**密级:**



**硕士学位论文**

**支持多工具协同的流程管理系统设计与实现**

**作者姓名： 杨涛**

**指导教师: （姓名、专业技术职务、工作单位）**

**学位类别: （学科门类或专业学位类别）**

**学科专业: 软件工程**

**研究所: 软件研究所**

**2017年 04月**

**Design and Implementation of Process Management System Supporting Multi-tool Collaboration**

**By**

**Yang Tao**

**A Dissertation/Thesis Submitted to**

**The University of Chinese Academy of Sciences**

**In partial fulfillment of the requirement**

**For the degree of**

**Master of Software Engineering**

**软件研究所**

**April, 2017**

**学位论文独创性声明**

本人郑重声明：我所呈交的学位论文是本人在导师指导下进行的研究工作及所取得的研究成果。尽我所知，除了文中已经标注引用的内容外，本论文中不含含其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中作了明确的说明或致谢。本人知道本声明的法律结果由自己承担。

学位论文作者签名： 日期：

**关于学位论文使用授权的说明**

本人完全了解中国科学院软件研究所有关保留、使用学位论文的规定，即：中国科学院软件研究所有权保留送交论文的复印件，允许论文被查阅和借阅；中国科学院软件研究所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。

（涉密的学位论文在解密后，按此规定执行。）

学生签名： 导师签名： 日期：

摘要

21世纪以来，随着计算机技术的飞速发展，软件工程也发生着日新月异的变化。在传统软件工程中，有各种特色的计算机辅助软件（Computer-aided Software，CAS）来帮助和支持软件开发过程，这些软件的集成在软件开发中扮演了十分重要的角色。然而在使用多种工具集成的情况下，开发过程中会遇到开发流程不可定制、流程数据不一致和流程管理效率低等问题。如何有效地将这些辅助软件应用在一起，让软件开发流程可定制化，来支持快速迭代、持续集成的软件工程开发是本文所研究的课题。鉴于此本文提出一种基于工作流的应用即服务应用集成框架（Workflow-based Software-as-a-service Application Integration Framework, WSAIF），致力于让处于软件开发过程中的用户可以根据自己的需求和习惯，自由定制软件开发流程。本文主要的工作如下：

首先，本文介绍了国内外该课题研究的现状和发展趋势，并详细介绍了工作流、面向服务的架构（Service-Oriented Architecture, SOA）和软件即服务（Software as a Service, SaaS）等相关概念；进而讨论了本文研究的意义，讨论将工作流应用到软件工程开发过程中的方案，以及怎样通过软件即服务的思想将工作流和软件开发辅助工具相结合。

其次，本文根据前面的研究，设计了基于工作流的软件即服务应用集成框架。从结构上将框架分为五层，分别是工作流引擎层、工作流API功能层、数据路由层、服务群和工具群。按层详细分析了框架的架构设计：工作流引擎和工作流API层对外提供功能应用程序接口（Application Program Interface, API），框架中下层架构通过调用API接口来使用工作流的功能，去推动软件开发流程的流转，以及实现实现流程相关的操作；框架中数据路由层的管理和收集软件开发流程中的数据，并为开发过程中的任务节点提供路由连接功能；服务层通过集中各种服务给工作流，最终在制定软件开发流程时选择调用；工具层通过添加不同工具的不同功能，将这些功能包装成符合交互标准的API，提供给服务层进行调用。

然后，本文在这个集成框架的基础上，基于实际需求完成了一个支持多工具协同的软件开发管理的解决方案，开发了支持软件开发的应用集成流程管理系统。该系统作为流程管理平台，同时兼有设计软件开发流程，软件开发辅助服务注册，软件开发过程管理以及数据分析等功能。

最后，该系统在测试组经过严格测试，并在七个开发组中推广试用，证实了该方案的可行性。根据管理组、开发组和测试组的反馈结果，该系统比传统方式节省了每周例会的时间，以及方便了开发任务的管理，从而提高了开发效率。

关键词：软件工程；计算机辅助软件；软件即服务；工作流；流程管理

Design and Implementation of Process Management System Supporting Multi-tool Collaboration

ABSTRACT

Since the 21st century, with the rapid development of computer technology, the development of software engineering has also undergone ever-changing changes. In the traditional software engineering, there are a variety of computer-aided softwares (CAS) to help and support the software development process, the software integration in software development plays a very important role. However, in the case of using a variety of tools, there are many problems encountered in the process of integration of various tools, including the non-customized of development process, inconsistency of process data and the low efficiency of process management. How to effectively use these CAS together, including achieving the customization of software development process to support fast iteration and continuous integration of software engineering development is the subject of this paper. In view of this, a software-as-a-service application integration framwork based on workflow is proposed, trying to help custom process flexibly according to different requirements and habits. The main work of this paper is as follows:

Firstly, this paper introduces the current situation and development trend of the research at home and abroad, and introduces the related concepts such as workflow, Service-Oriented Architecture (SOA) and Software as a Service (SaaS). In this part, this paper also discusses the solutions of applying workflow to the software engineering development process, and how to combine workflow and CAS by using software-as-a-service ideas.

Secondly, according to the previous research, this paper designs a software-as-a-service application integration framwork based on workflow. The framework will be divided into five layers, namely the workflow engine layer, workflow API layer, data routing layer, service group layer and tool group layer. The architecture of the framework is analyzed in detail: the workflow engine and workflow API layer provide the functional application interfaces (API), the lower layers uses the function of the workflow by invoking these APIs to promote the circulation of the software development process, and the realization of the process-related operations; the data layer’s design allows the data of software development process flows through the workflow, emboding the important role of data layer in the architecture; the service group layer serves software development process by providing various services to workflow; the tool group layer concentrates different functions of different tools, these functions will be packaged into an interactive standard API, provided to the service group layer to invoke.

Then, on the basis of this integration framework, this paper completes a solution of multi-tool collaborative software development management based on actual requirements, and a software development process management system that supports multi-tool collaboration is developed. The system supports the design of software development process, the registration of software development services, the management of software development process and data analysis and other functions.

Finally, the system was rigorously tested in the test group and tried in seven development teams to confirm the feasibility of this solution. The feedback from the managers, developers and testers shows that this system saves the time of weekly meetings, facilitates the management of development tasks, and significantly improves the development efficiency.

KEY WORDS: Software Engineering; computer-aided software; Software-as-a-Service; Workflow; Process Management

目录

[第一章 绪论 1](#_Toc478826021)

[1.1 国内外研究现状 1](#_Toc478826022)

[1.2 课题研究背景和意义 3](#_Toc478826023)

[1.3 课题研究内容和目标 5](#_Toc478826024)

[1.4 本文结构 6](#_Toc478826025)

[第二章 课题研究的相关技术 8](#_Toc478826026)

[2.1 工作流技术 8](#_Toc478826027)

[2.1.1 工作流技术理论研究 8](#_Toc478826028)

[2.1.2 工作流引擎 10](#_Toc478826029)

[2.1.3 工作流基本概念 13](#_Toc478826030)

[2.1.4 工作流管理系统 14](#_Toc478826031)

[2.2 SOA技术 15](#_Toc478826032)

[2.2.1 SOA模型介绍和SOA特征 15](#_Toc478826033)

[2.2.2 面向服务架构的软件即服务集成开发环境 17](#_Toc478826034)

[2.3 面向服务架构的工作流管理系统 19](#_Toc478826035)

[2.4 中间件技术 19](#_Toc478826036)

[2.5 数据存储技术 21](#_Toc478826037)

[2.6 Spring MVC技术 22](#_Toc478826038)

[2.6.1 Spring总体框架 22](#_Toc478826039)

[2.6.2 Spring MVC 23](#_Toc478826040)

[第三章 架构设计 25](#_Toc478826041)

[3.1 WorkFlow Engine & WorkFlow API 26](#_Toc478826042)

[3.2 数据路由层（Data Router） 27](#_Toc478826043)

[3.3 服务群（Services Group） 28](#_Toc478826044)

[3.4 工具群（Tools Group） 28](#_Toc478826045)

[第四章 需求分析与系统设计 29](#_Toc478826046)

[4.1 需求分析 29](#_Toc478826047)

[4.2 整体系统设计 30](#_Toc478826048)

[4.3 系统模块设计 32](#_Toc478826049)

[4.3.1 工作流后台 32](#_Toc478826050)

[4.3.2 多应用集成平台 32](#_Toc478826051)

[4.3.3 流程定制平台 35](#_Toc478826052)

[第五章 任务管理系统的实现 37](#_Toc478826053)

[5.1 JAVA开源工作流框架选用 37](#_Toc478826054)

[5.1.1 主流开源工作流框架比较 37](#_Toc478826055)

[5.1.2 四大开源工作流与统一开发平台集成比较 40](#_Toc478826056)

[5.1.3 版本选择与维护问题 41](#_Toc478826057)

[5.1.4 总结 41](#_Toc478826058)

[5.2 流程可定制化方案——流程定制平台 41](#_Toc478826059)

[5.3 流程管理方案——任务管理系统 44](#_Toc478826060)

[5.3.1 流程部署模块 44](#_Toc478826061)

[5.3.2 工具服务注册 45](#_Toc478826062)

[5.3.3 流程任务管理 48](#_Toc478826063)

[5.3.4 模板库管理 54](#_Toc478826064)

[5.4 数据流转方案——数据总线 55](#_Toc478826065)

[5.4.1 数据库扩展设计 55](#_Toc478826066)

[5.4.2 框架数据流转设计与实现 59](#_Toc478826067)

[5.4.3 数据统计 60](#_Toc478826068)

[第六章 应用和测试 65](#_Toc478826069)

[6.1 设备环境 65](#_Toc478826070)

[6.2 测试环境拓扑 67](#_Toc478826071)

[6.3 测试内容 68](#_Toc478826072)

[第七章 总结与展望 70](#_Toc478826073)

[7.1 总结 70](#_Toc478826074)

[7.2 展望 72](#_Toc478826075)

[参考文献 73](#_Toc478826076)

[致谢 79](#_Toc478826077)

图目录

[图 1简化开发流程图 14](#_Toc478653062)

[图 2工作流引擎结构参考图 19](#_Toc478653063)

[图 3工作流概念关系图 21](#_Toc478653064)

[图 4工作流模板样例 22](#_Toc478653065)

[图 5 SOA模型 24](#_Toc478653066)

[图 6简化软件工程开发模型 25](#_Toc478653067)

[图 7软件开发模型下的面向服务架构 26](#_Toc478653068)

[图 8中间件应用结构图 28](#_Toc478653069)

[图 9 Spring模块划分 31](#_Toc478653070)

[图 10 MVC之间的联系 32](#_Toc478653071)

[图 11 Spring MVC控制器 33](#_Toc478653072)

[图 12基于工作流的软件即服务集成框架图 34](#_Toc478653073)

[图 13工作流架构图 36](#_Toc478653074)

[图 14支持多工具协同的流程管理系统架构图 41](#_Toc478653075)

[图 15流程定制平台 53](#_Toc478653076)

[图 16流程部署模块 55](#_Toc478653077)

[图 17服务群与工具群 56](#_Toc478653078)

[图 18工具服务注册 57](#_Toc478653079)

[图 19工具信息展示 58](#_Toc478653080)

[图 20工具服务列表 59](#_Toc478653081)

[图 21待处理任务展示 60](#_Toc478653082)

[图 22已发起流程展示 61](#_Toc478653083)

[图 23流程列表 62](#_Toc478653084)

[图 24 LXSA-41流程执行详细信息 62](#_Toc478653085)

[图 25流程流转数据展示 64](#_Toc478653086)

[图 26模板库公有模板 65](#_Toc478653087)

[图 27简化开发流程 70](#_Toc478653088)

[图 28支持多工具协同的流程管理系统数据流转图 71](#_Toc478653089)

[图 29工具使用统计 72](#_Toc478653090)

[图 30工具服务使用统计 73](#_Toc478653091)

[图 31工具接口流量统计 74](#_Toc478653092)

[图 32模板发起统计 75](#_Toc478653093)

[图 33工作流功能测试测试环境拓扑图 79](#_Toc478653094)

表目录

[表 1 SOA协议栈 28](#_Toc478653701)

[表 2开源工作流与统一开发平台集成比较表 51](#_Toc478653702)

[表 3流程定义文件样例 53](#_Toc478653703)

[表 4支持多工具协同的流程管理关键数据表 67](#_Toc478653704)

[表 5 iscas\_user数据表 67](#_Toc478653705)

[表 6 iscas\_task\_defined数据表 68](#_Toc478653706)

[表 7 iscas\_task\_instance数据表 69](#_Toc478653707)

[表 8 iscas\_tool数据表 70](#_Toc478653708)

[表 9测试设备环境表 77](#_Toc478653709)

[表 10功能测试结果表 80](#_Toc478653710)

# 绪论

## 国内外研究现状

自从20世纪80年代互联网产生以来，计算机在人们生活中的扮演的角色越来越重要，各种各样的软件帮助人们实现各种各样的功能，大大提高了生活的便利程度。随着人们生活的改善以及需求的提高，不同的软件开发需求也将软件工程带入了一个全新的时代。面对纷繁复杂的开发需求和功能需要，完善的软件工程方法能够让开发事半功倍，大大改善和提高软件开发效率和软件开发质量。

在互联网技术飞速发展的今天，软件开发的模式也发生着日新月异的变化。在传统的软件工程中，软件开发生命周期是一个十分重要的概念。随着软件开发节奏日益加快，快速迭代式的开发逐渐成为主流。随着软件即服务概念的提出和普及，越来越多的开发者们在软件开发的过程中使用计算机辅助软件来帮助开发。

在这样环境下的计算机辅助软件日益增多，功能各异。针对软件工程开发领域，根据软件工程的不同阶段，可以对这些工具进行分类，每一类工具提供相同或者相似的一种服务。将软件即服务的思想应用到开发过程中，可将项目开发过程分成项目管理模块，开发模块，测试模块，文档管理模块，集成模块，验证模块等等。例如项目管理模块，就会有很多项目管理的软件（jira, git, jenkins等）来方便软件开发人员去定制项目，定制任务计划，确定项目人员以及其他工作等等。

但是这些软件或者系统基本上是独立运行的，相互之间缺乏必要的交互、协作与感知，从而使得它们在协调多个人的活动、以协同完成某个共同的任务方面有着比较严重的不足。在现代的软件开发环境下，多人多地点参与的项目也变得日渐增多，复杂情况下的软件开发需要考虑信息的对等、开发流程的定制、开发任务的分配以及管理等等问题。在协同开发过程中，面对的开发流程和工作环境各不相同，单纯依靠传统的软件开发模式，信息和数据的流转基本靠人工来进行，效率较低。这也就是工作流管理系统（Workflow Manag ement System, WfMS）自进入90年代以来吸引了来自研究机构与产业界广泛关注的重要原因[1-4]。通过研究软件开发流程，将开发项目进行解耦，降低各个模块之间的耦合度，来解决此类问题。

工作流技术是一种优秀的流程管理技术，能够实现高效并且完善的业务流程管理。就当前的社会发展来看，信息化变得越来越重要，信息流通更是人们所关心的话题。在软件开发过程中，信息流通的效率和成本将会大大影响软件开发的模式和质量。工作流作为最优秀的流程管理技术，对于信息流转可以提供高效的组织管理，对于业务流程可以实现进行标准化、规范化和定制化，可以实现不同结构和模块之间的信息高效流通。

工作流这项技术的起源可以追溯到20世纪70年代中期左右，在当时兴起的办公自动化领域能够找到它的身影。在当时互联网刚刚起步，计算机还没有全面普及的年代，工作流的思想萌生在帮助办公流程自动化流转之中。然而受网络交流的匮乏，理论基础的不坚固等因素的影响，这项新技术在当时并没有大放异彩。

在20世纪90年代之后，计算机网络发展迅速，个人计算机也逐渐普及开来，信息传输和共享逐渐成为人们关心和重视的话题。在这样的时代大背景下，工作流技术有了发展的空间和机会，第一个工作流技术标准化的工业组织也相应产生——工作流管理联盟（Workflow Management Coalition, WFMC）。工作流管理联盟制定了一系列技术标准，包括工作流参考模型。与此同时，专家学者们也对工作流的研究十分感兴趣，致力于开发各种不同的工作流引擎和工作流管理系统，针对不精通行业的不同需求进行设计和开发。经过一个发展的黄金期之后，工作流技术越来越成熟，相应的技术规范和模型已经被大多数人所认可，而工作流也从一个技术标准逐渐转变成为一种思想。成熟的工作流思想被应用在各种不同的场景下，结合各个行业相关的技术来解决不同的问题。

工作流在企业中的使用所带来的优点是显而易见的，企业在运营过程中，对业务流程的管理、企业资源的分配、工作效率的提高以及工作成果考核的量化等等都是企业十分关心的问题。怎样在处理这些问题上有一套灵活完成的解决方案来减少企业的损耗，同时增加企业的利润是问题的关键。

在企业中使用工作流，能够充分利用互联网的优势，通过计算机资源来完成企业业务流程的实施，减少人工的损耗和失误，同时能够大大地提高业务处理的速度和效率。工作流能给企业带来的优点和长处如下：

1）通过计算机的使用，员工能够在计算机上获知自己的任务和待处理事项；同时，新的待处理事项能够通过网络自动传递到计算机上。

2）对于企业的业务开发流程，使用工作流之后，不需要员工了解流程的每个细节，通过工作流自动流转能够简化对员工的业务流程培训。

3）工作流所带来的自动化和信息化业务处理，能够将处理数据记录和存档在相关设备上，做到对流程相关操作和产出有迹可循。

4）企业使用工作流之后，结合计算机能够对业务流程中的数据进行收集和分析，方便企业得到业务处理的效率和进度信息，同时能够提供一些成果考核量化指标。

5）由于计算机在处理流程中所扮演的重要角色，免去了靠纸张传递信息的方式，达到了无纸化办公的目的。

6）通过工作流实现的业务流程可以方便地进行调整和改进，有利于企业业务的提升和办公效率的提高。

工作流发明的初衷是办公自动化[5]。后来逐渐抽象成为工作流思想，并且应用在各个行业，例如金融行业、制造行业、土木行业、医药行业等等，来提高工作效率和规范流程。在一些大型企业中的应用更加普遍，带来的经济效益更加明显。例如刘龙等人研究的工作流在中职学校数字化校园管理中的应用研究[6]；冉晓蓉等人研究的基于审批单的工作流在施工企业信息化管理中的应用[7]；李志民等人研究的基于工作流的大型仓库物流监控系统[8]；李鹏等人研究的基于工作流技术的办公自动化系统在医院中的应用[9]；张宏等人研究的基于系统动力学的高速公路维护管理系统[10]，等等[11]。本文的研究目标是软件工程领域，针对软件开发日益成熟的开发模式和广泛使用的软件开发辅助工具，提供一种能够提高软件开发效率，有利于软件开发流程规范性的软件工程辅助工具集成方案。

随着科技的发展，企业业务的处理中所涉及的任务处理逐渐采用一些应用软件来支持和代替。这些应用软件或者系统基本上都是企业外部引进，用来帮助企业处理和解决相关问题的第三方工具。企业通过一定的方式直接采用或者通过API集成这些工具，来提高生产力和工作效率。工作流管理系统可以将企业的流程任务关联到第三方工具上，通过工作流来调用第三方工具提供的服务。基于工作流的企业级工具集成策略不失为一个很好的企业通过工具提高生产力的解决方案。

软件工程开发与工作流的结合，能够改善软件开发过程中的不同工具之间信息冗余的问题，能够提供流程可定制的软件开发模式，同时能够对软件开发过程中流转的数据进行记录、分析和实例化。工作流应用于软件工程行业，能够帮企业解决开发过程中任务和流程管理的问题，减少信息冗余，提供更加便捷的沟通方式，从而提高企业的经济效益。

## 课题研究背景和意义

在互联网尚不发达的单机系统时代，依靠个人的编程能力，一个程序员就能够完成一个功能完整软件的编写，而这种情况往往还比较常见，很多我们所熟知和经常使用的软件都是从这种开发模式发源起来，就像WPS，CCED这样的现在对我们生活有很大帮助的软件。但是随着互联网的飞速发展和信息大爆炸时代的来临，单兵作战的软件开发模式已经逐渐变得落伍，被更加高效的开发模式所取代。随着软件工程的崛起，软件开发逐渐趋向于规范化和流程化，各种各样的软件产品应运而生，包括很多计算机辅助软件，通过软件工程开发的软件来支撑和帮助软件工程更加高效的开发，形成一个相辅相成的良性循环。

随着分布式系统的出现和发展，多人多地点多协作的软件开发模式软件开发越来越收到人们的欢迎。同时，将软件开发模块化，更加有利于软件开发的效率和后期拓展维护。因此，不同开发小组和不同地点之间的开发人员的信息交流和相互协作成为软件开发过程中一个十分重要的环节。在一个复杂庞大的系统开发中，多个模块多个服务之间的交叉调用，不同网络之间的信息交互，软件开发人员资料交互安全等等问题，都随着分布式协同软件开发而产生。软件开发过程的难度和复杂度也随之提高。

尽管伴随着不同问题的产生，随着软件工程的发展，多人员多地点多模块的协同开发模式逐渐成为软件工程开发的主流，解决这些附带的问题成为软件工程领域的各位工程师和专家学者们所重点思考的问题。例如怎样更加方便的建立开发流程，怎样解决开发过程中不同人员的任务分配问题，怎样提高不同开发人员之间的信息交流效率和安全性，怎样让开发人员和项目管理人员更好掌握开发流程信息和开发项目的进度等等问题。

软件开发工具集成，即Tool Integration for Software Development（TISD），是指在软件开发生命周期中，通过使用各种不同的软件开发辅助工具，来支撑软件开发过程中的各种需求，例如信息管理、人员分配、任务调配、资料存储以及安全监控等等。并且随着开发过程和开发环境的改变，各种辅助工具和软件技术随之会进行变动、发展、完善和不断集成，软件开发工具集成逐渐变一套完整的方法论。在1968年NATO的学术会议上，“软件工程”这个术语第一次出现在人们的视野，意味着开始提倡软件开发的工程化，通过工程化的方法来实现各种不同软件的开发。

实现软件开发工程化需要有很多前提，其中包括：

首先，需要有一套科学合理的的软件开发理论，以及成熟的软件开发技术。在这个前提下，软件开发流程的合理性、软件开发过程模型以及软件开发质量保证方法与技术等等也是不可缺少的一部分。

其次，软件开发工程化需要有一定的开发规范来制约不同的开发过程。例如开发周期的制定，开发文档的撰写，开发进度的掌控以及开发产品质量的监督等等方法与技术。这些规范对于评价一个软件开发过程十分重要。

最后，利用计算机辅助软件来帮助软件开发过程工程化。通过建立软件开发辅助软件集成环境，来协助软件开发工程化。

对于开发一个复杂大型的系统来说，利用软件工程的方法至关重要。如果没有软件工程方法的支持，开发过程中的规范标准不统一，开发进度受影响，开发产出质量不能得到保证，在一定程度上会影响开发的进度和效率。采用软件工程开发方法，建立集成的软件辅助工具开发环境，通过辅助工具来对软件开发生命周期进行管理，同时在开发过程中提供软件开发所需要的不同的帮助，从“作坊式”而开发转变为“工厂式”在软开发，这会让庞大而复杂的开发任务变成井井有条的开发生命周期，会大大提高软件开发速度，提高开发产出质量，同时也有利于开发后期的质量监控和扩展维护。

## 课题研究内容和目标

在软件工程开发过程中，可以将软件的生命周期分成很多阶段，在不同的阶段，都有不同的软件辅助工具去支撑软件的开发[12]。在不同的软件开发环境和要求下，选择不同的软件开发辅助工具将会大大影响软件的开发过程。同时，在整个软件开发的过程中，不同开发过程之间信息和数据的流转传递是否高效，也将会对整个软件开发生命周期产生质的影响[13]。

在软件即服务的思想下[14]，可以将软件开发不同阶段的辅助工具抽象成一个个不同的服务，而这些服务可以由不同的工具来提供。从广义上来说，可采用四级模型来说明软件即服务（Software as a Service, SaaS）应用的成熟度，对于第二级成熟度而言，软件及服务需要可配置[15]。能够灵活选择某个软件提供的服务，同时能够选择不同的服务来构成整个软件开发生命周期将会对软件开发带来很大的裨益。

为了更简明地阐述，本文简化软件工程开发模型为三个步骤：项目管理，代码管理和缺陷管理。每个步骤下，都有各种特色的计算机辅助软件来辅助软件开发过程，这些软件的集成在软件开发中扮演了十分重要的角色[16]。

软件开发人员根据需求，设计的流程如图：



图 1简化开发流程图

软件开发者在设计开发流程的时候可以自由选择不同的节点来完成不同的功能，而这些节点的背后就是一个个计算机辅助软件提供的功能。在这样一个灵活定制的流程之下，高效地实现软件开发过程的自动流转。

现有研究中，软件开发辅助工具的集成已有相关研究，但是大部分的研究都是着力于辅助工具的硬性集成，偏向于研究一个支持某些软件开发辅助工具的开发平台，例如郭慎平研究的基于Eclipse的嵌入式开发工具的研究与实现[17]。在该平台上对软件开发过程中的任务和数据进行管理，包括任务分配和权限管理等。

面向服务开发框架致力于让软件开发过程更加灵活，根据不同的开发环境和实际情况可以定制最有利于当前开发生命周期的软件辅助工具。在该框架下，不同服务可以自由选择搭配，提供相同服务的工具可以自由注册和选择。同时，将工作流引入到框架中，来提高软件开发中信息和数据在软件生命周期中的流转的效率。

本文的研究目标是规范化软件开发流程，使之能够在软件开发过程中进行流程化的管理，基于工作流来定制软件开发流程，来提高软件开发的效率。在流程中对软件开发工具传输的数据进行传输和交互，降低每个开发软件开发工具之间的耦合性，定制可视化工作流。利用J2EE开发平台构建轻量级的工作流引擎，设计出工作流原模型和工作流定义模板，开发支持软件开发的多工具集成的工作流管理系统。在该系统中能够进行服务注册、接口对接、流程定制和流程实例的管理，并在实际情况下进行应用，验证该系统的功能。

## 本文结构

本文遵从以下的组织结构对论文进行撰写：

第一章为绪论。分别介绍了本文的研究背景、目标和意义，同时也总结了国内外现在的研究现状和发展趋势。

第二章为课题研究的相关技术。详细介绍了本文涉及的工作流技术，SOA技术，面向服务架构的工作流管理系统，中间件技术，数据存储技术以及Spring MVC技术。详细分析了各项技术的特征、原理以及相关运用。

第三章为架构设计。本章提出了本文的一个重点框架内容：基于工作流的软件即服务应用集成框架。该框架从五层架构上来解决软件开发过程中的计算机辅助软件协同工作的问题。本章详细介绍了该框架的构成，以及架构每层的详细作用。

第四章为需求分析与系统设计。基于第三章的框架设计，本章提出了软件开发过程中多工具柔性集成的解决方案，设计了一个支持软件开发的应用集成流程管理系统。本章内容包括该系统的实际需求分析，以及该系统的架构设计。本章针对该系统设计将整个解决方案分成了三个模块：工作流后台、多应用集成平台以及流程定制平台，方便后期的开发与实现。

第五章为支持软件开发的应用集成流程管理系统的实现。本章结合前面的系统需求分析和设计，以及第二章介绍所采用的相关技术，开发完成了该系统，并投入到实际环境中进行使用。本章还详细介绍了系统开发完成之后所具备的相关功能。

第六章为应用和测试。针对开发完成的系统，本章进行了完整规范化的测试，并给出了测试的结果总结。证明该系统的开发达到了设计的需求和标准。

第七章为总结与展望。对本文内容进行了一个总结性的概括，同时对本文所具备的不足和未来工作方向做了一个简单的计划。

# 课题研究的相关技术

## 工作流技术

工作流技术最早的出现是在二十世纪七十年代中期，萌发自办公自动化相关领域的技术出现和研究工作的进展。工作流思想的出现略早于工作流技术在实际生产中的应用，早在1968年，弗理兹•诺西克（Fritz Nordsieck）阐述过通过信息技术实现工作流程自动化的思想。

随着工作流技术的发展，一系列与工作流相关的标准也随之成为人们关心的话题。制定统一的标准有利于研究学者们的技术探索，也有利于工作流技术在实际中的运用。因此相关研究学者在1993年成立了工作流管理联盟 （Workflow Management Coalition）。作为工作流技术的标准化组织，WfMC给工作流的定义如下：工作流是一类能够完全或者部分自动执行的经营过程，根据一系列过程规则，文档、信息或任务能够在不同的执行者之间传递、执行[18]。

从WfMC组织给工作流作出的定义来看，工作流模型是一个将业务流程化，并且使用计算机去实现的过程模型，利用先进的计算机技术和网络信息交互方法来实现业务过程，使得业务过程中的任务、数据、信息便于管理，从一定程度上减少了人工操作，提高了工作效率。，而且能够让业务逻辑与流程逻辑相互分离，规范化了业务流程的实现，降低了业务流程复杂程度以及业务逻辑对使用人员的要求。通过实现流程管理系统，对企业开发流程进行一个比较好的管理与完善。

### 工作流技术理论研究

工作流技术的理论研究历时悠久，从工作流思想提出至今已经过去了接近五十年。在长时间的发展过程中，工作流技术的理论研究已经涉及到工作流思想的每一个方面，每一个细节，十分广泛。它包括了一个产品化的工作流所能够顺利工作的所有层面。接下来从五个方面来介绍工作流的技术理论研究。

* 工作流建模

工作流技术发展至今，最基础的技术从未动摇——流程建模。怎样将一个企业的实际业务通过工作流的形式表现出来？不可避免的就是根据企业的业务逻辑，建立工作流模型。通过建立成约定的工作流模型，才能够交与计算机进行识别、分析和处理。国内国际上很多专家学者都在研究关于流程建模的技术，其主流成果有以下四类。

第一类，基于活动网络图的工作流模型（Workflow model based on activity network diagram）[19-20]。这种工作流模型对用户十分友好，提供图形化显示，直观易懂，入门门槛不高，但是却不利于计算机进行语义分析。第二类，基于Petri网的工作流模型（Workflow model based on Petri net）[21-22]。这一类工作流模型入门门槛很高，因为基于Petri网的模型语义严格，专业性较高，需要一定的技术基础才能熟练掌握和使用。第三类，基于行为理论的工作流模型（Workflow model based on behavior theory）[23]。这类工作流模型是在行为理论的基础上，根据行为的过程定义，结合“人”这个实体在工作流中所扮演的角色，提出的一种协作过程的建模方法。第四类，基于事件驱动的过程链模型（Process chain model based on event-driven）[24-25]，这一类流程模型更加灵活和适应日常业务需要。根据流程中时间的产生和处理，驱动接下来流程的流转。流程中活动的设立以及活动的出发条件十分重要。

* 工作流模型验证

工作流模型验证已经成为工作流在实际应用中一个十分重要的步骤[26]。通过一定建模方法所建立的工作流流程模型必须得到验证之后才能够进行使用。好的验证方法能够帮助建模方法更加完善，好的建模思路也能够给验证方法一定的启发，两者相辅相成。如果没有对模型进行验证而将模型部署到实际生产环境之中，所带来的损失和风险将会是不可估量的。从验证严格性方面来划分，可以将工作流模型验证分为三个类型：语法验证、结构验证和语义验证[27]。在这三种验证类中之中，语法验证比较简单，实现难度较小，容易上手。而结构验证属于静态的方法，中外研究学者有很多这方面的研究，例如模块化可达图的验证方法[28]、化简方法的工作流过程模型验证[29]、邻接矩阵法[30-31]等等。语义验证比较复杂且困难。但是语义验证范围广泛、较为严格，使用场景广泛。Arpinar等研究学者给出了一个基于可用语义信息的工作流正确性标准的定义，并针对这个定义给出了判定方法[32]。

* 流程任务分配

基于工作流去处理企业相关业务的时候，工作流所推动的流程中的每一个任务的分配关系到流程实施的效果和企业产出的效率。设计完备的流程经过工作流引擎部署之后能够自动进行业务任务的分配，并且根据流程中所设定的调整规则进行调整。在中外学者的研究中，工作流任务分配方法大概有以下几种方法：直接分配法、基于负载平衡和经验值的分配方法[33]、面向团队的工作流任务分配方法[34]、基于位置的工作流任务分配方法[35]以及按照需求的工作流任务分配方法[36]等等。

* 权限管理

工作流所串联的是企业业务之间的任务，流程中的权限管理与业务息息相关。企业业务流程中，复杂的业务流程设计人员众多，任务繁杂，精密的权限管理就显得十分重要。工作流的访问控制模型主要分为以下几种：基于任务的访问控制模型[37-38]、基于角色的工作流权限访问控制模（Role-based Access Control, RBAC）型[39]、基于角色和任务的访问控制模型（Task-Role-based Access Control, TRBAC）[40]等等。

* 流程复用

不同的业务需要设计不同的流程，但是不同的流程之间会有一些共性，如果将流程按照小组进行分类的话，业务相似的流程应该属于同一个类别，并且设计出来的流程差别不会太大。为了解决这种流程重复设计的问题，采用流程复用将会大大提高流程设计的效率，同时也会对一些常见的业务流程提炼出规范化的模板。流程复用的思想起源于软件设计，将经常出现或者使用的业务处理单元可以被抽象成为独立的流程节点，这些流程节点可以看做是公共的流程单元，在设计流程的时候，通过使用这些公共的单元节点，省去了独立设计功能重复节点的工作。对于所有设计好的工作流流程，都将其存入工作流模板库，通过对模板库的管理来实现工作流模板复用。当用户需要设计一个流程的时候，可以从工作流模板库中的模板进行挑选，进行二次修改再利用。目前的研究中尚未有成熟的方案来给出模板库中的模板与用户需要设计流程相似度的匹配。

### 工作流引擎

工作流引擎是工作流在企业管理应用中最重要的核心部分，工作流所具备的流程部署、流程执行、任务分配、人员调整、工作流查询和监控等核心功能，以及工作流在一些场景下所提供的数据流转功能、数据统计和分析等等附加功能，都是由工作流引擎提供或者间接实现。一个工作流系统可能会包含多个工作流引擎，但是一个引擎会独立地处理独立的流程。工作流引擎的作用如图2所示。



图 2工作流引擎结构参考图

工作流引擎从类别上可以分为嵌入式和独立式两类。嵌入式的工作流引擎与业务系统关联在一起，工作流引擎连同业务系统中其他相关接口和软件一起构成了完整的业务系统。工作流引擎为业务系统中的业务部分提供工作流服务，业务系统结合工作流引擎和其他系统软件的功能服务对外提供业务服务。一般情况下，工作流与业务流程息息相关。工作流的设计依赖于业务流程，可以将工作流视为业务层流程的逻辑抽象。嵌入式的工作流引擎通过对业务层提供功能API接口，业务层通过API接口调用工作流的相关功能，来推动业务流程的执行和流转。

独立式的工作流引擎独立于业务系统之外，流程的设计和开发由业务人员完成。由于耦合性低，所以例如查询、分析、报表等基本功能都是工作流引擎定义完成，另外一些其他的通用功能也都是由工作流引擎制定。由于没有其他中间件提供的服务，引擎还需兼具多线程同步、网络间的消息通讯等功能，对开发人员的专业素质有一定的要求。由于独立式工作流引擎不与业务系统耦合，业务系统都是通过远程调用的方式去使用工作流引擎的服务，对引擎的响应速度和负载均衡有一定的要求。

从上个世纪发展至今，工作流已经得到了很大的发展，现在也有很多成熟的开源工作流引擎，本文建立的基于工作流的软件即服务应用集成框架（Workflow-based Software-as-a-service Application Integration Framework, WSAIF）准备采用开源工作流引擎作为工作流内核，来推动任务的流转和信息的传递。先介绍主流的工作流引擎。

* jBPM[41]

jBPM全称为java Business Process Management，是Jboss社区的开源项目，现在被业界广泛使用。jBPM具有很好的灵活性，是一个灵活可扩展的工作流管理系统。jBPM所使用的业务流程使用强大的可扩展标记语言（Extensible Markup Language, XML）表达，并且它能够在服务器端输入业务流程，然后在流程部署的时候保留在流程实例中。jBPM将工作流所带来的应用开发便捷性与成熟的企业应用相结合（EAI），它还可以与Hibernate、Spring集成，在java开发中使用十分广泛。

* OSWorkflow[42]

Osworkflow是完全用java语言编写的开放源代码的工作流引擎，具有显著的灵活性，完全面向有技术背景的用户。用户可以根据自身的需求利用这款开源软件设计简单或是复杂的工作流。通过使用OSWorkflow，用户就可以把工作中心放在业务和规则的定义上，而不需通过硬编码的方式实现。用户可以以最小的代价把Osworkflow整合到自己的程序中来。

* Enhydra Shark[43]

Shark是Lutris公司开发的开源工作流引擎，它遵从了WfMC指定的相关标准。Enhydra Shark的定义语言为XPDL，并且对用户提供一个可视化的流程创建工具，使用十分方便。

* Activiti5

Activit5继承了jBPM4的所有优点，支持最新BPMN2.0规范，实现了流程的可视化以及创新的Activiti Cycle协作组件，此外，通过与Mule的集成加强了其集成能力。

* OpenWFE[44]

OpenWFE是一个用java写的工作流管理系统，对外开源。它的组件包括一个工作流引擎、一个基于web的流程设计环境（Droflo）、一个web客户端以及一个web接口的资源管理系统（UMAN, for User MANagment）。

除此之外，还有其他的开源工作流引擎，如Werkflow、OFBiz、Flow4J、ObjectWeb Bonita、OBPM等等，限于本文篇幅，在此就不一一介绍。

### 工作流基本概念

图3展现了工作流各个概念以及这些概念之间的联系。



图 3工作流概念关系图

业务流程：实现业务最终目标的各种活动的合集。活动之间按照一定的逻辑相互连接，例如请假过程、项目开发过程、医院就诊过程等等。

过程定义：通过过程定义语言对过程进行形式化的描述。工作流通过对工程定义的读取和识别，完成流程运行的自动化。一个过程可以被分解成为多个子过程，每个过程包含不同的活动，过程以及子过程的定义就是描述这些活动起讫关系网络，同时也包含一些与个体行为有关的信息，例如相关操作人员信息，与活动有关的数据等等。

工作流模板：一个图形流程定义，代表工作流的流程逻辑元素以及各元素之间的关系。工作流模板跟流程图很类似，将一系列步骤按照一定的逻辑组织在一起，按照工作流定义语言形成一定的文本结构，能被工作流引擎所识别。工作流模板的定义方式最常见的就是图形拖拽，工作流定义语言也有很多种， XML语言比较常见。图4给出的就是工作流模板的样例。



图 4工作流模板样例

工作流管理系统：对工作流进行管理，包括工作流定义的解读、工作流实例的流转、工作流相关信息的管理和展示等等。

流程（活动）实例：每一条流程定义被工作流引擎部署之后会实例化为流程实例，同一条流程定义能够被部署和实例化多次。每个实例具有完整逻辑、能够独立执行的线程，与操作系统建立一个线程的概念类似，工作流管理系统可以对它进行读取、展示和管理。

活动：工作流中任务的描述。一般被分为手工操作和自动处理两种类型。工作流通过活动来完成相应的任务，手工操作需要人员的参与，自动处理一般是工作流引擎根据流程定义的逻辑和流转条件来自动流转完成任务。

子流程：父流程所包含的具有清晰完整逻辑的活动，可以建立子流程，方便任务的执行和管理。

### 工作流管理系统

工作流管理的主要目标是通过对相关的人力资源、信息资源的调用，来协调和完成业务过程中的各个环节，让业务按照一定的逻辑顺序依次执行，实现业务流转的自动化，从而提高业务完成的效率。

从计算机支持的协同工作的角度看，工作流管理系统是一种支持开发人员异地、异步协作的系统，它从支持同步的、非结构化协作（如各种桌面会议系统[45]、协同设计支持系统[46]、协同编辑系统[47]）向着支持形式化、结构化协作的进一步发展。

工作流管理系统主要具备以下三个功能特征：

* 工作流定义功能

主要是对业务处理过程的计算机定义，提供了一种或多种分析、建模、系统定义技术，将一个现实世界的业务处理过程转换成计算机可处理的定义。最终的定义叫作过程模型、过程模版或过程定义，可以表现为文本、图形或自然语言符号。

* 运行控制功能

对过程的定义进行解释，创建并控制过程的运行实例，调度过程的各种行为步骤，调用适当的人工和IT应用程序资源；工作流管理系统的核心部件就是工作流管理控制软件（工作流引擎）。

* 运行交互接口

提供与人员或IT应用程序工具进行交互接口来处理各种活动步骤，交互接口对于活动间的控制传递是必须的，如确定过程的状态，调用应用程序工具，传递应用程序数据等[48]。

## SOA技术

SOA，即Service-Oreinted Architecture，是一个组件模型，它将不同应用的不同功能定义成服务，通过服务之间定义的良好接口和协议进行对接和耦合，实现相关绑定之间的分离。

### SOA模型介绍和SOA特征

SOA模型有三个重要的组成部分：服务提供者、服务请求者和服务注册中心。如图5所示：



图 5 SOA模型

SOA是一种企业级的IT架构，用于按需请求和提供企业资源。在一个SOA架构中，企业资源可用于价值网、企业、业务线（通常跨越企业内或跨多个企业的多个应用和系统）中的参与者。它包括同一组业务中相一致的IT服务，它们共同实现组织的业务流程和目标，设计者可以将这些服务编排为复合应用程序，并通过指定相关的标准协议来调用它们。

服务是指具有外部化服务描述的软件资源，是由外部化描述定义的接口，接口的实现即为服务的本身。

服务提供者实现服务描述，并且向服务请求者传递服务质量要求。

服务请求者可以搜索、绑定和调用所定义的服务。服务自身也可能是调用者，通过调用其他的服务来实现自身服务的功能[49]。

服务应该由声明性策略来管理，并因此支持动态可重新配置的架构。业务敏捷性是由灵活的IT系统获得的，主要是通过分离SOA提供的接口，实现和协议绑定。

服务提供者需要在一个公开的地方发布他们的服务描述，这个地方就是服务注册中心。服务请求者可以在服务注册中心查询服务提供者所发布的服务描述，得到服务相关的绑定信息。

SOA有很多便于开发的特征，所以在软件开发过程中使用十分广泛。其中核心特征包括服务接口化、服务透明化、服务松耦合化、服务组合化以及服务注册化。基于这些特征，服务提供者能够便捷地提供服务，通过提供清晰明了的服务描述就能够让服务请求者明白所有服务所对应的功能；服务请求者能够根据相应的服务描述，在不用知道服务具体网络传输地址的情况下，可以透明地使用某项服务。而且通过服务注册中心的灵活设置，达到了不同服务之间的松散耦合，实现了对业务逻辑的解耦。

### 面向服务架构的软件即服务集成开发环境

WSAIF框架致力于让软件开发过程更加灵活，根据不同的开发环境和实际情况可以定制最有利于当前开发生命周期的软件辅助工具。在该框架下，不同服务可以自由选择搭配，提供相同服务的工具可以自由注册和选择。

软件开发过程中，至少有三个十分重要的部分：项目管理，代码管理，缺陷管理。项目管理包括项目的目标，项目的分工，项目进度的把控，项目质量的跟踪，以及项目资源的配置等等。代码管理包括代码的存放，代码的正确性验证，代码的安全性保证，代码的整合，代码的上线和维护等等。缺陷管理包括缺陷的发现，缺陷的记录，缺陷的跟踪和管理，缺陷的验证等等。每一阶段的资源管理，任务调度，信息流转都十分复杂，整个软件开发过程都围绕这些方面来进行。如何高效处理这些过程将会给软件开发带来很大裨益，大大减少过程之间的交流成本，降低资源消耗，同时提供更加安全可靠的开发环境。

我们可以简化软件工程开发模型，拟分为三个步骤：项目管理，代码管理和缺陷管理。软件开发人员根据需求，设计的流程如图6：



图 6简化软件工程开发模型

我们将该流程的三个步骤分别定义为三个服务（Service）：S1，S2，S3，每个组选择一种服务来支撑该组工作的进行。S1为项目管理服务，S2为代码管理服务，S3为缺陷管理服务。

在项目管理服务S1中，有不同的功能（Function），比如创建一个任务，结束一个任务等等，我们将这分别定义为F1，F2，F3...Fn。

每个功能，能够由不同的项目管理工具（Tool）来实现，比如JIRA或者Zen Tao，都能提供任务创建，任务结束等等功能，我们将这些工具分别定义为T1，T2，T3...Tn。

基于简化之后的软件工程开发模型，面向服务架构的基本结构如下：



图 7软件开发模型下的面向服务架构

S1：项目管理服务；S2：代码管理服务；S3：缺陷管理服务。

以代码管理S2为例：

不同功能F1...Fn包含了代码管理中需要的create、commit、pull、fork、delete、compare、merge等所有功能。

每个服务可以由不同的工具（Tool）T1、T2...Tn来实现。

以commit为例，可以使用gitlab、SVN或github实现，分别为T1、T2和T3。

S2-F1-T1：代码管理中，T1实现create功能F1；

S2-F2-T2：代码管理中，T2实现commit功能F2；

S2-F3-T1：代码管理中，T1实现pull功能F3；

……

S2-Fn-Tm：代码管理中，Tm实现服务Fn。

在这样灵活的配置下，软件开发者在设计开发流程的时候可以自由选择不同的工具来提供不同的服务，通过不同的服务来实现软件开发的自动流转。

## 面向服务架构的工作流管理系统

从面向服务的角度看，为了更好地在分布式异构环境下实现企业的业务流程，满足企业基于Internet的管理和企业间合作的要求，面向服务架构（SOA）的工作流管理系统也是近年来人们关心的研究热点[50-51]。

SOA能够对耦合度较松的系统提供支持，通过WEB服务技术提供灵活的服务方式。WEB服务是面向服务架构的技术基础，面向服务架构所实现的协议都应该与WEB服务技术体系相符[52]。SOA 协议栈是一个层次结构，下表给出了SOA协议栈，表左边介绍了SOA 的功能以及功能所对应的协议，表格右边展示了与其相关的体系结构的其他方面[50]。

表 1 SOA协议栈

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 协议规范 | 实现功能 |  | | |
| BPEL | 将服务聚集成流程 | Management  管理 | Security  安全 | QoS  服务质量 |
| UDDI | 服务发布与发现 |
| WSDL | 服务描述 |
| SOAP | 基于XML的服务消息 |
| HTTP,SMTP,FTP | 消息传输 |

## 中间件技术

中间件（MiddleWare），是连接系统软件和应用软件的软件，以便于软件各部件之间的沟通，能够解决软件之间的互操作问题。系统软件包括操作系统、数据库系统以及网络等等。中间件的应用如下图所示：



图 8中间件应用结构图

中间件根据不同的分类标准有很多分类方式，根据国际数据组织（IDC）的标准，中间件可以被分为六类。分别是终端仿真与屏幕转换中间件（TSCTM）、数据库中间件（DM）、远程过程调用中间件（RPC）、消息中间件（Message-Oriented Middleware, MOM）、对象中间件（OM）、基于对象请求代理的中间件（ORBM）[53]。

在这些分类中，面向消息的中间件采用异步通信模式，发送消息者不用等待接收者的回应，而接收者也不必在接到消息后立即对发送者的请求进行处理，双方是松耦合关系。消息之间的逻辑关系和结构由程序本身定义[54]。

中间件本质上是实现了整合异构系统中的各种资源，将分布式的应用对象的业务逻辑细节剔除，将不同分布式系统提供的服务在统一的层面提供给相关应用。所以，中间件的应用，能够让应用更好地专注到业务逻辑上，实现多工具协同工作。

## 数据存储技术

随着互联网技术的发展，信息数据量呈一个爆炸性的方式增长。庞大的数据量也给我们的存储设备带来了挑战。而且，现今的数据还具有多样性和高价值的特点，常规模式进行的数据存储已经渐渐被一些新的存储方式所取代。

传统的存储模式分为直接附加存储（Direct-Attached Storage, DAS）[55-56]、网络接入存储（Network-Attached Storage, NAS）[57-58]、存储区域网（Storage Area Network, NAS）[59-60]、对象存储技术（Object-Based Storage, OBS）[61-62]。接下来分别介绍下不同存储的特点和适用场景。

* 直接附加存储

在这种直接式存储方式下，存储设备一般是通过小型计算机接口连接到服务器。这种存储方式可靠度比较高，但是由于没有存储操作功能，无法进行数据的备份和共享，因此可维护性和可扩展性不高。

* 网络接入存储

这是一种专门为网络而生的存储方式，通过与网络媒介直接相连的方式实现数据存储。这种存储方式包括存储设备和内置系统，通过IP地址来提供数据存储的管理。这种存储方式安全性较高，比较适合一些中小企业，但是这种存储方式对网络也有一定的要求，对系统负担有一定的要求。但是随着网络带宽的增加，这种存储方案的发展也十分迅速。NAS采用TCP/IP协议进行数据传输，因此兼容性较好。

* 存储区域网

在这种存储方案下，不同的存储设备通过高速连接的网络连接到服务器上，形成一个存储网络。高速的网络会让这种存储方案十分高效。但是这种存储方案也有缺陷，存储设备十分昂贵，成本较高，再加上存储技术先进，对使用者有一定的门槛要求，所以在推广上收到一定的影响。

* 对象存储技术

这是一种新型的存储方案，使用不同的存储技术和存储设备。顾名思义，这种存储技术的存储单元是对象。这种存储结构决定了它支持高并行、可伸缩的数据访问[63]。

## Spring MVC技术

现在Web应用程序开发的竞争越来越激烈，需要更精确的开发时间和更经济高效的开发方式来开发应用程序。 如何提高生产力和降低复杂性是一个程序员开发J2EE Web应用程序的一个基本问题。如何在不使用企业Java Bean（EJB）的情况下创建符合J2EE的软件？最好的替代方法之一是Spring框架，它提供较少的服务，但它比EJB有更少的侵入。这种转变背后的驱动力是需要更高的生产率，降低Web应用软件开发与实现领域的复杂性[64]。

Spring框架是现在使用十分普遍的一种轻量级企业应用程序解决方案。在这个框架中，诸如消息处理、事务处理、远程方法等基础功能结构已经集成并实现，这一特点让程序开发人员能够将精力更加集中地放在设计和实现业务逻辑上[65]。

### Spring总体框架

Spring作为一个轻量级框架，基于J2EE构建，具有可扩展性高、可维护性强、响应速度快等优点，并且支持ORM（Object/Relation Mapping）和AOP（Aspect Oriented Programming）。图9描述了Spring主要的七个模块。



图 9 Spring模块划分

Spring作为轻量框架，能够在框架内实现所有服务。但是Spring也是组件化的，开发人员可以使用Spring的部分组件，而不用涉及其他的部分[66]。

### Spring MVC

模型视图控制器（MVC）是用户界面逻辑和业务逻辑之间分离的基本设计模式。 由于目前应用程序的规模非常大，MVC设计模式可以弱化不同应用程序层之间的耦合[67]。MVC通过分解模型、视图、控制在程序中的不同角色，将他们与具体业务逻辑分离开，实现应用程序的解耦。模型、视图、控制器三者之间的联系如图10所示：



图 10 MVC之间的联系

一般情况下，MVC会将应用程序返回的数据传递到前端，在视图层展示。View层接收到数据之后，将这些数据展示在前端，View层与后面逻辑层完全脱离，没有任何关联。Controller层会接收来自网络的请求（用户的请求），并根据这些请求的类型，调用后台的服务，来实现一定的业务逻辑，完成用户的这个请求。并将处理结束之后的结果返回。MVC模式关键思想就是业务逻辑与用户界面分离，通过数据传输的方式将界面展示与业务进行分离，使得这两者可以独自改变而不相互影响，实现了各层之间的解耦。

Spring MVC支持XML配置和Java注解配置。在Spring MVC应用程序中，控制器是一个十分重要的Servlet，Spring MVC框架以Dispatcher Servlet为核心，Dispatcher Servlet是一个Java Servlet，该Servlet接收到用户的请求之后会分发这些请求去不同的处理程序进行处理。以最简单的MVC应用程序为例，配置好Spring MVC的控制器Dispatcher Servlet即可，它接收用户不同的HTTP请求，并且每个HTTP请求都需要通过控制器，因此它才能对整个完整的请求生命周期进行管理[68]。



图 11 Spring MVC控制器

在Spring中，需要去定义一个类作为控制器类，[用Java注解@Controller标注。当控制器接收到HTTP](mailto:用Java注解@Controller标注。当控制器接收到HTTP)请求，会通过合适的Handler方法去处理这个请求。在Handler方法处理结束之后，它会根据Handler的返回值，将控制权转给名字与返回值相同的View。

# 架构设计

由于项目开发过程可以分成项目管理模块，开发模块，测试模块，文档管理模块，集成模块，验证模块等等，根据不同的模块，项目开发人员可以按照一定的逻辑进行开发流程的定制，针对不同的模块选择相应的服务，来支持软件开发的过程。充分利用不同计算机辅助软件所提供的功能来完成和支撑软件开发过程中在这个阶段的需求[69]。本文采用五层架构设计来处理不同模块之间的解耦合问题。



图 12基于工作流的软件即服务集成框架图

## WorkFlow Engine & WorkFlow API

在架构图中，工作流引擎(WorkFlow Engine)是一个核心部分。不同于张海军等人提出的基于面向服务架构(Service-Oriented Architecture, SOA)的企业应用集成框架[70]，工作流的加入对推动任务自动化流转有着非常高效的作用。使用工作流的好处是，高可维护性以及扩展性。如果一个系统中只用到了一个流程，且变动的可能性不大，基本不需要考虑工作流。在本文架构设计中，灵活的流程配置和服务选择是它的特点，工作流引擎在该框架下起到了核心的功能[71]。



图 13工作流架构图

工作流可以将整个流程中不同的操作以一定的逻辑衔接起来，并且让信息在其中自动流转。系统会处理自动流转的任务，剩下的任务用某种方式通知需要处理的人，让人介入，而不需要人主动去关注现在应该做哪个步骤。

工作流引擎跟灵活定制的流程相结合，并且流程可以灵活选择相应的服务，就必须将工作流当成一种服务来提供给软件开发辅助软件使用。对于制定的任何流程，以及选定的何种辅助软件，只要这些软件需要在流程中自动流转传递信息的时候，就需要跟工作流进行接口对接[72]。根据工作流引擎的特点，开放出应用程序接口(Application Programming Interface, API)来供服务进行调用和对接十分必要。

图13所展示的就是工作流引擎以及工作流引擎API之间的关系，对应架构中的WorkFlow Engine 和 WorkFlow API两层。将提供功能API与工作流引擎分离的另外一个好处就是提高了工作流自身的灵活性和可扩展性。由于流程的可定制性和服务的可选择性，未来的流程可能更加复杂，服务种类可能更多，对工作流的功能要求可能更复杂，单独将提供功能API作为架构的一层，会有利于后期的扩展和开发，同时在架构上这一层也是最接近数据层的，可以根据不同的需求提供相应的处理数据的服务，例如数据的存储、分析、追踪等等。

## 数据路由层（Data Router）

在流程的流转过程中，很重要的一个部分就是数据。流程所携带的信息对于整个流程中参与的人来说十分重要。一个流程的上游和下游的节点之间所需要的数据可能都会有所关联，辅助软件的服务跟工作流引擎之间会有数据交互等等。如果将数据所需要的功能以及数据流转同工作流的功能放在一起，第一会破坏上面提到的工作流引擎功能API层的独立性，第二会让数据层显得非常的冗余。所以在架构上将数据层独立出来为一层，用来支撑所有跟数据相关的服务。

其次，该层还能有发挥路由的作用。所谓路由，就是寻路的意思。在一个灵活自定义的流程中，在工作流引擎的支撑下，任务的自动流转依靠的是流程发起时，工作流引擎根据流程定义语言所自带的信息实例化的数据。每个节点之间的相互关联逻辑，流程中产生的数据等都会影响到流程的走向，决定下一个节点。该功能实际上是由工作流引擎API提供，但是更多的情况下，工作流的流转还是需要参考分析数据后才能得出结果。因此在数据层中，结合上工作流引擎的寻路功能和节点产生的而数据，可以提供下一个节点的寻路功能。而且这样的设计会让这个架构的逻辑更加清晰，对于工作流支撑下的软件开发过程的扩展性更强。

## 服务群（Services Group）

在面向服务集成的软件开发环境下，软件开发的每个过程都可以选择相应的辅助软件服务来帮助开发过程中的任务和数据进行更好的流转。服务是可选择的，同时，服务也是可以注册的，由于灵活定制的开发流程，不同的流程需要不同的服务[73]。

服务的可注册性会大大提升架构的可扩展性，而不仅仅是为了满足某一个或者某一类流程。比如软件开发过程中经常常见的项目管理服务，代码托管服务，代码code review服务，自动化测试服务，集成验证服务等等。对于这些服务，架构中属于单独一层，称为服务群。所有注册的服务都在这一层记录和管理。不同的服务注册进来之后，能不能正常提供服务，能不能符合架构所需要的要求，接口等数据结构是否兼容等等，都需要进行验证。对于服务的注册，管理和验证，都可以在这一层进行。

## 工具群（Tools Group）

服务层的下一层便是提供服务的工具集合层。针对不同的服务，都有多重多样的工具提供服务。由于软件开发过程中服务需求的多样性，选择不同的工具来提供不同的服务是必要且高效的。因此工具集合层会集中很多工具来提供服务，软件开发者在设计开发流程的时候，可以根据自己的需求去选择不同的工具来提供服务。例如前面提到的代码管理服务，可以选择Gitlab，Github，SVN等成熟的开源软件来进行代码管理，也可以选择其他软件进行管理。

在这里，为了实现更好的扩展性和灵活性，增加软件开发者的可选择性，工具集合中的工具也是需要进行注册的。不同工具介入到框架中，需要进行注册，只有通过注册验证的工具才被允许向工作流提供该工具的服务。工具的注册性质大大增加了底层提供服务的灵活性和扩展性。对于工具的管理和注册也集中在这一层。

# 需求分析与系统设计

采用基于工作流的软件即服务集成框架（WSAIF），本文计划设计一个任务管理系统。

## 需求分析

* 流程可定制

与传统的流程图定义不同，在WSAIF框架下对软件开发流程进行定义，需要完成拖拽生成流程图，允许用户在拖拽过程中自由选定计算机辅助工具，自由选定不同的功能节点。

* 流程模板管理

不同的开发流程由不同的流程模板决定，流程模板可以复用、更改和设置权限。不同的流程模板完成不同的流程，实现不同的需求。需要针对纷繁复杂的流程模板建立模板库，对模板进行集中的使用管理和权限管理。

* 流程自动化部署

自定义的模板完成之后，需要对模板进行部署才能完成后面的功能。采用工作流引擎能够自动化部署定义好的流程。由于本文使用的WSAIF需要结合注册的工具提供服务，因此在定义的流程中需要增加相关信息，在部署的时候这些信息需要携带到流程中，因此需要开发支持该种自定义功能的流程引擎，将流程中工具相关的信息和部署的数据添加到流程中。

* 流程任务管理和可视化

流程在部署之后，同一个流程会带有很多的任务。分配任务的用户需要知道这些任务的执行状态，流程的进度；被分配任务的用户需要知道自己被分配的任务，以及任务所携带的一些信息。对于这些需求，需要设计一个流程任务管理和任务可视化的功能部分，这一部分的主要作用是为了让软件开发流程在通过调用后台工作流功能处理任务的同时，能够在前台与用户进行交互，直观地显示给用户软件开发进展的状态和任务详细信息。用户通过前台便能够很好地进行软件开发任务的分配与管理。真正达到通过应用软件即服务集成框架来方便流程开发。

* 数据交互和统计

由于数据路由层将数据交由工作流处理，因此在工作流层数据如何与工具交互，工作流和工具之间如何流转数据成为一个重点研究的问题。在工作流层，对流转的数据进行持久化和统计，以便于后期进行分析和改进。

* 工具注册和对接

WSAIF框架的另外一个核心就是工具服务化。将软件开发过程所需要的软件支持抽象成一个个不同的服务，不同的服务通过不同的软件辅助工具来提供。能够为该框架提供服务的工具都必须先在任务管理系统进行注册，并通过对接检测，合格之后才能为框架提供服务。同时，工具所提供的每一个服务（功能）都需要通过API调用的形式开放，方便工具与工作流的对接。

## 整体系统设计

项目开发过程可以分成项目管理模块，开发模块，测试模块，文档管理模块，集成模块，验证模块等等。充分利用不同计算机辅助软件所提供的功能来完成和支撑软件开发过程中在这个阶段的需求[69]。

本文设计的WSAIF模型，是以工作流引擎为核心，来支撑软件开发过程中的各种需求。图14是该模型总体设计的基础架构。

本文的重点是工作流引擎的开发和流程管理与监控系统的的设计与实现。架构中的各类工具应用了面向服务和设计模式中代理模式的思想，通过服务代理中间件的方式实现与工作流最大限度的解耦。邵虹等人的工作中有详细介绍[74]。根据图4的系统设计分析，需要设计和实现以下两个方面：

* 工作流引擎

选择合适的工作流引擎

开发基于工作流引擎的RESTful API(REpresentational State Transfer Application Programming Interface)

* 流程管理

确定流程可定制化方案，提供流程定制服务

提供可视化的流程和任务管理服务

扩展设计数据库来支撑数据的存储和流转

图3 系统设计架构图

图 14支持多工具协同的流程管理系统架构图

## 系统模块设计

根据前面的架构设计，能够将系统分成几个模块，每个模块负责不同的功能，通过模块之间的相互配合和信息流转，来实现本文的设计。

### 工作流后台

工作流引擎是任务流转的核心。本文基于工作流设计了支持软件开发的服务集成框架，因此将工作流作为一个后台模块来开发，用来提供工作流功能。前台设计一个多应用集成平台，平台所完成的跟流程有关的功能支撑，都通过工作流后台提供。这样做到了核心功能与前台业务的分离，使得逻辑更加清晰，开发更加便捷。

在工作流后台中，提供了所有跟工作流有关的功能接口，同时根据不同的业务需要，结合多应用平台的需求，定制相应的API来实现指定的功能。但是流程定制这个跟工作流相关的功能与工作流后台分开。原因有三点：第一，该功能需要与用户交互，是一个前台功能；第二，流程定制对外提供的是流程模板的定制功能，该功能比较复杂且用户使用程度较高，独立出来也有利于根据用户的反馈和使用情况进行完善和维护；第三，该功能的阐述是工作流模板，工作流模板跟多应用集成平台息息相关，也涉及到模板的管理问题，部分功能需要在多应用平台中实现，将此功能独立出来有利于用户在多应用集成平台中对工作流模板进行直接的操作。

### 多应用集成平台

多应用集成平台是在前面系统架构设计下与用户交互最重要的平台系统。用户所的需求都通过该平台来完成。该平台集成了软件工程任务的创建、开发、管理、数据流转等等所有功能。

* 流程部署

当用户选定一个工作流模板进行部署时，工作流会根据工作流模板所携带的信息部署生成一个工作流实例，这是部署的基本步骤。在本文的设计中，针对软件开发过程的流程的部署会包含其他的信息。当给定一个工作流实例的时候，本文架构首先会分析这个工作流实例所需要完成的步骤，完成每个步骤所参与的辅助软件。除此之外，针对流程部署，采集一些其他的软件开发过程相关输入（如项目相关初始数据、流程节点负责人、软件开发流程时间限制以及一些阈值等等）。

* 流程管理

流程部署之后，流程实例中就包含一个个任务等待执行，这些任务会按照设定的执行逻辑，根据一定的条件进行执行。这些任务实质的内容就是通过任务节点关联的软件开发工具去完成相应的软件开发需求。软件开发过程的有序进行的前提就是对这些任务进行高效的管理，同时给用户提供便捷的服务。

1）待处理任务(Todo Task)

在设定好的软件开发流程（模板）中，每个节点即为一个任务，每个任务或是自动流转或是有任务负责人。如一个功能模块的开发任务被分配给A，那么A就能收到该任务并通过相应的工具去完成任务（例如在Gitlab中完成代码开发并提交）。同时A可能还在其他的开发流程中有其他的任务，那么查看待处理任务对于A来说很必要。因此，根据软件开发流程化的特性，给所有负责人发送相关的任务信息，并提供查询任务的接口。这样会大大提高流程中节点负责人的工作效率。

2）人工处理(Manual Handling)

软件开发过程中，走完所有的步骤，软件开发过程才算结算。反应到流程中，就是所有的任务都被执行完毕。但是不同的任务有不同的执行方式：与工具对接的自动流转和与用户操作相关的人工处理。与工具对接的自动流转任务执行类型在本文接下来即将介绍的数据中间件中就可以执行完毕。 与用户操作相关的人工处理需要用户的参与，从用户那里获得相关数据，流程再根据相应的逻辑进行流转。这里也体现了待处理任务功能的便捷性和必要性。

3）添加子流程(Add Sub-Process)

当一个节点负责人觉得当前任务过于复杂，以至于将这个任务单独处理为一个流程会比较高效。面对这种情况，在当前任务下面创建子流程是本文设计的一个亮点之一。子流程的创建机制与父流程相同，需要着重考虑和设计的是子流程与父流程执行状态的同步，数据的交互和节点负责人之间任务传递。

* 流程控制

一个流程实例(Instance)就是一个软件开发流程的简要概括。完善的流程控制会让软件开发过程进行的更加顺利。

1）已发起流程(Started Instance)

通过已发起流程，用户可以查看当前已经发起的流程，以及流程的一些信息。对应过来就是当前已经启动的软件开发项目，以及各个项目的进展。

2）流程监控(Instance Monitor)

软件开发流程当前处于什么状态，进行到哪一步，耗费了多少时间，有哪些产出，负责人是谁，接下来的任务是什么等等这些信息，都属于流程的状态信息，将这些信息提供给用户能够大大帮助用户对软件开发流程的掌控。

通过图表的形式将流程的状态和流程的数据，与流程执行过程中数据的传递直观地展现出来，让用户知道软件开发过程中的耗时，执行人以及成果。这对控制软件开发过程会有很大的帮助。

3）终止流程(Instance Termination)

对于废弃的项目或者需要重新作出调整的项目，有些流程可能需要及时终止，因此提供一个终止流程并保存之前相关数据的功能十分必要。

* 数据展示与分析

在本文设计的系统中，会收集软件开发过程中几乎所有的数据，对这些数据的分析将会大大帮助以后的软件开发过程。

1）工具使用统计

在用户设定模板进行软件开发的时候，模板中会带有使用何种软件工程辅助工具来帮助开发的信息。对这些信息的统计能够得出在不同开发阶段使用不同软件开发辅助工具的偏好，有利于优化和推广这些软件开发辅助工具。

2）数据流量统计

每个任务流转时所携带的信息可以在执行任务流转的功能中进行统计和记录。这些跟软件开发相关的数据，可以用来分析软件开发过程中的行为偏好，开发过程的侧重点等等，对于优化开发流程有一定的潜在价值。

3）模板使用统计

记录每一次模板的使用情况，针对经常使用的模板可以进行优化；以及抽象成样例模板，方便用户进行模板定制。

### 流程定制平台

本文出于用户使用角度的考虑和实际管理的需要，将该架构中的流程设计功能独立出来，设计了流程定制平台。

* 流程定义语言扩展

为了使工作流管理系统具有良好的互操作性，首先必须对工作流模型中公共的实体、属性以及实体间的关系作一个规范，这是实现工作流交互操作的第一步。工作流管理联盟 (WFMC)于1999年提出了工作流元模型 (Workflow Meta-Model)与工作流过程定义交互格式 (Workflow Process Definition Interchange)，并且提出了工作流过程定义语言 ( Workflow Process Definition Language,WPDL) ，流程设计者设计流程的时候，需要根据一定的语言，来定义流程的具体内容。包括流程的结点，流转方向，判断条件，负责人等等。有了流程设计语言，才是工作流引擎能够工作的第一步[75]。

合理的流程定义语言不仅能够帮助工作流引擎更好地工作，更能够让使用者轻松明确地制定自己需要的流程。根据本文设计的架构，需要对流程定义语言进行改进和设计，来支持服务和工具等更多的功能。

* 工作流模板库

1）模板制作(Template Design)

设计模板定制平台，向用户提供可操作的UI来定制属于自己的软件开发流程。设计合适的拖拽界面，根据需求定义适当的工作流模板节点和流程语言是该部分的重点。

2）模板管理(Template Management)

由于工作流模板的高复用性和可编辑性，建立一个模板库对于提高用户定制模板效率有着重要作用。每个用户建立的模板都可以被保存至模板库，用户在自定义模板的时候也可以从模板库直接导入相似模板进行修改，达到事半功倍的效果。

3）模板权限控制(Template Access Control)

模板库存了所有用户创建的模板，在用户保存模板的时候，需要根据用户的意愿以及用户权限设定该模板的保存权限。这个权限存在的意义就是在用户使用和复用该模板的时候，只有具有相应权限的人才能使用。在软件开发过程中，用户权限角色化对于软件开发过程管理是有帮助的，因此根据用户的权限来相应地建立模板库的权限对于保护软件开发过程中信息的安全性和私密性有着重要作用。

# 任务管理系统的实现

本文建立的基于工作流的软件即服务集成框架，旨在帮助软件开发过程中，帮助用户定制个性化开发流程，支持用户自由注册软件开发辅助工具并在开发过程中选用，同时帮助开发人员在开发过程中对开发流程进行管理，包括开发流程进度的管理、流程任务执行状态的监控、流程数据的收集分析等等。

## JAVA开源工作流框架选用

工作流引擎是本文的重点之一，工作流所能够提供的功能对于软件开发过程起到了决定性的作用。经过几十年的发展和完善，现在已经有很多成熟商用的开源工作流，提供稳定成熟的功能，且可扩展性比较高，可维护性很强。因此本文在实现阶段，根据开发需求，决定选择一款JAVA开源工作流引擎进行二次开发，根据软件工程的特点和本文设计所对应的需求，开发出符合需求的工作流引擎。

### 主流开源工作流框架比较

现在市面上所存在的工作流引擎有几十种之多，前文也进行了相应介绍。而在这些开源工作流引擎框架中，本文结合业务的需求和软件工程特点，拟定在JBPM、OSWorkflow、Enhydra Shark、Activiti5这四个主流工作流引擎中作出选择，在选定的这个开源工作流引擎上进行二次开发。下面就会重点对每个工作流引擎的优缺点进行分析，综合比较之后得出结论。

* JBPM

1）优点

a) JBPM可扩展性很强，在所有这些开源工作流引擎中，它是最适合被商业化应用的工作流引擎。

b) JBPM自己本身已经结合了开源框架Hibernate3，使用起来十分方便，而且JBPM对数据库的支持十分成熟且广泛，对大多数流行的数据库都有一个对应的文件脚本来初始化，支持当前大多数流行的数据库。

c) JBPM适宜商业化的一个特征就是数据分离。它将数据的管理功能与核心工作流功能分离，本身的功能部分专注于业务逻辑的实现和处理。

d) JBPM的流程定义语言使用Jpdl，对于流程的定义十分直观，通俗易懂，可读性很强。而且JBPM的流程定义语言可以自己保存，手工修改，也有利于针对不同需求进行扩展。并且JBPM在Eclipse有一个流程定义的插件，这会给本文设计的流程定义平台提供一定的参考，具有借鉴意义。

e) JBPM比较成熟，发展历史也比较久远，版本较多，文档丰富翔实，论坛以及开源组织都十分活跃，从长远未来上看，在被Jboss收购以后，JBPM的发展趋势不错。

2）缺点

a) JBPM在Eclipse上面的流程定义插件没有开源，但是这也不妨碍给我们提供的借鉴意义。

b) JBPM使用Hibernate3有利有弊，Hibernate3在做持久化层的时候，会产生一些冗余的数据表，导致会出现一定的数据冗余问题；

* OSWorkflow

1）优点

a) OSWorkflow是一款非常轻量型的工作流引擎，它是灵活工作流引擎的代表，它的定位是低级别的工作流引擎框架，在OSworkflow 里，可以用代码来实现可视化图标的流程。

b) OSWorkflow 可扩展性很强，它有着十分卓越的灵活性。它所提供的集成功能，能够给开发人员提供很大的便利，能够帮助应用程序开发人员与现有代码以及数据库进行集成。

c) OSWorkflow是基于Action驱动的工作流引擎框架，这十分符合现在应用程序开发人员的编程和操作习惯。

2）缺点

a) OSWorkflow实现一个工作流的入门难度较高。用它去实现一个工作流管理系统十分复杂，因为每一个流程的每一个步骤实现都需要设置状态字段，然后通过代码去改变该字段的状态，复杂而且容易出错。

b) OSWorkflow的组件不够丰富，如果基于OSWorkflow开发的流程管理系统需要支撑的流程十分复杂的话，后台的工作流引擎必须在OSWorkflow上做大量的二次开发，实用性较低。

c) OSWorkflow可维护性不高。OSWorkflow里有很多需要配置的内容，配置项与开发代码量来比，相对较多，不利于后期工作流引擎的维护和扩展。

* Enhydra Shark

1）优点

a) Enhydra Shark的可扩展性较高，它的工作流体系最为完善，也比较复杂。它具备“模块化”的设计思想，灵活度较高。

b) Enhydra Shark对于初学者来说门槛比较低，它的代码量较少，便于阅读。而且十分有助于二次开发和扩展维护。

c) Enhydra Shark自带一个基于Java的图形化工作流编辑器（JaWE），来图形化定义工作流流程。因此Enhydra Shark的图形化功能与其他几个开源工作流引擎相比相对较强，而且可以通过流程逻辑来控制属性。

2）缺点

a) Enhydra Shark与其他的几种开源工作流框架相比，它的体系最为庞大，而且工能很复杂，因此二次开发的可造性较差。而且在Shark2.0后，它的很多工能组件、技术文档都被商业化，不再开源，与本文思想有悖。

b) Enhydra Shark的版本更迭速度较慢，后期的代码开发方式也没有按照开源方式完成。

* Activiti5

1）优点

a) Activiti5采用了PVM（流程虚拟机），这是它一个很大的优势。流程虚拟机不定义流程语言，PVM的思想是工作流引擎应该支持多种工作流语言，因此应该多种工作流语言共存。流程虚拟机定义了多个流程执行语言共享的模型【流程虚拟机——jBPM产品思想分析】。Activiti5不仅支持BPMN2.0规范，还支持除了BPMN2.0规范之外的流程格式；

b) Activiti5的扩展性很高，它能够与外部服务有良好的集成，而且通过与Mule的集成，更加加强了Activiti5的集成能力；

c) Activiti5的作者是JBPM4的作者，它继承了JBPM4的很多优点，在流程可视化方面有所进步，而且Activiti5有一个创新的Activiti Cycle协作组件。

d) Activiti5对于所有在运行的流程实例，建立了一个流程实例管理平台，提供给用户进行流程的管理和监控。

2）缺点

a) Activiti5的数据持久层采用的是ORM MyBatis3，数据持久化标准没有遵循JPA规范。而且根据很多用户已经使用过去的反馈来看，工作流流程的“回退功能”的实现难度比较大。

b) Activiti5的核心是BPMN 2.0的流程引擎，但是BPMN2.0的规范发展十分缓慢，再加上其语言本身也很复杂，不易懂。

### 四大开源工作流与统一开发平台集成比较

表 2开源工作流与统一开发平台集成比较表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | JBPM | Activiti | OSWorkflow | Enhydra Shark |
| 流程定义插件集成 | 基于Eclipse图形化插件，基于Web的流程设计器 | 基于Eclipse图形化插件，基于Web的流程设计器 | 基于Eclipse开发的建模工具 | 定义工具JaWE |
| 核心包及jar包集成 | 轻量级工作流框架，JBPM.jar，1.06M | 轻量级工作流框架，activiti-engine-5.9，1.1MB | 轻量级工作流框架，osworkflow-2.8.0.jar，393KB | Jar包以来庞大，核心6M左右 |
| 部署方式 | 可以与项目集成，也可以独立部署 | 可以与项目集成，也可以独立部署 | 不可独立部署，推荐与Spring集成，方便事务管理 | 可以与项目集成，也可以独立部署 |

### 版本选择与维护问题

1、JBPM4发展历史悠久，技术文档积累十分丰富，用户众多，在网上也能够容易找到很多共享的技术资源进行学习和参考，是目前最稳定的版本。虽然目前已经停止了JBPM4的开发和更新，JBPM4的开源社区依然十分活跃，不同使用者在上面提出自己的一些见解和问题。JBPM5从版本上看是JBPM4的升级版本，但是实际情况是JBPM4的架构师离开JBoss公司之后，JBPM5基本上全盘抛弃了JBPM4以前的的代码和部分思想，然后开发了继承自JBoss原来的工作流产品Drools Flow，因此JBPM5与JBPM4并不能方便地升级转化，JBPM5的技术资料和相关文档也相对匮乏。

2、OSWorkflow曾经是发展良好的一个工作流框架，在ERP软件开发中的使用十分广泛，但是由于JBPM的横空出世分流走了Osworkflow的一部分用户，让Osworkflow后期发展乏力。但是OSworkflow是opensymphony下的一个开源项目，而且版本比较稳定。虽然文档不是很详细，但是开源社区也有一定的活跃度，网络上技术资源也比较丰富，学习起来较快。

3、Enhydra Shark2.0后，版本更迭速度较慢，而且不再开源，二次开发后可扩展性和可维护性都不高。

4、Activiti5是JBoss JBPM架构师加入Alfresco之后创建的工作流引擎，Activiti5的思想来自JBPM，它集中了JBPM4几乎所有的优点，目前来看上升势头比较明显，版本开发更新迭代速度较快，开源社区十分活跃，但是目前技术资料不是很多，正在逐步完善中。

### 总结

所选的四种开源工作流引擎均能够与开发平台集成，且成本较低。从快速项目开发角度来看，jBPM和Activiti5较为适合。

综合开发因素和网上参考资料丰富程度，本文最终选用jBPM作为本文的二次开发工作流引擎。

## 流程可定制化方案——流程定制平台

jBPM本身在Eclipse中的流程定义插件，可以通过拖拽进行流程的建立，自动导出XML文件，并且能够直接被流程引擎所实例化。结合框架的需求和用户友好性的考虑，从易用性和便捷性上进行思考，在实现过程中开发设计了自己的流程设计器，能够方便生成和导出XML文件。

流程定制方案上，用户可以自主选择不同的节点，包括开始节点，结束节点，任务节点，分支聚合节点，判断节点等等。值得一提的是，在案例中我们自己设计的流程定义平台上，任务节点是我们任务处理的核心所在，这个节点能够携带很重重要的信息，如我们前面提到的G1-S1-T1，表示任务到该节点是，使用的是什么服务提供的什么功能，是由某种工具提供的该功能服务。所以用户在构建流程的时候，会有选择用什么工具来提供该功能。因此我们也在基于XML的流程定义语言上扩充了自己的元素。

表 3流程定义文件样例

|  |
| --- |
| <process xmlns='http://jBPM.org/4.4/jpdl'>  <start g='15,16,48,48' name='开始' form='开始'>  <transition name='startTask' to='ProjectMgr'/>  </start>  <task assignee='yangtao@nfs.iscas.ac.cn' g='134,132,90,50' name='ProjectMgr ' tool\_id='4' function\_id='2002'>  <transition name='toCode' to='CodeMgr'/>  </task>  <task assignee='yangtao@nfs.iscas.ac.cn' g='349,82,90,50' name='CodeMgr' tool\_id='4' function\_id='2002'>  <transition name='finish' to='结束'/>  </task>  <end g='675,24,48,48' name='结束'/>  </process> |

将用户设计流程的步骤简化就是，选择所需要的服务，选择提供该服务的工具，将这些服务按照用户自己定义的开发逻辑链接起来成为一个流程图。完成这三步，用户的流程模板就已经建立起来，可以被流程引擎启动实例化，投入到实际中工作了。下图是是我们实现的流程定制平台。

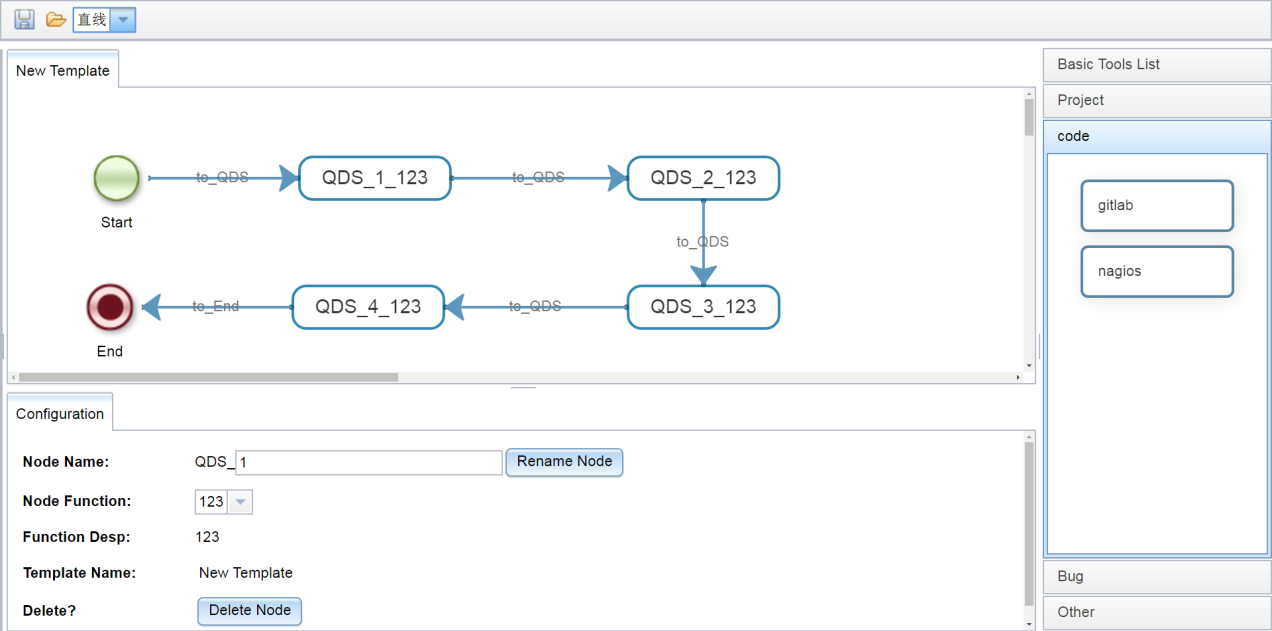


图 15流程定制平台

从图15中我们可以从右侧边栏选择不同的节点种类，在流程定制平台中，分为五大类：Basic Tools List、Project、Code、Bug、Other。其中基本工具列表（Basic Tools List）指的是支撑工作流基本功能的节点，包括开始节点（Start）、结束节点（End）、分支节点（Fork）、聚合节点（Join）、条件节点（Decision）等等。除了基本工具列表之外的节点，都是流程定制平台根据用户注册的工具和功能自动生成的节点。用户注册了某个工具的某些功能之后，这个工具就会自动被添加为流程定制平台的流程节点。这些节点的分类，也跟服务注册的服务群的分类一一对应。

用户在创建流程图的时候，在选定了某个节点之后就可以通过拖拽的方式放入左边的空白区域，完成创建流程最基本的一步。这个时候用户选定了节点，对应到流程开发中就是用户选定了某个工具去完成流程中该步骤的功能，但是使用这个工具的哪一个功能来完成需要给出明确的选择。用户可以选中流程图上的任何一个功能节点，流程图区域的下面就会显示出跟这个节点相关的信息，包括节点的名称、节点的功能、节点的描述等等。其中节点的功能是一个下拉菜单，里面对应该工具所具备的所有功能，这些功能与用户注册的工具下的功能一一对应，用户注册通过了一个新的工具功能，在流程定制平台这里都会进行更新。

在流程设计平台上，平台能提供多少种服务，每种服务能有那些功能，每个功能又有哪些工具提供，这些都是动态可配置的。前文所说的服务注册带来的灵活性就体现在这里。作为一个基于工作流的服务集成框架的架构平台，应当致力于不同服务的集成，如果有服务需要加入或者更改，可以通过架构中的服务注册功能，将该服务注册进来。同理，如果有不同的功能需要注册进来，该功能接口需要满足平台设计的数据格式要求。在通过统一数据收发验证之后就可以正式注册到平台中，并且在流程设计平台中能够直接看到新添加的服务和功能。

用户拖拽完所有的节点，选择完所有的服务之后，流程只要 符合定制平台的规则，就能进行保存。该流程就会作为一个工作流模板进入到模板库。用户可以在模板库中去发起自己建立的模板，部署自己的流程。

## 流程管理方案——任务管理系统

jBPM本身就提供了很多的用户查询交互接口，可以方便查询工作流引擎管理下的所有流程的相关信息，通过API调用的形式给出。在基于工作流的服务集成框架下，需要给用户给出一个清晰的任务管理界面来方便用户的操作。因此我们设计了一个任务管理系统，用来展示在这个框架下与用户定义流程相关的内容。除此之外，前文提到的工具和服务注册等问题，也集中设计到这个系统中。

因此从功能模块划分，将该系统分为四个主要部分，流程部署、工具服务注册、流程任务管理和模板库管理。

### 流程部署模块

系统可以根据开发模板的特性和开发流程的不同需求，传递相关数据来支撑开发过程中所需的功能。如图16所示。

在这里给用户提供的功能对于用户发起流程十分关键。众所周知，在软件开发过程中，对于时间和进度的控制往往关系到一个开发流程顺利与否。因此在部署开发的流程的时候，任务管理系统提供相应的功能，部署系统首先对流程开发模板进行解读，得到所有的开发任务，并将这些开发任务展示给流程部署者，让流程部署者提供每个任务的时间限制。在这里，时间限制这个数据会随着流程的部署被传递到后台，并且会伴随着每一次任务节点。在这里必须要提一下工作流引擎二次开发的重要性，针对像这样的定制化需求，完成任务节点完成时间的检查，以及针对超时或者死亡的任务节点如何处理的问题，必须要在二次开发中完成这样的功能，一个灵活的工作流引擎需要具备这样的可扩展性。

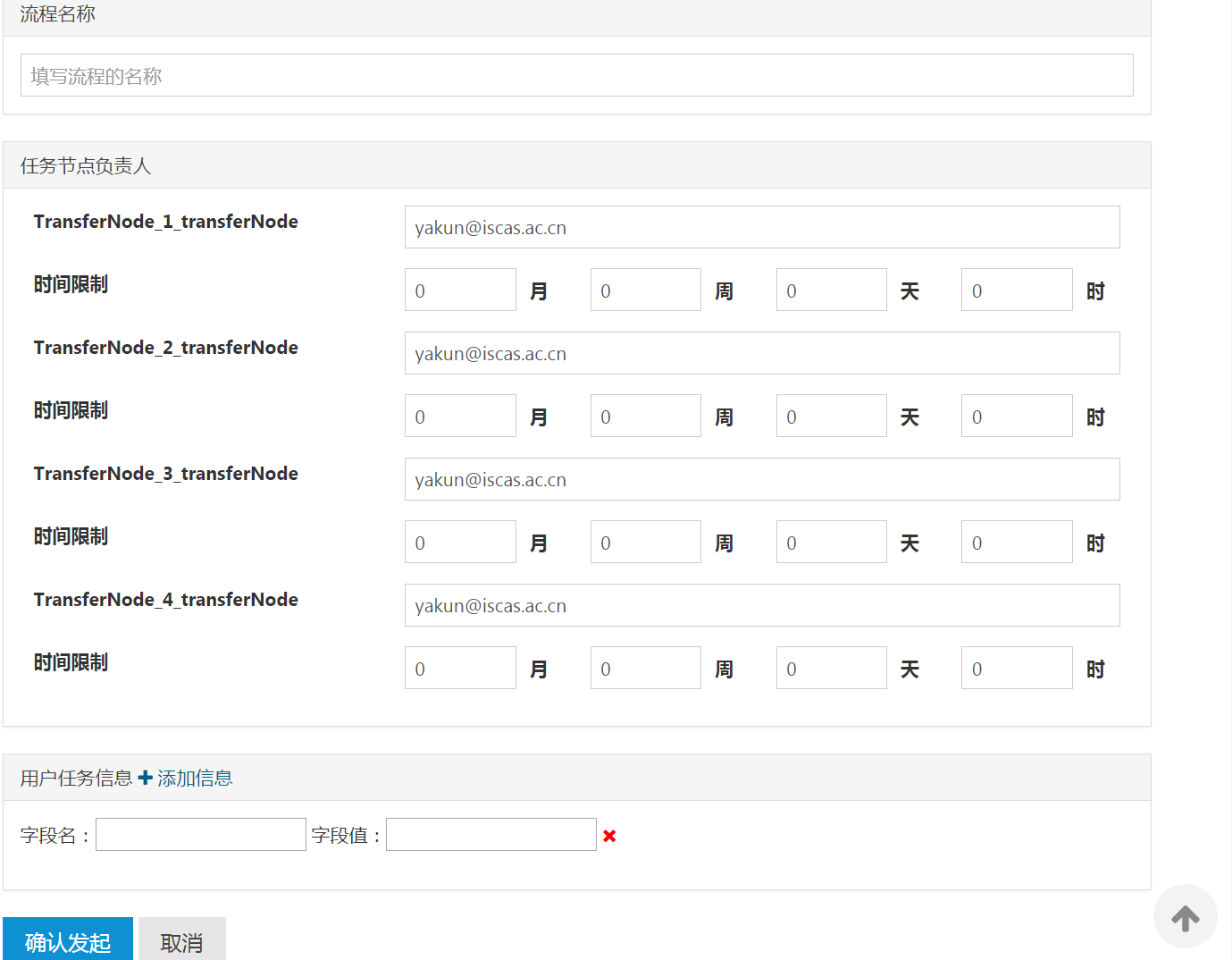


图 16流程部署模块

当然，除此之外，任务管理系统的任务部署模块还给用户提供了更多的便利。用户可以在流程发起的时候添加一些额外信息，例如这个项目的名字、所对应的基金号、关键负责人等等信息。这些信息的传入对于发起这个开发流程有着很重要的作用，在后期每个任务负责人完成任务的时候有一定的帮助。

### 工具服务注册

工具服务注册模块提供了框架所需的服务注册功能，针对不同的服务，提供工具注册功能，分配工具的ID，根据工具的ID配置工具提供的服务的地址，配置提供的功能信息等等。

总体上，对于服务群和工具群的分类，如图17所示。

这里的服务群分类与流程定制平台的任务节点分类相同，这里的工具群与流程定制平台的节点一一对应。可以看到每个工具都有自己的主页地址，这些工具可以是开源的软件开发辅助工具，对于他们提供的功能接口稍作处理就能与任务管理系统对接，也可以是自己开发的系统，通过开放API来与任务管理系统对接，从而被用户用来支撑软件开发流程。



图 17服务群与工具群

通过图17中的注册新工具功能，就可以进行新工具的注册。如图18所示。

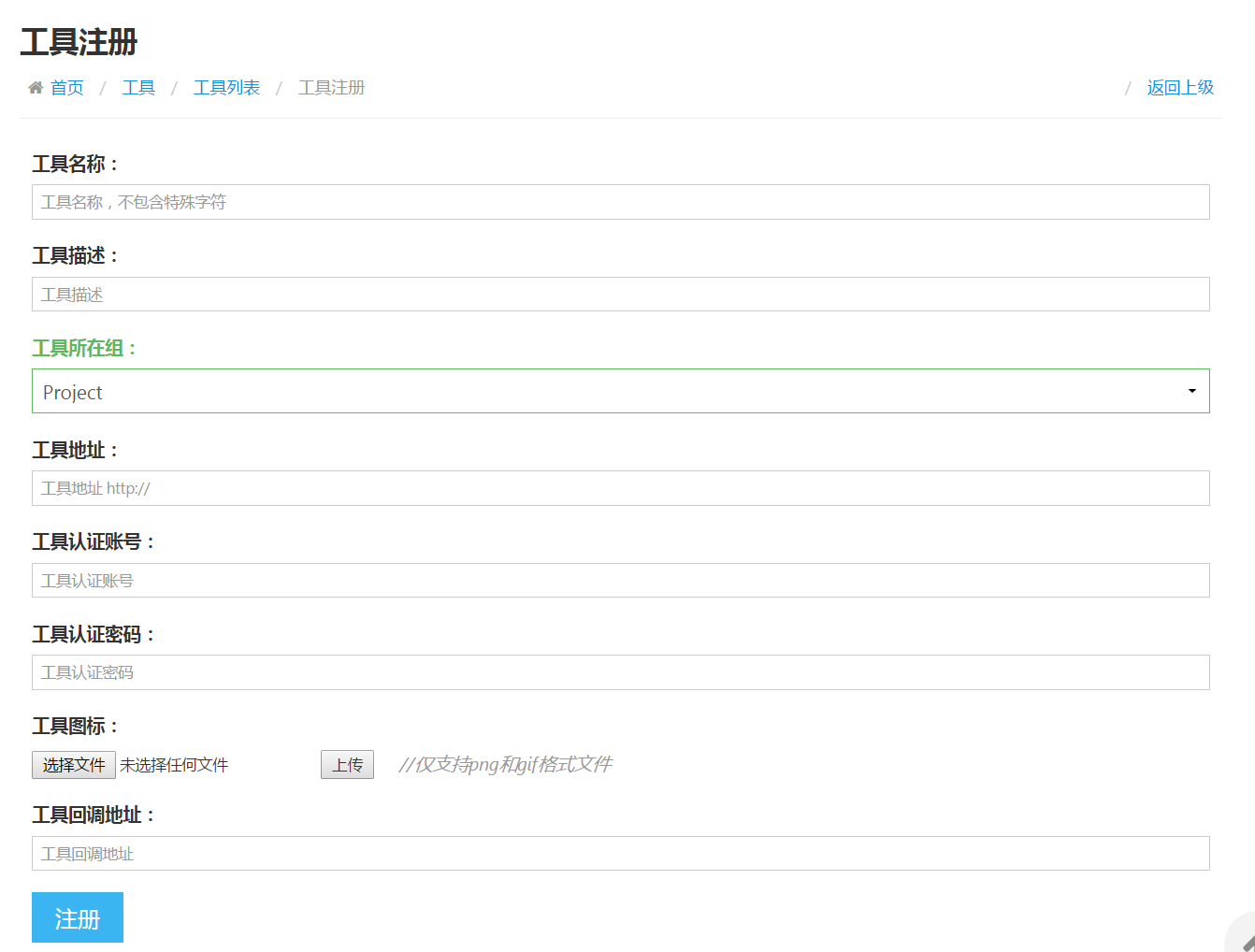


图 18工具服务注册

根据要求填写相应的信息，选择所在的分组，同时注册上工具所在的服务地址，加上一些工具本身的认证机制，就能够注册成功。在任务管理系统中，每个工具的服务提供之后，信息的返回是通过系统回调的方式返回到工具中进行处理，因此工具需要在注册的时候填写自己的回调地址，方便信息的流转与沟通。

在注册之后，系统会分配工具的ID，并在工具群中显示，如图19所示：



图 19工具信息展示

工具注册之后，会进行功能的注册。如果引入一个新的工具，而没有引入这个工具所提供的功能，工具注册将会变得毫无意义。因此功能注册是一个十分重要的模块。不同的工具都有注册属于自己的不同的功能，这些功能与流程定义平台中每个节点下拉菜单中的功能一致，这些功能是真正最后参与到软件流程开发中并提供相应服务的。例如例如代码管理软件Gitlab的功能如图20所示：

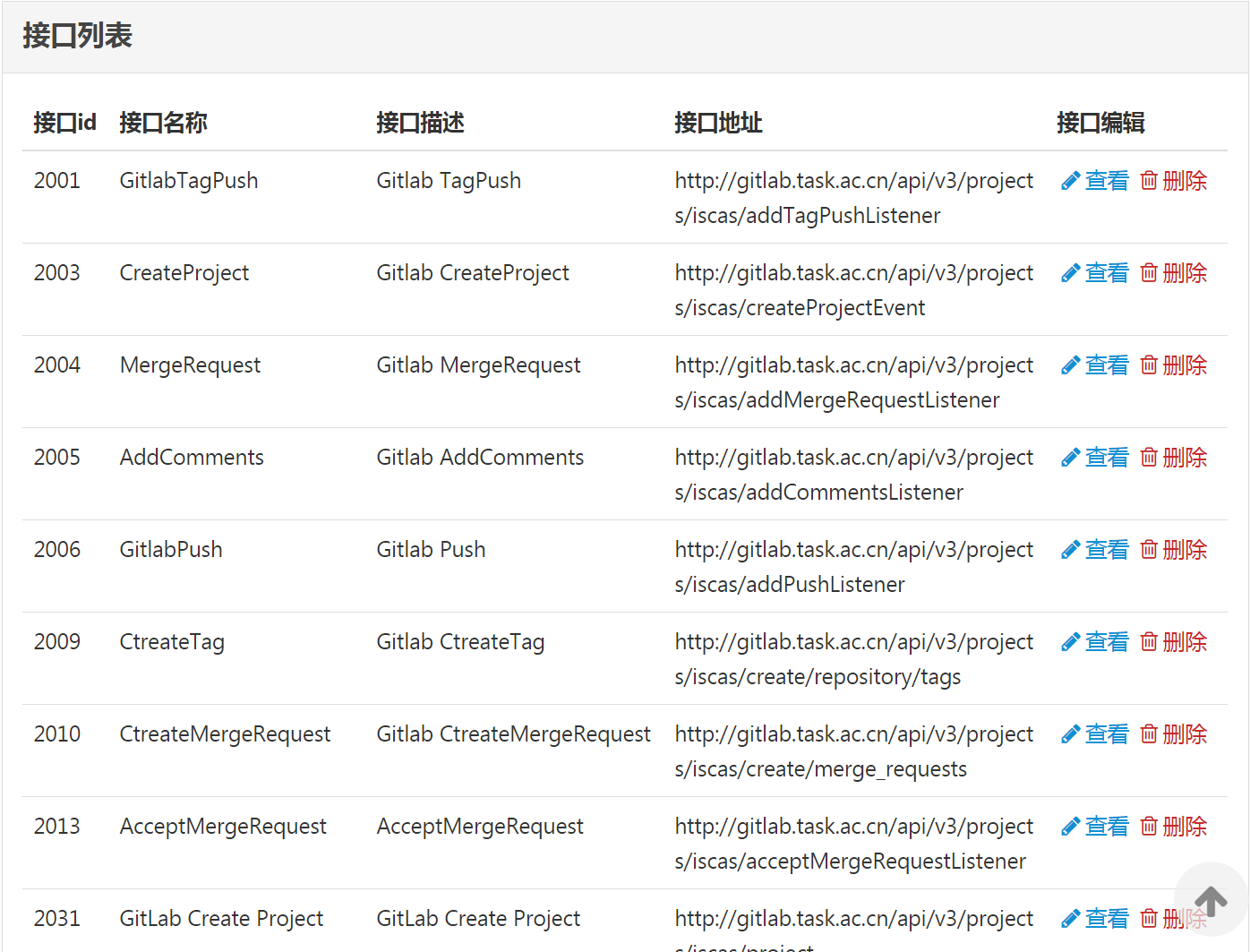


图 20工具服务列表

每个接口注册之后都会有一个特定的ID，用户在流程定义平台创建完成开发流程模板之后，所选择的的所有工具的所有功能都通过这些功能ID标记在XML文件中，工作流引擎在部署流程的时候通过解读开发模板的XML定义，就能调用相应的功能去完成不同的任务。

### 流程任务管理

流程任务管理包括待处理任务列表、已发起流程列表和流程监控。流程任务管理是任务管理系统的重中之重，也是用户参与流程管理的重要途径。流程监控向使用者展示包括流程进度，流程数据流等，流程当前状态以及提供终止流程等操作。

* 待处理任务列表

软件开发流程被部署之后，流程中的任务随着流程的进行而流转，每一个任务都一个或者多个负责人来负责完成。当流程到达某个任务时，该任务的负责人就会收到任务的通知和相关数据消息——这就是用户的待处理任务。负责人可以登录任务管理系统查看自己的待处理任务。一个用户可能同时又多个待处理任务，这些任务是所有发起流程中等待当前用户去处理的节点。每个任务的信息不一样，有些有时间限制，有些有数据传输等等。用户可以随时登录任务管理系统进行查询和处理。



图 21待处理任务展示

每个待处理任务都有唯一的ID进行标识，任务的相关信息也会在任务列表中进行显示。对于每个任务，用户除了能够进行常规的操作之外，还可以进行任务的细分。因为如果软件开发流程被设计为一级的话，有时候繁杂的任务或者开发过程中的变动，会给原定的开发流程带来一定的不合理性。因此对于流程中的每个任务节点，处理该节点的负责人都可以将该任务进行细分，创建一条子流程来完成这个任务。子流程与父流程意义相同，给予了任务处理人更大的发挥空间和处理自由度，也能够缓解不同解决的处理压力，能够灵活应对开发过程中出现的不同波动因素。用户创建的子流程会通过后台数据库与父流程进行关联，子流程是否成功结束会关系到创建该子流程的任务的完成情况，最后关联到父流程的流程进度上。

* 已发起流程列表

当软件开发流程被部署的时候，工作流引擎会记录流程的发起人，同时记录所发起流程的相关信息，如图所示。

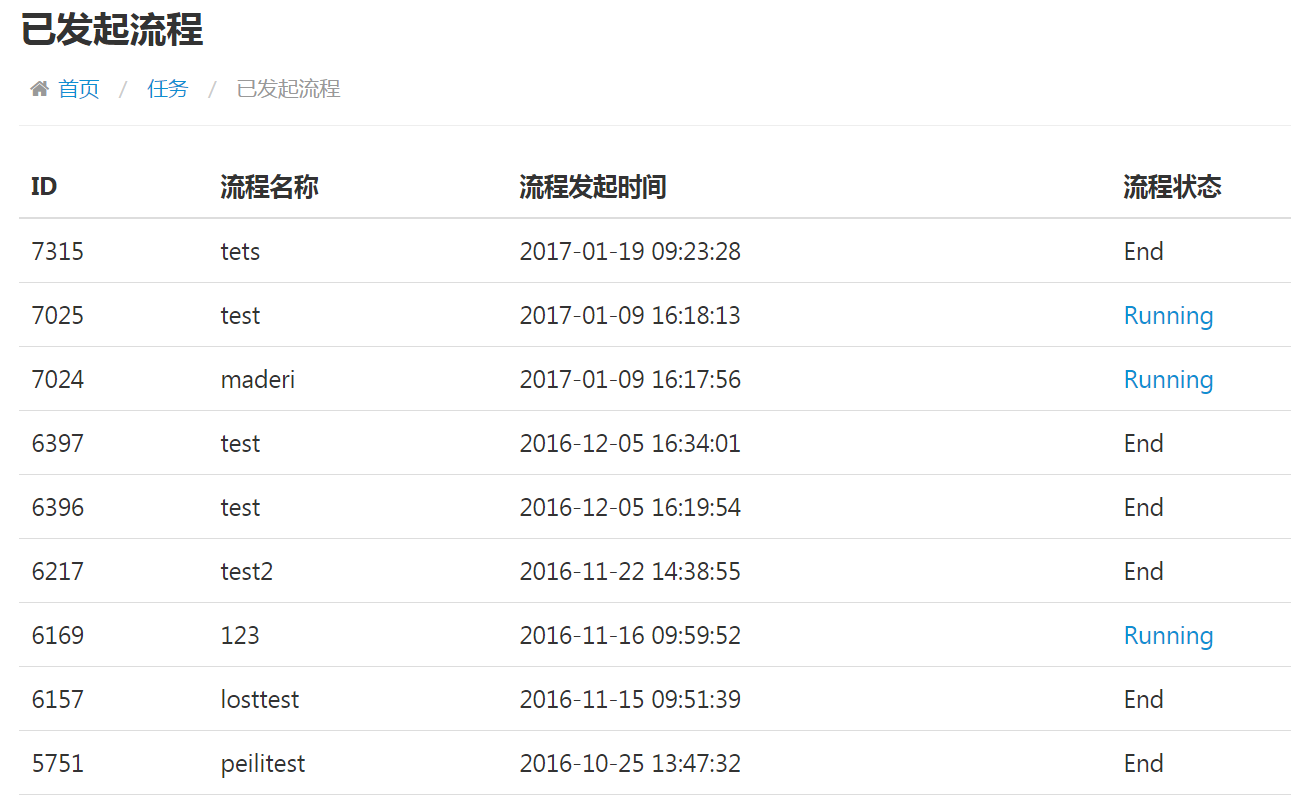


图 22已发起流程展示

用户登录任务管理系统，在已发起流程功能栏中，可以看见自己所发起的所有流程，但是不能看见别人所发起的流程。对于自己所有发起的流程，能够看到该流程的ID、流程的名称、流程发起的时间和该流程当前的状态。如果该流程还未结束，可以通过点击该流程的状态，了解流程进行到哪个任务，以及一些其他的相关信息。

* 流程监控

每个用户都有发起流程的权利和功能，对于所有发起的流程，任务管理系统提供一个监控和管理的角色，可以查看所有已经发起的流程和流程对应的当前状态。并且可以对流程进行操作和管理。

用户可以根据流程发起的时间，选择时间段进行查询，得到在该时间段内发起的流程。如图23所示。



图 23流程列表

系统也提供快捷查询方式，根据当前的系统时间，向前计算一周以内、一个月以内以及本月内所能发起的所有开发流程。在这里所能看到的流程信息，除了流程ID、流程名称和发起时间之外我，还能够看到流程的发起人，当前流程已经执行的时间以及流程的状态。同时，点击流程名称，系统还能够向用户展示当前流程已经运行到哪个具体的任务节点。例如上图中ID为3227，流程名称为LXSA-41的流程，如下图所示。

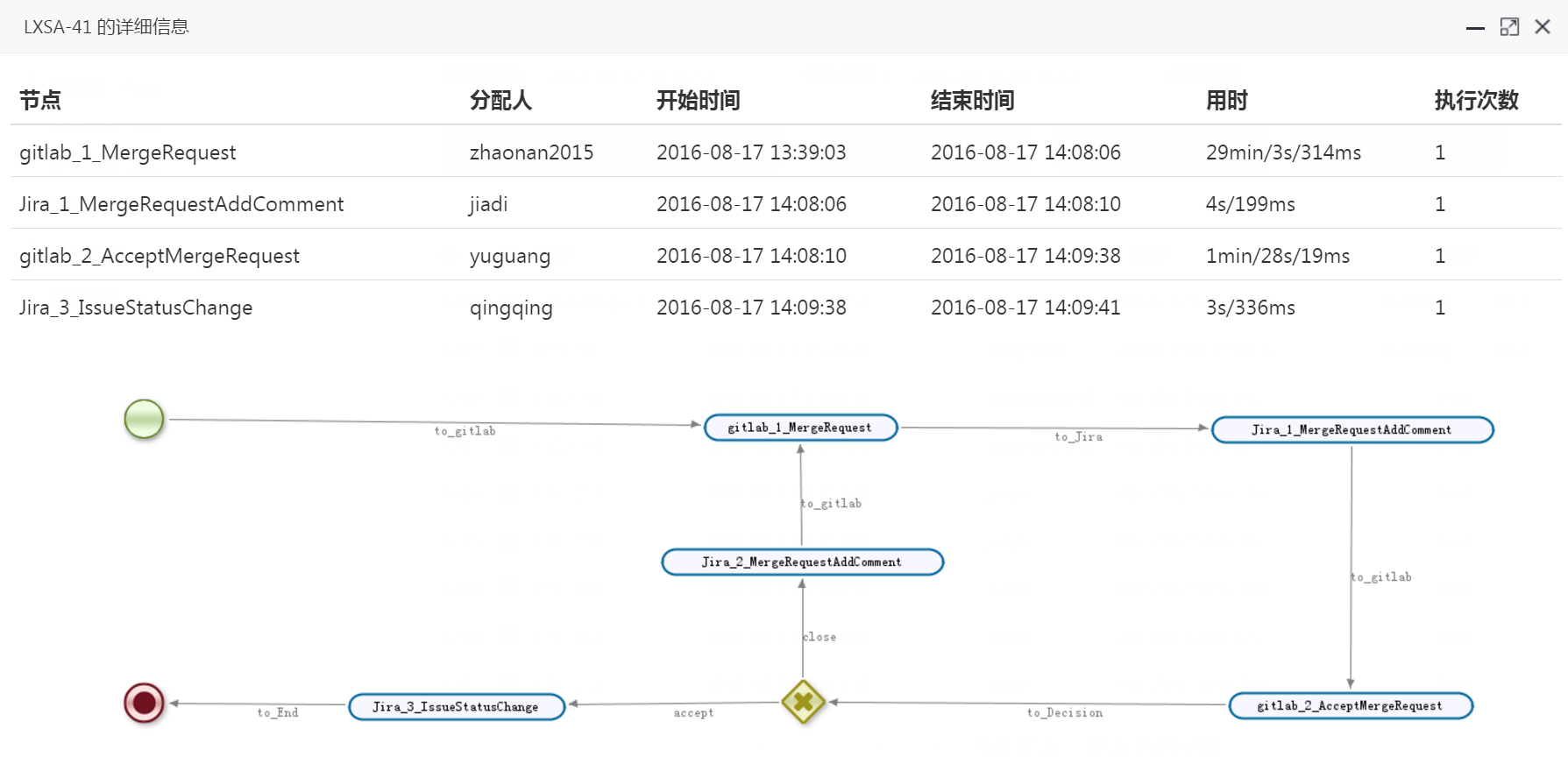


图 24 LXSA-41流程执行详细信息

通过点击流程名称，用户能够了解当前流程的初始流程设计，同时知道该流程每个流程的具体执行情况，包括每个任务的名称，任务负责人，任务的开始和结束时间，以及该任务被执行了多少次（涉及到流程中的回退功能等）。关于该流程执行情况的详细信息一目了然。如果当前流程尚未结束，存在未完成的任务，系统会在流程图中标记当前任务所进行到的位置。

流程的进行伴随着数据的流转，用户点击流程ID，可以看到当前流程进行到现在所有的数据流转。任务管理系统对流程数据进行了展示，以流程3227为例，如图25所示。

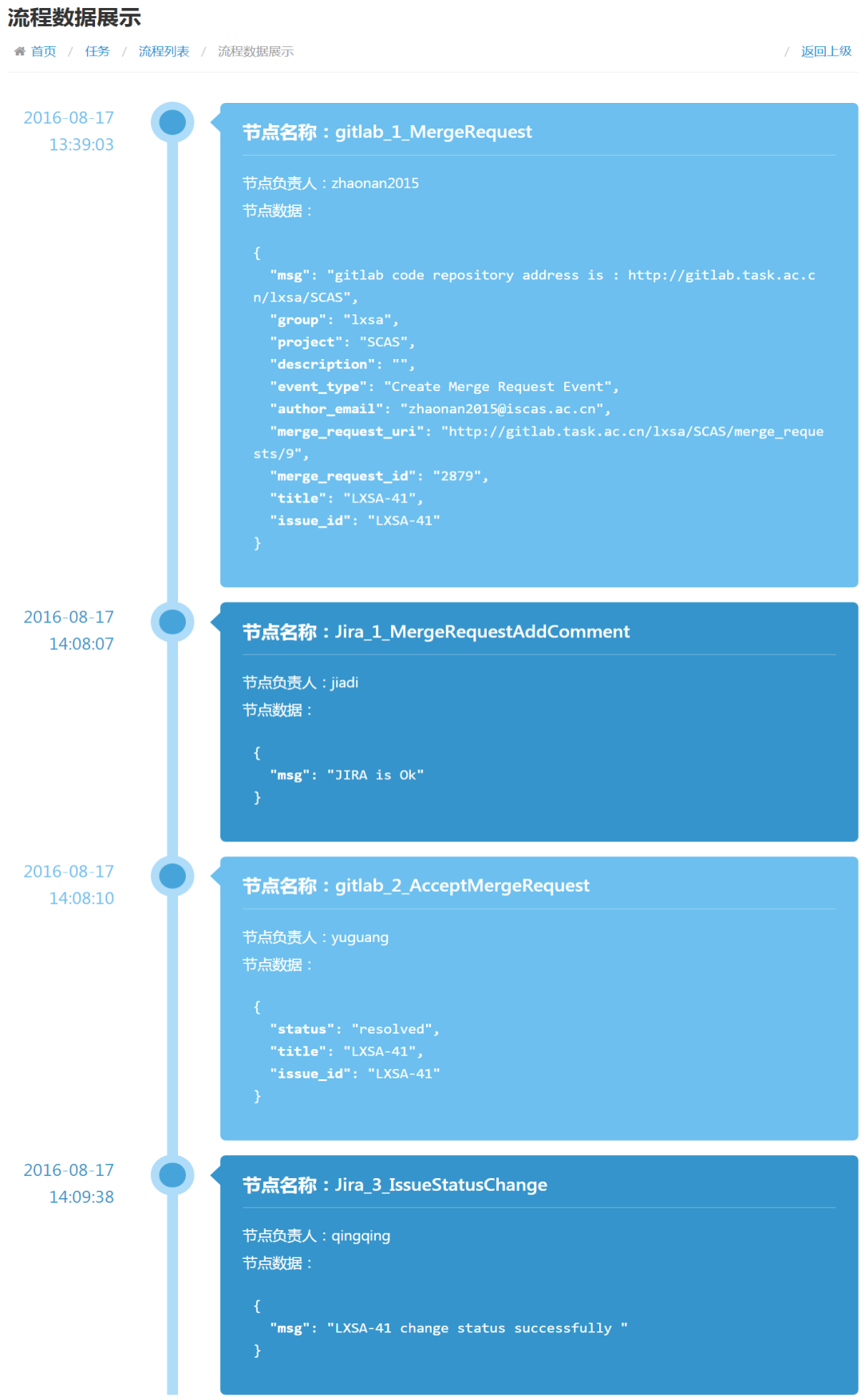


图 25流程流转数据展示

软件开发流程每个任务节点流转的有关信息都会经过工作流，工作流对这些数据进行分析和处理，按照一定的格式与工具进行交互。因此用户在查看一个开发的流程时，还可以查看在这个流程中流转的相关信息。

### 模板库管理

根据流程设计平台的设计方案，流程模板被创建完成之后，会进入模板库进行统一管理。由于用户与开发流程对接的主要交互界面就是任务管理系统，而且开发流程的发起也在任务管理系统中，因此将流程模板管理放在任务系统中实现比较合理。

模板库中的模板分为公有模板和私有模板。用户在流程设计平台创建完成开发流程模板，在保存的时候会选择保存成为公有模板和私有模板。公有模板创建出来是供所有用户使用的模板，大家都可以使用该模板来部署开发流程，只要该模板符合自己的需求。私有模板是只有相应权限的人才能查看和发起的模板。处于对开发流程管理的便捷和一定的保密性，对于模板的权限管理也十分重要。下图是公有模板库的示例。

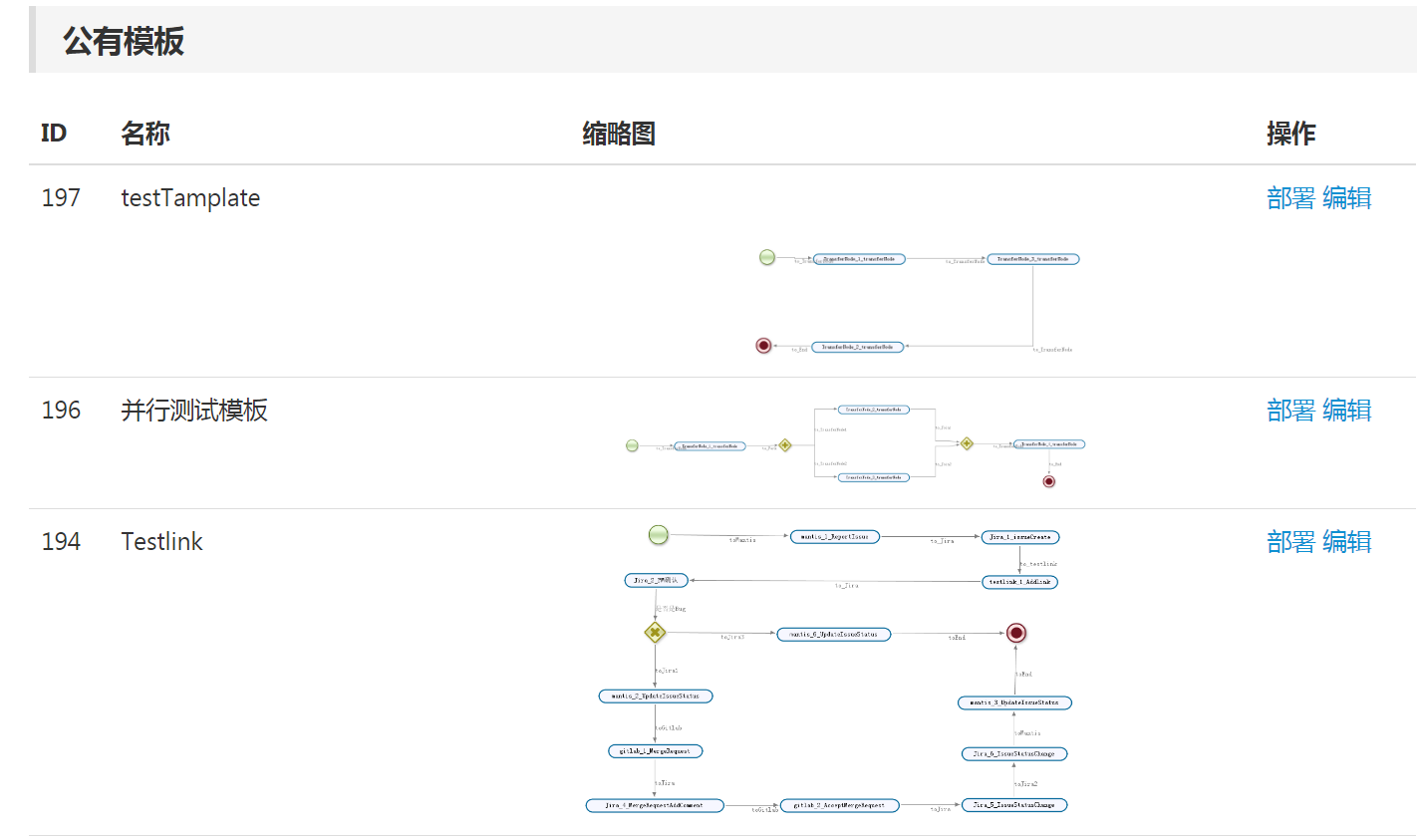


图 26模板库公有模板

每个模板都有自己的ID、名称和缩略图。在模板库中，用户可以选择相应的模板进行流程部署，或者对某个不符合需求的模板进行编辑。同理，在私有模板库中，用户所能够看见的模板知识当前自己有权限能够接触到的开发流程模板。

## 数据流转方案——数据总线

### 数据库扩展设计

本文实现采用的数据库是Mysql数据库，在JBPM内核数据库的基础上，根据实际需求和数据存储需要，设计了自己的数据表。其中主要关键的几个数据表如下所示。

表 4支持多工具协同的流程管理关键数据表

|  |  |  |
| --- | --- | --- |
| 序号 | 名称 | 用途与约束 |
| 1 | iscas\_user | 用户表 |
| 2 | iscas\_user\_tool | 工具表 |
| 3 | iscas\_task\_defined | 流程定义表 |
| 4 | iscas\_task\_instance | 流程实例表 |
| 5 | jBPM4\_\* | JBPM内核表 |

我们设计了保存用户的表iscas\_user，它用来存储用户的id、用户名、邮箱、密码、所在组、用户权限以及登录时间等信息。系统 中用户的必要信息都会存储在这张数据表中。

表 5 iscas\_user数据表

| 序号 | 字段名 | 字段说明 | 类型 | 长度 | NULL | DEFAULT | PK | UK | FK | Index | 备注 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id |  | bigint | 20 | NOT | 0 | Y |  |  | Y |  |
| 2 | username |  | varchar | 32 | NOT |  |  |  |  |  |  |
| 3 | password |  | varchar | 32 | NOT |  |  |  |  |  |  |
| 4 | email |  | varchar | 64 |  |  |  |  |  |  |  |
| 5 | group |  | tinyint | 1 |  |  |  |  | Y |  |  |
| 6 | update\_time |  | datetime |  |  |  |  |  |  |  |  |

本文设计了存储jBPM流程定义的表iscas\_task\_defined，它用来存储流程定义xml文件的名称，流程定义图片的名称，更新的时间，以及创建该流程定义的用户的ID和所在组号。根据存储的xml文件和图片文件的名称，我们可以在对应位置找到该文件，完成文件的定位，方便文件的读取。对于每一次用户在流程设计器中创建一个流程定义的模板时，我们在这张表中都会生成一条记录。

表 6 iscas\_task\_defined数据表

| 序号 | 字段名 | 字段说明 | 类型 | 长度 | NULL | DEFAULT | PK | UK | FK | Index | 备注 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id |  | int | 11 | NOT |  | Y |  |  |  |  |
| 2 | jpdl\_png\_file |  | varchar | 255 |  |  |  |  |  |  |  |
| 3 | jpdl\_xml\_file |  | varchar | 255 |  |  |  |  |  |  |  |
| 4 | update\_time |  | varchar | 255 |  |  |  |  |  |  |  |
| 5 | user\_id |  | int | 11 |  |  |  |  | Y |  |  |
| 6 | user\_workgroup |  | int | 11 |  |  |  |  |  |  |  |

本文设计了jBPM流程实例的表iscas\_task\_instance，它用来存储流程实例的相关信息。每条记录存储的信息包括流程实例ID、状态、标题、创建时间、创建该流程实例的用户ID、用户名以及该流程实例中所有task名称和对应task的assignee，还包括该流程实例对应的外部链接工具的ID、外部链接工具的链接、外部链接工具的名称。对于用户每次发起的一个流程实例，都是产生一条这样的记录，在展示流程实例进展状态图的时候，根据这张表中的内容能够知道该流程实例所有的task节点，每个节点的assignee是谁，然后将当前的task节点标出来。在流程实例结束的时候，将数据库中这条记录删除。

表 7 iscas\_task\_instance数据表

| 序号 | 字段名 | 字段说明 | 类型 | 长度 | NULL | DEFAULT | PK | UK | FK | Index | 备注 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id |  | int | 11 | NOT |  | Y |  |  |  |  |
| 2 | nodeNameAndAssginee |  | varchar | 255 |  |  |  |  |  |  |  |
| 3 | outerTaskId |  | varchar | 255 |  |  |  |  | Y |  |  |
| 4 | outerTaskLink |  | varchar | 255 |  |  |  |  |  |  |  |
| 5 | outerTaskTitle |  | varchar | 255 |  |  |  |  |  |  |  |
| 6 | process\_instance\_id |  | varchar | 255 |  |  |  |  | Y |  |  |
| 7 | status |  | int | 11 |  |  |  |  |  |  |  |
| 8 | time |  | varchar | 255 |  |  |  |  |  |  |  |
| 9 | title |  | varchar | 255 |  |  |  |  |  |  |  |
| 10 | user\_id |  | varchar | 255 |  |  |  |  | Y |  |  |
| 11 | username |  | int | 11 |  |  |  |  |  |  |  |

本文设计了存储对接外界工具的iscas\_tool表，它用来存储工具ID、工具描述、工具图标、工具唯一识别信息、工具名称、工具对接URL等工具相关的信息。每一个工具对接，都会记录到这张数据表中，每次工具的对接过程都会从这个数据表中读取数据进行操作。

表 8 iscas\_tool数据表

| 序号 | 字段名 | 字段说明 | 类型 | 长度 | NULL | DEFAULT | PK | UK | FK | Index | 备注 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | id |  | int | 11 | NOT |  | Y |  |  |  |  |
| 2 | tool\_desc |  | varchar | 255 |  |  |  |  |  |  |  |
| 3 | tool\_icon |  | varchar | 255 |  |  |  |  |  |  |  |
| 4 | tool\_identity |  | varchar | 255 |  |  |  |  |  |  |  |
| 5 | tool\_name |  | varchar | 255 |  |  |  |  |  |  |  |
| 6 | tool\_url |  | varchar | 255 |  |  |  |  |  |  |  |

除了上面设计的数据表之外，任务管理子系统主要还应用了jBPM的18张数据表，在此就不列入数据表设计的范畴。

### 框架数据流转设计与实现

假设简化开发流程如图：

图8 开发流程图

图 27简化开发流程

软件开发流程中，虽然使用了很多支持工具，但是有很多信息具备一致性的性质，例如项目名称、参与人员、开发组信息等等。在该开发流程之下，实现之后的数据流转过程如图：

图9 数据流转示意图

图 28支持多工具协同的流程管理系统数据流转图

数据的传输分为两个部分：Content和Taskdata。Content：全局数据，从发起流程开始，数据量会不断扩大。Taskdata：增量数据，每个任务节点产生的数据。

每个任务节点调用工作流接口结束任务的时候，会传递自己产生的数据到工作流。工作流将它合并到Content中，并传递到下一个任务节点。

在每个Taskdata传递回来的数据中，为确保数据能够合并到Content中并解决重复更新问题，必须对Taskdata传递的内容格式作出约束。系统中会设计通用词表，Taskdata根据通用词表来标志自己传递回的数据。

### 数据统计

根据本文的设计，基于工作流流转的软件开发流程所产生的部分数据会经过工作流的数据总线。根据上一小节的数据流转分析，通过分析Content和Taskdata中的数据，能够得到开发流程中的具体任务所流转的数据，对这些数据的统计和分析对于软件开发流程的改进有很大的帮助。

* 工具使用统计

用户在流程定义平台定义模板的时候，会选择服务群和工具群去完成流程模板的创建。相似的任务节点，用户可以选择不同的工具去完成该任务。对于所有发起的流程，通过数据统计可以分析出使用得比较多的工具，针对这些使用的工具可以进行一些接口完善和二次开发，使之符合开发人员的习惯，达到提高软件开发效率的目的。



图 29工具使用统计

* 工具接口统计

工具群中的工具通过开发接口的形式提供功能。在软件开发过程中，不同的功能有不同的使用频度。通过工具接口调用能够清楚地看到软件开发过程中使用的比较多的服务，有效地提高这些使用频度较高的服务会帮助软件开发过程。下图展示了一个月之内，gitlab工具提供的所有服务的使用情况。



图 30工具服务使用统计

从图中可以看出每个接口的使用频度，不同时间段的使用次数以及使用趋势。通过对服务使用的分析，对于使用频度较高的服务，提高服务的稳定性显得十分必要。

* 接口流量统计

除了从服务调用的角度来看每一项服务的重要性。通过统计不同服务之间流转的数据量，也可以对服务的重要性进行评估。以gitlab工具所提供的MergeRequest服务为例，如下图所示。

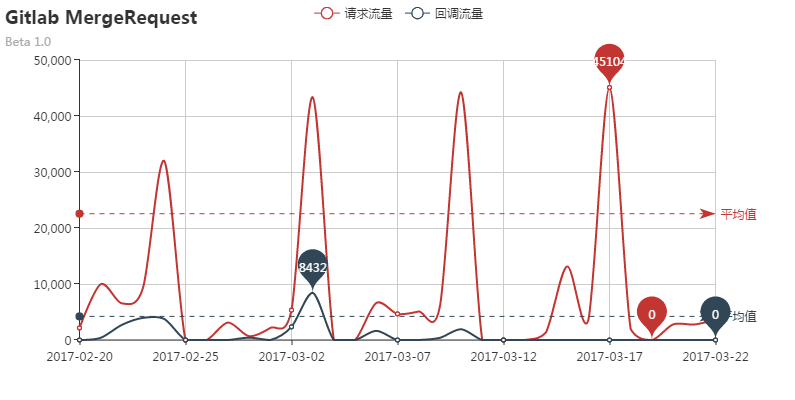


图 31工具接口流量统计

可以看到请求流量和回调流量在不同时间段的数据量。统计中会显示最大流量和最小流量，通过多个服务之间最大流量和平均流量的对比，能够分析出不同服务的重要性。

* 模板发起统计

用户在流程设计平台创建完流程模板，流程模板就会被存储在模板库。软件开发人员根据开发需求，在流程开发模板库中的选择模板进行部署。同一个模板能够被多次部署，对于模板被部署次数的统计，能够知道模板的受欢迎程度，也说明了模板契合需求的程度。下图展现了一段时间之内模板发起次数的统计。



图 32模板发起统计

通过对模板发起次数的统计，能够知道经常被使用的模板。模板本身的设计就有一定的可提升空间。对于经常使用的模板，就可以考虑对模板进行精简和规则化，定义为样例模板供用户使用。同时也能通过这些模板的发起，知道开发任务的需求和重点所在，帮助项目管理员更好地进行项目的管理。

# 应用和测试

根据框架所设计的系统在7个开发组中推广使用，并且委托测试组根据要求对系统主要功能进行了认真严格的测试。

## 设备环境

表 9测试设备环境表

| 检测参数 | 名称（编号） | | 配置类型 | 参数 |
| --- | --- | --- | --- | --- |
| 功能 | 工作流服务器  （FST/PCM-056） | | 硬件配置 | 型号：ThinkStation P300  CPU：Intel@Corei7-4796CPU @3.6GHz\*8  内存：7.7GB  硬盘：976GB |
| 软件配置 | 操作系统：Ubuntu 14.04  中间件：apache-tomcat-8.0.24 |
| Jira服务器宿主机  （FST/PCM-059） | | 硬件配置 | 型号：ThinkStation P300  CPU: Intel@Corei7-4796CPU @3.6GHz\*8  内存：7.7GB  硬盘：976GB |
| 软件配置 | 操作系统：Ubuntu 14.04  中间件：Virtual Machine Manager 0.9.5 |
| Gitlab服务器宿主机  （FST/PCM-061） | | 硬件配置 | 型号：ThinkStation P300  CPU: Intel@Corei7-4796CPU @3.6GHz\*8  内存：7.7GB  硬盘：976GB |
| 软件配置 | 操作系统：Ubuntu 14.04  中间件：Virtual Machine Manager 0.9.5 |
| 服务器  （委托方提供） | 宿主机  （委托方提供） | 硬件配置 | 型号：ThinkServer TD340  CPU: Intel (R) Xeon (R) CPU E5-2407 v2 @2.40Ghz  内存：6.6G  硬盘：1.8T |
| 软件配置 | 操作系统：LinuxMint  中间件：Virtual Machine Manager 0.9.5 |
| 网络账号虚拟机  （委托方提供） | 硬件配置 | CPU：2核  内存：2G  硬盘：10.71G |
| 软件配置 | 操作系统：Ubuntu14.04 |
| Secfile服务器虚拟机  （委托方提供） | 硬件配置 | 内存：2GB  硬盘：20GB |
| 软件配置 | 操作系统：ubuntu 14.04 |
| 测试机  （FST/PTEST-021） | | 硬件配置 | 型号：ThinkPad X1 Carbon  CPU:Intel(R)Core(TM)i5-5200U [CPU@2.20GHz](mailto:CPU@2.20GHz)  内存：8GB  硬盘：230GB |
| 软件配置 | 操作系统：Windows 10专业版  测试工具：Postman V0.8.1  浏览器：chromium 50.0.2661.102、firefox 47.0 |
| 路由器  （FST/RT-03） | | -- | 型号：TL-WR842N |

## 测试环境拓扑

本次测试在局域网环境下进行，共5台服务器，1台测试机，测试环境如图33所示：



图 33工作流功能测试测试环境拓扑图

## 测试内容

本次测试根据需求文档分析得到22个测试需求，设计780个测试用例，其中2个测试用例由于无法确认项目组的ID以及项目的排序规则未执行。

功能测试采用边界值、错误推测等黑盒测试用例设计方法设计用例，并手工执行测试用例，记录测试结果。功能测试如下表所示。

表 10功能测试结果表

| 指标要求 | 测试说明/要求 | 测试结果描述 |
| --- | --- | --- |
| 工具管理 | 根据需求分析3个测试需求，设计114个测试用例并执行，验证工具管理功能。  工具管理功能包括工具组的注册删除、工具的注册编辑删除、工具接口的注册编辑删除3个测试项。 | 执行通过61个测试用例，发现21个一般缺陷，提出3个建议及改进，详见《缺陷报告》 |
| 模板管理 | 根据需求分析2个测试需求，设计50个测试用例并执行，验证模板管理功能。  模板管理功能包括模板的创建编辑删除、模板分类2个测试项。 | 执行通过30个测试用例，发现13个一般缺陷，详见《缺陷报告》 |
| 任务操作 | 根据需求分析5个测试需求，设计38个测试用例并执行，验证任务操作功能。  任务操作功能包括根据模板生成工作流、为流中的各个任务节点执行负责人、任务节点的处理时间期限、任务跳转、驳回计数5个测试项。 | 执行通过23个测试用例，发现9个一般缺陷，提出5个建议及改进，详见《缺陷报告》 |
| 任务监控 | 根据需求分析2个测试需求，设计106个测试用例并执行，验证任务监控功能。  任务监控功能包括监控所发起流程状态、超级管理员监控并终结流程2个测试项。 | 执行通过85个测试用例，发现11个一般缺陷，提出3个建议及改进，详见《缺陷报告》 |
| 数据统计及可视化 | 根据需求分析6个测试需求，设计423个测试用例并执行，验证数据统计及可视化功能。  数据统计功能包括工具使用量统计、接口使用量、接口的请求流量和回调流量3个测试项。  可视化统计功能包括接口使用量、接口的请求流程和回调流量、工具使用量统计显示3个测试项。 | 执行通过371个测试用例，发现8个一般缺陷，提出6个建议及改进，详见《缺陷报告》 |
| 模板发起统计 | 根据需求分析3个测试需求，设计49个测试用例并执行，验证模板发起统计功能。模板发起统计功能包括日期、模板发起统计、模板使用统计3个测试项。 | 执行通过38个测试用例，发现4个一般缺陷，详见《缺陷报告》 |

# 总结与展望

## 总结

本文通过六章内容详细阐述了支持多工具协同的流程管理系统的设计与实现，从实际需求的角度出发，开发出能够为软件开发提供便利的系统。本文工作总结如下：

* 研究和学习工作流等相关技术

从确立通过工作流的思想来帮助软件开发过程开始，工作流在本文中就有着不可替代的作用，无论从架构还是功能上，工作流都发挥了重要的作用。通过介绍工作流的基本概念和工作流引擎，引入了工作流管理系统以及工作流思想和技术在不同行业的应用。并在学习工作流技术的过程中，了解并熟知了当前大多数的常用的工作流引擎。由于本文需要在开源工作流引擎上做二次开发，因此需要选择开源工作流引擎。通过对主流的Java开源工作流引擎的调研和研究之后，选择了JBPM作为本文架构使用的工作流引擎。在学习过程中，从不同工作流引擎的优缺点、与统一开发平台的集成、版本选择与维护等方面分析了不同的工作流引擎。

除了工作流技术之外，还研究学习了SOA架构和Spring MVC技术。软件即服务架构是本文的一个重点之一，通过将软件提供的功能虚拟化成一个个服务，通过软件开发流程调用的形式来进行使用。软件即服务的思想让软件开发流程定制成为可能，大大提高了软件开发流程的灵活性。由于本文系统的特性以及需要实现的的功能，采用Spring MVC技术来实现系统的开发。MVC所对应的三层架构，对开发者来说清晰明了，而且数据、逻辑和业务的分离，让系统的开发更加方便，也便于后期维护。

* 基于工作流的多工具集成框架设计

本文设计了一个基于工作流的多工具集成框架，在这个框架的基础上实现了支持多工具协同的流程管理系统。本文设计的这个框架分为五层，每一层都承担了不同的功能，耦合性较低。顶层工作流引擎层控制着软件开发流程的流转，API层由工作流引擎对外提供rest风格的工作流接口供软件开发辅助工具调用。数据路由层负责软件开发流程中工具数据之间的流转，根据流程模板确定流转的方向和路线。同时数据路由层还对数据进行持久化和分析。服务群集中了工具提供的各种服务供软件开发过程调用，这也是软件即服务思想发挥作用的地方。工具群集中了提供服务的各种工具。工具群和服务群的扩展和维护通过服务注册来实现。

该框架的设计，将软件开发辅助工具灵活服务软件开发过程的问题，通过工作流和软件即服务思想分成五层来进行解决。该架构各层之间的耦合度低，有利于开发时期的模块划分，也利于后期的扩展和维护。

* 系统需求分析

本文分析了软件开发流程中可以通过辅助软件和工作流自动化来提高开发效率的方式，对软件开发过程进行了需求分析。第一，为了提高流程的开发效率，流程可定制将会帮助开发人员自由定制不同的开发流程，在工作流推动流程流转的基础上成功让开发流程具备了多样化的特性，能够适应各种不同的开发团队的开发需求。第二，自由定制的开发流程可以作为开发模板反复使用，开发模板的管理将会帮助开发者和管理者来管理开发流程。第三，开发流程部署自动化能够帮助开发人员自动化部署开发流程，自动开启任务和通知相关负责人。第四，开发任务管理和可视化帮助开发人员和管理人员了解和熟悉当前的任务情况，以及当前开发所到达的阶段。第五，通过工作流和工具之间流转的数据的统计和分析，能够帮助开发者完善和修改开发流程，同时对开发过程中的数据做到有迹可循。第六，软件开发过程中所使用的开发辅助工具需要通过注册和对接的方式接入到系统中来发挥工具的功能和提供服务。工具注册和对接也是一个很重要的需求。

* 系统架构设计

基于前面的框架设计和需求分析，本文设计了一个流程管理系统。该系统能够支持软件开发辅助工具的柔性集成，通过工作流为软件开发过程提供自动化的功能服务。系统架构主要分为工作流核心模块、流程任务管理与监控模块以及工具注册管理以及数据传输模块。工作流核心模块对应多工具集成框架的第一和第二层的部分功能；流程任务管理模块对应框架的第二层的部分功能和第三层的部分功能；工具注册管理以及数据传输模块对应框架的三四五层。

* 系统实现

在系统需求分析和系统架构设计的基础上，本文实现了该系统。通过分别开发不同模块的方式，完成了不同模块的开发，最终完成了能够正常提供服务的系统。

* 系统测试

系统开发完成之后进行了严格的测试，通过单元测试和黑河测试等方式对系统的功能和用户体验进行了全面的测试。最终达到了使用的要求。

## 展望

本文所阐述和建立的基于工作流的支持多工具协同工作的软件开发过程管理模型致力于现代软件开发流程的自动化，提供自由可定制开发流程的方案，对软件开发流程进行全方位的管理与监控，对流程中的数据进行统一的统计、分析和追踪。它可以有效地解决现有的软件开发过程中的三个问题：

* 软件开发流程可配置

分析软件开发过程的步骤并分解，降低各个模块之间的耦合性，提升软件开发过程的灵活性和便捷性，使得软件辅助软件的参与不显得十分僵硬。

* 流程数据不一致

通过将工具与工作流对接的方式，将软件开发流程中的数据统一到工作流中进行流转，达成了流程数据的一致性。

* 支持多工具协同的流程管理

解决了软件开发过程中同时使用多种工具所带来工具数据和开发流程信息管理的问题。建立支持多种工具协同工作的软件开发过程管理平台，规范了多种工具下的开发流程管理，经理论分析和实践验证，在一定程度上提高了软件开发效率。

在目前架构设计中，工作流引擎对工具开放的API的规范性和通用性还有待提高，如何完善这样的工具与工作流对接标准是接下来的研究重点。这将会大大促进工具接入的效率，也有利于该模型的推广。

# 参考文献

1. 史美林，杨光信,向勇,等.WFMS:工作流管理系统[J]. 计算机学报. 1999, 22(3):325-334.
2. BAINAK, BENALIK, GODARTC. DISCOBOLE: a service architecture for interconnecting workflow processes [J]. Computers in Industry. 2006, 57(8/9):768-777.
3. 范玉顺.面向服务的企业建模和工作流管理研究[J].中国计算机学会通讯. 2006, 2(3):77-79+83.
4. LIU B, FAN Y S. Research on architecture and key technology for service-oriented workflow performance analysis [C]//Proceedings of International Workshop on Process Aware Information Systems. Berlin, Germany: Springer, 2007: 540-545.
5. <https://zh.wikipedia.org/wiki/>工作流技术. [2017-03-16]
6. 刘龙.“工作流”在中职学校数字化校园管理中的应用研究[J]. 当代教研论丛. 2015, 06:86-87.
7. 冉晓蓉.基于审批单的工作流在施工企业信息化管理中的应用[J]. 信息系统工程. 2015, 08:106-108+110.
8. 李志民，赵一丁.基于工作流的大型仓库物流监控系统设计[J]. 现代电子技术. 2016, 39(06):66-69.
9. 李鹏，辛海燕，苗元青，等.基于工作流技术的办公自动化系统在医院中的应用[J]. 齐鲁医学杂志. 2016, 31(01):96-98.
10. 张宏，任芳敏，郑荣贝.基于系统动力学的高速公路维护管理系统仿真研究[J].系统仿真学报. 2016, 28(03):676-682.
11. 高爱华，王寿喜，赵昆鹏，王佳伟，文林.天然气管道工程计算集成应用平台开发[J]. 天然气工业. 2015,35(7): 85-90.
12. 佟玉军，陈文实，赵悦，沈青松.CASE工具及其应用分析[J]. 辽宁工学院学报. 2003, 23(01):26-28.
13. 林云. CASE工具在软件工程过程管理中的运用[J]. 中国科技信息. 2008(04):94-95.
14. 耿冰，于修理.SaaS与传统软件的比较研究[J].沈阳师范大学学报（自然科学版）. 2009, 27(01):84-86.
15. 许四平.SaaS软件即服务模型研究[J]. 硅谷. 2009, 8(4):9-9.
16. 刘伟，朱晓非，聂亚平.浅谈软件工程和CASE工具的运用[J]. 中国科技信息. 2012, 24(14):90-90.
17. 郭慎平.基于Eclipse的嵌入式开发工具的研究与实现[D].武汉理工大学，2009
18. Workflow Management Coalition. Workflow Client Application Program Inteface (WAPI) Specification.WflVIC-TC-1009,Version1.2, October1996. http://www.wfmc.org/what-is-bpm. [2017-03-16]
19. 刘军，汤晓安，干哲，李晓慧.基于活动网络图的工作流模型研究[J]. 微计算机信息. 2009, (9):9-11.
20. 许可，杨路明.一种基于活动网络图扩展的工作流过程模型[J]. 微计算机信息. 2007, (27):120-122.
21. Salimifard K, Wright M. Petri net-based modelling of workflow systems: An overview[J]. European journal of operational research. 2001,134(3):664-676.
22. 谷建鑫,仇建伟.基于Petri网的工作流模型[J]. 计算机工程与设计. 2005, 26(2):513-515.
23. 钱纪蘅.浅谈基于行为理论的工作流模型[J]. 硅谷. 2011, (8):157-158.
24. 刘艳梅，郑国君，祁国宁.基于扩展事件驱动过程链(eEPC)的企业业务过程模型的仿真[J]. 制造业自动化. 2004, 26(2):18-22.
25. Scheer AW, Thomas O, Adam O. Process modeling using event-driven process chains[J]. Process-Aware Information Systems. 2005:119-146.
26. Zhou Jiantao, Shi Meilin, Ye Xinming. Formal verification techniques in workflow process modeling[J]. Journal of Computer Research and Development. 2005, 42(1):1-9
27. 李海波，战德臣，徐晓飞.工作流业务规则语义的完整性验证技术[J]. 计算机研究与发展. 2009, 46(7):1143-1151.
28. 杜彦华，范玉顺，李喜彤.基于模块化可达图的服务组合验证及 BPEL 代码生成[J]. 软件学报. 2010, 21(8):1810-1819.
29. 赵文，袁崇义，刘刚，张世琨，王立福.基于 P/T 系统化简方法的工作流过程模型验证[J]. 软件学报. 2004, 15(10):1423-1430.
30. 刘之强，李红燕，王磊，曲强.面向业务流程的数据模型异常检测方法[J]. 计算机学报. 2010, 33(8):1349-1358.
31. 张民，郭玉彬，李西明，蒋郁.工作流正确性问题综述倡[J]. 计算机应用研究. 2009, 26(5):1645-1649.
32. Arpinar, I.B., et al., Formalization of workflows and correctness issues in the presence of concurrency[J]. Distributed and Parallel Databases, 1999. 7(2): 199-248
33. 刘怡，张戡.基于负载平衡和经验值的工作流任务分配策略[J]. 计算机工程. 2009 , 35(21):57-59.
34. 曹健，張申生，周曉俊，張曉光.面向團隊工作的柔性工作流任務分配方法[J]. 計算機集成製造系統. 2003, 9(11):1006-1011.
35. 吉朝明，汪松.基于位置服务的工作流任务分配方法[J]. 计算机系统应用. 2017 , 15(2):227-230.
36. 王明微，周竞涛，敬石开.面向云制造的按需工作流任务分配方法[J]. 计算机辅助设计与图形学学报. 2012, 24(3):308-313.
37. 邓集波，洪帆.基于任务的访问控制模型[J]. 软件学报. 2003, 14(1):76-82.
38. 李凤华，苏铓，史国振，马建峰.访问控制模型研究进展及发展趋势[J]. 电子学报. 2012, 40(4):805-813.
39. 于春生，聂晶.基于组和角色的工作流权限访问控制模型[J]. 计算机应用. 2011, 31(03):778-780.
40. 张庆萍.基于任务和角色的访问控制模型研究[J]. 计算机安全. 2008, 13(9):74-75.
41. Koenig J. Jboss jBPM. White Paper, JBoss Labs, Atlanta, GA. 2004 Nov.
42. Lazo, Diego, and Diego Adrian Naya Lazo. OSWorkflow: A guide for Java developers and architects to integrating open-source Business Process Management. Packt Publishing Ltd, 2007.
43. 陈娴.开源工作流系统——Enhydra Shark分析[J]. 計算機與現代化. 2009, (5):73-77.
44. Wohed P, Russell N, Ter Hofstede AH, Andersson B, van der Aalst WM. Patterns-based evaluation of open source BPM systems: The cases of jBPM, OpenWFE, and Enhydra Shark[J]. Information and Software Technology. 2009, 51(8):1187-1216.
45. WU S G, SHI M L. Support environment for CSCW research-design and implementation of a desktop computer conferencing system[C]// Proceedings of International Conference on Communication Technology. Beijing, 1996. 187-190.
46. 史美林，杨光信. CODESIGN：一个协同设计支撑系统原型[J]. 清华大学学报（自然科学版）. 1998, 38(S1):30-35.
47. Ellis CA, Gibbs SJ. Concurrency control in groupware systems[C]. In: Proc ACM SIGMOD Confon Management of Data, Seattle, 1989: 399-407.
48. 赵瑞东，陆晶，时燕.工作流与工作流管理技术综述[J]. 科技信息. 2007, 24(08): 105-107.
49. 邓荣，张秀英.SOA 体系架构模型及特征分析[J]. 中国新技术新产品. 2010, (10):28-28.
50. 管红杰.SOA架构的工作流管理系统的研究与应用[J]. 计算机工程与设计. 2011, (3):23-26.
51. SUN H J, HUANG S X, FAN Y S. SOA based collaborative modeling method for cross-organizational business process integration [C]//Proceedings of International Workshop on Process Aware Information Systems. Berlin, Germany: Springer, 2007: 522-527.
52. 王浩鸣，冯博琴.传统工作流管理系统在电子商务应用中的改进[J]. 计算机工程. 2003, 29(9):10-11.
53. 张宏，佟玉军.中间件分类及标准[J]. 企业标准化. 2005, (11):20-21.
54. 秦璟，高文，储方杰.中间件技术研究[J]. 计算机应用研究. 2003, 20(8):35-37.
55. Haupt A, Leffhalm K, Wegner P, WiesandS. Joining the petabyt club with direct attached storag[J]. Journal of Physics: Conference Series, Volume 331, Part 1: Plenary,2011,33(1):1-6.
56. Roger D. Chamberlain, Berkley Shands. Direct-Attached Disk Subsystem Performance Assessment[C]//Fourth International Workshop on Storage Network Architecture and Parallel I/Os. Piscataway, NJ:IEEE Press, 2007:71-78
57. Garth A. Gibson, Rodney Van Meter. Network attached storage architecture[J]. Communications of the ACM, 2000, 43(11):37-45
58. De-zhi Han, Jian-zhong Huang. Security for the Storage Network Merging NAS and SAN[C]//Machine Learning and Cybernetics, 2006 International Conference on IEEE. Piscataway, NJ:IEEE Press, 2006:736-741
59. Ramani Routray, Sandeep Gopisetty, Pallavi Galgali, et al. iSAN: Storage area network management modeling simulation[C]// Networking, Architecture, and Storage, 2007. NAS 2007. International Conference on IEEE. Piscataway, NJ: IEEE Press, 2007:199-208
60. J. S. Glider, C. F. Fuente, W. J. Scales. The software architecture of a SAN storage control system[J]. IBM Systems Journal. 2003,42(2):232-249
61. Haiguo Zhu, Quanyi Huang, Jiming Guo, et al. Researches on Geographic Data Storage and Management Based on Object-Oriented Storage in the Distributed Software Architecture[J]. 2009 WRI World Congress on Software Engineering, 2009, (1):89-93
62. 苏勇，周敬利.基于iSCSI OSD存储系统的设计与分析[J]. 计算机工程与应用. 2007, 43(23):103-105
63. 吴明礼，张宏安.数据存储技术综述[J]. 北方工业大学学报. 2015, 27(1):30-35.
64. Gupta P, Govil MC. Spring Web MVC Framework for rapid open source J2EE application development: a case study[J]. International Journal of Engineering Science and Technology. 2010, 2(6):1684-1689.
65. 舒礼莲.基于SpringMVC的Web应用开发[J]. 计算机与现代化. 2013, (11):167-168+173.
66. ShengBin Yin, XiaoFan Bian. Construct the Component of the immature realm Using Data Mining[C]//Proceedings of 2004 International Conference on Machine Learning and Cybernetics, ShangHai, 2004:1509-1512.
67. Gupta P, Govil MC. MVC Design Pattern for the multi framework distributed applications using XML, spring and struts framework[J]. International Journal on Computer Science and Engineering. 2010, 2(4):1047-1051.
68. Web MVC framework. https://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html. [2017-03-16].
69. Rizescu, Gianina. Businesses Integration with Workflow and Web Service Technologies [J]. (2008).
70. 张海军，史维峰，刘伟.基于SOA企业应用集成框架研究与实现[J].计算机工程与设计. 2008, 29(8):2085-2088.
71. 刘明忠.支持跨企业协作的柔性工作流系统建模研究[J]. 计算机集成制造系统. 2004, 10(11):1356-1363.
72. 蔡孝武，韩永国，蓝科.一种轻量级工作流引擎的研究与设计[J]. 计算机工程. 2010, 36(20):78-82.
73. 李松犁，张型龙，肖俊超. 面向服务集成的自动化服务注册方法[J]. 计算机应用与软件. 2016, 33(06):59-63.
74. 邵虹，崔文成.面向服务的工作流管理系统研究[J]. 计算机集成制造系统. 2008, (14):2212-2216.
75. 张中林，殷新春，陈峻.基于XML的工作流定义语言的研究[J]. 扬州大学学报（自然科学版）. 2002, (03):48-52.

# 致谢