两个配置例子

会签	角色	Artifact	活动	业务规则(k/n)
配置 1	{ $director_1$, $director_2$, $director_3$, general $director_0$}	{id 408; subject “review of directors”; desc “relevant sectors”; $desc_1$ poor; $suggestion_1$ no; $desc_2$ good; $suggestion_2$ yes; $desc_3$ bad; $suggestion_3$ no; finalDesc bad; finalSuggestion no; state “finished”}	{$sign_1$, $sign_2$, $sign_3$, finalSign}	2/3
配置 2	{ $staff_1$, $staff_2$, $staff_3$, $staff_4$, $staff_5$, $staff_0$}	{id 732; subject “review of staffs”; desc “internal staffs”; $desc_1$ good; $suggestion_1$ yes; $desc_2$ bad; $suggestion_2$ no; $desc_3$ excellent; $suggestion_3$ yes; $desc_4$ good; $suggestion_4$ yes; $desc_5$ good; $suggestion_5$ yes; finalDesc good; finalSuggestion yes; state “finished”}	{$sign_1$, $sign_2$, $sign_3$, $sign_4$, $sign_5$, finalSign}	3/5

表 3.3 呈文模式的一个配置的例子

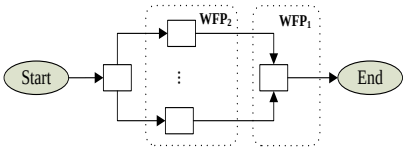
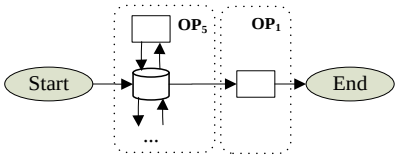
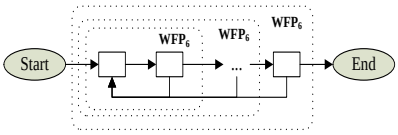
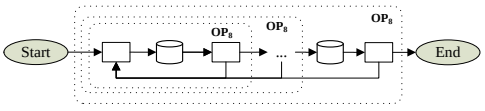
呈文	角色	Artifact	活动	业务规则(m)
配置 3	{team_leader, <project_leader, company_leader>}	{id 863; subject “manuscript review”; desc “2 steps’ review”; submitter Jack; accepter Rose; $desc_1$ nice; $suggestion_1$ yes; $desc_2$ awesome; $suggestion_2$ yes; state “finished”}	{creat_manuscript, sign1, sign2}	2

表 3.3 中，拟稿后面跟着 2 个签署意见的活动，首先由项目组长完成拟稿，然后由项目经理签署意见，最后由公司总裁签署意见。一旦遇到签署意见不通过，项目组长需要重新拟稿或修改原件再提交，或者停止流程。具体地，在签署的意见过程中均为通过，因此最后拟稿通过。

3.5 业务模式的转化

业务模式描述的是业务的行为，因此业务模型是在业务级别的描述，不能直接用于 IT 系统的实现。目前，业务流程建模的范式主要有两种主流的方式：传统以活动为中心的方式和最近提出的以数据为中心的方式。因此，为了使业务模型能够在这两种建模范式的执引擎上执行，可以将业务模式转化成工作流程模式的组合或运作模式的组合。这种转化是可行的，因为工作流模式和运作模式是比业务模式更加具体的流程结构描述，可表达流程更细节的内容。目前为止，一共有 43 种工作流模式[20,21]，归纳了常用的流程结构，其中包含 WFP_1：顺序模式、WFP_2：并行模式、WFP_3：同步模式、WFP_4：互斥分支模式、WFP_5：简单合并模式、WFP_6：循环模式等等。为方便基于数据的业务流程建模，Rong Liu 等人[38]提出了 9 种运作模式，包括 OP_1：管道模式、OP_2：仓库模式、OP_3：分支模式、OP_4：汇聚模式、OP_5：项目模式、OP_6：创建模式、OP_7：同步模式、OP_8：重做模式、OP_9：抢先办模式。

表 3.4 业务模式向工作流模式和运作模式的转化举例

BP	Workflow Patterns	Operational Patterns
CS		
DP		

根据工作流模式和运作模式的结构语义，会签模式可以转化为并行工作流模式和顺序工作流模式的组合，而呈文模式可表示为多个嵌套的循环模式的组合，具体的转化如表 3.4 所示。同样地，会签模式可以由项目运作模式和管道运作模式的组合来表示，而呈文模式可以多个嵌套的重做运作模式的组合来表示。

3.6 异构业务流程的监控

jBPM（Java Business Process Management）是一个开源、灵活、可扩展的业务流程管理工具，集成了丰富的功能。jBPM 允许业务流程建模、执行和监控，遍及整个业务流程管理的生命周期。jBPM5 及以上版本均集成了工作流引擎和 Drools 规则引擎，分别可用于执行工作流和描述业务模式中的业务规则。同时，jBPM5.4 支持图形化的 BPMN2.0 建模符号标准，业务模型中除业务模式结点以外的结点均可用 BPMN2.0 建模符号元素表示。BPMN2.0 是目前工业界业务流程建模的事实标准，在 jBPM 中业务建模人员和 IT 开发人员使用同一套建模语言来表达业务流程模型，有助于彼此的沟通，提高业务流程管理系统的开发效率。另外，jBPM5.4 支持 adHocSubProcess（即子流程）的表示和调用，每个业务模式结点可用子流程的形式来表达。然而，需要注意的是业务模式结点表示的子流程和普通的子流程不一样，它可以没有开始结点、终止结点以及连接活动的网关结点，内部包含的活动由业务规则产生的事件触发执行，因此更具有灵活性。由以上分析可知，jBPM 完全满足业务模型执行的前提条件，具体实现可参见[8,122]。

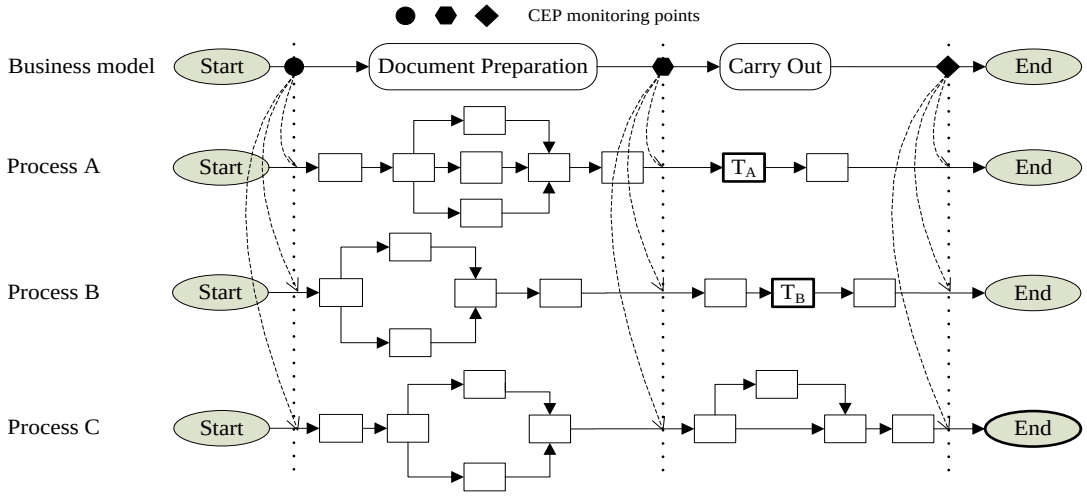


图 3.4 基于业务模型的异构业务流程监控

本节以三个异构的业务流程的监控为例来说明集中化监控的方法，如图 3.4 所示。Process A、B、C 可以是工作流模型或运作模型，它们均来自于同一个业务模型的配置，拥有统一的业务监控点。加粗的部分表示流程正在执行的阶段，因此由图可知 Process A 和 B 已经完成了呈文模式结点的任务，正在运行执行模

式结点，至于执行模式具体运行到哪一个步骤，业务人员并不关心。而 Process C 已经完成了整个业务流程的执行。因此，若同类异构业务流程拥有共同的业务模型，则业务人员可以很容易地统计各异构业务流程的进展情况，为领导阶层的决策提供支持。

3.7 方法评价

为了评价本章提出的方法，本章采用 Boot-strap 方法抽样进行评估。本章采用中国移动公司 19 个省市的发文、收文、签报、会议纪要这几个业务流程组成的数据集进行分析，并将发文流程作为测试集，总结出 5 个业务模式：会签、呈文、执行、抢先办、主控。基于这些业务模式和测试集，对发文流程的业务模型进行配置，其中 16 个发文流程可以直接由共同的业务模型进行配置得到，而其它 3 个发文流程需要增加额外的信息来完成。该实证研究表明对于本章抽取的样本，本章提出的方法可以应用于超过 80%的省市。既然样本来自于 4 类典型的且最常使用的流程类别之一，我们相信当本章的方法应用于由一些特定业务行为组成的大规模异构业务流程模型的整合时，可以得到类似的效果。通过调整业务模式使之适用于更多的场景，使得一个共同的业务模型可以描述所有同类别的异构业务流程。

以往“过程视图”的整合方法主要是通过标准化业务流程模型。这样的方法在很多场景下其实是难以实现的，因为不同的利益相关者采用了不同的流程建模语言，从一个转化到另一个容易带来语义上的歧义，且导致业务意图的歪曲表达。而且，即便它们都是统一标准化的流程模型，然而还是可以有不同的执行路径，仍然无法达到统一监控的目的。而业务模型的方法可以使得异构的业务流程拥有统一的业务模型，且该业务模型由不同的业务模式顺序串联而成，使得异构业务流程在业务模式级别拥有统一的执行路径，从而达到统一监控的目的。

3.8 本章小结

本章提出一种模式驱动的异构业务流程整合方法。业务模式描述了业务的行为，且可以被高度地重用。为了支持当前的两种主流的业务流程建模方法的执行引擎，业务模式可以转化为工作流模式或运作模式的组合，使得业务模型可以被执行。基于统一的业务模型，通过在业务模式的边界设置复杂事件处理监控点，异构业务流程可以被集中地监控。最后，实证研究验证了本章方法的有效性。

本章评价工作中涉及的流程数量相对较少，下一步工作可以在大范围内测试

更多业务流程模型来验证业务模式驱动的异构业务流程整合方法在实际应用中的有效性。另外，一般企业的业务流程中涉及的业务模式的数量是较少的，因此本文基于人工的业务模式抽取方法是适用的。特殊地，当业务流程中涉及的业务模式数量较多时，可通过挖掘业务流程日志的方法自动化地发现频繁的业务流程子结构，作为人工识别业务模式的一个参考。自动化的业务模式转化也是下一步要研究的工作。

第4章 基于生命周期的 Artifact-centric 业务流程可配置建模

随着企业的发展和变化，流程的变体可能出现在组织内部或者跨越不同的组织，因此这些组织可从可配置的业务流程模型中获益。一个可配置的业务流程模型是一个业务流程模型的多个变体的集中体现，它可用来设计新的业务流程模型。可配置的业务流程模型其实是带有可配置选择的业务流程模型。用户可通过配置可配置的业务流程模型进行业务流程的建模，而无需完全从零开始建立新的业务流程模型。因此，可配置的业务流程模型有利于提高业务流程模型的复用效率和建模效率，并提高流程模型的质量。第 3 章的研究内容也涉及配置建模，属于流程建模的一个方面，但是针对以活动为中心的业务流程模型且目标是解决异构业务流程的整合与监控。基于可配置建模的诸多优势，针对第 2 章中介绍的 Artifact-centric 建模范式的业务流程本章研究其可配置建模方法，解决设计阶段的 Artifact-centric 流程建模问题，其主要的研究思路是通过对流程中包含的所有 Artifact 生命周期模型的配置来实现 Artifact-centric 业务流程模型的整体配置。

4.1 引言

目前，有很多关于可配置的业务流程建模方法被提出来。这些配置方法在参考流程模型的基础上加入可配置点，并对多种流程建模语言，比如 EPC、YAWL、WF Net 等流程建模语言进行了可配置的流程建模研究。为了获得参考流程模型，一些业务流程模型的合并方法也被提了出来。然而，这些可配置流程模型的建模方法和合并方法主要集中于传统以活动为中心的业务流程模型。据我们的研究调查所知，目前还没有关于 Artifact-centric 流程模型配置的研究。与传统的以活动为中心的业务流程建模方法不同，控制流显示地体现在业务流程模型中，而在 Artifact-centric 流程建模方法中，控制流隐含地建模在业务规则中，因此传统的业务流程模型可配置建模方法不能适用于 Artifact-centric 流程模型的配置。Artifact-centric 业务流程建模方法具有相比传统以活动为中心的业务流程所不具备的优势，被研究者们认为是下一代业务流程建模的新趋势，因此研究 Artifact-centric 流程模型的配置在理论和实践上都具有深远意义。

4.2 可配置的 Artifact-centric 业务流程建模方法

在 Artifact-centric 业务流程建模方法中，一个业务流程可以通过用 Artifact

进行构建。一般而言，Artifact-centric 业务流程模型由 Artifact、服务和业务规则三类元素组成。假设有 N 个 Artifact-centric 业务流程模型变体$\Pi_1, \Pi_2, \cdots$，它们均源自于同一个集中的 Artifact-centric 业务流程模型Π，如图 4.1 所示。Π包含了衍生变体$\Pi_1, \Pi_2, \cdots$的所有信息，即有$A_{ij} \subseteq A_j$， $S_{ij} \subseteq S_j$，$V_i \subseteq V$，$R_i \subseteq R$。相反，若有了这 N 个 Artifact-centric 业务流程模型变体，则可通过将这些 Artifact-centric 业务流程模型变体进行合并来得到集中的 Artifact-centric 业务流程模型Π。

一个可配置的流程模型是多个相似的业务流程模型变体的综合描述，因此可配置流程模型的行为是不同流程模型行为的并集[48]。在合并 Artifact-centric 业务流程模型时，需要对其各个组成元素来做并集的操作。假设两个 Artifact-centric 业务流程模型为$\Pi_1 = (Z, V, R)$和$\Pi_2 = (Z, V, R)$，它们有相似的业务逻辑和共同的业务目标。两者可能有一些共同的 Artifact 类名，即$\Pi_1.Z.C_i = \Pi_2.Z.C_j$，它们来源于同一个 Artifact 类，然而它们可能有不同的数据属性和生命周期。设Π_m为Π_1和Π_2的合并，则两个 Artifact-centric 业务流程模型的合并算法如算法 4.1 所示。

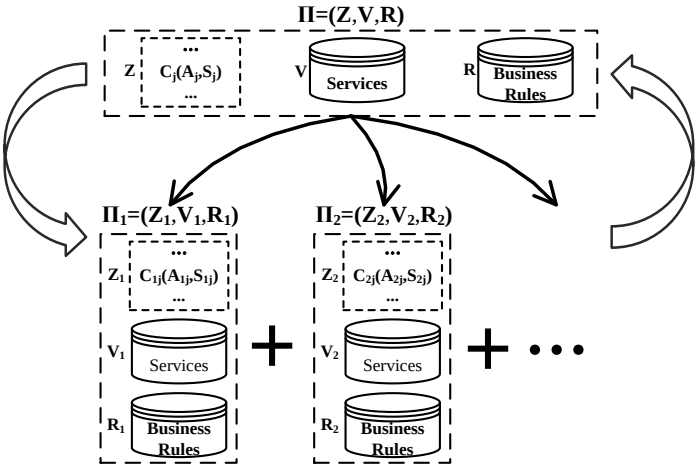


图 4.1 Artifact-centric 业务流程模型变体的产生与合并

算法 4.1：两个 Artifact-centric 业务流程模型的合并

输入： $\Pi_1 = (Z, V, R)$, $\Pi_2 = (Z, V, R)$
输出： $\Pi_m = (Z, V, R)$

1: $\Pi_m.Z = \emptyset$;
2: $\Pi_m.V = \Pi_1.V \cup \Pi_2.V$; // merging services
3: $\Pi_m.R = \Pi_1.R \cup \Pi_2.R$; // merging business rules
4: $Z_n = \Pi_1.Z \cap \Pi_2.Z$; // merging Artifact classes (line 4-11)
5: $\Pi_m.Z = \Pi_1.Z \cup \Pi_2.Z - Z_n$;
6: **for** each C_i in Z_n **do**
7: $C_m.A = \emptyset$; $C_m.S = \emptyset$;
8: $C_m.A = \Pi_1.Z.C_i.A \cup \Pi_2.Z.C_i.A$;
9: $C_m.S = \Pi_1.Z.C_i.S \cup \Pi_2.Z.C_i.S$;
10: $\Pi_m.Z = \Pi_m.Z \cup C_m$;
11: **end for**
12: **return** $\Pi_m = (Z, V, R)$;

事实上，由于 Artifact-centric 流程被建模为其中包含的 Artifact 生命周期的交互，因此可以通过分别对所有对应 Artifact 的生命周期模型的合并来实现两个 Artifact-centric 业务流程模型的合并，如图 4.2 所示。对于 Artifact 类C_1，我们合并来自两个 Artifact-centric 流程模型的两个 Artifact 生命周期模型ALM_{C_1}和ALM_{C_2}，分别对C_1的数据属性集合和状态集合、相关服务的集合以及相关业务规则的集合进行合并。对流程中包含的所有 Artifact 的生命周期模型分别进行这样的合并之后，则可以得到一个完整的 Artifact-centric 业务流程模型。

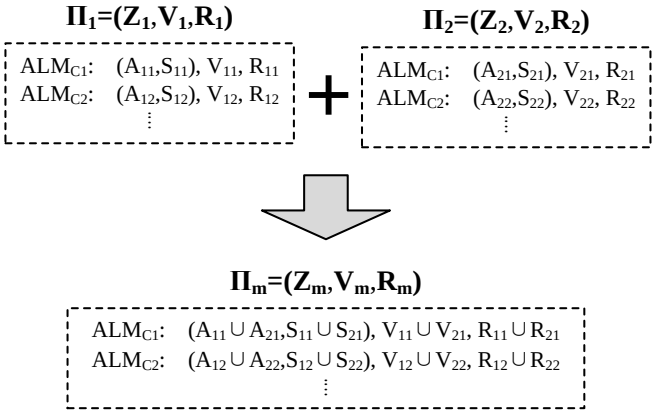


图 4.2 基于 Artifact 生命周期模型合并实现 Artifact-centric 流程模型的合并

算法 4.1 的 2-3 行是对服务和业务规则的集合进行合并，4-5 行是对不重叠的 Artifact 类进行合并，6-11 行对重叠的 Artifact 类进行合并，其中包括 Artifact 类的数据属性集合和状态集合的合并。算法的循环部分是对重叠的 Artifact 类分别进行合并操作。因此，假设重叠的 Artifact 类的数量为N，则算法 4.1 的时间复杂度为$O(N)$。

4.3 可配置的 Artifact 生命周期建模

基于以上分析，通过分别对流程中包含的所有 Artifact 生命周期模型的配置来实现整体的 Artifact-centric 业务流程模型的配置，这就是本章的主要研究思路。首先，提出 Artifact 生命周期图（Artifact Lifecycle Graph，ALG）的概念来对 Artifact 生命周期模型进行图形化描述；然后，提出 Artifact 生命周期图的合并方法；在合并的 Artifact 生命周期图中，加入配置选项，使其形成一个可配置的 Artifact 生命周期图；最后，对 Artifact 生命周期图进行配置，亦即 Artifact 生命周期模型的配置。在对所有 Artifact 生命周期模型对应的 Artifact 生命周期图进行这样的配置后，可得到一个完整的 Artifact-centric 流程模型。Artifact 生命周期模型的配置方法框架如图 4.3 所示。

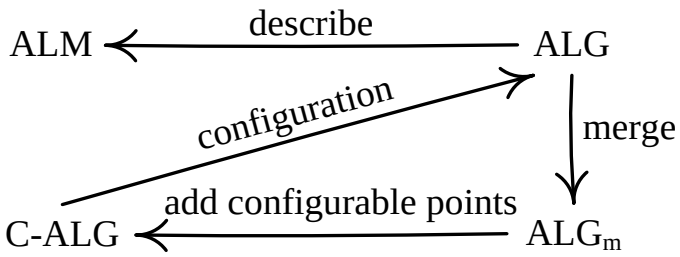


图 4.3 Artifact 生命周期模型的配置方法

4.2.1 Artifact 生命周期模型的描述

下面给出 Artifact 生命周期图的定义，并给出 Artifact 生命周期图结构化正确性的概念，如接下来介绍的定义 4.1 和定义 4.2 所示。

定义 4.1:(Artifact Lifecycle Graph,ALG)对于 Artifact C的生命周期模型ALM_C所对应的 Artifact 生命周期图为一个有向图，其中图中每条边赋予一个业务规则的非空集合，它可以表示为$ALG_C = (S_C, A_C, R_C, l_C)$，其中$S_C$为 Artifact C的状态集合，$A_C \subseteq S_C \times S_C$为有向边的集合，$R_C$为与 Artifact C相关的业务规则的集合，$l_C: A_C \to (2^{R_C} - \emptyset)$为一个映射，将每条有向边映射到一个非空的业务规则集合。

若对于$e = (s_x, s_y) \in A_C$，$|l_C(e)| > 1$意味着任何属于$l_C(e)$的业务规则在流程执行时均有可能被触发并调用相应的服务使 Artifact C的状态从s_x变迁到s_y。需要注意的是：Artifact 生命周期图和 Artifact 生命周期模型描述了相同的信息，只是使用了不同的形式，因此在本章使用时并不加具体的区分。

定义 4.2：（Artifact 生命周期图的结构化正确性）一个 Artifact 生命周期图$ALG_C = (S_C, A_C, R_C, l_C)$是结构化正确的，当且仅当它满足以下条件：

（1） ALG_C有且仅有一个初始状态；

（2） ALG_C至少有一个终止状态；

（3） 对于ALG_C中的每一个状态，均至少处在一条从开始状态到某个终止状态的路径上；

（4） 对于ALG_C中的每一个变迁，均附带了一个业务规则的非空集合。

如图 4.4 中 Artifact 生命周期图的例子所示，状态 A 为其中的初始状态，状态 G 和 H 均为终止状态，每个结点都至少处在一条从开始状态到某个终止状态的路径上，并且每条有向边均附带了一个业务规则的非空集合，因此该 Artifact 生命周期图是结构化正确的。

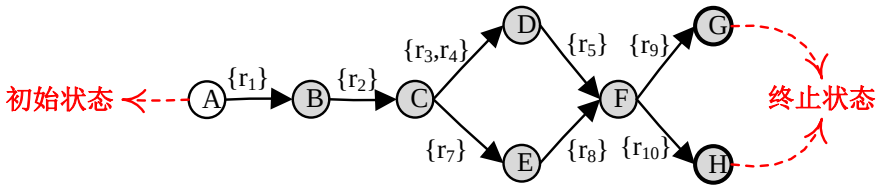


图 4.4 一个 Artifact 生命周期图的例子

4.2.2 Artifact 生命周期图的合并

同时，我们提出 Artifact 生命周期图的合并算法。对于图中的状态、有向边和业务规则可以根据集合的并操作直接进行合并。对于映射函数，假设两个生命周期图为$ALG_{C_i} = (S_{C_i}, A_{C_i}, R_{C_i}, l_{C_i})$和$ALG_{C_j} = (S_{C_j}, A_{C_j}, R_{C_j}, l_{C_j})$，$C_i$和$C_j$属于同类 Artifact。设$ALG_m = (S_m, A_m, R_m, l_m)$为$ALG_{C_i}$和$ALG_{C_j}$的合并，则有$S_m = S_{C_i} \cup S_{C_j}$，$A_m = A_{C_i} \cup A_{C_j}$，$R_m = R_{C_i} \cup R_{C_j}$，基于$l_{C_i}$和$l_{C_j}$，合并的映射函数$l_m$可定义为:

$$l_m(e) = \begin{cases} l_{C_i}(e) & e \in A_{C_i} \wedge e \notin A_{C_j} \\ l_{C_j}(e) & e \notin A_{C_i} \wedge e \in A_{C_j} \\ l_{C_i}(e) \cup l_{C_j}(e) & e \in A_{C_i} \wedge e \in A_{C_j} \end{cases} \qquad \text{（公式 4.1）}$$

基于以上映射函数的合并方法，可得到 Artifact 生命周期图合并算法的描述，如算法 4.2 所示。算法 4.2 的第 1 行是对图的状态结点集合、有向边集合及总的业务规则集合进行合并，2-10 行的循环部分是分别对每条有向边上的业务规则集合进行合并。因此，假设两个 Artifact 生命周期图中一共拥有N条有向边，即$|A_m| = N$，则算法 4.2 的时间复杂度为$O(N)$。

算法 4.2：两个 Artifact 生命周期图的合并

输入： $ALG_{C_i} = (S_{C_i}, A_{C_i}, R_{C_i}, l_{C_i})$, $ALG_{C_j} = (S_{C_j}, A_{C_j}, R_{C_j}, l_{C_j})$
输出： $ALG_m = (S_m, A_m, R_m, l_m)$
1: $S_m = S_{C_i} \cup S_{C_j}$; $A_m = A_{C_i} \cup A_{C_j}$; $R_m = R_{C_i} \cup R_{C_j}$;
// merging nodes/states, arcs, and business rules respectively
2: **for** each e in A_m **do** // merging functions (line 4-12)
3: **if** $e \in A_{C_i} \wedge e \notin A_{C_j}$ **then**
4: $l_m(e) = l_{C_i}(e)$;
5: **else if** $e \notin A_{C_i} \wedge e \in A_{C_j}$
6: $l_m(e) = l_{C_j}(e)$;
7: **else**
8: $l_m(e) = l_{C_i}(e) \cup l_{C_j}(e)$;
9: **end if**
10: **end for**
11: **return** $ALG_m = (S_m, A_m, R_m, l_m)$;

基于以上 Artifact 生命周期图的合并算法，我们定义两个 Artifact 生命周期图的合并操作如定义 4.3 所示。

定义 4.3：（Merger Operation for Artifact Lifecycle Graph，⊕）设两个生命周期图为$ALG_{C_i} = (S_{C_i}, A_{C_i}, R_{C_i}, l_{C_i})$和$ALG_{C_j} = (S_{C_j}, A_{C_j}, R_{C_j}, l_{C_j})$，$C_i$和$C_j$属于同类 Artifact。则这两个生命周期图的合并操作⊕可以定义如下：$ALG_m = ALG_{C_i} \oplus ALG_{C_j} = (S_m, A_m, R_m, l_m)$，其中合并结果$ALG_m$由调用算法 4.2 得到。

Artifact 生命周期图合并操作的结果也是一个 Artifact 生命周期图，其中包含了原来两个有向图中所有的结点、边、和业务规则。既然集合的并集操作满足结合律和交换律，因此 Artifact 生命周期图的合并操作同样也满足**结合律**和**交换律**，即有如下的两个公式成立。因此，基于以上分析，可以得到定理 4.1、定理 4.2。

$$ALG_1 \oplus ALG_2 = ALG_2 \oplus ALG_1 \qquad \text{(公式 4.2)}$$

$$(ALG_1 \oplus ALG_2) \oplus ALG_3 = ALG_1 \oplus (ALG_2 \oplus ALG_3) \qquad \text{(公式 4.3)}$$

定理 4.1：$< \mathbb{G}, \oplus >$为一个代数系统，其中$\mathbb{G}$为 Artifact 生命周期图的集合。

证明：$\mathbb{G}$为非空集合，⊕定义为$\mathbb{G}$上的二元运算且满足其运算封闭性，证毕。

定理 4.2：代数系统$< \mathbb{G}, \oplus >$为一个半群，其中$\mathbb{G}$为 Artifact 生命周期图的集合。

证明：$< \mathbb{G}, \oplus >$为一个代数系统且二元运算⊕是可结合的，证毕。

由公式 4.2、4.3 可知，n个 Artifact 生命周期图的合并结果与合并操作的顺序没有关系。当存在一个新的流程模型变体时，随后其对应的 Artifact 生命周期图便可以合并到已有的合并结果当中。注意，尽管此处提到的代数系统、半群在本章的后面并没有用到，但这一特性为 Artifact 生命周期模型的形式化分析奠定了的基础，因此在此特别提出来，有助于未来工作的开展。

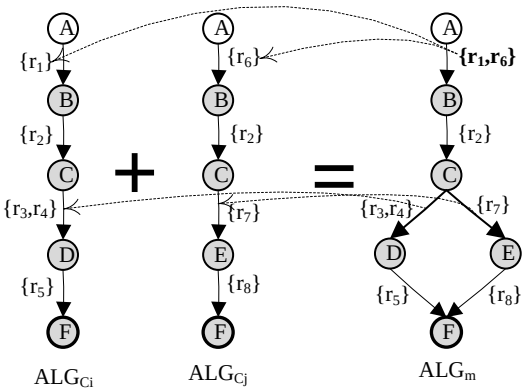


图 4.5 两个 Artifact 生命周期图合并的例子

下面采用一个例子来说明两个 Artifact 生命周期图的合并，如图 4.5 所示。可看到合并的图中包含了ALG_{C_i}和ALG_{C_j}中的所有元素信息。边(A, B)来自于两个

原始的图且赋予了不同的业务规则集合，在ALG_{C_i}中赋予了$\{r_1\}$，在ALG_{C_j}中赋予了$\{r_6\}$，因此在ALG_m中边(A,B)赋予了$\{r_1,r_6\}$。而边(C,D)仅来自于一个原始的图，因此该边在合并的图中所赋予的业务规则集合不变。

然而，在合并 Artifact 生命周期图之前，需要先确定两个生命周期中状态结点之间的映射关系，才能确定共同的结点和边，因为不同的变体中同一个 Artifact 的生命周期的状态命名可能是不一样的。同样，业务规则的命名也可能不一样。然而，它们之间的语义是相近的。因此，本章提出通过计算不同命名之间的相似度来匹配映射关系，并给出统一的命名。下面以状态结点的映射为例，说明不同命名的匹配方法。对于流程中一个 Artifact 类C，假设$s \in S_{C_i}$为流程模型变体P_1中生命周期ALG_{C_i}的一个状态，同样$s' \in S_{C_j}$为流程模型变体P_2中生命周期ALG_{C_j}的一个状态。为了计算s和s'之间的相似度Sim_S，本章结合语法和语义的相似度度量，该方法也常用在传统以活动为中心的业务流程模型活动的标签匹配中[32]。语法相似度采用 Levenshtein 距离（又称编辑距离），它通过计算将一个字符串转化为另一个字符串需要的最少编辑操作（比如：插入、删除、替代一个字符）次数来计算语法相似度[126]。语义相似度则可采用 WordNet 词汇数据库计算语义关系[127]。WordNet 包包含一系列返回两个单词同义程度的算法。具体地，可采用 WUP 算法[128]通过考虑两个单词在 WordNet 数据库中的深度来度量它们之间的相关性。通过预处理标签（比如：将所有字母转成小写，删除停用词等）进行标准化后，则两个标签的相似度为它们之间语法和语义相似度的平均，公式如下：

$$Sim_S(L(s),L(s')) = \frac{LD\left(L(s),L(s')\right)+WUP(L(s),L(s'))}{2} \qquad \text{（公式 4.4）}$$

其中，$0 \le Sim_S \le 1$，LD和WUP分别是返回$L(s)$与$L(s')$之间基于 Levenshtein 距离和基于 WordNet 的相似度。s'是s的最佳匹配，当且仅当$Sim_S(L(s),L(s')) \ge minSim_S$，并且不存在$s_x \in S_{C_j}$使得$Sim_S(L(s),L(s_x)) > Sim_S(L(s),L(s'))$，其中$minSim_S$为用户设置的阈值。

4.2.3 可配置的 Artifact 生命周期模型

接下来，给出可配置的 Artifact 生命周期图的形式化定义，如定义 4.4 所示。

定义 4.4：（Configurable ALG，C-ALG）可配置的 Artifact 生命周期图是一个元组$C-ALG_C = (S_C, A_C, R_C, l_C, A^C, l^C, RS^C)$，其中$S_C, A_C, R_C, l_C$的含义和$ALG_C$的定义一致，$A^C$是可配置的有向边集合，$l^C$是可配置的业务规则集合，$RS^C$是配置约束的集合。

在可配置的 Artifact 生命周期图中，可配置的有向边的配置可以是$blocked$或$selected$，若配置为$blocked$，则整个分支被去掉；而可配置的业务规则可以是

$removed$或$selected$。可配置的 Artifact 生命周期图的配置选择如图 4.6 所示，其中$\{r_1, r_6\}$有三个配置选项：$\{r_1, r_6\}$、$\{r_1\}$和$\{r_6\}$。配置点的配置约束可用逻辑表达式描述，原子命题为赋予可配置点一个具体值。例如：$e_0, e_1 \in A^C$，"$Conf(l_C(e_0)) = \{r_1\}$"和"$Conf(e_1) = blocked$"，其中$l_C(e_0)$一个可配置的业务规则集合，e_1是一条可配置的有向边。一个配置需求可能表达为"$Conf(l_C(e_0)) = \{r_1\}$" $\Longrightarrow$ "$Conf(e_1) = blocked$"，表示当第一个配置成立时，第二个配置也必须成立。

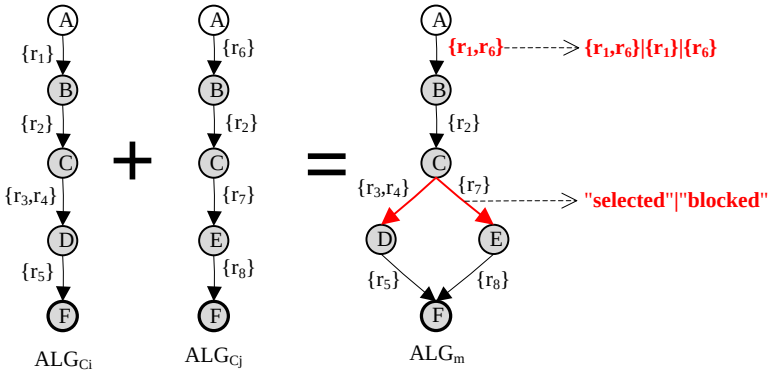


图 4.6 可配置的 Artifact 生命周期图中的配置选择

为了在 Artifact 生命周期图中加入可配置点，我们提出一个算法自动地在合并的 Artifact 生命周期图中识别可配置点，如算法 4.3 所示。

算法 4. 3：产生可配置的 Artifact 生命周期图

输入： $ALG_{C_i} = (S_{C_i}, A_{C_i}, R_{C_i}, l_{C_i})$, $ALG_{C_j} = (S_{C_j}, A_{C_j}, R_{C_j}, l_{C_j})$, $ALG_m = (S_m, A_m, R_m, l_m)$

输出： $ALG_m = (S_m, A_m, R_m, l_m)$ with configurable points

1: $N_m = \{n | n \in S_m \wedge |n\bullet| > 1\}$; // set of nodes whose outdegree is more than 1 in ALG_m
2: $F_{C_i} = \{n | n \in S_{C_i} \wedge |n\bullet| = 0\}$; // set of final sates in ALG_{C_i}
3: $F_{C_j} = \{n | n \in S_{C_j} \wedge |n\bullet| = 0\}$; // set of final sates in ALG_{C_j}
4: **for** n each in N_m **do** // generate configurable arcs (lines 4-15)
5: **for** each $e \in n\bullet$ **do**
6: **if** $(e \notin A_{C_i} \wedge e \in A_{C_j}) \vee (e \in A_{C_i} \wedge e \notin A_{C_j})$ **then**
7: e is configurable in ALG_m;
8: **end if**
9: **end for**
10: **end for**
11: **for** each $n \in (F_{C_i} \cup F_{C_j} - N_m)$ **do**
12: **if** $|n\bullet| > 0$ in ALG_m **then**
13: the outgoings of n are configurable in ALG_m;
14: **end if**
15: **end for**
16: **for** $e \in A_m$ **do** //generate configurable sets of business rules (lines 16-20)
17: **if** $(e \in A_{C_i} \wedge e \in A_{C_j}) \wedge (l_{C_i}(e) \neq l_{C_j}(e))$ **then**
18: $l_m(e)$ is configurable in ALG_m;
19: **end if**
20: **end for**
21: **return** $ALG_m = (S_m, A_m, R_m, l_m)$ with configurable points;

在算法 4.3 中，在合并的 Artifact 生命周期图中对于某一个结点有 2 个或多个输出边，若某一个输出边仅来自于一个原始图，则该输出边是可配置的，4-10 行即是对可配置的有向边的识别。而对于ALG_{C_i}或ALG_{C_j}中一个终结点，若在ALG_m中有输出边，则其输出边是可配置的，该识别过程对应算法的 11-15 行。16-20 行是对共同的有向边进行判断，若其上附着的业务规则集合不一样，则合并的业务规则集合是可配置的。因此，整个算法 4.3 主要是分别对不同类别的有向边进行循环判断。假设两个 Artifact 生命周期图中一共拥有N条有向边，即$|A_m| = N$，则算法 4.3 的时间复杂度为$O(N)$。

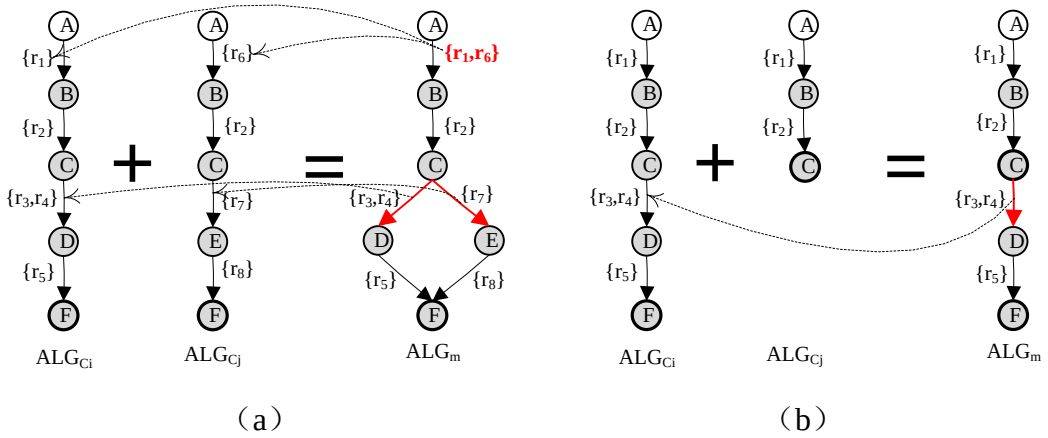


图 4.7 合并的 Artifact 生命周期图中的配置点的产生

例如：图 4.7（a）中，边(C, D)和(C, E)是可配置的。而对于某一终止结点，若其有 2 个或多个输出边，则所有的输出边均是可配置的，例如：图 4.7（b）中，边(C, D)是可配置的；对于合并的 Artifact 生命周期图中的业务规则，若某条边来自于不同的原始图且被赋予了不同的业务规则集合，则该边在合并的 Artifact 生命周期图中所赋予的业务规则均是可配置的。例如：图 4.7（a）中，边(A, B)所赋予的业务规则$\{r_1, r_6\}$均是可配置的。

4.2.4 Artifact 生命周期模型的配置

基于可配置的 Artifact 生命周期图，我们可以定义 Artifact 生命周期图的配置，如定义 4.5 所示。

定义 4.5：（Configuration）对于一个可配置的 Artifact 生命周期图$C - ALG_C = (S_C, A_C, R_C, l_C, A^C, l^C, RS^C)$，它的配置则为一个映射$Conf$，对其中的可配置点赋予具体的值。对于$x \in A^C$，$x$可配置为 *selected* 或者 *blocked*，即$Conf(x) = selected$或$Conf(x) = blocked$；对于$y \in l^C$，y可配置为它本身的一个非空集合，即$Conf(y) = y'$，其中$y' \subseteq y$且$y' \neq \phi$。

对于 Artifact 生命周期图的配置，边的配置应该先做，因为若一条可配置的边被配置为 *blocked*，则该边所赋的业务规则可直接去掉，而无需配置。然后，再对剩下的可配置的业务规则进行配置，直接赋予可配置选择的具体值即可。因此，基于以上分析，本章提出 Artifact 生命周期图的配置算法，如算法 4.4 所示。

算法 4.4：Artifact 生命周期图的配置

输入：$C-ALG_C=(S_C,A_C,V_C,l_C,A^C,l^C,RS^C)$
输出：$ALG_m=(S_m,A_m,R_m,l_m)$
1: $\{i\}=\{n|n\in S_C\wedge|\bullet n|=0\}$; // the initial sate in $C-ALG_C$
2: **for** each arc $e\in A^C$ **do** // e is formalized as $e=(s_x,s_y)$
3: assign a value for e according to configuration requirements;
4: **if** $Conf(e)=$"$blocked$" **then**
5: $A^C=A^C-\{e\}$; $l^C=l^C-l_C(e)$;
6: $A_C=A_C-\{e\}$; $l_C=l_C-l_C(e)$; // remove e and $l_C(e)$ from $C-ALG_C$;
7: **while** $\exists n\in S_C\wedge\nexists p: i\rightarrow n$ **do**
8: **for** each outgoing e of n **do**
9: $A^C=A^C-\{e\}$; $l^C=l^C-l_C(e)$;
10: $A_C=A_C-\{e\}$; $l_C=l_C-l_C(e)$; // remove e and $l_C(e)$ from $C-ALG_C$;
11: **end for**
12: **end while**
13: **end if**
14: **end for**
15: **for** each element l_C(e) in l^C **do**
16: assign a value for $l_C(e)$ according to configuration requirements;
17: **end for**
18: **return** the remaining $C-ALG_C$;

算法中，第 1-14 行为 Artifact 生命周期图中可配置边的配置。如果有向边配置为 *blocked*，则分别在可配置的有向边集合和所有的有向边集合中去除这条边（5-6 行）。配置为 *blocked* 的边去掉后，对后续连接的有向边若不存在从开始状态到有向边起始状态的路径，则无论该有向边是否可配置均从集合中去除（7-12 行）。15-17 行为 Artifact 生命周期图中可配置业务规则的配置。算法 4.4 中的循环主要是针对可配置的有向边和可配置的业务规则集合，而内层循环基本可以忽略不计。因此，假设$C-ALG_C$中拥有N条有向边，则算法 4.4 的时间复杂度至多为$O(N)$，因为可配置的有向边或业务规则集合的数目总和不超过N。

基于算法 4.4 Artifact 生命周期图的配置处理过程，可得到定理 4.3。另外，有时候出于更多个性化的考虑，对已配置的生命周期图可根据具体需求做适当的修改，比如：重命名或者添加状态、边和业务规则。

定理 4.3：配置算法 4.4 的配置结果将产生结构化正确的 Artifact 生命周期图。

证明：当可配置的有向边配置为 *blocked* 时，则该边被去掉，并对后续连接的有向边若不存在从开始状态到有向边起始状态的路径，则无论该有向边是否可配置均从集合中去除，故有向边的配置将不会破坏 Artifact 生命周期图的结构化正确性属性。而对于可配置的业务规则集合仅允许配置为其非空的子集，故业务规则

的配置同样不会破坏 Artifact 生命周期图的结构化正确性属性。因此，配置结果得到的 Artifact 生命周期图依然是结构化正确的。

4.4 实证分析

基于前面提出的 Artifact 生命周期图的配置方法，我们给出例子来说明方法的实施及有效性。如图 4.8 为两个 Artifact 生命周期图的配置例子。在例子（a）中，边(C,D)和边(C,E)均被配置为$selected$，r_1配置为$removed$，但r_6配置为$selected$；在例子(b)中，边(C,D)被配置为$selected$，但边(C,E)被配置为$blocked$，r_1和r_6均被配置为$selected$。箭头右边为配置后得到的结果图，可看出它们均为结构化正确的 Artifact 生命周期图。

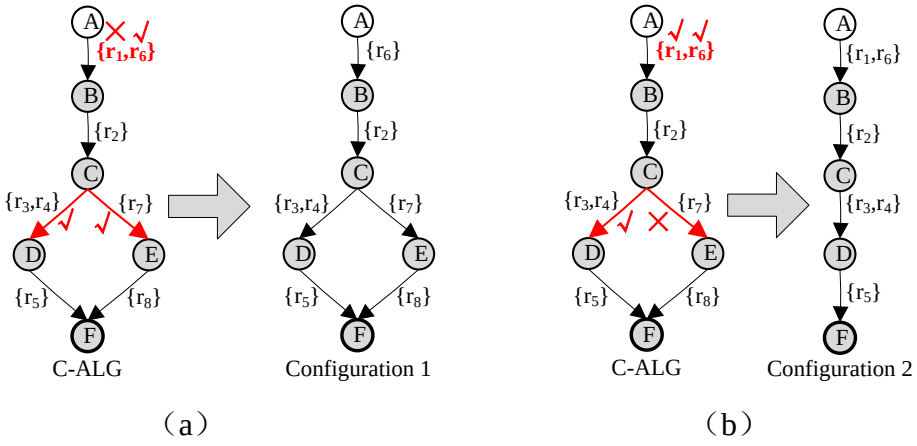


图 4.8 两个 Artifact 生命周期图的配置例子

这两个已配置的 Artifact 生命周期模型的相关信息可见表 4.1 所示，可知 Artifact、服务以及业务规则均已被具体化了。因此，Artifact-centric 流程模型的配置可以通过对流程中所有包含的 Artifact 生命周期模型的配置来实现。证实本章方法的有效性。

表 4.1 已配置的 Artifact 生命周期图

	Artifact, Services, Business rules
C-ALG	$C_m = (A_m, S_m),\ V_{C_m} = \{x \mid x = r.v \wedge r \in R_{C_m}\},\ R_{C_m}$
Configuration 1	$C_1 = (A_1, S_1),\ V_{C_1} = \{x \mid x = r.v \wedge r \in R_{C_1}\},\ R_{C_1} = \{r_2, r_3, r_4, r_5, r_6, r_7, r_8\}$
Configuration 2	$C_2 = (A_2, S_2),\ V_{C_2} = \{x \mid x = r.v \wedge r \in R_{C_2}\},\ R_{C_2} = \{r_1, r_2, r_3, r_4, r_5, r_6\}$

4.5 本章小结

本章针对 Artifact-centric 业务流程模型，提出可配置建模的方法。首先，采

用确定的有限状态机来形式化描述Artifact生命周期模型。进一步地，定义Artifact生命周期图来描述 Artifact 生命周期模型，并提出 Artifact 生命周期模型的合并以及产生可配置的 Artifact 生命周期模型的算法。最后，Artifact-centric 业务流程模型的配置通过流程中涉及的 Artifact 生命周期模型的配置来实现。该方法产生的可配置 Artifact-centric 业务流程模型是可逆的，可以通过指导规则记录合并元素的来源进而可配置得出原有的流程模型，同时本章提出的配置算法可产生结构化正确的 Artifact-centric 业务流程模型。因此，本章的可配置的 Artifact-centric 业务流程建模方法有利于提高业务流程模型的复用效率，提高业务流程建模的效率，并提高业务流程模型的质量。

本章 Artifact-centric 业务流程模型的可配置建模方法只考虑了 Artifact 生命周期的配置，这种情况在对于具有共通的同步约束的变体是没有问题的，然而对于不同变体存在不同的同步约束的情况，则本章的方法还需要增加同步方面的配置。另外，本章对 Artifact 的配置只涉及数据属性的配置，而没有对数据属性值的配置。因此，一个更完备的 Artifact-centric 业务流程模型可配置建模方法还需要考虑 Artifact 生命周期之间同步的配置和 Artifact 数据属性值的配置。同时，本章的 Artifact-centric 业务流程模型可配置建模将业务规则、数据属性、服务等元素看成一个整体元素来考虑，是在粗粒度的级别进行的，只适用于粗粒度的 Artifact 生命周期的概念模型，因此还需要一个在更加细粒度级别的 Artifact-centric 业务流程形式化模型可配置建模方法来完善流程的细节部分，该部分内容将在第 5 章中进行具体介绍。

第5章 Artifact-centric 业务流程形式化模型可配置建模框架

一项业务的流程模型伴随着应用的部署或演进将产生多个变体，建立可配置的业务流程模型是应对同一业务多处部署、未来业务推广不可预知的有效方法。可配置的业务流程模型也将通过重用设计而降低成本和缩短上市时间。然而，已有的业务流程模型可配置研究主要集中在传统的以活动为中心的业务流程，不能适用于新型的 Artifact-centric 业务流程建模范式。本文第 4 章中主要研究了单个 Artifact 生命周期概念模型的配置，是在粗粒度的级别进行的，为了给出一个更加完整的细粒度的可配置建模方法框架，本章提出 Artifact-centric 业务流程形式化模型可配置建模方法框架，考虑流程模型中所有的组成元素。为获得多个 Artifact-centric 业务流程模型变体的整合模型，提出 Artifact-centric 业务流程形式化模型的合并操作。根据模型元素的“可变”或“共同”特征，在整合模型中识别出可配置点，并为可配点设置相应的配置选项，最终得到可配置的 Artifact-centric 业务流程模型。基于可配置流程模型的行为选择配置选项，可配置出新的 Artifact-centric 业务流程模型。为了辅助业务流程模型的配置过程，基于流程模型元素关系图的概念提出了配置指南。通过保障性住房申请审批流程的真实案例分析，表明了本章方法的有效性。

5.1 引言

现今，企业的业务需求常常发生变化，使得业务流程模型的有效管理成为迫切需求。受“设计重用”思想的启发，可配置流程模型作为一个可定制的模型可表达相似流程模型的共同和可变部分，其获得了极大关注。近年来，Artifact-centric业务流程建模方法成为热点趋势。相比传统主要关注活动控制流的工作流模型，Artifact-centric方法建立在业务Artifact之上，结合了数据和流程两方面，为业务管理者提供了一个关于业务实体和操作的完整视图。目前，关于Artifact-centric业务流程的研究主要集中在形式化的描述语言与分析[61,129-131]、流程发现[106-108]等。然而，关于Artifact-centric业务流程可配置建模这一挑战问题还有待解决。为了提高Artifact-centric业务流程模型的建模效率和降低成本，Artifact-centric业务流程可配置建模成为了一项亟待解决的问题。

本章给出一个来自杭州房管局的一个保障房申请审批流程的例子，来说明可配置业务流程建模的现实需求。由于该业务需要在不同的城市进行部署，不同城市的不同组织机构及法律法规导致各自不同的业务需求，于是产生了不同的

流程模型变体，增加了建模的成本和时间开销，阻碍了该企业信息系统在不同城市的推广。

该业务是为特殊人群提供公租房申请审批服务，比如：低收入者、自然灾害受害者等。在申请提交（sub_App）后，申请表将经过几个审批阶段，即*dep_Rev*：部门审核、*res_Survey*：社区居委会调查、*str_Rev*：街道审批、*dis_Rev*：区局审核、*mun_Rev*：市局审核，最后备案（*archive*）。图5.1和图5.2为该公租房申请审批（*public housing application*，PHA）业务在杭州和衢州两个城市的Artifact-centric业务流程模型的简化描述。服务的执行和Artifact状态的变迁均与业务规则进行关联。两个业务流程模型非常相似，但有不同的信息模型、服务和业务规则集合。由于业务流程模型的不同，因此常常在业务推广过程中需从头建立新的业务流程模型，带来非常大的时间成本和人力成本。为提高流程建模效率和降低成本，Artifact-centric业务流程可配置建模成为业务推广过程中亟待解决的问题。

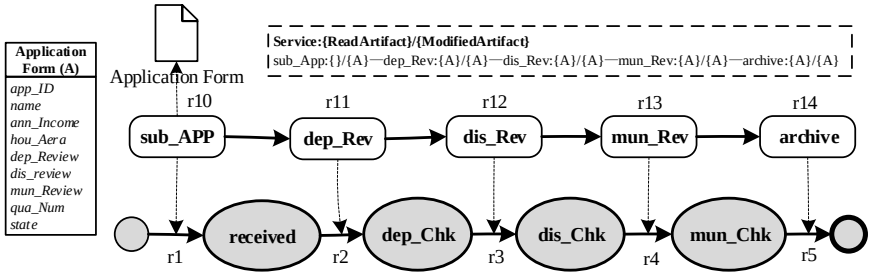


图 5.1 杭州的 PHA 业务流程

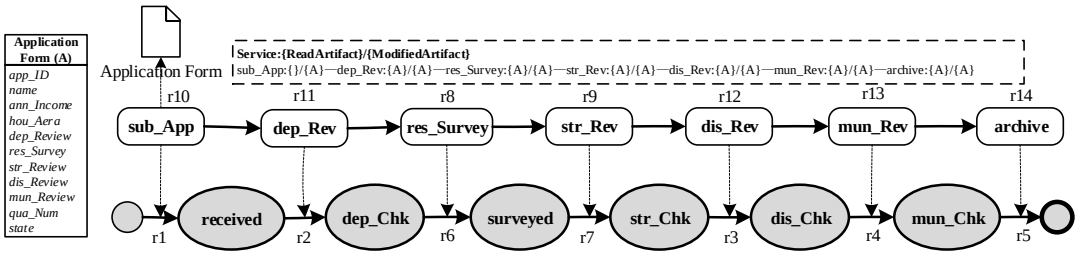


图 5.2 衢州的 PHA 业务流程

然而，已有的可配置流程模型研究主要集中在传统的以活动为中心的业务流程，不能适用于声明式的Artifact-centric业务流程建模范式。而且，在已往的方法中可配置流程模型元素的配置是互相独立的，除非它们在配置需求中被约束。然而，在Artifact-centric业务流程模型中，可配置的流程元素常常是互相关联的[132-135]，因此应对其进行分析并给出配置指南，以辅助业务建模人员的配置工作。

本文第4章中主要研究了单个Artifact生命周期概念模型的配置，是在粗粒度的级别进行的，而本章针对Artifact-centric业务流程形式化模型提出了一个完整

的细粒度的可配置建模方法框架，考虑流程模型中所有的各级别的组成元素。具体地，本章的主要贡献体现在以下四个方面：

（1）提出了Artifact-centric业务流程形式化模型变体的合并操作，从而产生关于多个流程模型变体的一个形式化整合描述；

（2）基于模型元素的“可变”或“共同”的特征，在整合模型中识别出可配置点，并为可配点设置相应的配置选项，最终得到可配置的Artifact-centric业务流程模型；

（3）为了使配置得到合适的流程模型，本章的方法允许提供模型配置的需求约束，并且提出流程元素关系图的概念为可配置流程模型的配置提供指南；

（4）基于杭州房管局保障房申请审批流程的真实案例，给出了配置过程的案例分析，证实了方法的有效性。

本章的剩余内容组织如下：第2节介绍Artifact-centric业务流程形式化模型；第3节提出Artifact-centric业务流程可配建模方法，包括Artifact-centric业务流程形式化模型变体的合并、可配置的Artifact-centric业务流程模型描述和附带配置指南的Artifact-centric业务流程形式化模型配置。第4节给出一个真实的案例分析，说明本章方法的有效性。第5节对本章研究工作进行总结并给出未来的研究工作。

5.2 Artifact-centric 业务流程形式化模型

本节引入文献[62]中提出的Artifact-centric业务流程形式化模型。基于该形式化模型，本章研究Artifact-centric业务流程的可配置建模。一个Artifact-centric业务流程模型包含三类组成构件：Artifact、服务和业务规则。因此，一个Artifact-centric业务流程形式化模型可模型化为一个三元组$M = (Z, V, R)$，其中Z为一个Artifact schema，V和R分别是建立在Z上的服务和业务规则的集合，更多细节的描述和模型语义可先参见文献[62]。注意，本章的形式化模型与2.2节的描述略有不同，具有更严格的形式化定义和描述。

5.2.1 Artifact 与 Schema

首先，假设存在以下不相交的无限集合：基本数据类型$\mathcal{T}_p$、Artifact class名$\mathbb{C}$、属性名$\mathcal{A}$、Artifact状态$\mathcal{S}$和Artifact class $C \in \mathbb{C}$的ID号$\boldsymbol{ID}_C$。一个类型是$\mathcal{T} = \mathcal{T}_p \cup \mathbb{C}$中的一个元素。$\mathcal{T}$中每个类型$t$的域表示为$D(t)$。

在Artifact-centric方法中，一个Artifact是业务流程中的一个关键业务实体。每个Artifact包含属性和状态的集合。每个属性具有一个名字和类型。基本类型

的取值采用标准的形式，比如：字符串、整型、浮点型、布尔型等。Artifact类型属性的取值为该类Artifact的一个ID号。在Artifact-centric业务流程中，同一类Artifact可能存在多个实例，因此定义Artifact class来描述同类Artifact实例的数据结构，如下：

定义5.1：（Artifact Class）一个Artifact class根据数据属性和状态抽象一组Artifact实例。一个Artifact class C表示为一个元组，即$C=(n,A,\tau,Q,s_0,F)$，其中$n\in\mathbb{C}$是一个Artifact类名，$A\subseteq\mathcal{A}$是数据属性的有限集合，$\tau:A\to\mathcal{T}$为一个全映射，$Q\subseteq\mathcal{S}$为状态的有限集合，$s_0\in Q$是初始状态，$F\subseteq Q$为终状态的集合。

一个 Artifact class C的对象或实例是一个元组，即$O=(id,\mu,q)$，其中$id\in\boldsymbol{ID}_C$是一个 ID 号；μ是一个部分映射，对属性进行赋值；$q\in Q$为当前的状态。当O处于初始状态时，$q=s_0$，且μ对于每一个数据属性来说均未定义；当O处于终状态时，$q\in F$。为了简便，在本章中 Artifact class 有时也简称为 Artifact。类似地，一个 Artifact-centric 业务流程可能涉及多个 Artifact 类，本章称流程中所有 Artifact class 构成的集合为 Artifact schema。由于 Artifact 之间可以互相引用，因此要求：（1）每个 Artifact class 有唯一的名字；（2）每个类中属性所引用的 Artifact 也存在于 Artifact schema 中（引用封闭性）。因此，得到 Artifact schema 的定义如下：

定义 5.2：（Artifact Schema）一个 Artifact schema Z是 Artifact class 的一个有限集合，这些 Artifact 具有唯一的名字，且每个 Artifact 所引用的 Artifact 均在Z当中，即$Z=\{C_1,C_2,\cdots,C_N\}$，其中$C_i(1\le i\le N)$为一个 Artifact class。

5.2.2 服务

一个服务是用于对Artifact进行读写操作的一个任务或软件模块，服务充当着组装业务模型的构件作用[136-138]。粗略地，一个服务可由输入变量、前置条件和后置效果来描述。这可视为OWL-S（Ontology Web Language for Services）语言描述的一个变种。在OWL-S中，服务有输入、输出、前置条件和后置效果。前置条件和后置效果可能引用输入和输出的内容。前置条件由调用服务的一组条件来描述。在本章的上下文中，后置效果主要是对Artifact进行写入新的值或改变其状态进行约束，因此不需再显式地描述服务的输出。

为了描述前置条件和后置效果，需定义Z上的项和原子公式。假定对于$\mathbb{C}$中的 Artifact class 存在不相交的无限的变量集合。一个$C\in\mathbb{C}$类型的变量持有$\boldsymbol{ID}_C$中的一个 ID 号。

定义 5.3：（项）Z上项的集合包含如下：（1）Z中一个类C的变量t；（2）$t.a$，

其中t是Z中的某个类C一个变量，a是C的一个属性。

定义 5.4：（原子公式）Z上的一个原子公式为如下形式中的某一种：（1）$t_1 = t_2$，其中t_1和t_2是Z中类C的项；（2）$defined(t,a)$，其中t是类C的一个项，a为C中的一个属性；（3）$new(t,a)$，其中t是类C的一个项，a为C中的一个 Artifact 类型的属性；（4）$instate(t,s)$，其中t是类C的一个项，s为C的一个状态。

一个原子公式c的否定形式记为 $\neg c$。Z上一个条件可用原子公式及其否定形式的合取表示。如果条件中不包含$instate$原子公式，则称其为无状态的。设I是Z的一个实例，则对I的一个赋值v是从变量到 ID 号一个映射。对于给定的实例I和对应的赋值v，所有的变量都赋予 Artifact 的 ID 号，于是条件的真值便可以自然地确定：（1）$defined(t,a)$为真，当且仅当 Artifact t中的属性a有值；（2）$new(t,a)$为真，当且仅当a持有一个不存在于I中的一个 ID 号；（3）$instate(t,s)$为真，当且仅当$q = s$，即 Artifact t当前处于s状态。对于实例I的一个赋值v，如果在v下I对于条件φ是可满足的，则记为$I \vDash \varphi[v]$。

在本章中，后置效果将从其如何影响 Artifact 属性成为被定义的或新的 Artifact 被创建，而不考虑属性从已定义到未定义的情况，这符合一般的真实情况。设$V = \{S_1, S_2, \cdots, S_z\}$为服务的集合，$T$是$Z$中类的变量集合。一个服务$S$可能对一个或多个 Artifact 的属性进行读或写操作。考虑服务是非确定性的，即服务的执行导致多个可能结果中的一个。假定存在一个无限的关于服务名的集合$\mathbb{S}$。基于以上分析，定义服务如下：

定义 5.5：（服务）一个Z上的服务S是一个元组$S = (n, S_r, S_w, P, E)$，其中$n \in \mathbb{S}$是一个服务名；S_r和S_w均是Z中类的变量的有限集合；P是T上一个无状态的条件；E为后置效果。

直观上，S_r和S_w分别是服务S读/写的 Artifact 集合，它们之间可能存在交集。注意，当$t \in S_r$，如果a是一个 Artifact 类型的属性，则$t.a.b$为一个项，用于引用赋值v下出现在S_r中之外的 Artifact 属性值，S_w中也会出现同样的情况，但这可通过语法的设置来限制这种情况的发生。

5.2.3 业务规则

基于 Artifact 和服务集合，一个业务模型可由业务规则进行控制。粗略地，业务规则描述了哪个服务在什么条件下作用在哪个 Artifact 之上。假定存在一个无限的关于业务规则名的集合$\mathbb{R}$。从技术上假定枚举出所有 Artifact 类型的变量。则对于服务S，采用记号$S(x_1, \cdots, x_\ell; y_1, \cdots, y_k)$来表示，其中$x_1, \cdots, x_\ell$为$S$写操作的

变量集合，而$y_1,\cdots,y_k$为S只读操作的变量集合。基于以上分析，定义业务规则如下：

定义 5.6：（业务规则）给定一个 Artifact schema Z和服务集合V，一个业务规则可形式化为一个元组$r=(n,con,action)$，其中$r\in\mathbb{R}$为业务规则名，con是执行该业务规则的条件，*action* 为调用服务或改变 Artifact 状态这一行为。因此，一个业务规则可表达为如下两种形式之一：

（1）**if** φ **invoke** $S(x_1,\cdots,x_\ell;y_1,\cdots,y_k)$；

（2）**if** φ **change state to** ψ。

定义 5.6 中，φ是对于变量$x_1,\cdots,x_\ell$和$y_1,\cdots,y_k$的一个条件，S是一个服务，$x_1,\cdots,x_\ell$为被写 Artifact 变量集合，$y_1,\cdots,y_k$为只读的 Artifact 变量集合，ψ为关于变量$x_1,\cdots,x_\ell$的一个条件。一般地，具有同样服务调用行为的多个业务规则被认为是同一个业务规则的多个变体。

5.3 Artifact-centric 业务流程可配置建模

可配置流程模型应包含同一业务的多个相似流程模型变体的行为或信息。基于这一观察，借鉴第4章的思路，本章将业务流程模型变体进行合并操作，形成一个整合的业务流程形式化模型描述。在整合的流程模型中识别可配置点，并为其设置配置选项，从而产生可配置的Artifact-centric业务流程形式化模型。最后，为了指导流程配置，基于流程模型元素之间的变化影响分析给出配置指南。因此，基于以上分析，本章的Artifact-centric业务流程形式化模型可配置建模方法框架如图5.3所示。

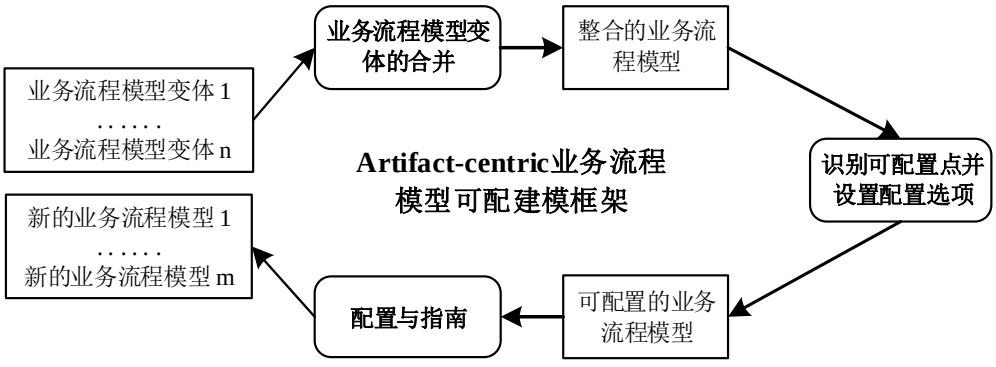


图 5.3 Artifact-centric 业务流程形式化模型可配置建模框架

5.3.1 Artifact-centric 业务流程模型变体的合并

对于一个业务，由于不同组织的不同需求，产生多个相似的Artifact-centric

业务流程模型变体。一个可配置的流程模型是相似流程模型变体的综合描述。因此，为了获得可配置流程模型，本章将流程模型变体合并起来形成整合的流程模型，从而可以在合并的流程模型中找出可变化点，即可配置点，这样得到可配置流程模型则包括了所有流程模型变体的行为。

设两个Artifact-centric业务流程模型$M_1 = (Z, V, R)$和$M_2 = (Z, V, R)$，它们是同一业务流程的两个相似流程模型变体，而$M = (Z, V, R)$为对M_1和M_2进行合并操作得到的Artifact-centric业务流程模型。为了进行合适的合并操作，本章给出如下**假设**：（1）一个Artifact的不同变体具有相同的名字和初始状态；（2）分别来自于同一个Artifact的两个变体的同一数据属性具有相同的属性名和类型，即它们在两个变体中是一样的；（3）一个服务的不同变体具有相同的名字；（4）一个业务规则的不同变体具有相同的名字；（5）一个Artifact-centric业务流程模型的不同变体采用同样的术语描述同一个概念。其中，第5个假设是为了简化本章的方法，否则需要对两个模型变体中的元素进行匹配映射的预处理。这里先定义两个映射的合并，再定义Artifact-centric业务流程模型的合并，因为形式化模型的合并要包含映射的合并。

定义 5.7：（映射的合并操作）设两个全映射分别为$f_1: A_1 \longrightarrow B_1$和$f_2: A_2 \longrightarrow B_2$，当$x_1 = x_2$时，$f_1(x_1) = f_2(x_2)$。则两个映射的合并定义为$f = f_1 \oplus f_2: A_1 \cup A_2 \longrightarrow B_1 \cup B_2$，其中

$$f(x) = f_1(x) \oplus f_2(x) = \begin{cases} f_1(x), & x \in A_1 \wedge x \notin A_2 \\ f_1(x) = f_2(x), & x \in A_1 \wedge x \in A_2 \\ f_2(x), & x \notin A_1 \wedge x \in A_2 \end{cases}$$

定义 5.8：（Artifact-centric 业务流程模型的合并操作）设两个相似的Artifact-centric 业务流程模型变体为$M_1 = (Z, V, R)$和$M_2 = (Z, V, R)$，则它们的合并操作可表示为：$M = M_1 \oplus M_2$，其中合并之后得到的流程模型为$M = (Z, V, R)$，该模型可根据算法 5.1 的合并过程得到。

算法 5.1 的时间复杂度主要由 Artifact 类的合并（3-11 行）、服务的合并（14-22 行）以及业务规则的合并（25-32 行）组成。任意两者在合并前，均需要对其命名进行判断是否相等，相等后再分别对其组成元素项进行合并操作。假设总的 Artifact 类、服务及业务规则的数目分别为N_1、N_2和N_3，则算法 5.1 总的时间复杂度为$O(N_1^2 + N_2^2 + N_3^2)$，因此算法时间复杂度的量级为$O(n^2)$。

定义5.8中，合并的流程模型包括了两个流程模型变体的所有信息，结果也是一个流程模型。同时，该合并操作还满足**交换律**和**结合律**，即有：$M_1 \oplus M_2 = M_2 \oplus M_1$和$(M_1 \oplus M_2) \oplus M_3 = M_1 \oplus (M_2 \oplus M_3)$。基于以上分析，可得出定理5.1，该证明可

参见定理4.1、4.2。因此，n个流程模型变体的合并结果与合并的操作顺序无关，当出现一个新的流程模型变体时，可将其合并到已有的合并结果当中。

定理5.1： $< M, \oplus >$是一个满足封闭性的代数系统，且该代数系统为一个半群，其中M为某个Artifact-centric业务流程模型变体的集合。

算法 5.1：Artifact-centric 业务流程模型变体的合并

输入：$M_1 = (Z, V, R)$， $M_2 = (Z, V, R)$ // 两个 Artifact-centric 业务流程模型变体
输出：$M = M_1 \oplus M_2$ // 合并的 Artifact-centric 业务流程模型
1: $M.Z = \emptyset$; $M.V = \emptyset$; $M.R = \emptyset$;
2: $Z_1 = M_1.Z$; $Z_2 = M_2.Z$; // Artifact class 的合并 (行 2-12)
3: **for** each C_i in $M_1.Z$ **do**
4: **for** each C_j in $M_2.Z$ **do**
5: **if** $C_i.n == C_j.n$ **then** // $C = (n, A, \tau, Q, s_0, F)$
6: $C_k.n = C_i.n$; $C_k.s_0 = C_i.s_0$; $C_k.\tau = C_i.\tau \oplus C_j.\tau$;
7: $C_k.A = C_i.A \cup C_j.A$; $C_k.Q = C_i.Q \cup C_j.Q$; $C_k.F = C_i.F \cup C_j.F$;
8: add C_k into $M.Z$; $Z_1 = Z_1 - \{C_i\}$; $Z_2 = Z_2 - \{C_j\}$; // 集合的差
9: **end if**
10: **end for**
11: **end for**
12: $M.Z = M.Z \cup Z_1 \cup Z_2$;
13: $V_1 = M_1.V$; $V_2 = M_2.V$ // 服务的合并(行 13-23)
14: **for** each S_i in $M_1.V$ **do**
15: **for** each S_j in $M_2.V$ **do**
16: **if** $S_i.n == S_j.n$ **then** // $S = (n, S_r, S_w, P, E)$
17: $S_k.n = S_i.n$; $S_k.S_r = S_i.S_r \cup S_j.S_r$; $S_k.S_w = S_i.S_w \cup S_j.S_w$;
18: $S_k.P = S_i.P \wedge S_j.P$; $S_k.E = S_i.E \wedge S_j.E$; // 合取操作
19: add S_k into $M.V$; $V_1 = V_1 - \{S_i\}$; $V_2 = V_2 - \{S_j\}$; // 集合的差
20: **end if**
21: **end for**
22: **end for**
23: $M.V = M.V \cup V_1 \cup V_2$;
24: $R_1 = M_1.R$; $R_2 = M_2.R$; // 业务规则的合并(行 24-33)
25: **for** each r_i in $M_1.R$ **do**
26: **for** each r_j in $M_2.R$ **do**
27: **if** $r_i.n == r_j.n$ **then** // $r = (n, con, action)$
28: $r_k.n = r_i.n$; $r_k.action = r_i.action$; $r_k.con = r_i.con \wedge r_j.con$; //合取操作
29: add r_k into $M.R$; $R_1 = R_1 - \{r_i\}$; $R_2 = R_2 - \{r_j\}$; //集合的差
30: **end if**
31: **end for**
32: **end for**
33: $M.R = M.R \cup R_1 \cup R_2$;
34: **return** $M = (Z, V, R)$;

5.3.2 可配置的 Artifact-centric 业务流程模型

为了获得完整的Artifact-centric业务流程可配置模型，首先在整合的流程模型中识别出可配点（即可配置元素）。正如第2节中提到的，一个Artifact-centric业务流程模型由三类构件构成：Artifact、服务和业务规则。一个Artifact由属性和状态组成，一个服务有读的Artifact集合、写的Artifact集合、前置条件和后置效果，一个业务规则由条件和行为组成。因此，这些构成元素在结果的合并流程模型中应被考虑是否是可配置的。

假设存在N个 Artifact-centric 业务流程模型变体$M_1,\cdots,M_N$，它们的合并模型为$M = M_1 \oplus M_2 \oplus \cdots \oplus M_N$。基于流程模型中的元素是**共同**的还是**可变**的特征，本章提出**可配置规则**来鉴别M中的可配置点，如下：（1）一个 Artifact（服务或业务规则）是作为一个整体可配置的，当且仅当它不包含在所有的流程模型变体中；（2）一个 Artifact 的一个属性或状态是可配置的，当且仅当它不包含在该 Artifact 的所有变体中；（3）一个服务所读或写的 Artifact 集合中的一个 Artifact 在该集合中是可配置的，当且仅当它不包含在该服务的所有变体所读或写的 Artifact 集合中；（4）一个服务的前置条件或后置效果中的一个原子条件是可配置的，当且仅当它不包含在该服务的所有变体的前置条件或后置效果中；（5）业务规则的条件中的一个原子条件是可配置的，当且仅当它不包含在该业务规则的所有变体的条件中。

以上可配置原则的依据是：变体$M_1,\cdots,M_N$可由最终的可配置流程模型通过配置得到，即它们应该是结果的可配置流程模型的部分实例[48]。本质上，可配置的总原则就是：一个模型的元素是可配置的，当且仅当它不包含在该模型的所有变体中。需要注意的是，一个Artifact作为一个整体是否是可配置的和其属性或状态是否是可配置的没有关系，即一个Artifact作为一个整体是可配置的但其属性或状态可能不是可配置的，反之亦然。因此，当说一个构件元素是可配置时，要么它作为一个整体是可配置的，要么其组成元素是可配置的。

一个可配置点应有其对应的配置选项。在本章中，一个可配置点可以配置为 *enabled* 或 *disenabled*，即一个可配置元素在配置中要么被选择，要么被移除。配置选择可用配置需求来约束。一个配置需求可用逻辑表达式来描述，比如：$t_1 = enabled \Rightarrow t_2 = disenabled$，其中$t_1$和$t_2$是两个可配置点。同时，为了避免配置冲突，可以用偏序关系来表达可配置点应遵循的配置顺序。此外，除了配置需求，配置指南对于指导配置过程也是需要的。配置指南同样可以用逻辑表达式来描述。可以把配置需求看作是硬性的约束，而把配置指南看作软性的约束。硬性的约束是配置过程中必须遵行的，而软性的约束则只是作为建议指导。

为了区分可配置的元素和不可配置的元素，将它们在合并的形式化流程模型中分开，从而得到可配置的Artifact-centric业务流程形式化模型，形式化定义如下：

定义 5.9：（可配置的 Artifact-centric 业务流程模型）一个可配置的Artifact-centric 业务流程模型可形式化定义为这样一个元组：$M^C = (Z \cup Z^C, V \cup V^C, R \cup R^C, \mathbb{O}^C, \mathbb{R}^C, \mathbb{G}^C)$，其中$Z \cup Z^C$是 Artifact schema，$S \cup S^C$和$R \cup R^C$分别是$Z \cup Z^C$上的服务和业务规则。$X^C$（$X$为某类模型元素）表示对应的可配置元素的集合，$X^C$可以是空集。具体地，$Z^C$中的一个 Artifact 可形式化为$C = (n, A \cup A^C, \tau, Q \cup Q^C, s_0, F)$；$V^C$中一个服务可形式化描述为$S = (n, S_r \cup {S_r}^C, S_w \cup {S_w}^C, P \cup P^C, E \cup E^C)$；$R^C$中的一个业务规则可形式化描述为$r = (n, con \cup con^C, action)$。$\mathbb{O}^C \subseteq X^C \times Y^C$（$X^C$和$Y^C$分别是可配置元素的集合）是一个偏序关系，表示可配置点的配置顺序约束，例如：(x, y)表示x的配制应先于y的配置。$\mathbb{R}^C$是配置需求的集合，$\mathbb{G}^C$表示配置指南，它们均可用逻辑表达式来描述。

配置顺序和配置需求是配置约束，它们需要领域知识进行定义。然而，对于M^C的可配置顺序$\mathbb{O}^C$，本章提出一般的**配置顺序规则**：

（1）对于$C^C \in Z^C$，$X^C \neq \emptyset$（其中，$X \in \{A, Q\}$），则有$(C^C, X^C) \in \mathbb{O}^C$；

（2）对于$S^C \in V^C$，$X^C \neq \emptyset$（其中，$X \in \{S_r, S_w, P, E\}$），则有$(S^C, X^C) \in \mathbb{O}^C$；

（3）对于$r^C \in R^C$，$con^C \neq \emptyset$，则有$(r^C, con^C) \in \mathbb{O}^C$。

以上配置顺序规则可应用于所有的可配置 Artifact-centric 业务流程模型，然而其它流程相关的配置顺序可根据领域知识进行添加。该配置顺序规则的依据是：如果一个流程模型元素及其组成元素均是可配置的，则该元素的配置应先于其组成元素的配置，因为当一个模型元素被移除时，它的组成元素也将直接被移除而不需要配置。

5.3.3 配置与指南

根据可配置的Artifact-centric业务流程模型，业务建模人员很容易通过配置得到新的Artifact-centric业务流程模型。实际上，配置就是对可配点选择具体的配置选项。因此，本章定义可配置的Artifact-centric业务流程模型的配置如下：

定义 5.10：（配置）设一个可配置的 Artifact-centric 业务流程形式化模型为$M^C = (Z \cup Z^C, V \cup V^C, R \cup R^C, \mathbb{O}^C, \mathbb{R}^C, \mathbb{G}^C)$，该模型的一个配置是从可配置点到具体配置选项的一个函数映射：$l^C \in (X^C \to \{enabled, disenabled\})$，其中$X^C$为可配点的集合。

在配置中，配置为 *enabled* 的可配置元素直接被选择保留，而配置为 *disenabled* 可配置元素直接被移除。然而，在可配置的 Artifact-centric 业务流程模型中，一个可配置模型元素的具体配置可能对其它可配置点的配置产生直接或间接的影响。因此，一个可配置元素的配置应先于被其影响的可配置元素的配置。为了计算一个可配置元素在配置中对其它可配置元素的直接或间接影响范围，本章提出**流程模型元素关系图**的概念。图中的结点为流程模型元素，包括构件元素及其组成元素，比如：属性、服务等。影响范围的计算是基于元素之间关联关系。根据第 2 节中的 Artifact-centric 业务流程模型的形式化定义，图 5.4 展示了一个一般的流程元素关系图。该流程元素关系图形成的依据如下：

（1）Artifact 由属性和状态组成，服务具有读 Artifact 集合、写 Artifact 集合、前置条件和后置效果；

（2）一个 Artifact 的一个属性可能在服务的前置条件或后置效果中被使用；

（3）一个 Artifact 可能被一个服务读或写；

（4）一个业务规则的行为可能调用一个服务或改变一个 Artifact 的状态；

（5）一个 Artifact 的属性或状态可能在业务规则的条件中被使用。

因此，这些关联关系（即**组成关系**和**被使用关系**）就形成了流程元素关系图中的有向边，如图 5.4 所示。加粗的有向边表示组成的关系，细的有向边表示被使用的关系。从该元素关系图中可看出，一个元素的变化可能对其它元素产生直接或间接的影响，但业务规则组成元素（即 condition 和 action）的变化对其它元素没有影响，因其没有到其它元素的有向边。把结点A到结点B的连接记为：$A \rightarrow B$，表示A的变化（即配置）对B有直接的影响。那么，在配置中A的存在应该先于B的存在，即A的配置应先于B的配置。

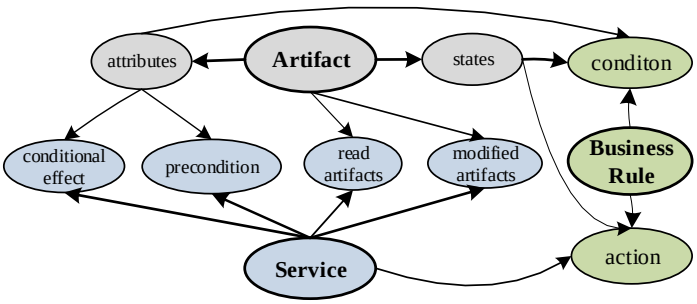


图 5.4 Artifact-centric 业务流程模型元素关系图

类似地，根据一个具体的可配置的 Artifact-centric 业务流程模型，可对应产生具体的流程元素关系图。由于配置中变化的元素只涉及可配置的元素，因此

在产生流程模型元素关系图时可只考虑可配置的元素。依据得到的流程元素关系图，当对M^C中的一个可配置元素进行配置时，可定位其直接影响范围，即该元素结点所连接到的元素结点；同样，可定位影响该可配置元素的元素，即连接到该元素的所有元素。基于具体的流程元素关系图，本章提出以下**配置指南**：

（1）如果一个可配置元素被配置为 *disenabled*，则它的组成元素直接被移除，而不管它们是否是可配置的；

（2）如果有$A \to B$，则建议A的配置先于B的配置；

（3）对于$A \to B$，若A被配置成 *disenabled*，则B也必须被配置成 *disenabled*；

（4）对于$A \to B$，若B被配置成 *enabled*，则A也必须被配置成 *enabled*。

这些配置指南对于一般的建模人员来说是特别有帮助的，因他们缺乏对应业务流程相关的领域知识。同时，为去除配置中多余的元素或信息，针对如下两种情况，本章提出自动化的去除操作：（1）如果一个 Artifact 的可配置属性被配置为 *disenabled*，则与之相关的映射也需要从 Artifact 的描述中自动去除；（2）如果一个 Artifact 的可配置的终状态被配置为 *disenabled*，则将其从 Artifact 描述中的终状态集合中自动去除。

基于配置顺序、配置需求和配置指南，新的 Artifact-centric 业务流程模型可以容易地通过对可配置的 Artifact-centric 业务流程模型进行配置产生。该配置的方法建模业务流程模型时，只需要进行简单的配置操作。显然，相比从零开始建模业务流程模型或从业务流程模型变体进行改编，配置的方法可大大地改进流程建模的效率。配置完成之后，根据具体的业务需求也可能需要对配置得到的业务流程模型进行后期处理，比如改变或添加某个流程模型元素，使得配置得到的业务流程模型更加个性化。这些后期工作的完成由业务建模人员确定，属于配置之外的工作。最后，得到的 Artifact-centric 业务流程模型还需要进行验证的工作，这将在本文的第 6 章中提到。

5.4 案例分析

根据本章的Artifact-centric业务流程可配置建模方法，我们将引言中的两个PHA流程模型合并，并找出可配置点，得到形式化的Artifact-centric业务流程可配置模型。由于该形式化模型对应的图形化表示仍然是一个待研究的问题，而已有的图形化表示方法主要集中在Artifact生命周期的表示，属于概念模型，对于本章形式化模型的内容不能完整地进行描述。因此，本节采用软件产品线中

的特征模型[139,140]来图形化描述Artifact-centric业务流程可配置模型中的可配置模型元素，而忽略其不可配置的元素，从而降低图形的复杂度。

图5.5中的特征模型组成了结果的可配置Artifact-centric业务流程模型中的可配置模型元素，其中可配置元素用矩形框表示，它们是配置操作的对象。这些可配置元素对应被合并的流程模型变体中的非共同模型元素，称之为可变元素。当可配置点的数量相对不多的时候，这种表示方法是可行的，因为得到的特征模型相对简单。根据图5.5，业务建模人员容易对可配置模型元素进行配置。相应地，图5.6展示了该可配置流程模型中的可配置元素构成的流程模型元素关系图。其中，灰色表示Artifct及其组成元素，蓝色表示服务及其组成元素，绿色表示业务规则及其组成元素，可配置的流程元素用虚线的圆圈表示。根据图5.6，可以得到可配置的模型元素之间的影响关系，从而指导可配置流程模型的配置过程。图5.7为可配置Artifact-centric业务流程模型对应的一个可能的配置，该配置满足图5.6描述的配置指南。图5.7的配置结果对应的业务流程模型的简化描述如图5.8所示。类似地，用户可以根据自己的具体需求得到其它的配置，从而建立适应其新环境的流程模型。从该配置可以看出，基于配置的流程建模方法简单、有效，能够提高流程建模的效率。

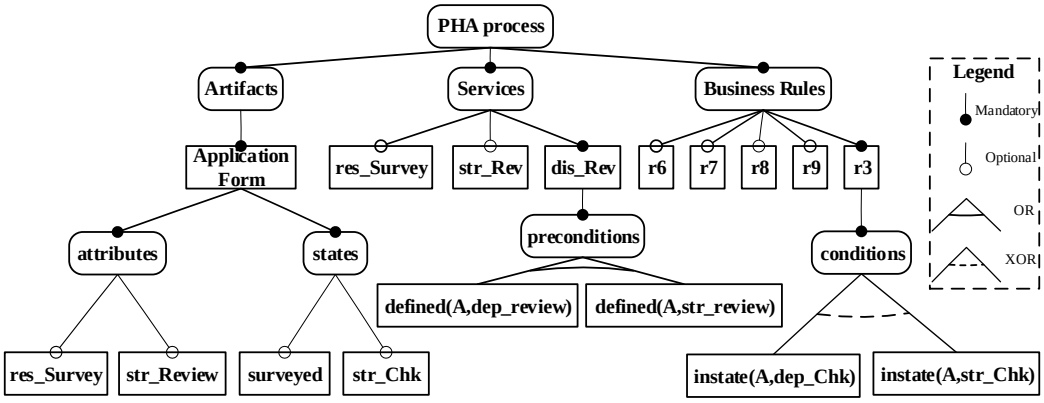


图 5.5 特征模型方法描述的可配置流程元素

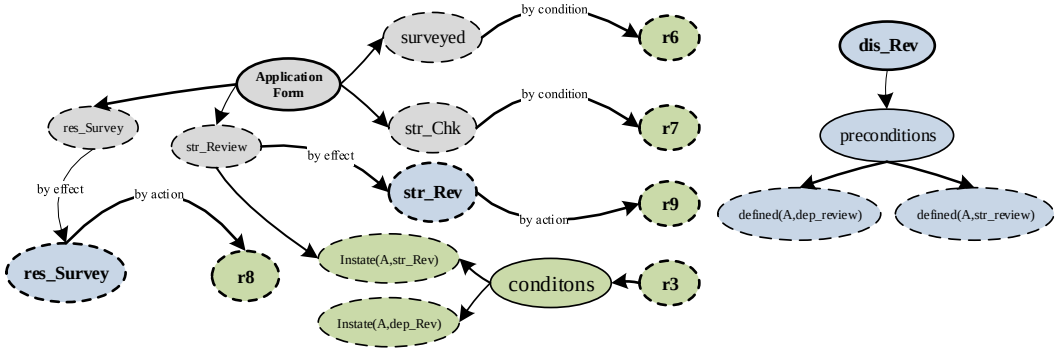


图 5.6 可配置 Artifact-centric 业务流程模型对应的可配置流程元素关系图

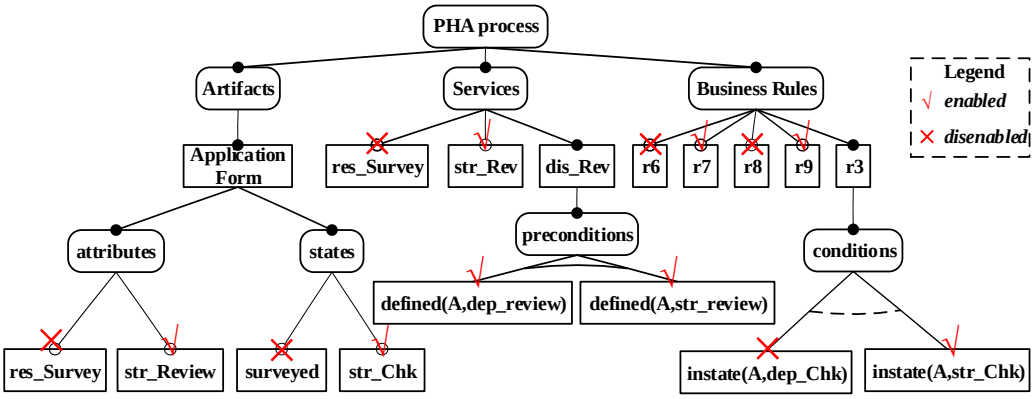


图 5.7 可配置 Artifact-centric 业务流程模型对应的一个可能的配置

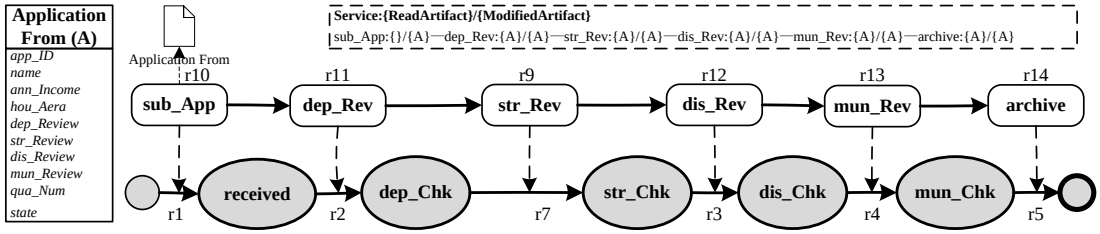


图 5.8 图 5.7 配置结果的业务流程模型对应的简化描述

5.5 本章小结

本章提出了一个可配置的Artifact-centric业务流程建模方法框架。为了获得整合的业务流程模型，提出了Artifact-centric业务流程模型变体的合并操作。进一步地，根据可配置的原则，识别出可配置元素，可配置点的配置选项为*enabled*和*disenabled*。于是，得到了可配置的Artifact-centric业务流程模型。在可配置的Artifact-centric业务流程模型中，可设置配置顺序和配置需求。为了更有效地指导流程配置，可分析具体的可配置的Artifact-centric业务流程模型的可配置元素关系图，得到配置指南。本章的研究工作可促进Artifact-centric业务流程模型的复用，并使得Artifact-centric业务流程建模更加快速和有效。然而，为了确保建模结果的正确性需要对配置建模的结果模型进行行为正确性验证，该部分内容将在第6章中进行具体介绍。

本章的Artifact-centric业务流程模型可配置建模方法是建立在形式化模型之上，具有通用性。为使该方法更加具有实用性，可配置的Artifact-centric业务流程模型的图形化表示和配置的软件实现是下一步研究需要解决的问题。另外，本章将配置视为业务流程设计时的行为约束，在未来工作中我们将研究Artifact-centric业务流程模型在运行时的可变性管理。

第6章 Artifact-centric 业务流程模型同步的行为正确性验证

近十多年来提出的 Artifact-centric 业务流程建模范式隐含地将控制流描述在业务规则当中，所以在处理动态和跨组织的流程方面具备优势。在 Artifact-centric 方法中，将业务流程建模为涉及的 Artifact 生命周期的交互，交互使得流程的执行变得复杂，因为 Artifact 生命周期之间可能存在各种各样的同步，这就为一个基本的正确性验证问题带来了挑战，即行为正确性问题。第 4、5 章探讨了如何通过配置的方式进行 Artifact-centric 业务流程建模，为了保证业务流程模型的正确性在流程模型设计完成之后部署之前需要对流程模型的正确性属性进行验证。本章针对带复杂同步条件的 Artifact-centric 业务流程模型的行为正确性验证这一问题提出解决方案。首先，将流程中涉及的每个 Artifact 生命周期分别映射为 Petri Net 的等价描述。然后，基于同步约束提出规则将这些 Petri Net 合成描述为一个集中的 Workflow Net。基于得到的 Workflow Net，计算其可达图以获得原 Artifact-centric 流程模型描述的所有可能的服务（即活动）执行序列。最后，行为的正确性（即正常完成）的验证通过验证每个可能的服务执行序列是否在控制流和数据流两方面均正常完成来实现。一个真实的案例分析展示了本章提出的方法的有效性。

6.1 引言

在 BPM 中，业务流程模型传达了业务意图，且在业务中作为多个利益相关者之间互相交流的基础。业务流程模型的正确性在 BPM 中起到非常重要的作用，因为不正确的流程模型会导致业务的错误决策以及不符合要求的系统实现[17]。在一个 BPM 实践报告中显示在建立的流程模型库中有 20%是错误的[18]，所以需要对建模流程的正确性进行验证。

在传统以活动为中心的业务流程管理中，可基于 Petri Net 分析技术及其工具对各种业务流程建模语言（比如：Workflow Net[68]、EPC[44]、YAWL[70]、BPMN[71]）描述的流程模型关于正确性方面的不同属性进行验证。这些方法的共同特点是检查流程模型正常完成（即行为正确性）的不同方面。在这些工作中，通常需要执行三个步骤：（1）将流程模型转换成 Petri Net 描述的 Workflow Net；（2）创建该 Workflow Net 的可达图；（3）使用可达图检查流程模型控制流的正确性。因此，如果该 Workflow Net 是正确的，那么原始的流程模型也是正确的，反之亦然[69]。

最近一些年，提出了一个结合控制流和数据流描述的 Artifact-centric 业务流程建模方法，其中流程建模为 Artifact 生命周期的交互[35]。在诸多的案例分析中均记述了该方法的优点[67]。然而，目前在流程验证方面已有的研究工作主要集中在 Artifact-centric 业务流程执行的形式化分析[61-63,78,79]，其中业务流程的正确性依赖于数据库的完整性约束。这些工作主要讨论形式化验证的可判定性和复杂性结果。然而，它们缺少考虑 Artifact-centric 业务流程模型的行为正确性，尤其是多个 Artifact 生命周期之间存在复杂同步的情况。在 Artifact-centric 方法中，控制流隐含在业务规则中。另外，在一个现实的流程中可能涉及多个 Artifact。例如：一个在线的购物流程就涉及 *Order*，*Shipment* 和 *Invoice* 三个 Artifact 类，这些 Artifact 的行为依照同步约束互相制约。Artifact 生命周期之间一般存在一些同步约束，因此带来复杂的流程执行。这些特征使得 Artifact-centric 业务流程模型描述的隐含服务执行序列难以确定，给流程模型的行为正确性验证带来挑战。

本章针对同步约束的 Artifact-centric 业务流程模型的行为正确性问题提出其验证方法以检查流程模型的质量问题，比如：控制流方面的死锁、数据流方面的数据缺失。具体地，本章的主要贡献体现在以下三个方面：

（1）形式化定义了同步 Artifact 生命周期的概念，并定义了 Artifact-centric 业务流程模型行为正确性的概念；

（2）提出了带各种同步约束的 Artifact-centric 业务流程模型的行为正确性的验证方法；

（3）给出了一个案例分析证实了方法的有效性。

本章的剩余内容组织如下：第 2 节定义了同步 Artifact 生命周期的概念。第 3 节定义了 Artifact-centric 业务流程模型行为正确性的概念，并提出了带各种同步约束的 Artifact-centric 业务流程模型的行为正确性的验证方法；第 4 节展示了一个案例分析；第 5 节总结本章工作。

6.2 同步 Artifact 生命周期

在一个 Artifact-centric 业务流程中，Artifact 生命周期之间通过同步互相联结。图 6.1 显示了一个带各种同步约束的 Artifact-centric 业务流程模型的例子，其中忽略了其数据模型。

图 6.1 中带箭头的虚线边表示同步，其中双向的同步边表示变迁的同步，即分别来自两个 Artifact 生命周期中的两个 Artifact 变迁之间的同步；单向的同步

边表示状态的同步，即分别来自两个 Artifact 生命周期中的两个 Artifact 状态之间的同步。为了简单起见，在该例子中 Artifact 生命周期的变迁边上均只附带了一个业务规则，一般情况下变迁边上可以附带一个具有任意有限多个业务规则的集合。本章要求 Artifact 生命周期之间不存在循环的同步，这是 Artifact-centric 业务流程模型结构化正确的一个前提条件。在该例子中有四个 Artifact，即 A、B、C 和 D，其中 A 是关键 Artifact，或称为主 Artifact，它是由外部环境触发而创建的，而 B、C 和 D 则是在流程执行过程中被创建的。在该例子中，有 4 个变迁同步和 2 个状态同步。

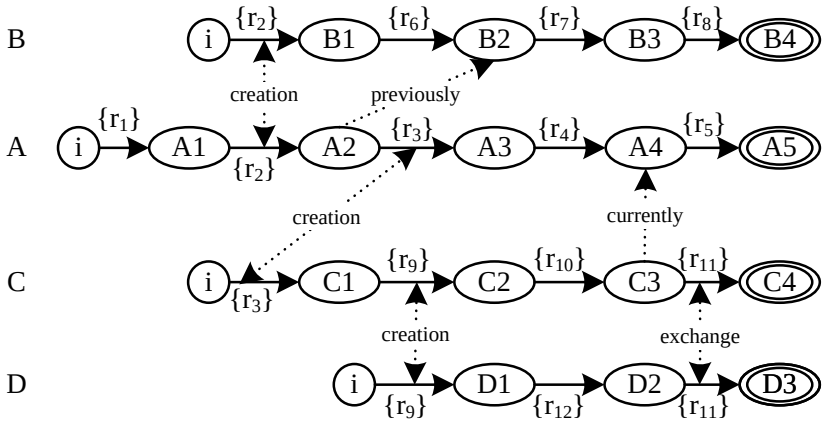


图 6.1 A、B、C、D 四个 Artifact 类的生命周期之间同步的例子

为了描述和处理这些 Artifact 生命周期之间的互相依赖，本章引入同步 Artifact 生命周期的概念，一个同步通过定义同步边来描述，正如定义 6.1 所示。

定义 6.1：（Synchronization Edge）同步边$se = (src, tgt, dep)$由源src、目标tgt，以及依赖类型dep组成，其中dep连接两个 Artifact 生命周期以同步它们之间的状态变迁。dep要么连接来自两个 Artifact 生命周期的两个变迁（即变迁同步）要么连接其中的两个状态（即状态同步），保证连接变迁的共同执行或定义 Artifact 状态变迁的前提条件。

具体地，变迁同步边形式化为$\mathrm{se}_T = (t_1, t_2, dep)$，其中$t_1 \in T_{C_1}$，$t_2 \in T_{C_2}$且$C_1 \neq C_2$，$t_1$为源变迁，$t_2$为目标变迁，$dep \in \{creation, exchange\}$描述了变迁之间的依赖类型，即“创建”和“交换”。而状态同步可形式化为$\mathrm{se}_S = (s_1, s_2, dep)$，其中$s_1 \in C_1.S$，$s_2 \in C_2.S$且$C_1 \neq C_2$，$s_1$为源 Artifact 状态，$s_2$为目标 Artifact 状态，$dep \in \{currently, previously\}$描述了 Artifact 状态之间的依赖类型，即“当前”和“之前”。图形化地，一个变迁和状态同步边分别表示为双向和单向的虚线边，正如图 6.1 所示。

两个 Artifact 变迁由同步边连接合并在一起，因此它们一起执行，这个属性是可传递的。针对流程模型，这意味着流程调用一个服务但同时改变两多或多个 Artifact 的状态。对于变迁同步，如果依赖类型是$creation$，则在源变迁src的输入满足的情况下服务执行，然后一个与目标变迁tgt关联的新的 Artifact 实例被创建；如果依赖类型是$exchange$，则两个 Artifact 在执行该服务时交换信息。对于状态同步，如果依赖类型是$currently$，意味着若要触发转向目标状态的变迁，则源src相关的 Artifact 生命周期必须正处于src状态；如果依赖类型是$previously$，意味着若要触发转向目标状态的变迁，则源src相关的 Artifact 生命周期必须在过去某个时间处于src状态。若L表示流程中的 Artifact 生命周期模型集合，SE表示所有涉及的同步边集合，则L和SE可一同组成带同步约束的 Artifact-centric 业务流程模型，即同步 Artifact 生命周期的概念。

6.3 Artifact-centric 业务流程模型的行为正确性

一个流程模型的行为可以由所有可能的任务或活动执行序列的集合来描述[2]。在传统以活动为中心的流程模型中，行为的正确性定义为所有可能的活动执行序列的正常完成。如果所有的活动执行序列完成时均到达理想的终止标记，则流程模型在控制流方面是行为正确的。在 Artifact-centric 业务流程模型中，特定的控制流隐含在业务规则中，同时被 Artifact 生命周期所约束，其中的数据流又被视为与控制流具有同等重要的地位。而 Artifact 生命周期的变迁通过触发相应的业务规则并调用服务来实现。因此，Artifact-centric 业务流程模型的行为过程涉及多个方面。在此，本章提出包含控制流和数据流两方面的 Artifact-centric 业务流程模型的行为正确性的概念，正如定义 6.2 所示。基于定义 6.2，则容易得到定理 6.1。

定义 6.2：（Behavioral Soundness of APM）对于隐含描述在 Artifact-centric 业务流程模型$\Pi = (Z, V, R)$中的任一服务执行序列，如果它使得所有 Artifact 生命周期均到达它们的终止状态（即控制流方面），并且在流程执行过程中每个服务的输入都是依次满足的（即数据流方面），则该 Artifact-centric 业务流程模型是行为正确的。

定理 6.1：一个 Artifact-centric 业务流程模型是行为正确的，当且仅当它在控制流和数据流两方面均是行为正确的。

控制流和数据流之间存在着同步的交互关系，任一个行为不正确都会导致流程行为的不正确。控制流强调对活动的流转，而数据流强调对数据的变化。只有

保证活动使能，才能对数据进行操作，因此流程的行为正确性首先需要保证控制流的行为正确性。当然，可以同时验证控制流和数据流，但由于控制流的行为正确性是数据流的行为正确性的前提，因此本章先验证控制流的行为正确性，这样可以简化验证过程。当控制流行为不正确的情况下，就不必再验证数据流的行为正确性，而直接给出流程行为不正确的验证结果。

验证 APM 的行为正确性面临着两大挑战。第一是如何识别出流程中所有隐含描述且满足同步约束的服务执行序列，并确定它们是否均能使得所有的 Artifact 生命周期均到达它们的终止状态。第二是如何衡量一个服务执行序列中服务的输入需求在实际流程执行中是否依次满足以确定其是否可正常完成。

本章将服务数据的具体输入需求抽象为其存在性而不是其实际的取值，既然实际值的可满足性是在业务规则中保证的。本章采用 Petri Net 表示 Artifact 生命周期模型[141]，并根据同步约束将它们合成为一个集中描述的 Workflow Net。然后，创建该 Workflow Net 的可达图以获得所有隐含描述且满足同步约束的服务执行序列。控制流的行为正确性可通过 Petri Net 分析技术来验证[142-146]，因为转换后的控制流语义遵循 Petri Net 的语义[147]。最后，数据流的行为正确性可通过检查服务执行序列中服务的数据输入需求是否依次满足来确定其是否可正常完成。

6.3.1 控制流的行为正确性

本节首先将流程中涉及的 Artifact 生命周期模型分别映射为 Petri Net，然后基于同步约束提出合成规则将这些 Petri Net 合成为一个集中描述的 Workflow Net。最后，创建该 Workflow Net 的可达图以获得所有流程描述的服务执行序列。因此，控制流的行为正确性可通过验证其服务执行序列的正常完成来实现。

首先，将流程中涉及的 Artifact 生命周期模型分别映射为 Petri Net。单个 Artifact 生命周期模型可向 Petri Net 进行映射是基于这样一个事实：当每个变迁只有一个前序和一个后序库所时，Petri Net 本身就是一个状态机[68,148,149]，因此可用 Petri Net **等价**描述一个自动机，即保证语义的**一致性**。基于等价的语义，本章提出以下的映射规则：

（1）Artifact 生命周期的每个状态映射为 Petri Net 中的库所；

（2）Artifact 生命周期的每个 Artifact 状态变迁由调用相应的服务来实现，每个变迁实现对应 Petri Net 中的一个变迁连接前后的两个库所，即状态。

图 6.2 是将一个 Artifact 生命周期模型映射为一个 Petri Net 的例子。从图中可以看出，每个业务规则均映射为 Petri Net 中的一个变迁，其中$r_i.v$是与业务规则r_i关联的服务，既然每个业务规则对应执行状态变迁的一个服务，因此信息的映射是一一对应的，保证信息的映射是**无损的**，即保证信息的**完整性**。

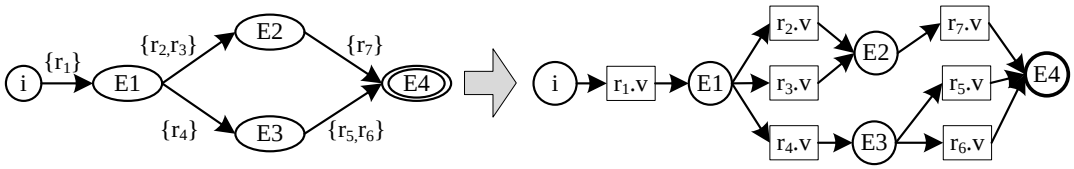


图 6.2 一个 Artifact 生命周期模型映射为一个 Petri Net 的例子

注意，在映射过程中，Artifact 生命周期之间的同步边仍然保留在映射后的 Petri Net 中，比如：图 6.1 中多个 Artifact 生命周期之间存在同步，当它们均映射为 Petri Net 之后，同步边依然保持不变，因此保证信息的**完整性**。如图 6.3 所示，其中业务规则转换成对应的服务作为变迁上附带的标记。

然后，根据同步边集合合成映射后的 Petri Net。为了达到这一目的，基于同步的语义本章提出如下 5 个合成规则，也正如图 6.4 中的图形化描述所示。

规则 1：对于$creation$类型的变迁同步边，将两个变迁（即源变迁和目标变迁）合并为一个变迁。同时，删除创建 Artifact 的初始状态及其连接边；

规则 2：对于$exchange$类型的变迁同步边，简单地合并两个变迁（即源变迁和目标变迁）为一个变迁即可；

规则 3：对于$currently$类型的状态同步边，两个新的库所p_1、p_2增加到 Petri Net 中，其中一个库所（即p_1）的前序变迁集合由代表源状态库所的前序变迁集合组成，且该库所的后序变迁集合由代表目标状态库所的前序变迁集合组成。而另一个库所（即p_2）的前序变迁集合由代表目标状态库所的前序变迁集合组成，且该库所的后序变迁集合由代表源状态库所的后序变迁集合组成。特殊地，如果源状态为一个终状态，则只需要增加第一个库所即可；

规则 4：对于$previously$类型的状态同步边，一个新的库所p_1增加到 Petri Net 中。该库所的前序变迁集合由代表源状态库所的后序变迁集合组成，且该库所的后序变迁集合由代表目标状态库所的前序变迁集合组成；

规则 5：最后，增加一个新的终库所p_f和一个新的变迁t_f（即终变迁），并连接每个 Petri Net 的终库所到新增的变迁，以及连接新增的变迁到新增的终库所。

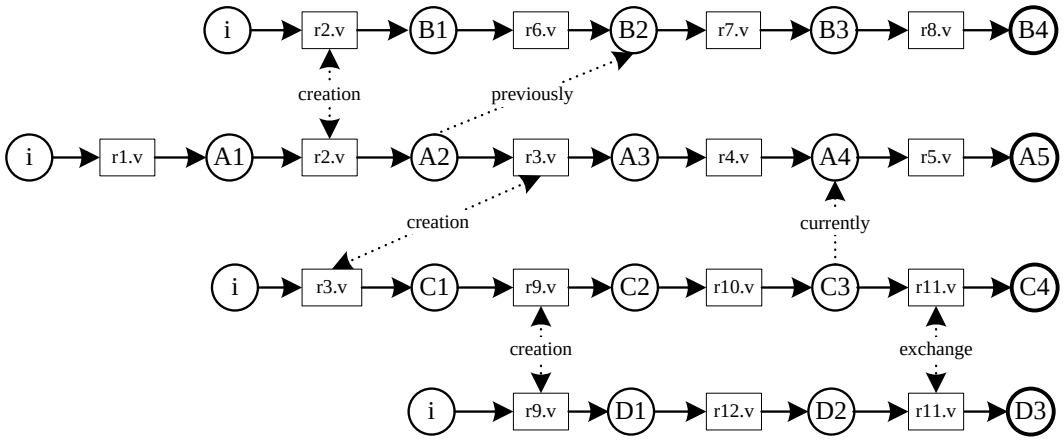


图 6.3 图 6.1 中的同步 Artifact 生命周期模型转化后的 Petri Net 结果

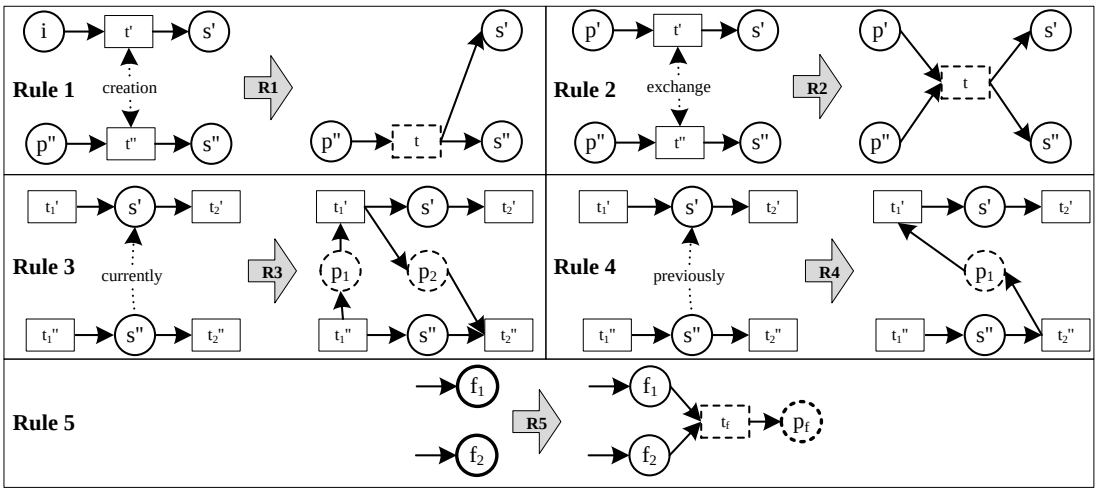


图 6.4 合成规则

在图 6.4 中，虚线的矩形框和圆圈分别代表新增的变迁和库所，其中变迁t是变迁t'和变迁t''的合并。对所有 Artifact 生命周期模型转化后的 Petri Net 结果，应用以上的合并规则后，则得到一个集中描述的 Workflow Net，其中有且仅有一个开始库所和结束库所。该 Workflow Net 不会改变原有 Artifact-centric 业务流程模型的行为，因为它们在行为上是**等价**的，因此对该 Workflow Net 的行为正确性验证即是对 Artifact-centric 业务流程模型的行为正确性验证。Workflow Net 常用于流程信息系统中的一致性检查，因其有良好的形式化语义。在此，本章要求 Artifact-centric 业务流程模型是结构化正确的，其中结构化正确性的概念可参见文献[16]，因此在 Petri Net 的合成中不会而引入循环的同步。图 6.3 中的 Petri Net 合成的 Workflow Net 结果如图 6.5 所示，其中为了处理 3 个$creation$类型的变迁同步（**规则 1**）和 1 个$exchange$类型的变迁同步（**规则 2**），4 对变迁分别被合并了；库所p_1，p_2和p_3是新增的，用于处理状态同步（**规则 3** 和**规则 4**）；t_f和p_f是新

增的变迁和库所，用于将多个终库所合并为一个终库所（**规则 5**），使得结果的 Workflow Net 满足其结构化的正确性而不改变其行为语义。

根据 Workflow Net，基于 Petri Net 分析工具创建其可达图RG（Reachability Graph），因此可得到流程模型中所有描述的服务执行序列。在RG中，可能有多个终状态，然而只有p_f是合法的。这里，本章省略图 6.5 的可达图，在案例分析中将看到一个真实流程例子的可达图。事实上，从集中的 Workflow Net 的特征可知，一般真实案例中服务执行序列的数量是很有限的，这在一些真实的案例分析中均有显示，该结论可参见文献[150]。因此，在现实流程中计算可达图时，不会出现状态爆炸问题。基于RG和服务执行序列，可得到如下关于 APM 控制流方面的行为正确性定理。

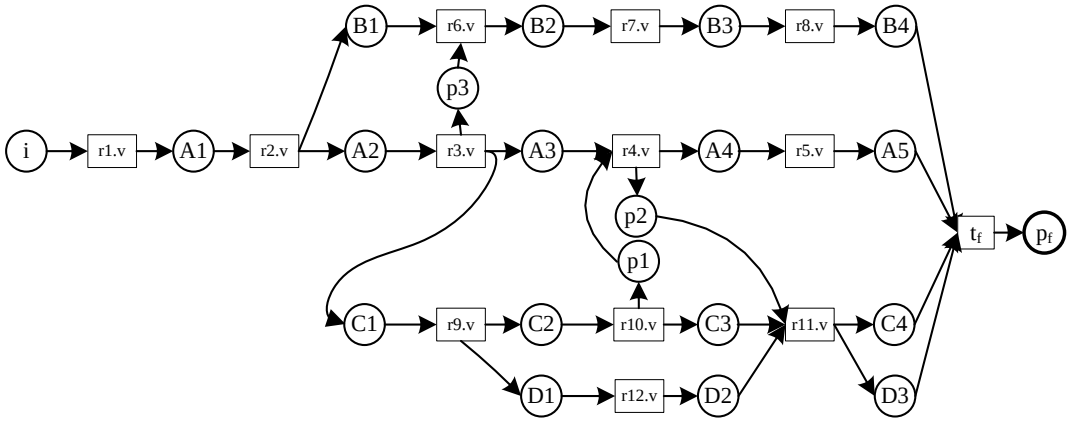


图 6.5 合并图 6.3 中的 Petri Net 得到的 Workflow Net

定理 6.2：在一个 Artifact-centric 业务流程模型$\Pi = (Z, V, R)$中，对于其可达图中的任一服务执行序列，如果它都是以终变迁t_f结束的，则该 APM 在控制流方面是行为正确的。

证明：一个服务执行序列以终变迁t_f结束，意味着它是一条从初始状态到理想终状态p_f的路径，当中没有遇到死锁，即它执行完毕后使得所有的 Artifact 生命周期均到达了它们的终状态，因为 Workflow Net 中的终状态p_f是汇集了所有 Artifact 生命周期的终状态之后的状态，因此它的执行完成是正常的。如果对流程中所有描述的服务执行序列均满足正常执行完成的特性，则自然该 APM 在控制流方面是行为正确的。

6.3.2 数据流的行为正确性

为了判定关于数据流方面服务执行序列的执行过程是否可以正常完成，必须了解服务被调用的条件。在 Artifact 的操作过程中，Artifact 状态的变迁是通过调

用与业务规则关联的服务来实现。在本章中，简化为当输入数据准备好时，则服务可调用，然后提供输出。本章不考虑服务的可用性和可靠性，该问题超出了本章的研究内容。因此本章假定在流程执行过程中业务规则关联的服务总是可用的，当业务规则的前置条件为真且对应服务的输入数据准备好，则该服务可成功地被调用并满足其后置效果。

服务输入的数据源来自于流程中的 Artifact。因此，本章为 Artifact-centric 业务流程定义 Artifact 的赋值，如定义 6.3 所示。服务调用之后，Artifact 的赋值被更新。Artifact-centric 业务流程模型整体的动态行为必须考虑所有涉及的 Artifact 行为。APM 的状态由当前涉及的 Artifact 实例状态集合组成，正如定义 6.4 所示。基于涉及的 Artifact 赋值和 APM 的状态，本章提出 APM 的变迁规则，正如定义 6.5 所示。

定义 6.3：（Assignment of Artifacts）对一个 Artifact-centric 业务流程模型 $\Pi = (Z, V, R)$，单个 Artifact 的赋值定义为该 Artifact 中所有带值的数据属性集合，而流程的 Artifact 赋值AS定义为当前所有涉及的 Artifact 中带值的数据属性集合，即由流程中所有涉及的单个 Artifact 的赋值组成。

定义 6.4：（State of APM）对一个 Artifact-centric 业务流程模型$\Pi = (Z, V, R)$，APM 的流程状态PS定义为 APM 执行时当前所有 Artifact 状态的集合，其中 Artifact 状态分别记录在其数据模型中的属性$state$中。

定义 6.4：（Transition Rule of APM）对一个 Artifact-centric 业务流程模型 $\Pi = (Z, V, R)$，当前 Artifact 的赋值为AS，流程状态为PS。如果当前业务规则r_i的前置条件满足，且对应服务的输入数据均存在，即$r_i.v.r \subseteq AS$，则服务$r_i.v$可被调用，即变迁被触发。变迁完成后，AS更新到一个新的 Artifact 的赋值 $AS' = AS \cup r_i.v.w$，并且 APM 更新到一个新的流程状态PS'，其中包含所有 Artifact 的状态且至少有一个 Artifact 的数据属性被变迁更新。

在 Artifact-centric 业务流程中，一个服务的输入信息可能来源于多个 Artifact。关联的 Artifact 赋值应该一同满足服务的输入需求以实现变迁。否则，该服务必须等待所有的输入信息准备完毕，这种情况尤其在 Artifact 之间存在同步时可能会发生。依据流程的 Artifact 赋值定义，它包含了涉及的 Artifact 中所有带值的数据属性。因此，当服务的输入信息包含在流程的 Artifact 赋值中时，说明服务的输入数据已就绪，服务可被调用。在此处忽略了的业务规则前置条件的判断，因为本章数据流正确性的验证主要强调数据的存在性方面。

基于 APM 的初始状态、流程的 Artifact 赋值以及变迁规则，对给定的来源于可达图RG的服务执行序列$< v_1, v_2, \cdots, v_k >$，先假定它在控制流方面是行为正确的，我们通过检查在服务的依次执行过程中所有服务的输入数据是否依次可满足被调用来验证服务执行序列的正常完成特性。本章提出验证算法，如算法 6.1 所示，其中$v_i.r$为服务v_i的输入涉及的 Artifact 数据属性集合（第 5 行），$v_i.w$为输出涉及的 Artifact 数据属性集合（第 8 行）。对于终变迁t_f，输入和输出均为空集。通过对可达图中的每个服务执行序列应用算法 6.1，根据流程设计中描述的服务输入和输出数据项，在服务的依次执行过程中不断更新流程的 Artifact 赋值和流程的状态，可判定一个具体的服务执行序列是否可正常完成（4-13 行）。算法的时间复杂度主要由循环判断各服务的输入满足性及更新流程的状态（即$state$）与 Artifact 赋值（即$assignment$）组成。假设服务执行序列中有N个服务，则 4-13 行需循环执行N次，因此算法 6.1 的时间复杂度为$O(N)$。

算法 6.1 验证数据流方面服务执行序列的正常完成特性

输入：a service execution sequence $< v_1, v_2, \cdots, v_k >$ from the reachability graph, the initial process state PS of APM, and the assignment of Artifacts AS
输出：$True$ or $False$ // whether $< v_1, v_2, \cdots, v_k >$ completes properly or not

```
1: state = PS;   // the initial process state of APM
2: assignment = AS;   // the assignment of Artifacts
3: decision = True;
4: for v_1 to v_k do   // v_i is associated with r_i in line 5
5:     if r_i.α == True and v_i.r ⊆ assignment then
6:         update the states of Artifacts associated with v_i;
7:         state = update(state);   // update the process state of APM
8:         assignment = assignment ∪ v_i.w;   // update the assignment of Artifacts
9:     else
10:        decision = False;
11:        break;
12:    end if
13: end for
14: return decision;
```

迭代应用算法 6.1，可循坏地判定是否所有的服务执行序列均可正常完成。因此，可得出关于数据流方面流程模型行为正确性的定理，如定理 6.3 所示。

定理 6.3：在一个控制流方面行为正确的 Artifact-centric 业务流程模型$\Pi = (Z, V, R)$中，对于其映射产生可达图中的任意一条服务执行序列，应用算法 6.1 验证其数据流方面的正常完成特性，如果返回的结果均为真，则该 APM 在数据流方面是行为正确的。

基于 6.3.1 和 6.3.2 节的讨论，可从控制流和数据两方面分别验证 Artifact-centric 业务流程模型的行为正确性。一个 Artifact-centric 业务流程模型是行为正确的，意味着它在控制流和数据两方面均是行为正确的。这种分析在解决设计阶段模型的不一致性是很重要的。这样的质量保证可降低后续再去修正流程模型的代价，并提高信息系统实现时流程模型的可用性。

6.4 案例分析

本节案例分析中的 Artifact-centric 业务流程模型例子改编于一个在线的购物流程[36]，其中增加了各样的同步类型，如图 6.6 所示。这类 Artifact-centric 业务流程模型的行为正确性不能被已有的研究方法所验证，既然它们仅针对单个 Artifact 实例且忽略了多个 Artifact 实例之间的同步交互。该流程当客户通过网站提交包含账单信息的订单时开启。然后，订单被发送到制造工厂，在那里订购的产品被装备、测试和打包。最后，产品被快递运输到客户。图 6.6 的左边部分显示了该 Artifact-centric 业务流程模型涉及的 Artifact，即 *Order*（主 Artifact），*Shipment* 和 *Invoice*。

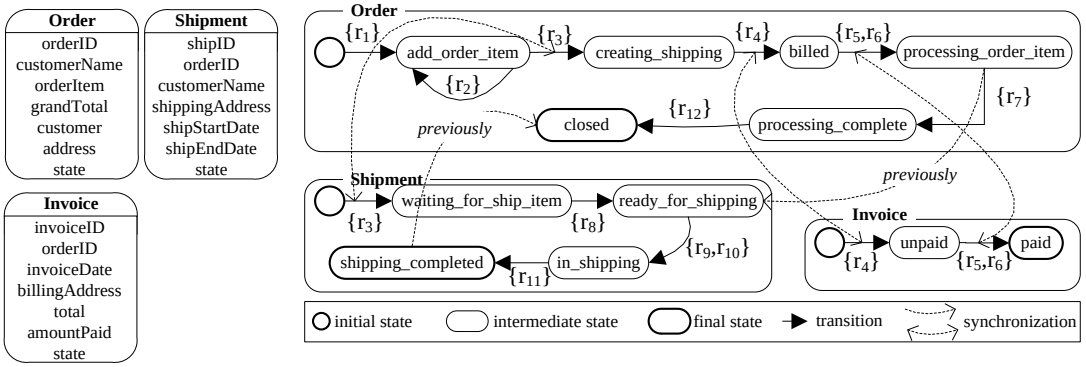


图 6.6 Artifact-centric 业务流程模型案例

Artifact 生命周期图形化表示为状态图。该案例中 Artifact 生命周期之间有 3 个变迁同步和 2 个状态同步，如下所示：

$$se_T^1 = (t_1, t_2, creation)$$

$$se_T^2 = (t_3, t_4, creation)$$

$$se_T^3 = (t_5, t_6, exchange)$$

$$se_S^1 = (processing\_order\_item, ready\_for\_shipping, previously)$$

$$se_S^2 = (shipping\_completed, closed, previously)$$

其中，

$$t_1 = (add\_order\_item, r_3, creating\_shipping)$$

$$t_2 = (init, r_3, waiting\_for\_ship\_item)$$

$$t_3 = (creating\_shipping, r_4, billed)$$

$$t_4 = (init, r_4, unpaid)$$

$$t_5 = (billed, r_5/r_6, processing\_order\_item)$$

$$t_6 = (unpaid, r_5/r_6, paid)。$$

一旦 *Order* 被创建，其状态从 *init* 变到 *add_order_item*。在这个状态，客户可向订单中循环地添加产品。当客户完成添加产品，则生命周期从状态 *add_order_item* 变到状态 *creating_shipping*。这个时候，*Shipment* 被创建。然后，客户提交账单，*Order* 的状态就转到 *billed* 状态。客户付款成功之后，订单就送到工厂处理。这一阶段描述为 *processing_order_item* 和 *processing_complete* 两个状态。*ready_for_shippin*g、*in_shipping*、*shipping_completed* 三个状态描述了产品的递送阶段。*closed* 状态表示整个流程的完成。

表 6.1 展示了流程中涉及的所有业务规则的描述。以业务规则r_1为例，前置条件是 *Order* o处于 *init* 状态，且它的数据属性 *orderID*、*customerName*、*customerAddress* 均未定义。当前置条件满足时，服务 *createOrder* 被调用，且 Artifact o被更新。之后，后置条件需要满足，即 Artifact o从状态 *init* 变到状态 *add_order_item*，且其数据属性 *orderID*，*customerName* 和 *customerAddress* 被定义同时被赋值。

表 6.1 Artifact-centric 业务流程模型案例涉及的业务规则

r_1: Customer requests to make an *Order* o	
Pre-condition	$instate\ (o, init) \wedge \neg defined\ (o.orderID) \wedge \neg defined\ (o.customerName) \wedge \neg defined\ (o.customerAddress)$
Service	$createOrder\ (o)$
Post-condition	$instate\ (o, add\_order\_item) \wedge defined\ (o.orderID) \wedge defined\ (o.customerName) \wedge defined\ (o.customerAddress)$
r_2: Add items to *Order* o	
Pre-condition	$instate\ (o, add\_order\_item) \wedge (\neg defined(o.grandTotal) \vee (defined(o.grandTotal) \wedge o.grandTotal > 0)) \wedge defined\ (o.orderID) \wedge defined\ (o.customerName) \wedge defined\ (o.customerAddress)$
Service	$addItem\ (o)$
Post-condition	$instate\ (o, add\_order\_item) \wedge defined(o.grandTotal) \wedge o.grandTotal > 0$
r_3: Create *Shipment* s for *Order* o	
Pre-condition	$instate\ (o, add\_order\_item) \wedge instate(s, init) \wedge defined(o.grandTotal) \wedge o.grandTotal > 0 \wedge \neg defined(s.customerName) \wedge \neg defined(s.shippingAddress) \wedge \neg defined(s.shipID) \wedge \neg defined(s.orderID)$
Service	$createShipping(s, o)$

Post-condition	$instate(o, creating\_shipping) \wedge instate(s, waiting\_for\_ship\_item)$ $\wedge defined(s.customerName)$ $\wedge defined(s.shippingAddress) \wedge defined(s.shipID)$ $\wedge defined(s.orderID)$
r_4: Create *Invoice* i for *Order* o	
Pre-condition	$Instate(i, init) \wedge instate(o, creating\_shipping) \wedge \neg defined(i.invoiceID)$ $\wedge \neg defined(i.orderID) \wedge \neg defined(i.billingAddress)$ $\wedge \neg defined(i.invoiceDate) \wedge \neg defined(i.total)$
Service	$createInvoice(i, o)$
Post-condition	$instate(i, unpaid) \wedge instate(o, billed) \wedge defined(i.invoiceID)$ $\wedge defined(i.orderID) \wedge defined(i.billingAddress)$ $\wedge defined(i.invoiceDate) \wedge defined(i.total) \wedge i.total$ $= o.grandTotal$
r_5: Pay for *Order* o	
Pre-condition	$instate(o, billed) \wedge instate(i, unpaid) \wedge defined(i.invoiceID)$ $\wedge defined(i.orderID) \wedge defined(i.billingAddress)$ $\wedge defined(i.invoiceDate) \wedge defined(i.total) \wedge i.total$ $= o.grandTotal \wedge \neg defined(i.amoundPaid)$
Service	$WeChatPay(i, o)$
Post-condition	$instate(o, processing\_order\_item) \wedge instate(i, paid) \wedge defined(i.total)$ $\wedge i.total = o.grandTotal \wedge defined(i.amoundPaid)$
r_6: Pay for *Order* o	
Pre-condition	$instate(o, billed) \wedge instate(i, unpaid) \wedge defined(i.invoiceID)$ $\wedge defined(i.orderID) \wedge defined(i.billingAddress)$ $\wedge defined(i.invoiceDate) \wedge defined(i.total) \wedge i.total$ $= o.grandTotal \wedge \neg defined(i.amoundPaid)$
Service	$Alipay(i, o)$
Post-condition	$instate(o, processing\_order\_item) \wedge instate(i, paid) \wedge defined(i.total)$ $\wedge i.total = o.grandTotal \wedge defined(i.amoundPaid)$
r_7: Prepare ordered items for *Order* o	
Pre-condition	$instate(o, processing\_order\_item) \wedge instate(i, paid)$ $\wedge defined(s.shipStartDate)$
Service	$prepareItems(o)$
Post-condition	$instate(o, processing\_complete)$
r_8: Pack ordered items for *Order* o	
Pre-condition	$instate(o, creating\_shipping) \wedge instate(i, paid)$ $\wedge instate(s, waiting\_for\_ship\_item)$ $\wedge defined(s.customerName)$ $\wedge defined(s.shippingAddress) \wedge defined(s.shipID)$ $\wedge defined(s.orderID)$
Service	$packItems(s)$
Post-condition	$instate(s, ready\_for\_shipping)$
r_9: Load ordered items for *Order* o	

Pre-condition	$instate(s, ready\_for\_shipping) \wedge \neg defined(s.shipStartDate)$
Service	$sf\_Express(s)$
Post-condition	$instate(s, in\_shipping) \wedge defined(s.shipStartDate)$
r_{10}: Load ordered items for *Order* o	
Pre-condition	$instate(s, ready\_for\_shipping) \wedge \neg defined(s.shipStartDate)$
Service	$EMS\_Express(s)$
Post-condition	$instate(s, in\_shipping) \wedge defined(s.shipStartDate)$
r_{11}: Ship items for *Order* o	
Pre-condition	$instate(s, in\_shipping) \wedge defined(s.shipStartDate) \wedge \neg defined(s.shipEndDate)$
Service	$shipItem(s)$
Post-condition	$instate(s, shipping\_complete) \wedge defined(s.shipEndDate)$
r_{12}: Close order for *Order* o	
Pre-condition	$instate(o, processing\_complete) \wedge instate(i, paid) \wedge instate(s, in\_shipping) \wedge defined(s.shipEndDate)$
Service	$closeOrder(o)$
Post-condition	$instate(o, closed)$

基于第 3 节中介绍的验证方法，我们研究图 6.6 中 Artifact-centric 业务流程模型的行为正确性。该流程模型导出的 Workflow Net 如图 6.7 所示，为方便作图其中状态的名字用简单的符号所代替。当计算该 Workflow Net 的可达图时，本章限制其循环的次数为 1 次以避免状态的组合爆炸。尽管如此，这不会影响其验证结果。因此，结果导出的可达图如图 6.8 所示。

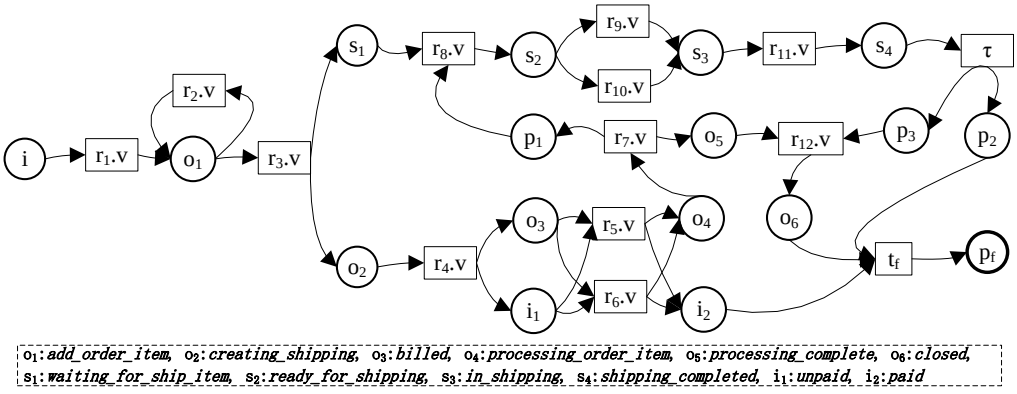


图 6.7 图 6.6 中 Artifact-centric 业务流程模型转化得到的 Workflow Net

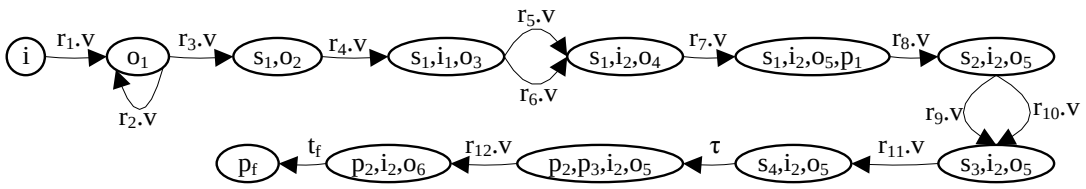


图 6.8 图 6.7 中 Workflow Net 导出的可达图

通过使用本章的验证方法，可以判定图 6.6 的 Artifact-centric 业务流程模型在控制流上是行为正确的，服务执行序列均以t_f结束，但它在数据流上不是行为正确的，因为$r_7.v$的输入不能被满足。具体地，在该模型中，由于存在$previously$类型的同步约束，$r_7.v$的执行必须在$r_8.v$之前，但是$r_7.\alpha$要求$s.shipStartDate$已定义，然而$s.shipStartDate$需要在完成$r_8.v$之后再执行$r_9.v$或$r_{10}.v$时才定义且赋值。这类情况不能被已有的研究工作所验证，已有工作会认为该流程模型是行为正确的，因为它们只关注单个 Artifact 生命周期，而忽略来自其它 Artifact 的数据需求。因此，为了保证流程模型的行为正确性，本章对该流程模型给出以下两个修正建议：

（1） 从$r_7.\alpha$中删除条件$defined(s.shipStartDate)$；

（2） 将其中的一个状态同步$se_S = (s_{poi}, s_{rfs}, previously)$修改为：$se_S = (s_{poi}, s_{is}, currently)$，其中涉及的状态分别为：$s_{poi} = processing\_order\_item$，$s_{rfs} = ready\_for\_shipping$，$s_{is} = in\_shipping$。

然而，考虑到在现实的业务场景中，产品应先在工厂生产、组装，然后打包，再快递运输。因此，第二个建议不合适现实的业务场景，应该采纳第一个建议。因此，严格上来说，流程模型的正确性其实不只是考虑模型的结构和语义上的正确性、行为的正常完成，还必须考虑对具体业务场景的合理性。这是与具体的业务相关的，需要对业务知识有非常清楚的了解。

在实际的流程执行中，一个服务执行序列应该与 Workflow Net 或可达图的描述保持一致。因为合法的服务执行序列应在流程图中被描述。否则，该服务执行序列可能违反了 Artifact 生命周约束或同步约束。比如：一个服务执行序列$< r_1.v, r_2.v, r_3.v, r_8.v, r_4.v, r_5.v, r_7.v, r_9.v, r_{11}.v, r_{12}.v >$，它不是行为正确的，因$r_8.v$在$r_7.v$之前执行，即它与流程的可达图描述不一致。然而，这种情况无法被已有的工作所验证，因为它们忽略了对 Artifact 生命周之间的同步约束的验证。

6.5 本章小结

本章提出了 Artifact-centric 业务流程模型同步的行为正确性问题及其验证方法。本章应用 Petri Net 来表示 Artifact 生命周期模型，并基于同步约束将它们合成为集中描述的 Workflow Net。通过计算该 Workflow Net 对应的可达图，可识别出流程中隐含描述的服务执行序列。因此，Artifact-centric 业务流程模型的行为正确性可通过检查每个服务执行序列是否正常完成来验证。本章的方法为 Artifact-centric 业务流程模型的行为正确性问题从控制流和数据流两方面提供一

个完整性的验证。针对 6.3.1 节和 6.3.2 节中介绍的属性违反问题，可以识别出是控制流还是数据流方面的问题，因此可以验证行为正确性问题。

本章 Artifact-centric 业务流程模型同步的行为正确性验证方法集中在验证流程是否存在控制流和数据流方面的问题。然而，问题检测出来之后，流程模型该如何修正属于下一步的工作，未来可以考虑添加领域知识来指导模型的修正。另外，我们计划在下一步工作中研究同步的 Artifact-centric 业务流程模型的一致性检查问题，以进行对流程模型执行后的异常行为检测。

第7章 结论与展望

7.1 结论

现今快速变化的业务环境使得公司的业务和相关信息系统处于不断的变化当中，因此有效的业务流程建模是支撑企业及其信息系统的重要手段。本文针对传统以活动为中心的业务流程建模范式下异构业务流程环境中的流程整合与流程监控问题，以及新型 Artifact-centric 业务流程建模范式下流程可配置建模与同步流程模型行为正确性验证问题展开研究，具体总结如下：

1. 异构业务流程的整合建模与监控

企业的发展通常通过合并、收购的方式进行扩大，这种自底向上的发展方式常常导致企业内部业务流程模型的异构、冗余和不一致等问题。本文提出了一种模式驱动的异构业务流程整合与监控方法，一个业务可以模型化为抽象的业务模式的交互。每个业务模式是可配置的，可以根据具体的应用场景进行配置为具体的业务模式。为执行业务模型，将各业务模式转化为工作流模式或运作模式的描述，使得业务模型可以运行于目前大部分的工作流程引擎之上。为提供企业业务的全局视图，在业务模式的边界设置复杂事件处理监控点，这些监控点恰好与业务阶段对齐，因此可以监控业务的进展。最后，方法的实证研究验证了该方法的有效性。本文的流程整合与监控方法有助于提高业务流程建模的效率，简化业务流程的管理，并为业务决策提供依据。

2. 基于生命周期的 Artifact-centric 业务流程模型可配置建模

一个流程的变体可能出现在组织内部或者跨越不同的组织，因此这些组织可以从可配置的业务流程模型中获益。一个可配置的业务流程模型是一个业务流程模型的多个变体的集中体现，它可以用来设计新的业务流程模型。本质上，可配置的业务流程模型其实是带有可配置点的业务流程模型。本文提出通过对流程中包含的所有 Artifact 生命周期模型的配置来实现 Artifact-centric 业务流程模型的整体配置。首先，提出 Artifact 生命周期图的概念来对 Artifact 生命周期模型进行图形化描述；然后，提出 Artifact 生命周期图的合并方法；在合并的 Artifact 生命周期图中，加入配置点和配置选项，使其形成一个可配置的 Artifact 生命周期图；最后，对 Artifact 生命周期图进行配置，亦即 Artifact 生命周期模型的配置。在对所有 Artifact 生命周期模型对应的 Artifact 生命周期图进行配置后，可

得到一个完整的 Artifact-centric 业务流程模型，其中包含 Artifact 数据模型和相关的服务、业务规则。本文的可配置建模方法有助于提高业务流程模型的复用，降低业务流程建模的难度，并提高流程建模的效率。

3. Artifact-centric 业务流程形式化模型可配置建模框架

一项业务的流程模型伴随着应用的部署或演进将产生多个变体，建立可配置的业务流程模型是应对同一业务多处部署、未来业务推广不可预知的有效方法。可配置的业务流程模型也将通过重用设计而降低成本和缩短上市时间。然而已有的业务流程模型可配置研究主要集中在传统的以活动为中心的业务流程，不能适用于新型的 Artifact-centric 业务流程建模范式。第 4 章中主要研究了单个 Artifact 生命周期概念模型的配置，是在粗粒度的级别进行的，为了给出一个完整的细粒度的可配置建模方法框架，本文提出 Artifact-centric 业务流程形式化可配置建模方法框架，考虑了流程模型的所有组成元素。为获得多个 Artifact-centric 业务流程模型变体的整合模型，提出 Artifact-centric 业务流程形式化模型的合并操作。根据模型元素的“可变”或“共同”特征，在整合模型中识别出可配置点，并为可配点设置相应的配置选项，最终得到可配置的 Artifact-centric 业务流程模型。基于可配置流程模型的行为选择配置选项，可配置出新的 Artifact-centric 业务流程模型。为了辅助业务流程模型的配置过程，基于流程模型元素关系图的概念提出了配置指南。通过保障性住房申请审批流程的真实案例分析，表明了本文方法的有效性。

4. Artifact-centric 业务流程模型同步的行为正确性验证

正确的业务流程模型是 BPM 实现业务目标的关键。近十多年来提出的 Artifact-centric 业务流程建模范式将其控制流隐含描述在业务规则当中，所以在处理动态和跨组织的流程中具备优势。在 Artifact-centric 方法中，业务流程建模为涉及的 Artifact 生命周期的交互，交互使得流程执行变得更加复杂，因为 Artifact 生命周期之间存在各种各样的同步，这就为一个基本的正确性验证问题带来了挑战，即行为正确性问题。本文针对带复杂同步条件的 Artifact-centric 业务流程模型的行为正确性验证这一问题提出解决方案。首先，将流程中涉及的每个 Artifact 生命周期分别映射为 Petri Net 的等价描述。然后，基于同步约束提出规则将这些 Petri Net 合成描述为一个集中的 Workflow Net。基于得到的 Workflow Net 计算其可达图以获得原 Artifact-centric 流程模型描述的所有可能的服务（即活动）执行序列。最后，行为正确性（即正常完成）的验证通过验证每个可能的服务执行序列是否在控制流和数据流两方面均正常完成来实现。一个真实的案例分析展示了本文提出的方法的有效性。本文的行为正确性验证方法有助于为 Artifact-centric

业务流程模型的修正提供基础，降低后续系统修改带来的维护成本。

7.2 未来工作展望

异构业务流程的整合与监控问题对一些大型企业是一大难点，因其各分公司的业务流程分别各自建立使得流程变体繁多，执行引擎也可能不一样，难以形成统一标准的流程模型。对 Artifact-centric 业务流程模型的配置，ACP 模型提供了相对直观的图形化描述，因此本文的部分内容是基于 ACP 模型的流程模型配置。进一步地，为了解细粒度级别的模型配置，本文研究了 Artifact-centric 业务流程形式化模型的可配置建模框架。而关于 Artifact-centric 业务流程模型的行为正确性验证问题，主要考虑了流程中存在多个 Artifact 且之间存在各种同步条件的验证。然而，本文这些提出的方法依然还欠完备，还有许多挑战性的问题尚待解决，下面对本文的后续工作提出几点设想：

（1）第 3 章中的评价工作中涉及的流程数量相对较少。下一步可以在大范围内测试更多业务流程模型来验证模式驱动的异构业务流程整合方法在实际应用中的有效性。同时，可通过挖掘业务流程日志，自动化地发现业务模式，作为人工识别业务模式的一个参考。自动化的业务模式转化也是下一步要研究的工作。

（2）第 4 章中的 Artifact-centric 业务流程模型的可配置建模只考虑了 Artifact 生命周期的配置。这种情况在对于具有共通的同步约束的变体是没有问题的，然而对于不同变体存在不同的同步约束的情况，则本文的方法还需要增加同步方面的配置。另外，本文对 Artifact 的配置只涉及数据属性的配置，而没有对数据属性值的配置。因此，一个更完备的 Artifact-centric 业务流程模型可配置建模方法还需要考虑 Artifact 生命周期之间同步的配置和 Artifact 数据属性值的配置。

（3）第 5 章中的 Artifact-centric 业务流程模型可配置建模方法是建立在形式化模型之上，具有通用性。为使该方法更加具有实用性，可配置的 Artifact-centric 业务流程模型的图形化表示和配置的系统实现是下一步研究需要解决的问题。另外，第 5 章将配置视为业务流程设计时的行为约束，在未来工作中我们将研究 Artifact-centric 业务流程模型在运行时的可变性管理。

（4）第 6 章中的 Artifact-centric 业务流程模型同步的行为正确性验证方法集中在验证流程是否存在控制流和数据流方面的问题。然而，问题检测出来之后，流程模型该如何修正以及系统实现属于下一步工作，未来可以考虑添加领域知识来指导模型的修正。另外，我们计划在下一步工作中研究同步的 Artifact-centric 业务流程模型的一致性检查问题，以进行对流程模型执行后的异常检测。

参考文献

[1] Michele Cantara, David Norton, and Bill Swanton. Systems of Differentiation and Innovation Require Different Types of Model-Driven Application Platforms. Gartner, 2013.

[2] Manfred Reichert, and Barbara Weber. Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer, 2012.

[3] Peter Fettke, and Peter Loos. Classification of Reference Models: A Methodology and Its Application. InformationSystems and E-Business Management, Vol. 1, No. 1, pp. 35-53, 2003.

[4] Helen Schonenberg, Ronny Mans, Nick Russell, Nataliya Mulyar, and Wil Mp Van Der Aalst. Towards a Taxonomy of Process Flexibility. Proceedings of CAiSE Forum. pp. 81-84, 2008.

[5] Wei Song, and Hans-Arno Jacobsen. Static and Dynamic Process Change. IEEE Transactions on Services Computing, Vol. 11, No. 1, pp. 215-231, 2018.

[6] Remco M Dijkman, Marcello La Rosa, and Hajo A Reijers. Managing Large Collections of Business Process Models-Current Techniques and Challenges. Computers in Industry, Vol. 63, No. 2, pp. 91-97, 2012.

[7] Xiang Gao, Yurong Chen, Zizhe Ding, Meng Wang, Xiaonan Zhang, Zhiqiang Yan, Lijie Wen, Qinlong Guo, and Ran Chen. Process Model Fragmentization, Clustering and Merging: An Empirical Study. Proceedings of Business Process Management Workshops. Springer, pp. 405-416, 2013.

[8] 田朝阳, 康国胜, 杨丽琴, 张亮, 张笑楠, and 高翔. 基于 jBPM5 的业务模型执行方法与实现. 计算机工程与科学, Vol. 37, No. 04, pp. 726-733, 2015.

[9] Economist Intelligent Unit. Organisational Agility: How Business Can Survive and Thrive in Turbulent Times. Economist, 2009.

[10] Michael Rosemann, and Wil Mp Van Der Aalst. A Configurable Reference Modelling Language. Proceedings of International Conference on Business Process Management. Springer, pp. 1-23, 2005.

[11] Florian Gottschalk, Wil Mp Van Der Aalst, and Monique H Jansen-Vullers. Configurable Process Models—a Foundational Approach. Reference Modeling, Springer, pp. 59-77, 2007.

[12] Wil Mp Van Der Aalst. Business Process Configuration in the Cloud: How to Support and Analyze Multi-Tenant Processes? Proceedings of 2011 Ninth IEEE European Conference on Web Services (ECOWS). IEEE, pp. 3-10, 2011.

[13] Michael Rosemann, and Wil Mp Van Der Aalst. A Configurable Reference Modelling Language. Information Systems, Vol. 32, No. 1, pp. 1-23, 2007.

[14] Jjcl Vogelaar, Hmw Verbeek, B Luka, and Wil Mp Van Der Aalst. Comparing Business Processes to Determine the Feasibility of Configurable Models: A Case Study. Proceedings of Business Process Management Workshops. Springer, pp. 50-61, 2012.

[15] Marcello La Rosa, Wil Mp Van Der Aalst, Marlon Dumas, and Fredrik P Milani. Business Process Variability Modeling: A Survey. ACM Computing Surveys (CSUR), Vol. 50, No. 1, pp. 2, 2017.

[16] Guosheng Kang. Correctness of Artifact-centric Business Process Models. International Journal of Process Integration and Management, Vol. 8, No. 3, pp. 172-181, 2017.

[17] Jan Mendling, Marcello La Rosa, and Arthur Hm Ter Hofstede. Correctness of Business Process Models with Roles and Objects. Queensland University of Technology, 2008.

[18] Jussi Vanhatalo, Hagen Völzer, and Frank Leymann. Faster and More Focused Control-Flow Analysis for Business Process Models through Sese Decomposition. Proceedings of International Conference on Service-Oriented Computing. Springer, pp. 43-55, 2007.

[19] Andreas Meyer, and Mathias Weske. Activity-Centric and Artifact-centric Process Model Roundtrip. Hasso Plattner Institute at the University of Potsdam, 2013.

[20] Wil Mp Van Der Aalst, Arthur Hm Ter Hofstede, Bartek Kiepuszewski, and Alistair P Barros. Workflow Patterns. Distributed and Parallel Databases, Vol. 14, No. 1, pp. 5-51, 2003.

[21] Nick Russell, Arthur Hm Ter Hofstede, and Nataliya Mulyar. Workflow Controlflow Patterns: A Revised View. Proceedings of International Conference on Business Process Management. pp. 6-22, 2006.

[22] Nick Russell, Arthur Hm Ter Hofstede, David Edmond, and Wil Mp Van Der Aalst. Workflow Resource Patterns: Identification, Representation and Tool Support. Proceedings of International Conference on Advanced Information Systems Engineering. Springer, pp. 216-232, 2005.

[23] Nick Russell, Arthur Hm Ter Hofstede, David Edmond, and Wil Mp Van Der Aalst. Workflow Data Patterns: Identification, Representation and Tool Support. Proceedings of Conceptual Modeling–ER 2005. Springer, pp. 353-368, 2005.

[24] Nick Russell, Wil Van Der Aalst, and Arthur Ter Hofstede. Workflow Exception Patterns. Proceedings of International Conference on Advanced Information Systems Engineering. Springer, pp. 288-302, 2006.

[25] Thomas Gschwind, Jana Koehler, and Janette Wong. Applying Patterns During Business Process Modeling. Proceedings of International Conference on Business Process Management. Springer, pp. 4-19, 2008.

[26] Barbara Weber, Stefanie Rinderle, and Manfred Reichert. Change Patterns and Change Support Features in Process-Aware Information Systems. Proceedings of International Conference on Advanced Information Systems Engineering. Springer, pp. 574-588, 2007.

[27] Barbara Weber, Stefanie Rinderle, and Manfred Reichert. Aktuelles Schlagwort: Process Change Patterns. Proceedings of EMISA Forum. pp. 45-51, 2007.

[28] Marcello La Rosa, Marlon Dumas, Reina Uba, and Remco Dijkman. Business Process Model Merging: An Approach to Business Process Consolidation. ACM Transactions on Software Engineering and Methodology (TOSEM), Vol. 22, No. 2, pp. 11-44, 2013.

[29] Jussi Vanhatalo, Hagen Völzer, and Jana Koehler. The Refined Process Structure Tree. Data & Knowledge Engineering, Vol. 68, No. 9, pp. 793-818, 2009.

[30] Jussi Vanhatalo, Hagen Völzer, and Jana Koehler. The Refined Process Structure Tree. Proceedings of International Conference on Business Process Management. Springer, pp. 100-115, 2008.

[31] Ana Karla Alves De Medeiros, Antonella Guzzo, Gianluigi Greco, Wil Mp Van Der Aalst, Ajmm Weijters, Boudewijn F Van Dongen, and Domenico Saccà. Process Mining Based on Clustering: A Quest for Precision. Proceedings of Business Process Management Workshops. Springer, pp. 17-29, 2008.

[32] Remco Dijkman, Marlon Dumas, Boudewijn Van Dongen, Reina Käärik, and Jan Mendling. Similarity of Business Process Models: Metrics and Evaluation. Information Systems, Vol. 36, No. 2, pp. 498-516, 2011.

[33] Manfred Reichert, Alena Hallerbach, and Thomas Bauer. Lifecycle Management of Business Process Variants. Handbook on Business Process Management 1, Springer, pp. 251-278, 2015.

[34] Reza Motahari-Nezhad, Moshe Chai Barukh, Ahmed Gater, and Seung Hwan Ryu. Process Analytics-Concepts and Techniques for Querying and Analyzing Process Data. Springer, 2016.

[35] Anil Nigam, and Nathan Caswell. Business Artifacts: An Approach to Operational Specification. IBM Systems Journal, Vol. 42, No. 3, pp. 428-445, 2003.

[36] Kan Ngamakeur, Sira Yongchareon, and Chengfei Liu. A Framework for Realizing Artifact-centric

Business Processes in Service-Oriented Architecture. Proceedings of International Conference on Database Systems for Advanced Applications. Springer, pp. 63-78, 2012.

[37] Kamal Bhattacharya, Richard Hull, and Jianwen Su. A Data-Centric Design Methodology for Business Processes. Handbook of Research on Business Process Modeling, IGI Global, pp. 503-531, 2009.

[38] Rong Liu, Kamal Bhattacharya, and Frederich Wu. Modeling Business Contexture and Behavior Using Business Artifacts. Proceedings of International Conference on Advanced Information Systems Engineering. Springer, pp. 324-339, 2007.

[39] Guohua Liu, Xi Liu, Haihuan Qin, Jianwen Su, Zhimin Yan, and Liang Zhang. Automated Realization of Business Workflow Specification. Proceedings of International Conference on Service-Oriented Computing. Springer, pp. 96-108, 2010.

[40] Maroun Abi Assaf, Youakim Badr, Kablan Barbar, and Youssef Amghar. On the Integration of Artifact Lifecycles. Proceedings of The 7th International Conference on Management of Computational and Collective IntElligence in Digital EcoSystems: MEDES'15. ACM, 2015.

[41] Wil Mp Van Der Aalst, Alexander Dreiling, Florian Gottschalk, Michael Rosemann, and Monique H Jansen-Vullers. Configurable Process Models as a Basis for Reference Modeling. Proceedings of Business Process Management Workshops. Springer, pp. 512-518, 2006.

[42] Florian Gottschalk, Teun Ac Wagemakers, Monique H Jansen-Vullers, Wil Mp Van Der Aalst, and Marcello La Rosa. Configurable Process Models: Experiences from a Municipality Case Study. Proceedings of International Conference on Advanced Information Systems Engineering. Springer, pp. 486-500, 2009.

[43] Marcello La Rosa, Marlon Dumas, Arthur Hm Ter Hofstede, Jan Mendling, and Florian Gottschalk. Beyond Control-Flow: Extending Business Process Configuration to Roles and Objects. Proceedings of International Conference on Conceptual Modeling. Springer, pp. 199-215, 2008.

[44] Jan Mendling, Jan Recker, Michael Rosemann, and Wil Van Der Aalst. Generating Correct EPCs from Configured C-EPCs. Proceedings of ACM Symposium on Applied computing. ACM, pp. 1505-1510, 2006.

[45] Florian Gottschalk. Configurable Process Models[D], Eindhoven University of Technology, The Netherlands, 2009.

[46] Jan Mendling, and Carlo Simon. Business Process Design by View Integration. Proceedings of

Business Process Management Workshops. Springer, pp. 55-64, 2006.

[47] Shuang Sun, Akhil Kumar, and John Yen. Merging Workflows: A New Perspective on Connecting Business Processes. Decision Support Systems, Vol. 42, No. 2, pp. 844-858, 2006.

[48] Dennis Mm Schunselaar, Eric Verbeek, Wil Mp Van Der Aalst, and Hajo A Raijers. Creating Sound and Reversible Configurable Process Models Using Cosenets. Proceedings of International Conference on Business Information Systems. Springer, pp. 24-35, 2012.

[49] Chen Li, Manfred Reichert, and Andreas Wombacher. Discovering Reference Models by Mining Process Variants Using a Heuristic Approach. Proceedings of International Conference on Business Process Management. Springer, pp. 344-362, 2009.

[50] Florian Gottschalk, Wil Mp Van Der Aalst, and Monique H Jansen-Vullers. Mining Reference Process Models and Their Configurations. Proceedings of the OTM Confederated International Workshops and Posters on On the Move to Meaningful Internet Systems. Springer, pp. 263-272, 2008.

[51] Joos Cam Buijs, Boudewijn F Van Dongen, and Wil Mp Van Der Aalst. Mining Configurable Process Models from Collections of Event Logs. Proceedings of International Conference on Business Process Management. Springer, pp. 33-48, 2013.

[52] Wil Mp Van Der Aalst, Niels Lohmann, and Marcello La Rosa. Ensuring Correctness During Process Configuration Via Partner Synthesis. Information Systems, Vol. 37, No. 6, pp. 574-592, 2012.

[53] Marcello La Rosa, Wil Mp Van Der Aalst, Marlon Dumas, and Arthur Hm Ter Hofstede. Questionnaire-Based Variability Modeling for System Configuration. Software & Systems Modeling, Vol. 8, No. 2, pp. 251-274, 2009.

[54] Marcello La Rosa, Johannes Lux, Stefan Seidel, Marlon Dumas, and Arthur Hm Ter Hofstede. Questionnaire-Driven Configuration of Reference Process Models. Proceedings of International Conference on Advanced Information Systems Engineering. Springer, pp. 424-438, 2007.

[55] Ying Huang, Zaiwen Feng, Keqing He, and Yiwang Huang. Ontology-Based Configuration for Service-Based Business Process Model. Proceedings of 2013 IEEE International Conference on Services Computing. IEEE, pp. 296-303, 2013.

[56] Akhil Kumar, and Wen Yao. Design and Management of Flexible Process Variants Using Templates and Rules. Computers in Industry, Vol. 63, No. 2, pp. 112-130, 2012.

[57] Emanuel Santos, Joao Pimentel, Jaelson Castro, Juan Sánchez, and Oscar Pastor. Configuring the Variability of Business Process Models Using Non-Functional Requirements. Enterprise,

Business-Process and Information Systems Modeling, Springer, pp. 274-286, 2010.

[58] Nour Assy, and Walid Gaaloul. Extracting Configuration Guidance Models from Business Process Repositories. Proceedings of International Conference on Business Process Management. Springer, pp. 198-206, 2015.

[59] Nour Assy, and Walid Gaaloul. Configuration Rule Mining for Variability Analysis in Configurable Process Models. Proceedings of International Conference on Service-Oriented Computing. Springer, pp. 1-15, 2014.

[60] Santhosh Kumaran, Rong Liu, and Frederich Wu. On the Duality of Information-Centric and Activity-Centric Models of Business Processes. Proceedings of International Conference on Advanced Information Systems Engineering. Springer, pp. 32-47, 2008.

[61] Cagdas Gerede, and Jianwen Su. Specification and Verification of Artifact Behaviors in Business Process Models. Proceedings of International Conference on Service-Oriented Computing. Springer, pp. 181-192, 2007.

[62] Kamal Bhattacharya, Cagdas Gerede, Richard Hull, Rong Liu, and Jianwen Su. Towards Formal Analysis of Artifact-centric Business Process Models. Proceedings of International Conference on Business Process Management. Springer-Verlag, pp. 288-304, 2007.

[63] Cagdas Gerede, Kamal Bhattacharya, and Jianwen Su. Static Analysis of Business Artifact-centric Operational Models. Proceedings of IEEE International Conference on Service-Oriented Computing and Applications. IEEE, pp. 133-140, 2007.

[64] Sira Yongchareon, Chengfei Liu, and Xiaohui Zhao. An Artifact-centric View-Based Approach to Modeling Inter-Organizational Business Processes. Proceedings of Web Information System Engineering–WISE 2011. Springer, pp. 273-281, 2011.

[65] Sira Yongchareon, and Chengfei Liu. A Process View Framework for Artifact-centric Business Processes. Proceedings of On the Move to Meaningful Internet Systems: OTM 2010. Springer, pp. 26-43, 2010.

[66] Richard Hull. Artifact-centric Business Process Models: Brief Survey of Research Results and Challenges. Proceedings of On the Move to Meaningful Internet Systems: OTM 2008. Springer, pp. 1152-1163, 2008.

[67] Kamal Bhattacharya, Nathan Caswell, Santhosh Kumaran, Anil Nigam, and Frederich Wu. Artifact-Centered Operational Modeling: Lessons from Customer Engagements. IBM Systems Journal,

Vol. 46, No. 4, pp. 703-721, 2007.

[68] Wil Mp Van Der Aalst. Verification of Workflow Nets. Application and Theory of Petri Nets, Springer, pp. 407-426, 1997.

[69] Mathias Weske. Business Process Management: Concepts, Languages, Architectures, Springer, 2012.

[70] Wil Mp Van Der Aalst, and Arthur Hm Ter Hofstede. YAWL: Yet Another Workflow Language. Information Systems, Vol. 30, No. 4, pp. 245-275, 2005.

[71] Michele Chinosi, and Alberto Trombetta. BPMN: An Introduction to the Standard. Computer Standards & Interfaces, Vol. 34, No. 1, pp. 124-134, 2012.

[72] Harald Störrle. Semantics and Verification of Data Flow in UML 2.0 Activities. Electronic Notes in Theoretical Computer Science, Vol. 127, No. 4, pp. 35-52, 2005.

[73] Jochen M Küster, Ksenia Ryndina, and Harald Gall. Generation of Business Process Models for Object Life Cycle Compliance. Proceedings of International Conference on Business Process Management. Springer, pp. 165-181, 2007.

[74] Ksenia Ryndina, Jochen M Küster, and Harald Gall. Consistency of Business Process Models and Object Life Cycles. Proceedings of International Conference on Model Driven Engineering Languages and Systems. Springer, pp. 80-90, 2007.

[75] Andreas Meyer, Artem Polyvyanyy, and Mathias Weske. Weak Conformance of Process Models with Respect to Data Objects. Proceedings of ZEUS. Citeseer, pp. 74-80, 2012.

[76] Andreas Meyer, and Mathias Weske. Weak Conformance between Process Models and Synchronized Object Life Cycles, Hasso Plattner Institute at the University of Potsdam, 2014.

[77] Andreas Meyer, and Mathias Weske. Weak Conformance between Process Models and Synchronized Object Life Cycles. Proceedings of International Conference on Service-Oriented Computing. Springer, pp. 359-367, 2014.

[78] Elio Damaggio, Alin Deutsch, Richard Hull, and Victor Vianu. Automatic Verification of Data-Centric Business Processes. Proceedings of International Conference on Business Process Management. Springer, pp. 3-16, 2011.

[79] Alin Deutsch, Richard Hull, Fabio Patrizi, and Victor Vianu. Automatic Verification of Data-Centric Business Processes. Proceedings of International Conference on Database Theory. ACM, pp. 252-267, 2009.

[80] Richard Hull, Elio Damaggio, Fabiana Fournier, Manmohan Gupta, Fenno Heath, Stacy Hobson, Mark Linehan, Sridhar Maradugu, Anil Nigam, Piyawadee Sukaviriya, and Roman Vacul ń. Introducing the Guard-Stage-Milestone Approach for Specifying Business Entity Lifecycles. Proceedings of International Workshop on Web Services and Formal Methods. pp. 1-24, 2010.

[81] R. Hull, E. Damaggio, R. De Masellis, F. Fournier, M. Gupta, F. Heath Iii, S. Hobson, M. Linehan, S. Maradugu, and A. Nigam. Business Artifacts with Guard-Stage-Milestone Lifecycles: Managing Artifact Interactions with Conditions and Events. Proceedings of International Conference on Distributed Event-Based Systems (DEBS). pp. 51-62, 2011.

[82] Pavel Gonzalez, Andreas Griesmayer, and Alessio Lomuscio. Verifying GSM-Based Business Artifacts. Proceedings of International Conference on Web Services. IEEE, pp. 25-32, 2012.

[83] Francesco Belardinelli, Alessio Lomuscio, and Fabio Patrizi. Verification of GSM-Based Artifact-centric Systems through Finite Abstraction. Proceedings of International Conference on Service-Oriented Computing. Springer, pp. 17-31, 2012.

[84] Dmitry Solomakhin, Marco Montali, Sergio Tessaris, and Riccardo De Masellis. Verification of Artifact-centric Systems: Decidability and Modeling Issues. International Conference on Service-Oriented Computing, Springer, pp. 252-266, 2013.

[85] Niels Lohmann. Compliance by Design for Artifact-centric Business Processes. Proceedings of International Conference on Business Process Management. Springer, pp. 99-115, 2011.

[86] Dirk Fahland, Massimiliano De Leoni, Boudewijn F Van Dongen, and Wil Mp Van Der Aalst. Behavioral Conformance of Artifact-centric Process Models. Proceedings of International Conference on Business Information Systems. Springer, pp. 37-49, 2011.

[87] Dirk Fahland, Massimiliano De Leoni, Boudewijn F Van Dongen, and Wil Mp Van Der Aalst. Conformance Checking of Interacting Processes with Overlapping Instances. International Conference on Business Process Management, Springer, pp. 345-361, 2011.

[88] 王建民, and 闻立杰. 工作流管理——模型、方法和系统. 清华大学出版社, 北京, 2004.

[89] Kees Max Van Hee. Workflow Management: Models, Methods, and Systems. The MIT press, 2004.

[90] Wil Mp Van Der Aalst, Arthur Hm Ter Hofstede, and Mathias Weske. Business Process Management: A Survey. Proceedings of International Conference on Business Process Management. pp. 1-12, 2003.

[91] Jan Vom Brocke, and Michael Rosemann. Business Process Management. Springer, 2015.

[92] 梅杰. 以 artifact 为中心的工作流技术研究及其应用[D], 东华大学, 上海, 2011.

[93] Peter Forbrig. Generic Components for BPMN Specifications. Proceedings of International Conference on Business Informatics Research. Springer, pp. 202-216, 2014.

[94] Alexandre Alves, Assaf Arkin, Sid Askary, Ben Bloch, Francisco Curbera, Yaron Goland, Neelakantan Kartha, Dieter König, Vinkesh Mehta, and Satish Thatte. Web Services Business Process Execution Language Version 2.0, 2006.

[95] James Rumbaugh, Grady Booch, and Ivar Jacobson. The Unified Modeling Language Reference Manual. Addison Wesley, 2017.

[96] Wil Mp Van Der Aalst. Process Mining: Discovery, Conformance and Enhancement of Business. Springer, 2011.

[97] Artem Polyvyanyy, Sergey Smirnov, and Mathias Weske. Reducing Complexity of Large EPCs. Proceedings of MobIS. pp. 195-207, 2008.

[98] Nahla Haddar Ouali, and Mohamed Tmar. Efficiency of Artifact-centric Paradigm: A Literature Study. Proceedings of the 9th International Conference on Information Management and Engineering. ACM, pp. 50-54, 2017.

[99] Jyothi Kunchala, Jian Yu, and Sira Yongchareon. A Survey on Approaches to Modeling Artifact-centric Business Processes. Proceedings of Web Information Systems Engineering–WISE 2014 Workshops. Springer, pp. 117-132, 2014.

[100] Esra Kucukoguz, and Jianwen Su. On Lifecycle Constraints of Artifact-centric Workflows. Proceedings of International Workshop on Web Services and Formal Methods. Springer, pp. 71-85, 2010.

[101] Serge Abiteboul, Pierre Bourhis, Alban Galland, and Bogdan Marinoiu. The AXML Artifact Model. Proceedings of 16th International Symposium on Temporal Representation and Reasoning (TIME 2009). IEEE, pp. 11-17, 2009.

[102] Niels Lohmann, and Martin Nyolt. Artifact-centric Modeling Using BPMN. Proceedings of Service-Oriented Computing-ICSOC 2011 Workshops. Springer, pp. 54-65, 2011.

[103] 王颖. 以 artifact 为中心的业务流程建模与分析[D], 燕山大学, 秦皇岛, 2011.

[104] Haihuan Qin, Guosheng Kang, and Lipeng Guo. Maxinstx: A Best-Effort Failure Recovery

Approach for Artifact-centric Business Processes. Proceedings of International Conference on Service-Oriented Computing. Springer, pp. 558-566, 2013.

[105] Giuseppe De Giacomo, Riccardo De Masellis, and Riccardo Rosati. Verification of Conjunctive Artifact-centric Services. International Journal of Cooperative Information Systems, Vol. 21, No. 02, pp. 111-139, 2012.

[106] Viara Popova, Dirk Fahland, and Marlon Dumas. Artifact Lifecycle Discovery. International Journal of Cooperative Information Systems, Vol. 24, No. 01, pp. 1550001, 2015.

[107] Xixi Lu. Artifact-centric Log Extraction and Process Discovery[D], Eindhoven University of Technology, Eindhoven,, 2013.

[108] Erik Hj Nooijen, Boudewijn F Van Dongen, and Dirk Fahland. Automatic Discovery of Data-Centric and Artifact-centric Processes. Proceedings of Business Process Management Workshops. Springer, pp. 316-327, 2013.

[109] Diana Borrego, Rafael M Gasca, and María Teresa Gómez-López. Automating Correctness Verification of Artifact-centric Business Process Models. Information and Software Technology, Vol. 62, No. 2015, pp. 187-197, 2015.

[110] Zakaria Maamar, Youakim Badr, and Nanjangud C Narendra. Business Artifacts Discovery and Modeling. Proceedings of International Conference on Service-Oriented Computing. Springer, pp. 542-550, 2010.

[111] Kan Ngamakeur, Sira Yongchareon, Veronica Liesaputra, Chengfei Liu, and Jian Yu. An Artifact-centric Business Process Execution Platform. Proceedings of 2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW). IEEE, pp. 1-4, 2016.

[112] Kan Ngamakeur. On Realization of Artifact-centric Model for Business Process[D], Swinburne University of Technology, Melbourne, 2013.

[113] Manoj Mahabaleshwar. Distributed Execution of Artifact-centric Workflows in Publish/Subscribe Infrastructure[D], TECHNISCHE UNIVERSITÄT MÜNCHEN, 2015.

[114] Jyothi Kunchala, Jian Yu, Sira Yongchareon, and Yanbo Han. Towards Merging Collaborating Processes for Artifact Lifecycle Synthesis. Proceedings of the Australasian Computer Science Week Multiconference. ACM, pp. 50, 2017.

[115] Dirk Fahland, Massimiliano De Leoni, Boudewijn F Van Dongen, and Wil Mp Van Der Aalst. Many-to-Many: Some Observations on Interactions in Artifact Choreographies. Proceedings of the 3rd

Central-European Workshop on Services and their Composition (ZEUS 2011). pp. 9-15, 2011.

[116] Youakim Badr, Nanjangud C Narendra, and Zakaria Maamar. Business Artifacts for E-Business Interoperability. Electronic Business Interoperability: Concepts, Opportunities and Challenges, IGI Global, pp. 670-690, 2011.

[117] Mengfei Yi. Managing Business Process Variability in Artifact-centric BPM[D], Eindhoven University of Technology, Eindhoven, 2015.

[118] Giovanni Meroni, Claudio Di Ciccio, and Jan Mendling. An Artifact-Driven Approach to Monitor Business Processes through Real-World Objects. Proceedings of International Conference on Service-Oriented Computing. Springer, pp. 297-313, 2017.

[119] Rong Liu, Roman Vacul ń, Zhe Shan, Anil Nigam, and Frederich Wu. Business Artifact-centric Modeling for Real-Time Performance Monitoring. Proceedings of International Conference on Business Process Management. pp. 265-280, 2011.

[120] Viara Popova, and Marlon Dumas. Discovering Unbounded Synchronization Conditions in Artifact-centric Process Models. Proceedings of Business Process Management Workshops. Springer, pp. 28-40, 2014.

[121] Wei Xu, Jianwen Su, Zhimin Yan, Jian Yang, and Liang Zhang. An Artifact-centric Approach to Dynamic Modification of Workflow Execution. Proceedings of OTM Confederated International Conferences "On the Move to Meaningful Internet Systems". Springer, pp. 256-273, 2011.

[122] 田朝阳. 基于 jBPM5 的业务模式执行方法与实现[D], 复旦大学, 2015.

[123] Marcello La Rosa, Marlon Dumas, Reina Uba, and Remco Dijkman. Merging Business Process Models. Proceedings of OTM Confederated International Conferences "On the Move to Meaningful Internet Systems". Springer, pp. 96-113, 2010.

[124] 闾亚, 雷建坤, 杨丽琴, and 张亮. Bpdetector:一种可扩展的业务模式识别框架. 计算机集成制造系统, Vol. 22, No. 2, pp. 287-293, 2016.

[125] 王颖, 刘国华, 高尚, 赵丹枫, and 刘海滨. Artiflow 中 artifact 生命周期的可满足性问题. 小型微型计算机系统, Vol. 33, No. 6, pp. 1176-1182, 2012.

[126] Vladimir I Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. Soviet Physics Doklady, Vol. 163, No. 4, pp. 845-848, 1965.

[127] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet:: Similarity: Measuring the Relatedness of Concepts. Proceedings of Demonstration Papers at HLT-NAACL 2004. Association for

Computational Linguistics, pp. 38-41, 2004.

[128] Zhibiao Wu, and Martha Palmer. Verbs Semantics and Lexical Selection. Proceedings of the 32nd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, pp. 133-138, 1994.

[129] Yutian Sun, Wei Xu, and Jianwen Su. Declarative Choreographies for Artifacts. Proceedings of International Conference on Service-Oriented Computing. pp. 420-434, 2012.

[130] N. Lohmann, and K. Wolf. Artifact-centric Choreographies. Proceedings of Service-Oriented Computing. pp. 32-46, 2010.

[131] Yutian Sun. Management of Data and Collaboration for Business Processes[D], University of California, Santa Barbara, 2015.

[132] Yi Wang, and Ying Wang. Change Analysis for Artifact-centric Business Processes. Proceedings of International Conference on Business Information Systems. Springer, pp. 98-109, 2014.

[133] Yi Wang, Jian Yang, Weiliang Zhao, and Jianwen Su. Change Impact Analysis in Service-Based Business Processes. Service Oriented Computing and Applications, Vol. 6, No. 2, pp. 131-149, 2012.

[134] Khubaib Amjad Alam, Rodina Binti Ahmad, and Maria Akhtar. Change Impact Analysis and Propagation in Service Based Business Process Management Systems Preliminary Results from a Systematic Review. Proceedings of 2014 8th Malaysian Software Engineering Conference (MySEC). IEEE, pp. 7-12, 2014.

[135] Yi Wang, Jian Yang, and Weiliang Zhao. Change Impact Analysis in Service-Based Business Processes. Proceedings of 2010 International Conference on Service Oriented Computing and Applications. IEEE, pp. 1-8, 2010.

[136] Guosheng Kang, Jianxun Liu, Mingdong Tang, Buqing Cao, and Yu Xu. Improved Active Web Service Recommendation Based on Usage History. Applied Mathematics & Information Sciences, Vol. 10, No. 3, pp. 903-913, 2016.

[137] Guosheng Kang, Mingdong Tang, Jianxun Liu, Xiaoqing(Frank) Liu, and Buqing Cao. Diversifying Web Service Recommendation Results Via Exploring Service Usage History. IEEE Transactions on Services Computing, Vol. 14, No. 99, pp. 35-48, 2015.

[138] Guosheng Kang, Jianxun Liu, Mingdong Tang, Buqing Cao, and Yu Xu. An Effective Web Service Ranking Method Via Exploring User Behavior. IEEE Transactions on Network and Service Management, Vol. 12, No. 4, pp. 554-564, 2015.

[139] Edson Alves De Oliveira Junior, Itana Gimenes, Elisa Hatsue Moriya Huzita, and José Carlos Maldonado. A Variability Management Process for Software Product Lines. Proceedings of the 2005 Conference of the Centre for Advanced Studies on Collaborative Research. IBM Press, pp. 225-241, 2005.

[140] Klaus Pohl, Günter Böckle, and Frank J Van Der Linden. Software Product Line Engineering: Foundations, Principles and Techniques. Springer, Berlin, Heidelberg, 2005.

[141] Niels Lohmann, and Karsten Wolf. From Artifacts to Activities. Web Services Foundations, Springer, pp. 109-135, 2014.

[142] Zhou Conghua, and Chen Zhenyu. Model Checking Workflow Net Based on Petri Net. Wuhan University Journal of Natural Sciences, Vol. 11, No. 5, pp. 1297-1301, 2006.

[143] Anne Ratzer, Lisa Wells, Henry Lassen, Mads Laursen, Jacob Qvortrup, Martin Stissing, Michael Westergaard, Søren Christensen, and Kurt Jensen. CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets. Proceedings of International Conference on Application and Theory of Petri Nets. pp. 450-462, 2003.

[144] Khodakaram Salimifard, and Mike Wright. Petri Net-Based Modelling of Workflow Systems: An Overview. European Journal of Operational Research, Vol. 134, No. 3, pp. 664-676, 2001.

[145] Sea Ling, and Heinz Schmidt. Time Petri Nets for Workflow Modelling and Analysis. Proceedings of IEEE International Conference on Systems, Man, and Cybernetics. IEEE, pp. 3039-3044, 2000.

[146] Nabil R Adam, Vijayalakshmi Atluri, and Wei-Kuang Huang. Modeling and Analysis of Workflows Using Petri Nets. Journal of Intelligent Information Systems, Vol. 10, No. 2, pp. 131-158, 1998.

[147] Tadao Murata. Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, Vol. 77, No. 4, pp. 541-580, 1989.

[148] Wmp Van Der Aalst, and Km Van Hee. Business Process Redesign: A Petri-Net-Based Approach. Computers in Industry, Vol. 29, No. 1, pp. 15-26, 1996.

[149] 袁崇义. Petri 网应用. 科学出版社, 北京, 2013.

[150] Kristian Bisgaard Lassen, and Wil Mp Van Der Aalst. Complexity Metrics for Workflow Nets. Information and Software Technology, Vol. 51, No. 3, pp. 610-626, 2009.

攻读博士期间发表的论文与参与的项目

发表论文

第一作者

1. **Guosheng Kang**, Liqin Yang, and Liang Zhang. "Towards Configurable Modelling for Artifact-centric Business Processes". Concurrency and Computation: Practice and Experience, 2019, DOI: 10.1002/cpe.5367
2. **Guosheng Kang**, Liqin Yang, and Liang Zhang. "Verification of Behavioral Soundness for Artifact-centric Business Process Model with Synchronizations". Future Generation Computer Systems, Vol. 98, No. 2019, pp. 503-511, 2019.
3. **Guosheng Kang**, Jianxun Liu, Mingdong Tang, Buqing Cao, Yu Xu. "Improved Active Web Service Recommendation Based on Usage History". Applied Mathematics & Information Sciences, Vol. 10, No. 3, pp. 1-11, 2016.
4. **Guosheng Kang**, Liqin Yang, Wei Xu, Zhaoyang Tian, and Liang Zhang. "Artifact-centric Business Process Configuration". International Journal of High Performance Computing and Networking, Vol. 9, No. 2, pp. 93-103, 2016.
5. **Guosheng Kang**, Jianxun Liu, Mingdong Tang, Buqing Cao, and Yu Xu. "An Effective Web Service Ranking Method via Exploring User Behavior". IEEE Transactions on Network and Service Management, Vol. 12, No. 4, pp. 554-564, 2015.
6. **Guosheng Kang**, Mingdong Tang, Jianxun Liu, Xiaoqing(Frank) Liu, and Buqing Cao. "Diversifying Web Service Recommendation Results via Exploring Service Usage History". IEEE Transactions on Services Computing, Vol. 14, No. 99, pp. 35-48, 2015.
7. **Guosheng Kang**, Zhaoyang Tian, Xiao Zhang, Liang Zhang, Lixin Ma, Xiang Gao, Xiaonan Zhang, and Zizhe Ding. "Heterogeneous Business Process Consolidation: A Pattern-Driven Approach". Proceedings of the 2014 International Conference on Service Sciences. IEEE Computer Society, pp. 136-141, 2014.

第二作者

1. 杨丽琴, **康国胜**, and 张亮. "基于临床实践指南的诊疗过程建模方法". 计算机集成制造系统, Vol. 23, No. 5, pp. 1040-1049, 2017.
2. 杨丽琴, **康国胜**, 郭立鹏, 田朝阳, 张亮, 张笑楠, and 高翔. "一种适用于多样性环境的业务流程挖掘方法". 软件学报, Vol. 26, No. 3, pp. 550-561, 2015.
3. 田朝阳, **康国胜**, 杨丽琴, 张亮, 张笑楠, and 高翔. "基于 jBPM5 的业务模型执行方法与实现". 计算机工程与科学, Vol. 37, No. 4, pp. 726-733, 2015.
4. Haihuan Qin, **Guosheng Kang**, and Lipeng Guo. "MaxInsTx: A Best-Effort Failure Recovery Approach for Artifact-centric Business Processes". Proceedings of International Conference on Service-Oriented Computing. Springer, pp. 558-566, 2013.

参与项目

1. “以案件为中心的检察业务协同支撑技术研究”之课题“案件驱动的跨时空域通用检查业务系统及数据供应链建模方法”（国家重点研发计划项目）

   项目编号：2018YFC0831402

2. “跨界服务融合理论与关键技术”之课题“跨界服务质量管理与价值工程”（国家重点研发计划项目）

   项目编号：2017YFB1400604

3. IT 支撑网中智能业务流程管理工具研究与实现（教育部-中国移动科研基金项目）

   项目编号：MCM20123011

4. SOA 的等级化服务替换理论与机制研究（国家自然科学基金）

   项目编号：60873115

致谢

在此论文完成之际，我无比的感恩，谨在此衷心地向所有给予我指导、帮助、支持和关心的师长、同学、朋友和家人们表示深深的感谢。

首先要感谢我尊敬的导师张亮教授。在博士期间，张老师为我创造了良好的科研环境和广阔的学术交流平台。他学识渊博，对我悉心的指导，使我在科研上不断地进步。他常向学生分享读论文、做研究的方法。张老师诚实守信、公正不阿、诲人不倦，在为人处世上给我做了表率，使我学会了许多做人的道理。同时，他严谨的治学态度、奋进的工作精神也为我树立了榜样。在生活中，我也得到张老师的诸多关心和帮助。在此，向张老师表达我最深的敬意和感谢！

感谢顾宁教授、汪卫教授、李银胜副教授对我的博士论文提出的许多中肯的修改意见。感谢上海交通大学曹健教授和中科院上海高等研究院祝永新研究员对我的学位论文提出的宝贵评审意见。还有感谢几位校外盲审的老师对我的论文给出的评审意见。他们的评审建议使我的学位论文在修改后又上了一个台阶。

感谢实验室的小伙伴们为实验室共同创造了良好的学习氛围。特别感谢实验室与我有过讨论和对我有过帮助的同学。感谢实验室朝夕相处的同学们，让这个实验室不觉得空旷，尤其是在节假日期间依然看到他们还在实验室奋斗。让我在此记录他们的名字：秦海焕、徐玮、郭立鹏、杨丽琴、周征奇、吴海双、王龙、汤瀑、田朝阳、张校、陈斌、宇菲、雷建坤、刘进、白如帆等。

最后，要感谢我的家人，他们的鼓励和支持是我前进路上最坚强的后盾。感谢父母含辛茹苦的养育，让我有了坚忍不拔的毅力和阳光正直的性格，可以让我在正确的道路上稳步前行。特别感谢妻子对我一如既往的支持，在我求学期间独自辛苦地照顾两个孩子。感谢两个可爱的女儿，她们的到来是我最大的喜悦，她们的欢声笑语总是让我感到满足。感谢我的岳父岳母在我求学期间对我的支持和理解，使我可以坚持完成学业。

论文独创性声明

本论文是我个人在导师指导下进行的研究工作及取得的研究成果。论文中除了特别加以标注和致谢的地方外，不包含其他人或其它机构已经发表或撰写过的研究成果。其他同志对本研究的启发和所做的贡献均已在论文中作了明确的声明并表示了谢意。

作者签名：_____ 日期：_____

论文使用授权声明

本人完全了解复旦大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其它复制手段保存论文。保密的论文在解密后遵守此规定。

作者签名：_____ 导师签名：_____ 日期：_____