



南京大學

研究生毕业论文 (申请工学硕士学位)

论 文 题 目 基于 Hyperledger Fabric 的
区块链云化框架的研究与实现

作 者 姓 名 张富利

学科、专业名称 软件工程

研 究 方 向 DevOps、区块链

指 导 教 师 张贺 教授

2022 年 2 月 28 日

学 号：**MG1932016**

论文答辩日期：xxxx 年 xx 月 xx 日

指导教师： (签字)

by

Fuli Zhang

Supervised by

Professor He Zhang

A dissertation submitted to
the graduate school of Nanjing University
in partial fulfilment of the requirements for the degree of
MASTER OF SCIENCE
in
Software Engineering



Software Institute
Nanjing University

February 28, 2022

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：_____

软件工程 专业 2019 级工学硕士生姓名：张富利
指导教师（姓名、职称）： 张贺 教授

摘 要

领域驱动设计是一种针对复杂业务流程的软件设计方法，它可以帮助架构师和软件开发人员提炼业务流程和构建复杂软件系统。近年来，越来越多的团队将领域驱动设计应用到大型分布式系统的设计与实现中。

然而，作为领域驱动设计的核心内容，战术建模的应用却存在诸多问题与挑战。其一，战术建模模式的实践过程仍缺少标准且统一的规范，使得实践者无法准确地使用这些模式建模业务流程；其二，战术建模实践过程所缺少的规范，也使得开发人员和架构设计人员在使用这些建模模式时具有不同的理解，从而带来了一系列项目团队沟通问题；其三，现有的大多数建模平台对领域建模过程的支持仍不完善，极大地阻碍了领域战术建模的应用。

为了解决上述问题与挑战，本文首先调研了现阶段有关领域驱动设计战术建模的理论文献，对具有相关实践经验的从业者进行访谈，提炼出一系列面向战术建模过程的实践规范作为战术建模指南。所提炼的建模指南包含一系列战术建模中使用到的战术模式、这些战术模式的属性、使用时机与实现技术。然后，基于 UML 扩展机制定义标准的领域驱动设计战术建模元模型作为战术建模语言。上述建模指南和战术建模语言共同组成本文所提出的战术建模支持方法。最后，基于所提出的战术建模支持方法，本文还实现了一个可视化战术建模工具 DDDD (Draw for Domain-Driven Design)，该工具支持标准化的战术建模、对建模结果的验证、复用以及扩展，帮助实践者更规范地进行战术建模。

本文采用案例研究对提出的建模支持方法及工具进行验证。结果表明本文提出的战术建模支持方法及工具，能够帮助领域驱动设计的实践者开展更规范化战术建模过程，从而解决战术建模实践过程缺少规范带来的一系列挑战，一定程度上促进了战术建模的应用。

关键词：领域驱动设计；战术建模；建模工具；特定领域建模语言

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Modeling Support Method and Tool for
Domain-Driven Design

SPECIALIZATION: Software Engineering

POSTGRADUATE: Fuli Zhang

MENTOR: Professor He Zhang

Abstract

As a software design approach for complex business process, Domain-Driven Design (DDD) enables architects and developers building complex software system. In recent years, more and more teams have applied DDD to the design and implementation of large-scale distributed systems.

However, as the core means of DDD, tactical modeling are facing many challenges in its application. Firstly, the practicing process of tactical modeling patterns still lacks standard and unified specifications, making it hard for practitioners to use these patterns to model business processes accurately; secondly, the lack of specifications also makes developers and architects have different understandings when using the modeling patterns, leading to a series of communication problems in project team; thirdly, most of the existing modeling platforms have incomplete support for the tactical modeling process, severely impeding the application of DDD.

In order to solve the aforementioned challenges, this thesis first reviews theoretical literature on tactical modeling, then interviews practitioners with relevant practical experience, and finally extracts a series of practical specifications for the tactical modeling process. As a guide for tactical modeling, the specifications contain a series of tactical patterns used in tactical modeling, their attributes, time of using, and implementation techniques. Furthermore, the standard tactical modeling metamodel is defined based on UML profile mechanism, as a tactical modeling language. The above modeling guide and tactical modeling language together constitute the tactical modeling support method proposed in this thesis. Based on the proposed method, this thesis also implements a visual tactical modeling tool called Draw for Domain-Driven Design, or DDDD. The tool

supports standardized tactical modeling, verification, reuse and extension of modeling results to help practitioners develop tactical modeling in a more standardized way.

This thesis uses case studies to verify the proposed modeling support method and tool. Results show that the proposed method and tool can help DDD practitioners to conduct a more standardized tactical modeling process, thus solving the challenges caused by lack of specification in the tactical modeling process, and promoting the application of tactical modeling to a certain extent.

keywords: Domain-driven design, Tactical modeling, Modeling tool, Domain-specific modeling language

目 录

目 录	iv
插图清单	vi
附表清单	viii
第一章 绪论	1
1.1 研究背景及意义	1
1.2 区块链云化研究现状	3
1.3 本文主要研究工作	4
1.4 本文组织结构	4
1.5 本章小节	5
第二章 理论与技术支持	6
2.1 区块链技术	6
2.1.1 区块链基本概念	6
2.1.2 Hyperledger Fabric	8
2.1.3 Blockchain as a Service	10
2.2 云原生	11
2.2.1 云原生基本概念	11
2.2.2 Kubernetes	15
2.3 其他相关技术	17
2.3.1 Helm	17
2.3.2 Istio	18
2.4 本章小结	19
第三章 基于 Hyperledger Fabric 的区块链云化框架	20
3.1 区块链基础设施现状	20
3.2 设计原则	21
3.3 基于 Hyperledger Fabric 的区块链云化框架	22
3.3.1 Custom Resource Definition	23
3.3.2 Manager	25

3.3.3 Fabric 网络	28
3.4 本章小结	28
第四章 原型工具设计与实现	30
4.1 概述	30
4.2 需求分析	33
4.2.1 工具总体功能	33
4.2.2 可视化建模模块需求分析	34
4.2.3 模型校验模块需求分析	35
4.2.4 模型存储与转化模块需求分析	37
4.3 工具设计与实现	38
4.3.1 总体设计	38
4.3.2 可视化建模模块详细设计与实现	40
4.3.3 模型校验模块详细设计与实现	42
4.3.4 模型存储与转化详细设计与实现	45
4.4 本章小结	48
第五章 建模支持工具测试与案例研究	49
5.1 建模支持工具测试	49
5.1.1 可视化建模测试	49
5.1.2 建模约束校验测试	50
5.1.3 模型存储测试	51
5.1.4 生成项目测试	52
5.1.5 性能测试	52
5.2 案例研究	53
5.2.1 验证步骤	54
5.2.2 验证过程	54
5.3 本章小结	58
第六章 总结与展望	59
6.1 总结	59
6.2 展望	60
致 谢	61
参考文献	62
简历与科研成果	66

插图清单

1-1 区块链云化框架及其原型工具	3
2-1 区块链示例图	6
2-2 Hyperledger Fabric 网络架构	8
2-3 Fabric 账本结构	9
2-4 BaaS 与 PaaS 以及 SaaS 的对比	10
2-5 BaaS 通用架构	11
2-6 云原生技术发展趋势	12
2-7 云原生技术范畴	13
2-8 国内外云原生技术现状	14
2-9 Kubernetes 架构图	15
2-10 Helm 架构图	18
3-1 基于 Hyperledger Fabric 的区块链云化框架	23
3-2 Manager 监听 CRs	25
3-3 controller 循环监听	26
3-4 区块链云化框架的安全性	27
4-1 建模支持工具业务流程	30
4-2 工具总体功能	34
4-3 可视化建模模块用例图	34
4-4 模型校验模块用例图	36
4-5 模型存储与转化模块用例图	37
4-6 支持工具整体架构图	39
4-7 支持工具前端整洁架构图	40
4-8 可视化建模模块类图	41
4-9 可视化建模模块顺序图	42
4-10 初始化 ToolBar 代码	43

4-11 模型校验模块类图	44
4-12 模型校验部分实现代码	46
4-13 模型存储与转化类图	46
4-14 模型存储与转化顺序图	47
4-15 模型存储部分实现代码	48
5-1 图形化建模测试结果图	50
5-2 建模约束校验测试结果图	51
5-3 工具帮助页面图	54
5-4 工具主页	55
5-5 可视化建模过程	55
5-6 模型校验	56
5-7 保存导出模型	56
5-8 加载历史模型	57
5-9 建模结果图	57

附表清单

1-1 主要公有云的 BaaS 平台	1
2-1 区块链类型	7
3-1 CRD 描述	24
4-1 可视化建模用例表	35
4-2 模型校验用例表	36
4-3 模型存储与转化用例表	37
5-1 可视化建模测试用例	50
5-2 建模约束校验测试用例	51
5-3 模型存储测试用例	51
5-4 生成项目测试用例	52
5-5 测试环境	52
5-6 接口性能测试	53
5-7 案例验证结果	58

第一章 绪论

1.1 研究背景及意义

区块链 (Blockchain) 是一种按照时间顺序将数据区块以链条的方式组成的特定数据结构, 被视为一个分布式的共享账本和数据库。它能够使用户无需相互信任与可信第三方的条件下完成可信的价值传输 [1]。作为区块链 2.0 的以太坊^①重新将智能合约描述为图灵完备的、部署于区块链网络中合同条款代码。这意味着传统的合同条款可以进入实体计算机中, 且在区块链网络去中心化、不可伪造、不可篡改的特性下严格执行。区块链推进了人与企业之间和线上与线下之间的全方面互联, 其被成为下一代的新型生产关系。随着区块链的快速发展, 智能合约极大地丰富和扩展了区块链应用场景。它们为供应链、金融等传统领域带来重大变革, 已经快速渗入到人们生活的方方面面。

表 1-1: 主要公有云的 BaaS 平台

BaaS 平台	Ethereum	Quorum	Corda	Fabric
AWS	Y	Y	Y	Y
Azure	Y	Y	Y	Y
IBM				Y
阿里云区块链服务	Y			Y
腾讯云区块链服务 TBAAS				Y
华为云区块链服务 BCS				Y

与此同时, 云原生 (Cloud native) 作为一种基于云的基础之上的软件架构思想, 以及基于云进行软件开发实践的一组方法论。因其弹性和分布式的优势成为当今流行的软件服务模式。云原生技术有利于各组织在公有云、私有云和混合云等新型动态环境中, 构建和运行可弹性扩展的应用, 借助平台的全面自动化能力, 跨多云构建微服务, 持续交付部署业务生产系统。区块链即服务 (Blockchain as a Service, 简称 BaaS) 则是基于云原生技术体系的一种构建、管理、托管和运维区块链网络及其应用的云服务平台 [2]。BaaS 支持将任何企业级区块链实施

^①以太坊白皮书

到云环境，而无需任何 IT 专业知识。这大大降低了区块链技术的使用门槛，是促使区块链技术更广泛、更深入地渗透到各个行业和企业的催化剂，其市值预计从 2018 年的 6.23 亿美元猛增 2023 年的 150 亿美元^①。其中，云厂商提供了大多数的 BaaS 平台 [3]，如表 1-1 所示，其底层区块链支撑技术大多数选择 IBM 开源的跨企业级联盟链 Hyperledger Fabric，这也是本文选择 Hyperledger Fabric 的原因。

然而，当前 BaaS 平台的发展尤其是底层的区块链基础设施的建设依旧存在诸多挑战。

第一，基础商业化应用工具并不完善^②。虽然市场上存在多种可选择的 BaaS 平台解决方案，但是这些 BaaS 平台由商业巨头把控，行业马太效应明显 [3]。BaaS 平台的构建需要专业的区块链以及云原生的技术能力，只有实力雄厚的云厂商进行 BaaS 平台的研发，这些 BaaS 服务往往与云计算节点捆绑，可用性限制通常会迫使其他企业为来自各种云提供商的基础设施即服务 (Infrastructure as a Service，简称 IaaS) 付费，这导致了基于多云的网络 [4]。

第二，现阶段 BaaS 平台利用云能力对区块链基础设施赋能乏力。对于 BaaS 平台市场规模的不断膨胀，需要找到解决当前区块链网络部署、备份、升级以及数据存储以及可扩展性的方法。区块链与云原生两种技术都相对不成熟，但两者的集成可能会在技术方面产生新的复杂性 [2]。当前 BaaS 平台仅提供了一种基于开源区块链平台的一键化自动部署管理方案，未深入到区块链与云基础设施的底层。这并不是有效的云化方式，浪费了云原生技术的潜力。区块链如何和云原生快速深度结合，利用原生伸缩性、可移植性和高可用性价值来提供“高质量”的区块链服务，是当前仍然遗留的挑战。

针对上述问题，本文选择面向联盟链场景开源框架 Hyperledger Fabric 和 Kubernetes 针对 BaaS 平台的区块链基础设施提出了一种基于 Hyperledger Fabric 的区块链云化框架，具体工作如图 1-1 所示。该框架通过对 Fabric 网络中的 Ca、Orderer、Peer 组件进行抽象适配并利用 Kubernetes Operator 对其进行完整生命周期管理。本框架利用原生 Kubernetes 安全性、可扩展性等机制使用户通过专家提供的领域知识对 Fabric 网络获得类似云的自我管理。此外，本文还实现了基于区块链云化框架的原型工具，支持更简单、更原生的管理 Fabric 网络。

^①Global Blockchain-as-a-Service Market 2018-2022

^②区块链基础设施研究报告 (2021 年)

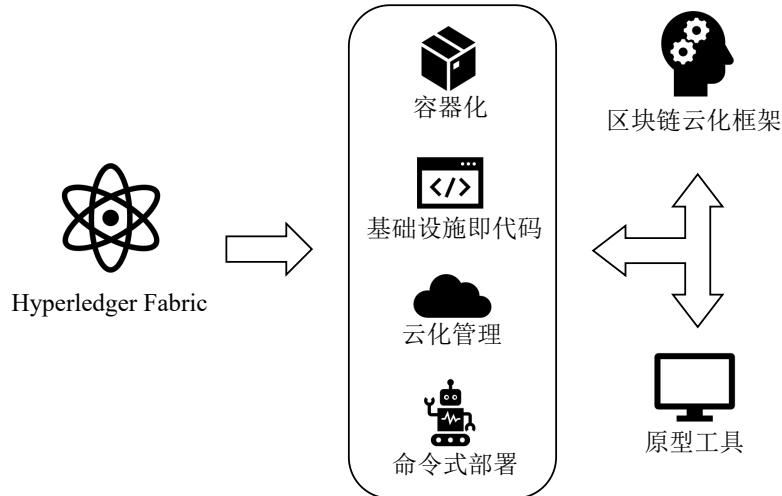


图 1-1: 区块链云化框架及其原型工具

1.2 区块链云化研究现状

云原生背景下的区块链自诞生以来就受到了学术界与工业界的广泛关注。除了利用区块链的特性提升云的能力外 [5][6][7], 研究人员也都期望自动化地构建出易于弹性扩展、高可用的区块链平台, 在区块链基础设施与云原生结合方面都开始了一定的探索。

在学术研究方面, 研究人员在探究如何有效利用云平台来部署区块链平台。Gerrits 等人 [4] 在 Kubernetes 中部署了分布式账本 Hyperledger Sawtooth^①, 并运行了一个用例。他们旨在探讨该用例在真实场景云部署中的可行性和可扩展性。Liang 等人 [8] 针对教育领域数据共享和信息欺骗的问题构建了一个高可用的教育联盟区块链平台, 并实现了基于 Kubernetes 的 Fabric 部署, 实现了将链码纳入 Kubernetes 环境管理的目标。

然而, Wan 等人 [9] 指出当前主流的 BaaS 提供商通常采用 API 进行用户访问, 或者简单地将区块链应用迁移到云, 这会侵蚀不可信的机制并带来锁定风险。他们随后提出了一种新的服务范式来克服现有 BaaS 的局限性。基于 Hyperledger 的实施表明, 该范式可以缓解当前 BaaS 对区块链特征的侵蚀。在云原生底层基础设施方面研究人员关注与区块链结合的 Kubernetes 调度问题。才 [10] 在 Kubernetes 上面对 PBFT 和区块链的本身特性提出了静态调度和自适应算法。Shi 等人 [11] 为了解决云端现实 PoS 区块链工作负载的高效调度问题, 首

^①Hyperledger Sawtooth github 地址

次在云计算中设计和实现了基于 Kubernetes 的解决 PoS 区块链应用程序迁移成本的系统,最大限度地减少了使用的 Kubernetes 工作节点数量以降低总体费用,而且还提出了一种高性能的 Kubernetes 调度方案 HPKS 以最大限度地利用工作节点进行在线 pod 管理。

相比于学术研究,工业领域的探索更加注重自动化实践。Hyperledger Cello^①支持在多种底层基础设施上从头快速构建 BaaS 平台,提供管理区块链网络的生命周期、自定义区块链网络配置等功能帮助人们以更高效的方式使用和管理区块链。Hyperledger Cello 当前阶段重点关注在 Docker 安装,对于 Kubernetes 支持方面的仍处在相对初级阶段,配置项简单灵活性不足且老旧。Blockchain Automation Framework^②提供了一个自动化框架,利用 Ansible^③以及 Helm^④快速地、一致地将生产就绪的分布式账本技术 (Distributed ledger technology, 简称 DLT) 平台部署到云基础设施。虽然,Blockchain Automation Framework 提供了一个自动化框架将区块链平台部署于 Kubernetes,但本质上还是描述为一个需要希望远程主机执行命令的方案,或者一组 IT 程序运行的命令集合。这大大提升了自动化程度,但其远没有发挥 Kubernetes 的潜力。

尽管学术界以及工业界目前已有一些关于区块链云化的探索与研究,研究重点多为如何自动化地将区块链平台向云上迁移。总体来说目前的研究缺少构建支持云和区块链一体化的有效服务模型 [11]。

1.3 本文主要研究工作

1.4 本文组织结构

本文组织结构如下:

第一章 绪论。介绍了本文的研究背景及意义、国内外研究现状、工业界的主要探索以及本文主要的研究工作;

第二章 理论与技术支持。介绍区块链尤其是 Hyperledger Fabric 的相关理论和概念;同时介绍云原生的基本概念发展历程,并对云原生基础设施 Kubernetes 进行了详细介绍;

^①Hyperledger Cello github 地址

^②Blockchain Automation Framework github 地址

^③Ansible github 地址

^④Helm github 地址

第三章 基于 Hyperledger Fabric 的区块链云化框架。介绍区块链基础设施的现状及挑战，基于这些现状区块链云化框架应当具有的设计原则，并详细介绍了本文提出的基于 Hyperledger Fabric 的区块链云化框架。

第四章 原型工具设计与实现。；

第五章 框架检验与评估。；

第六章 总结与展望。

1.5 本章小节

第二章 理论与技术支持

本章将介绍区块链及云原生相关概念和理论知识,还将介绍本文区块链云化框架所涉及到的其他技术与工具。

2.1 区块链技术

2.1.1 区块链基本概念

区块链是以比特币等数字加密货币体系的核心支撑技术,它是一种全新的去中心化基础架构与分布式计算范式 [12]。区块链通常被当作分布式账本,具有去中心化、持久性、匿名性、不可篡改性、可追溯性的特点。

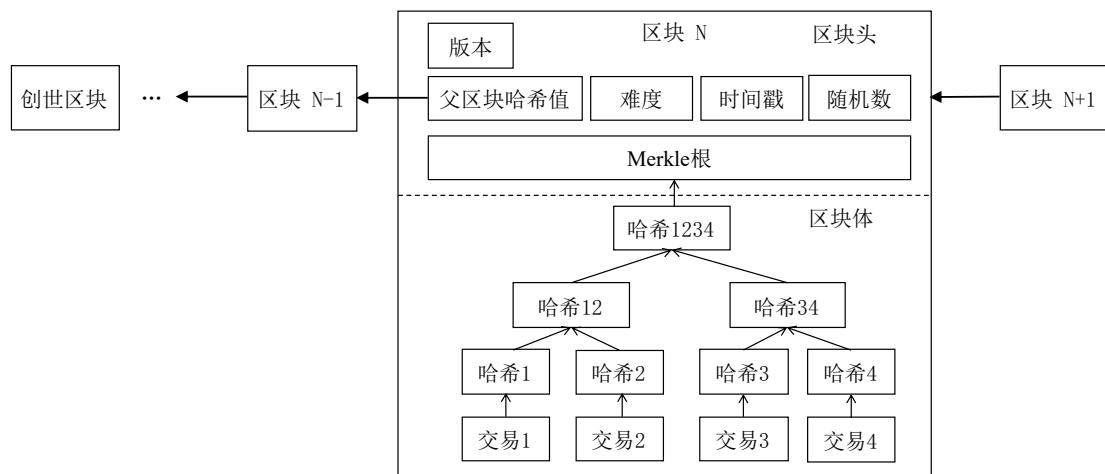


图 2-1: 区块链示意图

区块链典型示例如图 2-1 所示,可以看作是一种按照时间顺序将数据区块以顺序相连的方式组合成的一种链式数据结构,并以密码学方式保证的不可篡改和不可伪造的分布式账本。每个数据区块包含区块头 (Block header) 和区块体 (Block body) 两部分,区块头主要用来存储本区块的一些相关属性,区块体则用来存储真实的交易数据记录。区块头主要由三组数据组成,第一组是父区块的哈希值,用来将该区块与它的前一区块相连接;第二组数据和矿工竞争挖矿有关,

即难度、时间戳和随机数 (Nonce); 第三组是由区块体中计算出来的根哈希值, 即默克尔 (Merkle) 根。区块体包括当前区块经过验证的、区块创建过程中生成的所有交易记录。这些记录通过默克尔树的哈希过程生成唯一的默克尔根并记入区块头。整个区块链到第一个区块称为创世区块 (Genesis block)。如果网络中大多数节点通过共识机制就新区块中交易的有效性和区块本身的有效性达成共识, 则可以将新区块添加到链中。

表 2-1: 区块链类型

	公有区块链	私有区块链	联盟区块链
准入限制	无	有	有
读取者	任何人	仅限受邀用户	相关联用户
写入者	任何人	获批参与者	获批参与者
所属者	无	单一实体	多方实体
交易速度	慢	快	快

当前, 区块链分为公有区块链、联盟区块链和私有区块链。如表 2-1 所示, 公有链没有准入限制, 没有监管方可以组织参与, 任何人都可以参与共识, 常见的两种共识协议为工作量证明机制 (Proof of work, 简称 PoW) 和权益证明机制 (Proof of stake, 简称 PoS)。由于任何人都可以自由加入, 因此公有链网络具有高度分布式的拓扑结构。但是, 公有链在安全性和性能方面也进行了权衡。公有链上的许多服务器遇到了扩展瓶颈, 吞吐量相对较弱; 与公有区块链的无准入限制形成鲜明对比的是, 私有区块链建立了准入规则, 规定谁可以查看和写入区块链。因为在控制方面有明确的层次结构, 私有链也不是去中心化系统。在某些私有链中, 具备安全模型的背景下, 共识协议是多余的。因此在私有区块链中, 不使用 PoW 并不会造成很严重的威胁, 因为每个参与者的身份都是已知的, 是手动进行管理的; 联盟区块链是介于公有链和私有链之间的, 结合了两者的特征要素。在共识方面, 联盟链将少数同等权力的参与方视为验证者, 而不是像公有链那样开放的系统, 让任何人都可以验证区块, 也不是像私有链那样, 通过一个封闭的系统, 只允许某一个实体来任命区块的生产者。对于从事各类活动的个人和企业来说, 存在大量的区块链选择。即使在公有链、私有链和联盟链中, 根据复杂性的不同, 也会出现许多不同的用户体验。根据实际使用情况, 企业可以选择最适合实现自己目标的产品。供应链、电商、医疗等需要彼此之间需要相互沟通的场景下, 联盟链可减轻私有链中交易对手的风险, 并且较少的节点数通常可使它们能够比公共链更有效率的运行, 通常选择联盟链作为企业

级区块链的底层。

2.1.2 Hyperledger Fabric

Hyperledger Fabric^①是一种企业级联盟链解决方案，因其可插拔模块化、可伸缩、可扩展的架构、多编程语言的智能合约受到业界广泛追捧。

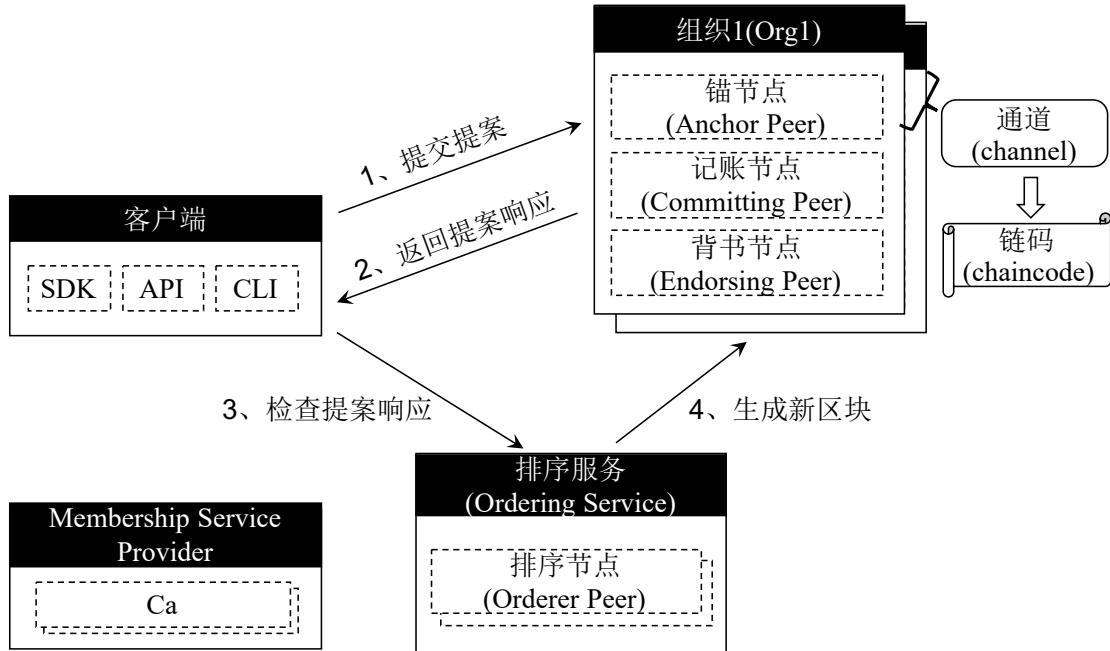


图 2-2: Hyperledger Fabric 网络架构

如图 2-2 所示，Fabric 网络通过组织划分，每个组织内包含多种不同角色的 Peer 节点，每个 Peer 节点又可以担任多种角色，所有的组织共用排序服务。Fabric 多种节点通过网络相互链接组成联盟链网络完成链上交易。

网络节点

(1) 客户端节点: 在 Fabric 网络外部用于主动与区块链交互实现区块链操作的组件。常见的包含软件开发工具包 (Software Development Kit, 简称 SDK)^②、fabric-cli^③、REST API^④；

(2) Ca 节点: fabric-ca^⑤是一个官方可选的 Membership Service Provider 组件，对 Fabric 网络中各实体 (Identity) 的数字身仹证书进行管理管理。完成实体身份

^①Hyperledger Fabric github 地址

^②Hyperledger Fabric Go 版本的 SDK

^③fabric-cli github 地址

^④fabric api github 地址

^⑤fabric-ca github 地址

注册、数字证书的签发续签或吊销;

(3) Peer 节点: Fabric 网络的每个组织都包含一个或多个 Peer 节点, 每个 Peer 节点可以通过配置文件担任一个或同时担任多种角色。

- 锚节点 (Anchor Peer): 负责与其他组织的锚节点进行通信;
- 记账/提交节点 (Committing Peer): 负责对区块及区块交易进行验证, 验证通过后将区块写入账本中, 同时提交节点会定期与其他节点通过 Gossip 协议进行信息交换;
- 背书节点 (Endorsing Peer): 负责对客户端发送的提案进行签名背书。背书节点与具体的链码 (chaincode) 绑定, 其通过调用链码模拟执行交易并向生成提案的客户端返回提案响应。背书节点是动态的, 在客户端发起提案时才会根据背书策略 (Endorsement policy) 成为背书节点, 其他时候为记账节点。

(4) Orderer 节点: 排序服务节点接收经过背书签名的交易并对未打包的交易进行排序生成新区块, 最终通过原子广播到记账节点。排序服务采取支持可插拔设计, 支持 Solo、Kafka 等分布式共识协议。

Fabric 区块链支持在通信节点之间启用传输层安全性协议 (Transport Layer Security, 简称 TLS) 保证两通信节点的数据保密性和完整性。TLS 采用 X.509 证书进行身份验证并生成会话密钥, 不仅支持客户端节点对服务节点的身份验证, 同时也可以支持服务节点来验证客户端的身份的双向验证。

账本结构

Fabric 中智能合约被称为链码, 通过通道 (channel) 允许参与者同时履行不同的链码。Fabric 网络子链通常按照由“1 个通道 +1 个账本 +N 个成员”组成。不同组织及其成员能够在通道中完成特定的交易, 限制信息传播范围。建立一个通道就相当于创建了一条子链, 这条子链上只拥有唯一的一份账本。

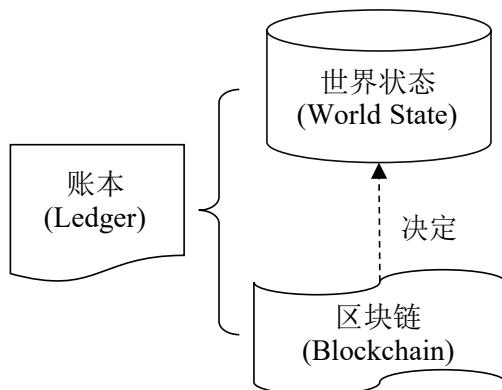


图 2-3: Fabric 账本结构

如图 2-3 所示, Fabric 账本由区块链以及世界状态 (World State) 组成, 其中世界状态由区块链决定。首先, 世界状态是一个可插拔的键值对 (key-value, 简称 k-v) 数据库, 提供简单、快速、丰富的账本状态的检索和存储方式。通过世界状态, 客户端能够直接定位访问账本状态的某个值, 不需要遍历计算整个交易日志。为解决不同类型的问题, Fabric 提供了 LevelDB 和 CouchDB 来保障账本状态类型的灵活性。当账本状态是简单的键值对时, 使用 LevelDB 合适; 当账本状态结构为 JSON 时, 使用 CouchDB 合适。其次, 这里的区块链指的是交易日志, 是指区块形成的链。区块记录了世界状态改变的历史, 并以文件的方式进行持久化。交易数据一旦写入区块链就无法篡改。

2.1.3 Blockchain as a Service

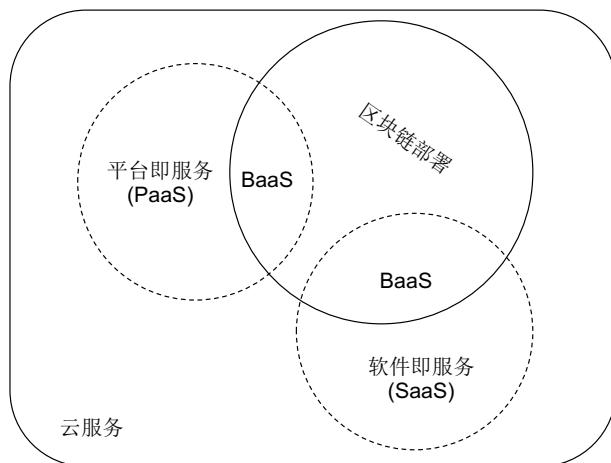


图 2-4: BaaS 与 PaaS 以及 SaaS 的对比

随着区块链以及云原生的发展, BaaS 也悄然兴起。云提供了一种抽象、汇集和共享整个网络中的按需的、虚拟的、资源可伸缩的 IT 环境, BaaS 则提供了一种基于云的区块链服务。BaaS 能够在云上构建、管理、托管和运维区块链技术的各个方面, 提供快速部署区块链网络及其开发环境、编写智能合约、构建去中心化应用 (即区块链应用, Decentralized application, 简称 Dapp)。基于云基础设施, BaaS 屏蔽了底层区块链与云原生的逻辑, 消除了用户构建开发去中心化应用的壁垒, 尤其是部署区块链网络所需的大量硬件和专业知识的前期成本, 为用户提供便捷的、一体化的区块链的能力。如图 2-4 所示, 根据 BaaS 的实施方式, BaaS 在云环境中的位置会有所不同 [2]。BaaS 可以从使用平台即服务 (Platform as a Service, 简称 PaaS) 获得基础设施支持的同时也可以通过软件即服

务 (Software as a Service, SaaS) 获得软件服务。

微软在推出由 Azure 云驱动的开放式区块链平台 Bletchley, 该项目保证服务对于所有平台、合作者和客户来讲都是开放的、灵活的 [13]。IBM 推出了名为 Bluemix 的云计算平台, 依托于 PaaS 云将帮助开发者更快的进行应用开发和部署。随后 AWS、Google、阿里云等也相继推出自家的区块链即服务平台。以 Hyperledger Fabric 为例, BaaS 平台通用的架构如 2-5 所示, 本质上 BaaS 以计算、存储资源等资源为基础, 联合上层的区块链基础设施以及相关能力, 如共识能力、记账能力、智能合约等转化为可编程接口, 使得区块链网络的部署以及去中心化应用开发过程简单而高效。同时, BaaS 通过底层标准化的云基础设施能力为上层的区块链及其去中心化应用提供安全可靠的支撑, 解决弹性、网络、安全性、性能等难题。本文重点关注于基础设施层中 Kubernetes 上的区块链云化服务模型的研究。

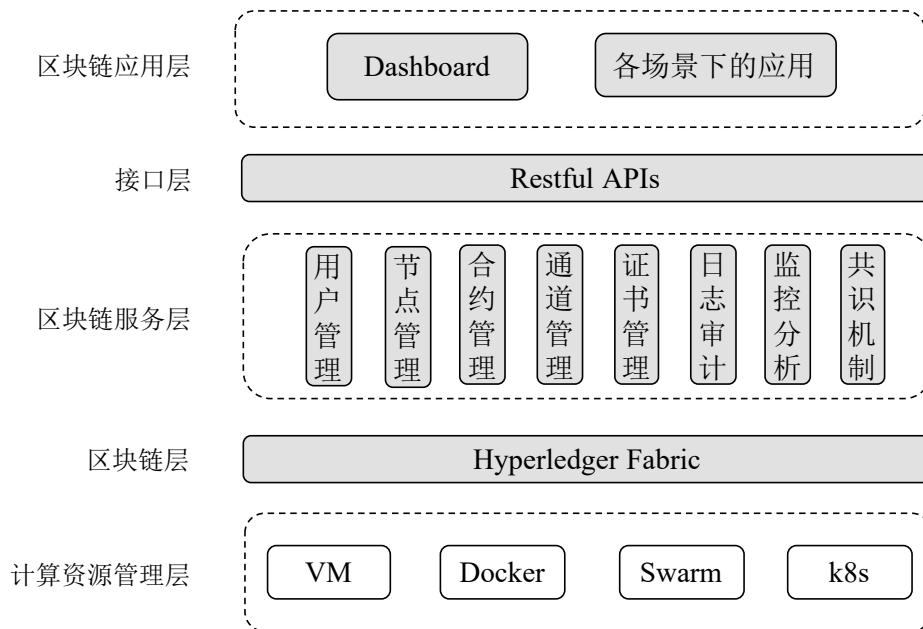


图 2-5: BaaS 通用架构

2.2 云原生

2.2.1 云原生基本概念

云原生, 即云原生计算。从发展历程来说, 云原生是云计算的升级。云计算最早由 Dell 公司在 1996 年提出 [14], 亚马逊公司在 2006 年率先推出的弹性计

算云 (Elastic Compute Cloud, 简称 EC2) 服务对云计算产生了深刻影响, 越来越多的企业开始逐步接受云计算这一概念, 并将应用逐步迁移到云端, 享受这一新型计算方式带来的技术红利。此后, 软件系统规模、软件开发方式驱动着技术不断升级。在云计算的时代, 云端只是用于计算的场所, 应用无须重新编写, 只需重新部署, 应用的迁移从物理机到虚拟机, 存储选用兼容的块存储或文件存储。但几乎所有分布式场景中都需应用自行解决稳定性、数据同步、容灾等方面的问题。要解决这些问题, 只能从根本上寻求解决方案。即从迁移到云转变为诞生于云。2013 年 Docker 开源, 之后 Pivotal 公司提出了云原生的概念, 这是对云计算概念的全面升级。Pivotal 指出云原生由容器、微服务、DevOps 以及持续交付等技术 [15] 组成, 并充分利用云计算优势构建和运行应用。2014 年, 容器编排技术 Kubernetes 发布。容器技术日趋成熟, 在业界开始广泛应用。在 2015 年, 云原生计算基金会 (Cloud Native Computing Foundation, 简称 CNCF) 成立。而到了 2021 年, CNCF 已经孵化了超过 120 个项目、740 名成员以及 142000 的贡献者^①。除了工业界, 云原生在学术界也引起了关注, 《计算机学报》发起了以“云原生”为主题的专刊征文^②。

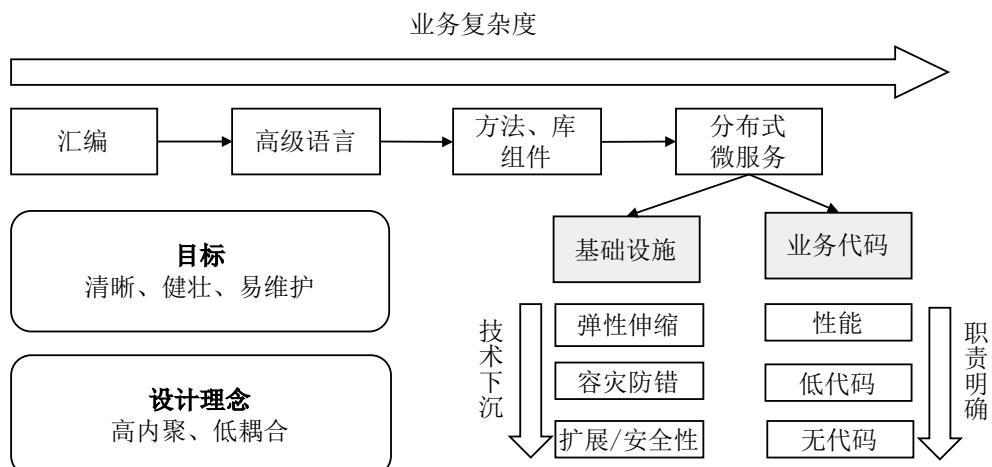


图 2-6: 云原生技术发展趋势

以软件工程的视角来看, 如图 2-6 所示, 随着软件规模、软件复杂程度在不断增大, 软件上线速度需要不断加快, 软件体量不断增大, 软件稳定性的要求在不断提高。围绕着“高内聚、低耦合”的设计理念, 软件制品进一步深层次抽象, 从单体架构演化为分布式微服务架构, 从可复用的方法、库、组件进一步下沉到底

^①CNCF2021 年年度报告

^②《计算机学报》云原生软件技术与工程实践专刊征文通知-CCF2021 中国软件大会

层的基础设施。在这些客观需求的驱动下,敏捷进一步向运维端延伸,继瀑布开发、敏捷开发之后,开发运维一体化 (Development and Operations, 简称 DevOps) 成为又一新兴的软件开发理念和愿景。由于软件体量巨大,多个不同职责明确的团队负责整体软件项目的运行。DevOps 旨在通过一系列文化及技术手段(尤其是自动化 IT 工具链)打破开发和运维团队之间的壁垒,改善团队之间的协作关系,实现更加频繁快速、可靠的软件产品交付 [16]。DevOps 不仅将敏捷向扩展到运维端,其更涉及到软件全生命周期中的人、流程与平台。可以说,DevOps 是当下软件工程的第一生产力,其是一种普世的价值观。云原生作为一种基于云基础设施的技术体系涵盖了云应用定义、开发、构建与运行时的所涉及到的各工具或平台。云原生是 DevOps 生产力下的现阶段生产工具的体现,是 DevOps 的价值具象,“DevOps 时代下的云原生”就如“蒸汽时代的蒸汽机”。

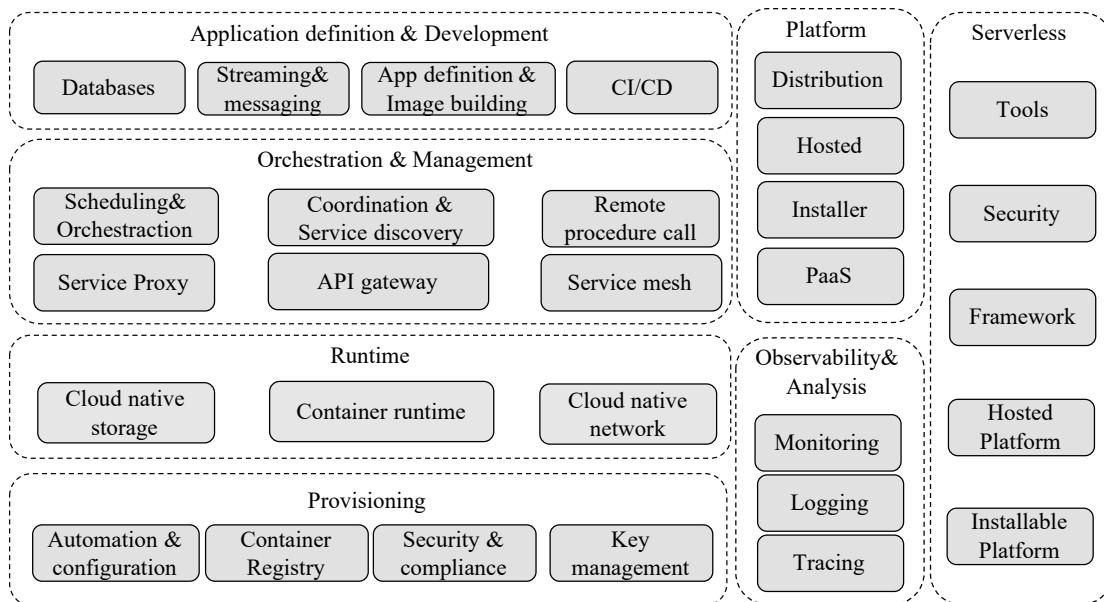


图 2-7: 云原生技术范畴

云原生技术囊括了 DevOps 的各环节。如图 2-7 所示,CNCF 定义了云原生的技术范畴,主要包括:

- 供应层 (Provisioning): 涉及云原生应用运行基础环境所涉及的自动化基础设施;
- 运行时 (Runtime): 指保障云原生应用程序正常运行所需的沙盒;
- 云应用编排与管理 (Orchestration and Management): 为云原生应用提供自动化编排和弹性伸缩能力,让云原生应用天然地具备可扩展性;

- 云应用定义与开发 (Application definition and Development): 开发、构建、部署和运行应用程序的工具;
- 可观测性与分析 (Observability and analysis): 全方位监控和分析云原生应用层的工具;
- 平台 (Platform): 主要指 Kubernetes, 将多类工具有机组合在一起解决庞大的工程问题;
- 无服务器 (Serverless): 提供函数级别的更细粒度部署的一种新的云原生计算模型。

随着云架构的不断普及,“未来的软件一定生长于云上”的理念被越来越多的人所接受。云提供了一种面向企业应用按需进行资源分配的模型,以一种全新的高效的方式来部署应用。云原生是一系列基于云技术体系和企业管理方法的集合,既包含了实现应用云原生化的方法论,也包含了落地实践的关键技术。云原生应用利用容器、服务网格、微服务、不可变基础设施和声明式 API 等代表性技术,来构建容错性好、易于管理和便于观察的松耦合系统,结合可靠的自动化手段可对系统做出频繁、可预测的重大变更,让应用随时处于待发布状态。Gartner 指出,到 2022 年全球公共云服务市场预计将增长至约 3546 亿美元,60% 的组织将使用外部服务提供商的云管理服务 [17]。企业纷纷开始云化转型,希望将传统应用迁移到云端。

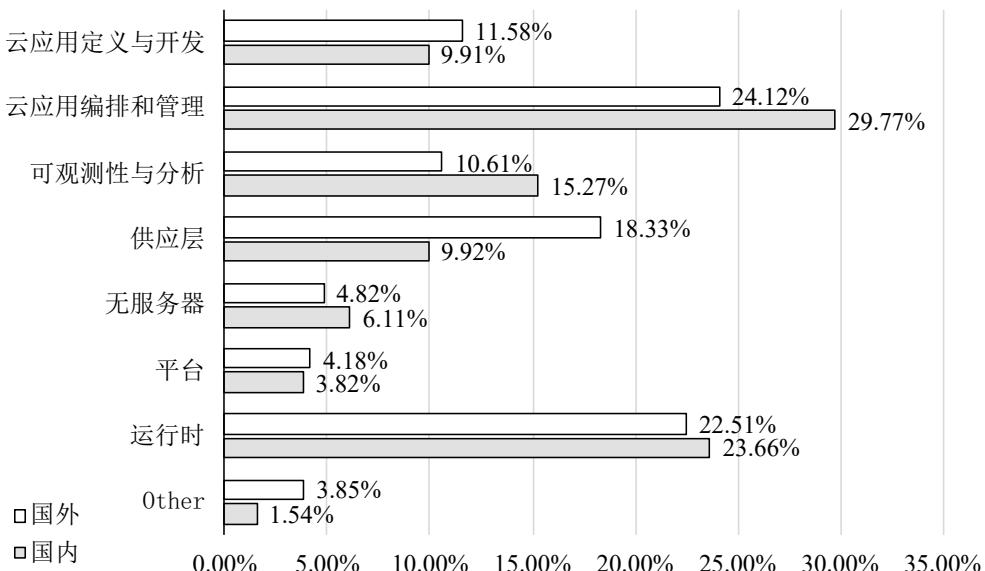


图 2-8: 国内外云原生技术现状

如图 2-8 所示,本文根据上述 7 类云原生技术范畴,对 2020-2021 年国内

外重点技术会议的共 442 场分享进行了系统化 Review, 其中包括国内云原生社区 MeeUp 城市站以及 Cloud Native+Open Source Virtual Summit China 共 131 场分享, 国外(欧洲、北美)KubeCon+CloudNativeCon Europe 以及 KubeCon+CloudNativeCon North America 共 311 篇场分享。从表中可知, 目前国内外研究重点关注于云应用编排以及运行时, 国内对云应用编排与管理的讨论要大于国外并且云应用定义与开发流程、可观测性与分析、平台与无服务器国内外相关分享在比例上差距并不是很大, 云原生技术体系已经成为当代软件工程技术发展的主流。

2.2.2 Kubernetes

Kubernetes(简称 k8s) 是 Google 开源的容器集群管理平台。在 Docker 容器化技术之上, k8s 为容器化的云原生应用提供部署运行、资源调度、服务发现、弹性伸缩等一系列基础功能, 提升了大规模容器集群管理的便捷性。自开源来, k8s 成为一种全新的基于容器技术的分数时架构解决方案, 在云原生领域具有举足轻重的地位。2019 年, 在我国 72% 的工程师已经在云原生生产环境中使用大规模使用 k8s 来进行容器^①。

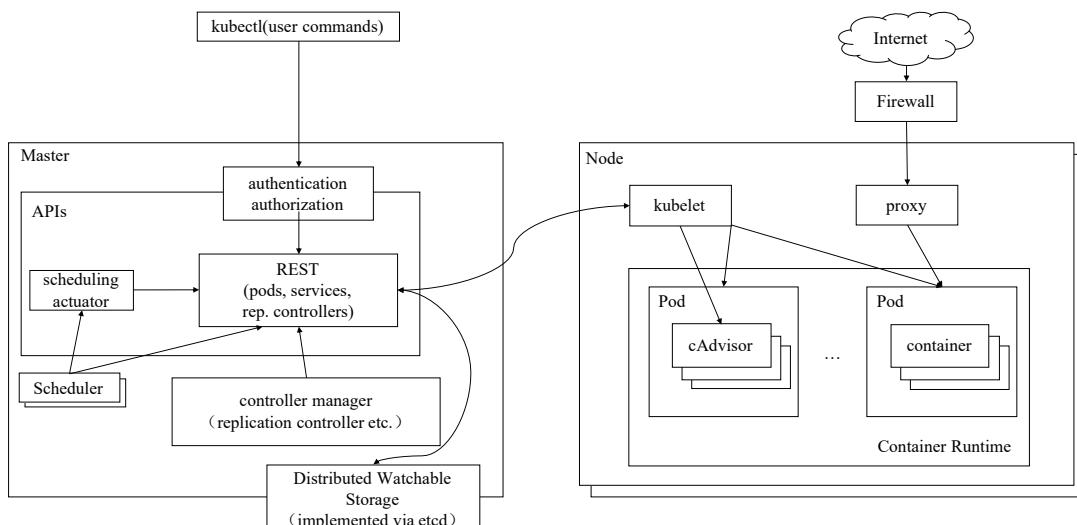


图 2-9: Kubernetes 架构图

k8s 是基于主从式的分布式存储系统, 由节点代理 (kubelet) 和 Master 组件组成。如图 2-9 所示, k8s 主要由以下核心组件构成:

^①CNCF Cloud Native Survey China 2019

- Etcd: 用于保存集群中一切网络配置和状态信息;
- ApiServer: 是资源配额控制的入口, 具备完备的集群安全机制并且提供用于集群管理的 REST API 接口;
- Controller Manager: 集群内部的所有资源的管理控制中心, 负责集群内的 Node、Pod 副本、Endpoint、Namespace、ServiceAccount、ResourceQuota 的管理, 实现自动扩展、自动滚动更新、故障检测等功能;
- Shaduler: 根据特定的调度算法将 Pod 调度到指定的工作节点;
- kubelet: 定时检查获取节点上 Pod、容器的期望状态, 并调用对应的容器平台接口达到这个状态;
- Container Runtime: 负责镜像管理以及 Pod 和容器的真正运作 [18];
- kube-proxy: 为 Service 提供集群内部的负载均衡和服务发现。

k8s 拥有独特的声明式 API 设计, 即声明式地告诉 k8s 所需资源的状态, 而不是告诉它如何做。在 k8s 中对象 (Object) 是持久化的实体, k8s 使用这些实体去表示整个集群的状态, 同时这些对象可以在 Yaml[19] 文件中作为一种声明式 API 类型来灵活的创建并配置。典型的, k8s 的对象主要有以下几种:

- Namespace: 能够隔离资源。k8s 集群可以拥有多个命名空间, 这些命名空间在逻辑上彼此隔离, 实现了对多用户的资源隔离;
- Pod: k8s 中最基本的操作单元, 包含一个或多个容器。k8s 为每个 Pod 分配一个唯一的 IP 地址, Pod 内部的多个容器共享该 IP 地址。并且每个 Pod 都能设定自己的计算资源, 即 CPU 和 Memory;
- Replication Controller(简称 RC): 确保任意时间 k8s 集群中运行指定数量的 Pod 副本;
- Deployment: 保证 Pod 的数量和健康, 绝大多数的功能与 RC 完全一样, 能够被当作全新一代的 RC;
- Service: 定义了一个 Pod 逻辑集合以及访问 Pod 的策略, 它提供一种桥梁会为访问者提供一个固定的 Pod 访问地址用于在访问时重定向到相应的后端;
- Label: 通过键值对的方式被附加到任何资源对象上, 用于配置资源;
- Secret: 不需要将敏感数据外露, 解决密码、token、密钥等敏感数据的配置问题;
- Role: 一组权限的集合, 给某个 NameSpace 中的资源进行鉴权;
- ConfigMap: 为了让镜像和配置文件解耦, 应用程序会从配置文件、命令

行参数或环境变量中读取配置信息,以便实现镜像的可移植性和可复用性;

- **Volume**: Pod 中能够被多个容器访问的持久化共享目录;
- **Persistent Volume(简称 PV)**: 类似于 Volume, k8s 提供的存储资源的抽象管理集群存储,其 API 内包含存储的细节实现;
- **Persistent Volume Claim(简称 PVC)**: 对存储资源的请求声明, PVC 不关心底层存储实现的细节,只消耗 PV 资源。

随着 k8s 生态的持续发展,上述常规的资源类型仅代表通用的对象,并不能适应多变的业务需求。为提升自身的扩展能力,k8s 提供用户资源定义 (Custom Resource Definition, 简称 CRD) 向 k8s API 中增加定制化的资源类型。用户的自定义资源 (Custom Resource, 简称 CR) 创建并注册到 k8s API 后,其与常规通用资源对象都是原生的、存在于 etcd 中的同等资源,可以采用 k8s 原生方式进行创建、查看。CRD 本质上用于声明用户自定义资源对象,开发人员还需要针对 CRD 提供关联的 Operator 对 CR 进行完整生命周期管控。

Operator 主要负责有状态应用 (即 CR) 及其组件的部署、更新、自动扩展、维护、数据备份等,保证其可用性。其作为 k8s 的一种扩展形式,基于 k8s 的资源和控制器 (controller) 概念构建,遵循 k8s 原则,功能类似于 Controller Manager,但同时又包含了特定领域知识。

2.3 其他相关技术

2.3.1 Helm

Helm 是一种基于 k8s 云计算平台的打包和部署复杂软件应用程序的技术 [20]。随着云原生及微服务架构的发展,开发人员倾向于将大的单体应用分解成多个可以独立开发、部署、运维的微服务部署于 k8s 之上。服务数量的增加为 k8s 编排带来了复杂性,Helm 则通过软件打包的方式极大的简化了 k8s 应用部署和管理的复杂性。

Helm 存在三个核心概念:

- **chart**: 即一系列包含创建 k8s 应用实例必要信息的文件;
- **config**: 包含应用发布的相关配置信息;
- **release**: chart 及其配置的运行实例。

如图 2-10 所示, Helm 将 k8s 资源 (如 Deployment、Service) 打包到 chart 中,

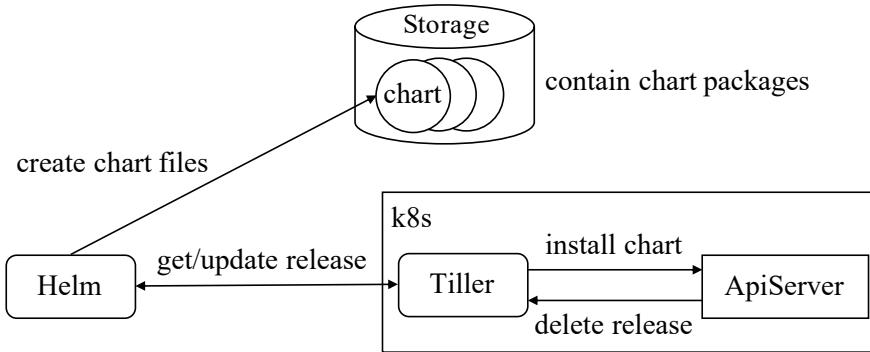


图 2-10: Helm 架构图

chart 将会被保存到仓库中。Tiller 是部署于 k8s 中的 Helm 的服务端, 负责接受 Helm 请求并于 ApiServer 进行交互, 根据 chart 生成并管理 release。开发者使用 Helm 可以简化应用配置及版本管理, 使得在 k8s 上部署、升级、回滚、卸载应用程序更加方便。

2.3.2 Istio

随着技术下沉, 传统由软件本身负责的负载均衡、路由等功能下沉到云原生基础设施。服务网格 (Service Mesh) 通过在不断变化的条件和拓扑结构面前强制执行所需的网络行为, 为网络连接的工作负载提供基于策略的网络服务 [21]。istio^①是 Service Mesh 架构的一种实现方式, 其是一个用于保证容器服务间连接、安全、控制和观测的网络代理组件, 具有负载均衡、服务间认证、监控等功能。

istio 分为两个逻辑部分 [22]: 数据平面 (Data Plane) 与控制平面 (Control Plane)。数据平面由代理程序 Envoy(通常被称为 Sidcar) 组成, 其受控制平面组件控制, 通常与业务容器捆绑, 来劫持业务应用容器的流量完成针对特性应用程序的控制与治理; 控制平面提供服务发现、配置和证书管理等功能, istio 对其进行了进一步细分:

- Mixer: 负责策略、访问控制和请求追踪;
- Pilot: 提供服务发现的功能;
- Citadel: 负责证书颁发;

^①istio github 地址

2.4 本章小结

本章 2.1 节首先介绍了区块链基本概念、架构、分类，其次重点介绍联盟链解决方案 Hyperledger Fabric 的网络节点、账本结构，最后介绍区块链即服务的诞生和通用架构；2.2 节介绍云原生的发展历程、技术范畴以及 Kubernetes；2.3 节介绍本文区块链云化架构涉及到的其他相关技术。

第三章 基于 Hyperledger Fabric 的区块链云化框架

本章将介绍去中心化应用在价值交付过程中区块链基础设施建设存在的问题,随后阐述区块链云化的原则,最后提出基于 Hyperledger Fabric 的区块链云化框架。

3.1 区块链基础设施现状

由于区块链智能合约的无法删除、修改、历史可追溯、去中心化、严格执行等特点,越来越多的研究人员想挖掘区块链的潜能,将区块链的能力应用到各种应用场景。McCorry 等人 [23] 将区块链应用在电子投票领域,实现了一个基于以太坊的去中心化互联网公开投票协议,公开投票网络是一种自助协议,每个选民都控制着自己投票的隐私,只有在所有人都参与的情况下才会被破坏,这是第一个实现的不依赖任何可信的权威机构来计数和保护选民隐私的去中心化应用。Chang 和 Chen[24] 在供应链领域进行了系统文献综述,表明传统的供应链活动涉及多个中介、信任和性能问题,利用区块链的潜力可以更好地扰乱供应链运作性能、分布式治理和过程自动化。Zhang 和 Wen[25] 提出了一种物联网电子商务模型,旨在重新设计传统电子商务模型中的许多元素,并借助区块链和智能合约技术在物联网上实现智能财产和付费数据的交易。Leka 等人 [26] 相信区块链技术将是下一个技术革命,同时表明区块链研究现阶段在物联网 [27]、医疗、教育、政府各个领域都有涉及。

目前,去中心化应用逐步从概念验证阶段转变为工程化、商业化阶段。由于区块链的复杂性给网络的构建以及智能合约的部署、运维工作带来了严重的时间成本。智能合约的部署采用的是纯手工或脚本部署的方式,交易方对生产部署环境有不同的要求,缺乏提高部署效率、缩短产品交付周期的有效工具与方法。同时,区块链领域频繁的出现安全问题可能会动摇人们对去中心化应用的信任。虽然智能合约安全事件占比并不高,但是每次发生安全事件造成的损

失能达到数百万乃至上亿美元。最著名的有发生在 2016 年 6 月的 The DAO[28] 安全漏洞,造成了 1.5 亿美元的损失,也造成了以太坊的永久性硬分叉。在去中心化应用价值交付过程中,存在对于区块链底层技术的易用性、部署效率、安全性等多方面的挑战。

BaaS 基于云的弹性可伸缩的能力屏蔽了底层区块链技术,为上层去中心化应用的开发运维人员提供便捷的构建区块链网络等功能。“Each coin has two sides”, BaaS 促进去中心化应用落地的同时也引入了许多问题。

首先,基础商业化应用工具并不完善。BaaS 平台发展还处于初级阶段,商业运行模式仍处在探索阶段。由于 BaaS 平台研发投入大,目前市场上存在如 AWS、IBM、阿里云区块链服务等多种 BaaS 平台解决方案,这些 BaaS 平台计费高昂^①并且与其他云服务捆绑销售。传统的企业业务或已存在稳定合作的云服务商,使用不同云服务商的合作伙伴就需要跨云部署,由于缺乏统一的顶层规划,各云厂商的 BaaS 导致不同应用底层异构形成技术、信息孤岛,存在不可复用的跨解决方案的资源和工具,给跨云部署带来了诸多问题。同时,BaaS 平台几乎都由互联网云服务巨头企业把控,最终行业马太效应明显 [3]。

其次,BaaS 平台利用云能力对区块链基础设施赋能乏力。作为一种广泛部署的技术,云计算是实施区块链技术的合适目标。然而,行业缺少既定的标准或最佳实践范例,各云厂商的 BaaS 解决方案基本都采取已有的开源解决方案,业务同质化严重,仅提供一种基于开源区块链平台的一件化自动部署管理方案,问题仍然在于如何使区块链和云兼容 [29]。区块链服务比常规云服务更复杂,有许多未解决的问题需要回答。除区块链网络自动部署外,在数据存储备份及可扩展性方面,传统自动化部署方案无法利用云的弹性伸缩能力且随着交易数量的增加,区块数据最终会占满磁盘空间,出现存储膨胀问题。BaaS 的数据可扩展性受到了的限制,需要结合云的能力寻找数据备份、升级及可扩展性的方法。

3.2 设计原则

BaaS 致力于一站式构建、管理、托管和运维区块链网络及其应用,本文提出的区块链云化框架重点关注于 Kubernetes 与 Hyperledger Fabric 的高度融合,需要满足以下原则:

- 易用性: 在 Kubernetes 上启动管理 Hyperledger Fabric 网络并部署链码需

^①阿里云区块链服务 BaaS 规格与定价

要专业的领域知识, Fabric 各组件的配置项繁多且与 Kubernetes 适配极易出错。区块链云化框架需屏蔽底层 Fabric 配置与 Kubernetes 逻辑, 帮助用户采用命令行配置的方式提供完整的 Fabric 各组件的全生命周期管理;

- 可迁移性: 区块链云化框架需要具备基础架构的云独立性, 本框架依托 Kubernetes, 可以方便的迁移到支持 Kubernetes 的任何云;
- 安全性: 区块链云化框架需要具备有效的认证、鉴权、准入机制来确保区块链系统的安全性; 具备可插拔的共识算法取保区块链的去中心化、可追溯等特点, 支持完善的用户密钥授权、保存、隔离处理以及提供可靠的故障恢复能力;
- 可扩展性: 区块链云化框架需按照模块化配置, 将 Kubernetes 底层的计算资源、存储资源、网络资源等供给 Fabric, Fabric 的 Ca、Peer、Orderer 基本组件的证书认证、共识、TLS 加密、存储等功能模块作为可配置项进行命令式配置; 该框架能够确保系统核心底层逻辑稳定运行的同时, 对外提供小而精的扩展边界, 实现系统的高内聚与低耦合;
- 透明性: 区块链云化框架需保证区块链上所有的交易记录是可追溯的、不可篡改的, 区块链交易过程以及获取交易产生的记录需要专业人员才能获取操作, 非专业人员难以理解, 框架需通过简单透明的方式获取交易记录;
- 可视运维: 当前区块链系统缺乏一套涵盖不同层面的标准方法来监控区块链及其智能合约的运行 [30], 区块链云化框架需有效利用云上监控方案为区块链系统提供 7*24 小时可视化资源监控能力。

3.3 基于 Hyperledger Fabric 的区块链云化框架

基于 3.2 所述原则, 如图 3-1 所示, 本文提出了基于 Hyperledger Fabric 的区块链云化框架。k8s API 是云原生容器管理系统的大脑, 它是一个复杂的 API, 具有多个层与各种资源 [31]。k8s operator 则是将领域知识集成到 Kubernetes API 编排过程中的最新方法 [32]。该框架使用 k8s operator 整合 2.1.3 节所展示的 BaaS 区块链层 Fabric 的联盟链能力以及计算资源层 k8s 的计算资源、存储资源等资源为上层的区块链服务层提供完备的去中心化应用开发的能力, 该框架命令式启停 Fabric 网络及其链码、隐式管理密码文件、按需配置调度 k8s 资源, 解决 Fabric 部署效率、安全性、弹性等运营难题, 节约开发人员时间成本, 使得其更加专注于去中心化应用的逻辑。

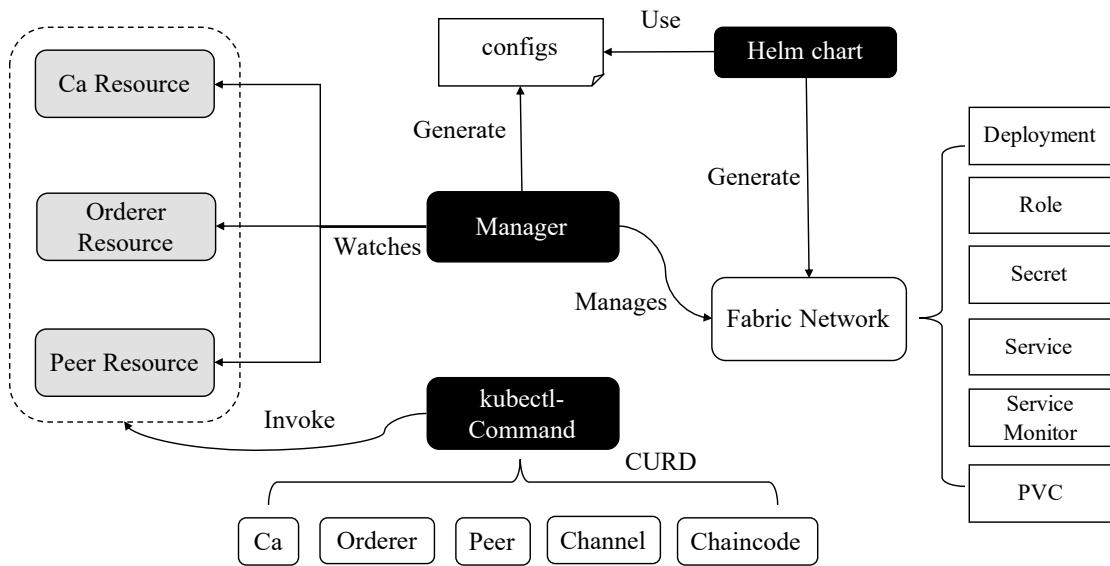


图 3-1: 基于 Hyperledger Fabric 的区块链云化框架

Operator 应该管理单一类型的应用程序, 遵循 UNIX 原则: 只做一件事, 并把它做好 [33]。本框架必须解决的主要任务是屏蔽 Fabric 及 k8s 底层细节, 简化 Fabric 网络的各节点的部署, 以及有效利用 k8s 代码化及云化的管理基础设施, 所以本框架需要分别对 Ca、Orderer、Peer 进行完整生命周期管理。本框架的整体工作流程将分为几个步骤。首先, CRD 作为本框架的输入, 通过自定义 kubectl 命令完成静态 CRD 相关属性的配置, 这些属性包含 Fabric 网络中 Ca、Orderer、Peer 各节点所具备的功能、性能、监控等各方面。其次, Manager 是本框架的处理单元, 其被设计成一个黑盒 [34], 用户无需关心 Manager 的内部逻辑设计。Manager 自动生成部署配置文件, 使用生成的配置文件并结合 Helm chart 可以轻松的将 Fabric 网络各节点部署进目标 k8s 集群。除了部署之外, Manager 将持续监控这些节点及其存储资源。根据持续监测结果并调用 k8s 及 Fabric API 将各节点调整到期望状态以维持 Fabric 网络的稳定。最后, Fabric 网络是本框架的输出, 通过自定义 kubectl 命令完成动态通道、链码的增删查改。除 Fabric 网络各节点基本 Deployment、Service 外, 本框架采用 istio 完成 TLS 通信的负载均衡; 采用原生 Role、Secret、PVC 的方式管理 Fabric 网络的权限、密码及存储。

3.3.1 Custom Resource Definition

CRD 通过扩展 k8s API 将具备领域知识资源类型注入进 k8s 集群中。k8s 提供了标准资源 ConfigMap, 也可用于使配置数据项供应用程序使用, 但这两种针

对不同的情况。ConfigMap 擅长为在集群上的 pod 中运行的程序提供配置, 应用程序通常希望从 pod 中读取此类配置, 例如文件或环境变量的值, 而不是从 Kubernetes API 中读取。CRs 由标准的 k8s 客户端创建和访问, 遵守 k8s 规范。通过自定义 controller 可以监控 CRs 运行, 这些代码可以反过来创建、更新或删除其他集群对象, 甚至集群外的任意资源。

本框架基于 Fabric 网络各节点自身功能及配置^{①②③}设计三种 Fabric 静态资源类型作为输入, 篇幅原因仅展示包括但不限于如表 3-1 所示的属性。额外的, 除上述针对不同网络节点的特殊属性外, 本框架需要为每个节点提供如副本数、镜像、Hosts、日志、ServiceMonitor 等基本属性维持上述网络节点的基本运行状态。

表 3-1: CRD 描述

CRD 名称	属性	描述
Ca Resource	CRLSizeLimit	可接受证书撤销列表 (Certificate Revocation List, 简称 CRL) 的大小限制
	TLS	服务器侦听 TLS 端口以及证书等信息
	CA	包含与证书颁发机构相关的信息
	Database	用作数据存储
	CFG	配置身份允许的错误密码尝试次数
	CSR	控制根 CA 证书的创建, 如根 CA 证书的过期时间配置
	Registry	部分控制 fabric-ca 服务器执行验证包含用户名和密码的注册和检索标识的属性名称、值的方式
Orderer Resource	BCCSP	用于选择要使用的加密库实现
	Genesis	初始区块相关配置
	BootstrapMethod	指定了获取引导块系统通道的方法
	ChannelParticipation	通道管理对系统链码的依赖
Peer Resource	Secret	包含 Orderer 的数字签名以及与 Ca 通信所需的基本信息
	Gossip	确保 Peer 间通过 Gossip 协议来达到所有账本的最终一致性
	LevelDB/CouchDB	Fabric 提供 levelDB 与 CouchDB 用以保存 Fabric 账本信息, 用以灵活适应 Peer 不同数据库之间的转换
	CouchDBExporter	采集 CouchDB 的监控数据
	ExternalChaincodeBuilder	提供外部链码构建的能力
	Secret	包含 Peer 的数字签名以及与 Ca 通信所需的基本信息

^①Ca Config^②Orderer Config^③Peer Config

3.3.2 Manager

CRs 本身仅为特定应用程序提供声明式 API 的数据项的集合, controller 负责对 CRs 的不同事件做出反馈, 管理 CRs 的完整生命周期。

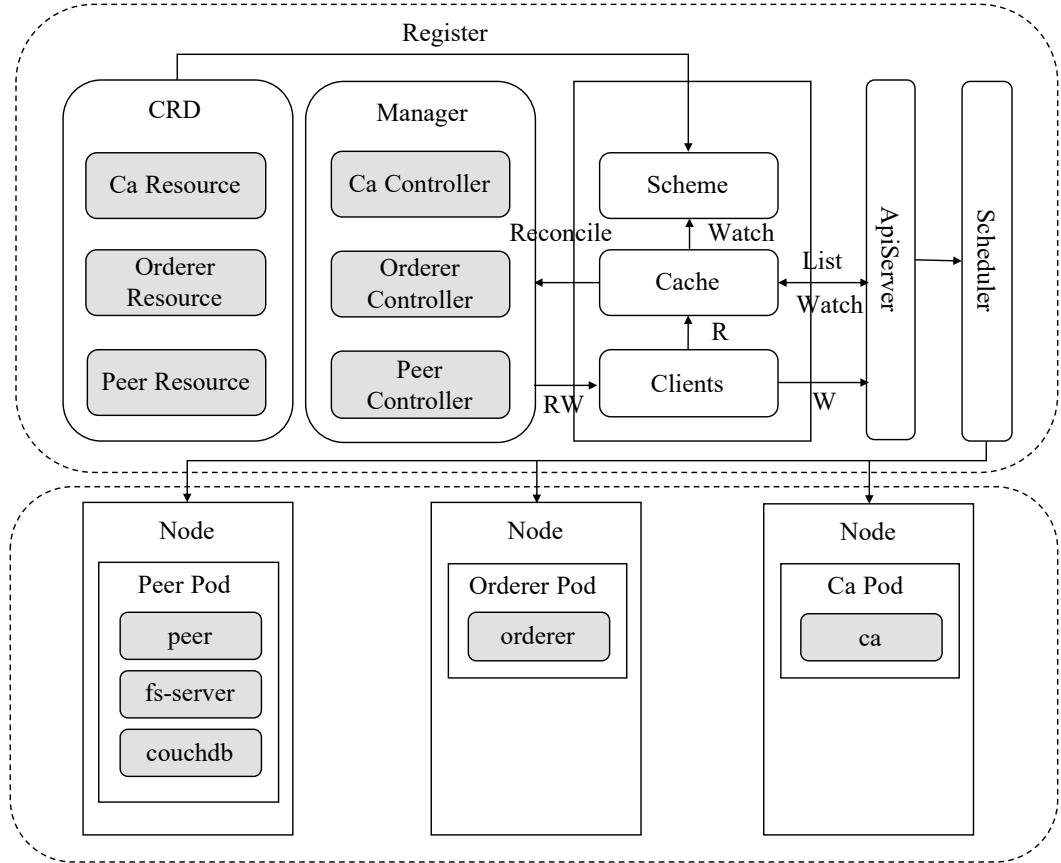


图 3-2: Manager 监听 CRs

如图 3-2 所展示的是 Manager 监听 CRs 的全过程。首先, 需要在 CRs 中指定 Fabric 网络各节点所期望的状态, CRD 会注册进入 Scheme, 其提供了 ApiServer 对应的集群中 GVK(Group Version Kind, k8s 集群资源定义方式) 与 CRs 资源类型的映射, 通过资源类型 controllers 就能获取 CRs 所定义的期望状态; 其次, cache 通过 List-Watch 机制与 ApiServer 进行通信用以同步监听 Fabric 网络各节点在 k8s 集群中的创建、删除、更新等操作, cache 可以获取 Fabric 网络各节点的实际状态; 最后, controller 循环监听期望状态与实际状态, 若期望状态与实际状态不一致, 则通过调用 clients 更新、缩放、扩展、备份等操作进行协调一致。

Helm 通过调用 k8s 的 ApiServer 逐个将 helm chart 中的 yaml 推送给 k8s, 当且仅能进行安装。所以 helm 的弊端是缺乏对资源的全生命期监控, 只有 CRD

才能持续的监听 k8s 资源对象的变化事件, 进行全生命周期的监控响应, 高可靠的完成部署交付。区块链云化框架设计了一套针对 Fabric 网络各节点 helm 的通用 CRD 与 controller, 直接将提前配置好的 helm 随 CRD 以及 controller 一直部署进 k8s。一旦创建新的 CR, controller 根据对应的资源对象更新 helm 的模板参数并重新部署入 k8s 集群。图 3-3 以 spec.size 为例展示了 controller 更新 helm 的流程。本框架不仅通过 helm 简化部署流程, 并且还能实现带全生命周期管理的 helm 效果。

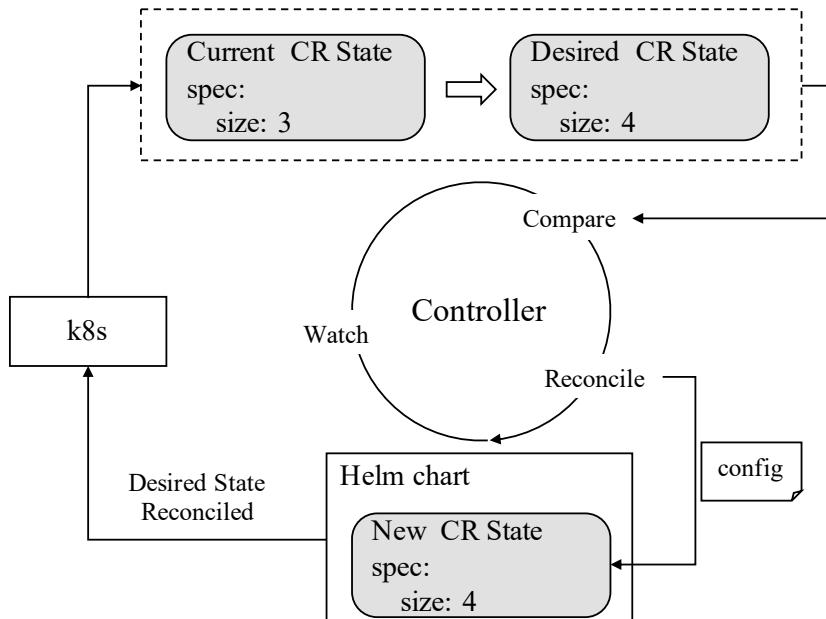


图 3-3: controller 循环监听

容器化在云原生应用程序中的拥有更高的部署效率 [35] 以及更高的可迁移性, 容器化将应用程序及其库、配置文件和其他依赖项封装在一起, 确保了环境兼容性, 从而使用户能够轻松地在集群之间移动和部署程序。区块链云化框架遵循标准化原则, 复用 Fabric 网络各节点镜像并利用 k8s 进行编排和管理底层的物理资源。这可以确保区块链系统基础架构的云独立性, 取消 BaaS 平台对云提供商的强依赖性, 提升 BaaS 平台的灵活性与通用性。

在安全性方面区块链云化框架复用 k8s 的原生安全保障体系, 主要涉及到两方面。一方面是框架对 k8s 集群操作的安全性, 区块链云化框架会生成很多清单文件向 k8s 集群中部署 Fabric 网络, 同时 k8s 集群需要向已部署的区块链云化框架授予在 Fabric 网络中生命周期内执行各种任务的权限。k8s 没有以用户身份进行身份验证, 如图 3-4-I 所示, 本框架采用基于角色的权限控制

(Role-Based Access Control, 简称 RBAC) 将对 k8s 资源操作的最小权限映射到框架中的 Manager 及 Fabric 网络节点, 在允许框架正常工作的同时, 应尽可能限制访问; 另一方面是 Fabric 网络运行环境均受 k8s 安全容器保护。在密码管理方面, 框架避免使用直接向节点镜像中注入环境变量的方式管理密码信息。如图 3-4-II 所示, 以注册 Fabric 网络用户为例, 本框架采用 Secret 配合 x509[36] 存储管理敏感数据, 这种方式不但能提高灵活性而且增加了密码的传输、存储、访问安全, 增强隐私保护。

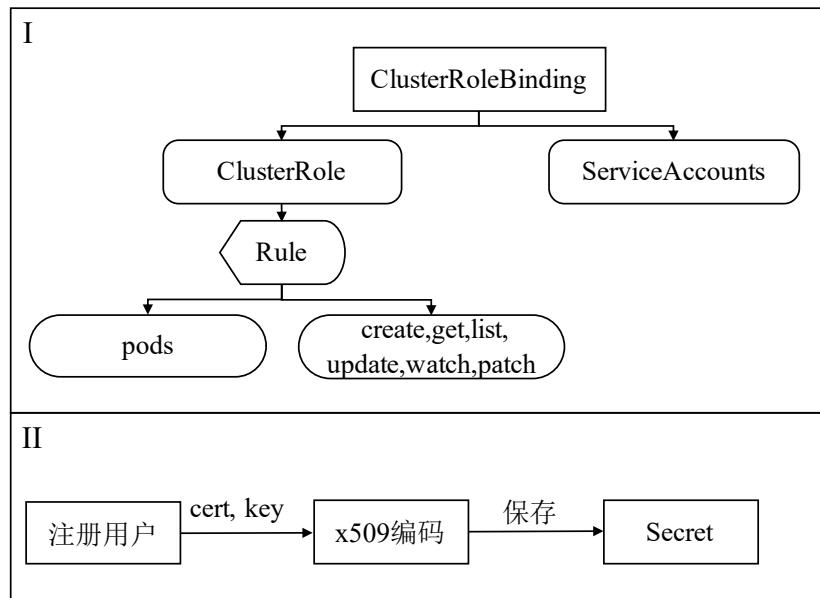


图 3-4: 区块链云化框架的安全性

除 3.3.1 所提到的利用 CRD 对 Fabric 网络配置进行模块化设计外, 本框架为每个运行中的 Fabric 网络节点选择配置的 PVC, 并为每个 PVC 中预留出一定的额外存储资源。相较于对所有持久存储的服务使用一个 PVC 而言, 虽然分配存储远超必要范围增加额外的存储资源冗余, 但用拥有更对的 PVC 能够保障每个网络节点拥有足够的存储空间, 以便在不缺乏存储资源的情况下正确运行节点。拥有更多的 PVC 增加了首次部署难度及过度调配的风险, 但多 PVC 能够灵活针对不同节点运行情况利用 k8s 进行有针对性的存储扩容, 增强框架对于存储的扩展性。尽管多 PVC 在管理方面存在一定的复杂性, 但在选择多 PVC 更加符合最佳实践, 并且效率更高 [33]。

Prometheus[37] 作为云原生领域的监控事实标准, 有着强大的功能和良好的生态。Grafana 可以与各种其他类似于 Prometheus 的数据源进行交互并进行可视化。本框架采用非介入式的云上监控方案 Prometheus 以及 Grafana 进行可视

运维,在 CRD 中预留 exporter、ServiceMonitor 等属性,对应的在 helm chart 中定制抓取周期的相关配置对 Fabric 网络中的 Ca、Orderer、Peer、CouchDB 等进行可视化监控。同时,Grafana 开源特性能够创建自定义插件,一定程度上能提升本框架的可视运维的灵活度。

3.3.3 Fabric 网络

Fabric 网络一个复杂的分布式系统,需要权衡速度、性能等条件对不同网络节点的部署状态进行合理设计。静态节点 Ca、Orderer、Peer 在首次启动网络时就需要部署在 k8s 集群中并以 Pod 形式运行。Pod 是 k8s 中可以创建和部署的最小单位。在 k8s 集群中,Pod 有两种运行状态:

- Pod 中运行单容器: 每个 Pod 一个容器是最常见的状态,在这种状态下,可以将 Pod 当作单容器进行封装,但 k8s 管理仍然是 Pod 而不是容器;
- Pod 中运行个容器: 当容器间需要紧密协作时可以在同一 Pod 中运行多容器。

Ca 是 Fabric 的证书授权中心,Orderer 负责交易的排序,这两个节点在配置上需要满足可插拔式设计。但在网络运行时,需要各自在一个 Pod 中运行即可,同时在一个 Pod 中运行可以有效的利用 k8s 的自动缩放功能进行弹性伸缩。

Peer 是 Fabric 中被使用最多的模块,是 Fabric 网络的基石,其负责区块链数据的存储以及运行链码。由于 Peer 节点需要频繁的账本存储单元如 CouchDB 进行交互,所以 peer 容器应当与 couchdb 容器存在于同一 Pod 中。在同一个 Pod 中,peer 容器与 couchdb 紧密协作,拥有相同的存活周期,更优的,相同 Pod 中的不同容器共享进程、IP 地址和数据卷,可以进行频繁的文件和数据交换。Peer 支持外部链码部署,即链码拥有自己独立的 Pod 运行环境与 Peer 解耦,能够纳入智能合约微服务化流程 [30] 进行快速响应与监控。

3.4 本章小结

本章首先详细介绍了目前区块链基础设施的现状,其中包括基础商业化应用工具不完善以及当前云能力对区块链基础设施的赋能乏力。其次,本章重点介绍了基于 Hyperledger Fabric 的区块链云化框架所应满足的 6 条设计原则;最后,介绍了云化框架的工作流程、输入单元 CRD、处理单元 Manager 以及输出单元 Fabric 网络节点运行状态。本章首先详细介绍了目前区块链基础设施的现

状,其中包括基础商业化应用工具不完善以及当前云能力对区块链基础设施的赋能乏力。其次,本章重点介绍了基于 Hyperledger Fabric 的区块链云化框架所应满足的 6 条设计原则;最后,介绍了云化框架的工作流程、输入单元 CRD、处理单元 Manager 以及输出单元 Fabric 网络各节点运行状态。

第四章 原型工具设计与实现

4.1 概述

云原生具有资源按需配置, 动态伸缩的特性。当前, BaaS 虽然能够基于云平台构建区块链系统, 但仅提供脚本化的方式部署区块链网络及智能合约, 仍未深入云基础设施平台的底层有效利用云的特性管理区块链平台, 这导致了 BaaS 平台对云特性的严重浪费。

因此, 一个支持区块链有效云化的工具十分重要。本章基于第三章提出基于 Hyperledger Fabric 的区块链云化框架提供配套的区块链云化原型工具。原型工具建立在基础 BaaS 平台的基本功能之上, 对外应提供 Fabric 网络及链码标准化的、便捷化启停构建方式; 对内应当将区块链领域知识注入进云平台 k8s 中, 有效利用 k8s 的安全性、扩展性的能力为 Fabric 网络赋能。

本章实现的建模支持工具使用流程如下图 4-1 所示。开发人员或架构师使用支持工具开始建模, 首先可以通过“模型存储与转化”模块的模型加载功能继续从上一次建模保存的结果开始继续建模; 使用“可视化建模”模块进行建模时, 开发人员或架构师可以随时使用“模型校验”模块进行对建模结果的验证, 确保建模的正确性和规范性, 工具也将对不符合规范的建模结果给出提示和警告, 帮助开发人员和架构师进行修改; 如果建模结果符合规范和约束, 工具将通过“模型存储与转化”模块保存建模结果到数据库, 或者以 XML、图片或 JSON 格式导出, 还可以根据建模结果生成框架项目文件。

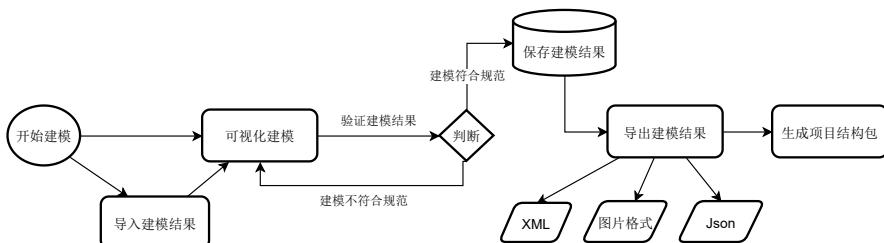


图 4-1: 建模支持工具业务流程

本章的剩余部分将对建模支持工具进行需求分析, 根据分析得到的结果对

工具进行设计与实现。首先对工具按功能模块进行划分，按模块介绍功能需求与非功能性需求；然后对工具进行总体设计；最后介绍各模块详细设计与实现。

Finalizer 在一般情况下，如果资源被删除之后，我们虽然能够被触发删除事件，但是这个时候从 Cache 里面无法读取任何被删除对象的信息，这样一来，导致很多垃圾清理工作因为信息不足无法进行，K8s 的 Finalizer 字段用于处理这种情况。在 K8s 中，只要对象 ObjectMeta 里面的 Finalizers 不为空，对该对象的 delete 操作就会转变为 update 操作，具体说就是 update deletionTimestamp 字段，其意义就是告诉 K8s 的 GC“在 deletionTimestamp 这个时刻之后，只要 Finalizers 为空，就立马删除掉该对象”。

所以一般的使用姿势就是在创建对象时把 Finalizers 设置好（任意 string），然后处理 DeletionTimestamp 不为空的 update 操作（实际是 delete），根据 Finalizers 的值执行完所有的 pre-delete hook（此时可以在 Cache 里面读取到被删除对象的任何信息）之后将 Finalizers 置为空即可。

数据卷扩容 <https://blog.fleeto.us/post/k8s-1.11-resizing-pvc/> <https://cloud.tencent.com/developer/article/1710700>
<https://www.cnblogs.com/fat-girl-spring/p/15176272.html>

最后部署 operator 的时候在 Kubernetes 集群中部署 « 操作员 » 时，使用了 Helm 工具 [17]。该工具允许使用一个命令部署应用程序（图 6）。我们可以为 Helm 开发特殊的清单（图表），以表示 Kubernetes 中的 « 操作员 » 应用程序。这些清单包含有关应用程序的所有必要信息，因此 Kubernetes 可以正确地部署它。例如，我们可以指定应该在 Kubernetes 集群上部署多少个应用程序副本

基础架构的云独立性，取消对云提供商的依赖

计算资源管理层支持公有云、专有云以及混合云，为区块链服务及上层应用提供所需要的云基础资源。

区块链底层平台是 BaaS 平台的核心，构建于云容器服务集群之上，支持超级账本、以太坊等不同区块链底层架构。区块链服务层依托底层区块链的支持，抽象封装一系列服务模块，简化开发工作，帮助企业快速部署区块链应用，降低区块链开发门槛。管理服务为用户提供基本管理功能，包括平台用户权限管理功能、服务使用计费管理功能、通知功能等。运维服务提供图形化的区块链管理运维服务能力，实时监控区块链网络运行数据，帮助运维人员及时发现并解决问题。

区块链云化降低开发门槛。区块链技术与其他技术不同之处在于它是融合

密码学、P2P 网络、分布式存储等多种技术的组合体，技术门槛高致使其开发成本高。而区块链云化产品 BaaS 平台将区块链技术封装在底层，使功能模块化，开发人员直接调用封装后的 API 接口即可完成一键部署，降低中小企业用区块链技术的门槛，从而推动区块链应用的落地。

区块链云化实现个性化定制。区块链云化产品 BaaS 平台可依托云服务商强大的业务能力，在提供标准服务基础上再根据开发者业务需求提供不同的配置，扩展开发者自定义的功能，满足其个性化需求，提高灵活性。同时通过 BaaS 平台可以沉淀出一层标准的区块链应用解决方案模板，为用户快速匹配建链场景。

市场前景广阔，发展迅速。区块链技术的不断成熟加速了区块链行业应用落地，不断扩大 BaaS 市场规模，对全球云计算的服务市场促进作用明显。特别是云服务开放性和资源可扩展性使其成为区块链应用落地的最佳载体，区块链与云计算结合愈发紧密。云链协同在加快区块链产业发展的同时，也成为云计算产业发展的关键性新动能。

负责实现云资源的管理调度，该模块会调用云资源管理适配模块的统一接口，所以底层不同云平台接口的差异性对该模块是透明的。该模块的主要功能有创建及删除虚拟机（Docker 容器）和网络资源、进行初始化配置、对已有资源进行扩容或缩容等操作。

对区块链节点的跨云部署支持，需要由该模块来实现对不同公有云、私有云的虚拟机、Docker 容器等资源调度 API 的封装，屏蔽各种云平台 API 的差异性，对上层调用模块提供统一的资源管理接口。

数据存储数据存储的核心任务是把数据账本高效地读写到持久化介质中。JD Chain 把数据账本模型映射为“键值”结构，为数据的存储提供更好的伸缩性。另外，还定义了标准的持久化服务 SPI，能够适配不同的数据库引擎，更好地复用企业现有的 IT 基础设施，满足企业的多样化需求。

华为云区块链服务基于可信、开放、服务全球的华为云上运行，华为云产品和服务具有华为独有的新技术，以降低成本、弹性灵活、电信级安全、高效自助管理等优势惠及用户，BCS 可以和华为云技术产品和行业解决方案无缝对接，帮助企业在安全、高效、不可篡改等基础上轻松跨入云时代，快速部署新解决方案和应用。

本文还实现了基于 Hyperledger Fabric 的云化工具

为解决上述挑战，本文 Hyperledger Fabric Operator：云原生时代下 Fabric 管

理工具利用 Kubernetes, 72% 的工程师在云原生生产中使用 Kubernetes[3], 可以方便的迁移到支持 Kubernetes 的任何云更简单、更原生的: 使用 kubectl 命令管理 Fabric, 显示于 k8s-dashboard; 复用 Kubernetes, API 公共功能如 CRUD、watch、内置认证管理 Fabric: 声明式自动化配置静态组件, 如: CA、HF 网络组件. 命令式创建动态通道、链码

由 linux 基金会牵头, 包括 IBM 等 30 家初始企业成员共同成立的 Hyper-ledger Fabric 项目成为流行的面向联盟链场景开源框架之一。该项目定位是面向企业的分布式账本平台, 引入权限管理, 设计上支持可插拔、可扩展, 自开源依赖广受欢迎, github 上已经有 12.7k star。然而云原生时代下, Fabric 缺乏一个成熟的、一站式的解决方案来解决在云计算平台上构建区块链联盟链(私有链)。

区块链作为一种点对点的信息和价值交换的“桥梁”, 通过定义一套标准的操作接口和数据结构, 能够提升多方业务对接的效率, 降低应用落地的复杂度。遵循标准化原则, 要求在系统设计时数据模型及操作模型独立于系统实现, 让数据“系于链却独于链”, 可在链下被独立地验证和运用, 更好地支持企业进行数据治理, 提升区块链系统的灵活性和通用性

4.2 需求分析

4.2.1 工具总体功能

DDDD (Draw for Domain-Driven Design) 建模支持工具总体功能规划如图 4-2 所示, 主要包括“可视化建模”模块、“模型校验”模块以及“模型存储与转化”模块, 各模块详细功能需求描述如下:

- “可视化建模”模块: 支持开发人员从工具栏中选择需要使用的战术模式, 并灵活地将模式图形拖拽到建模绘图面板上进行绘制, 还支持根据建模需要对不同的模式进行连线, 表达模式间的关系, 完成可视化建模过程。
- “模型校验”模块: 在建模过程中或建模完成时, 可以依据定义的战术建模元模型中的约束对建模的结果进行验证, 并给出修改意见。
- “模型存储与转化”模块: 建模完成后, 将建模结果进行存储或导出到本地, 也可以从数据库或本地读取模型文件, 根据建模结果, 生成符合领域驱动设计规范的框架项目代码包。

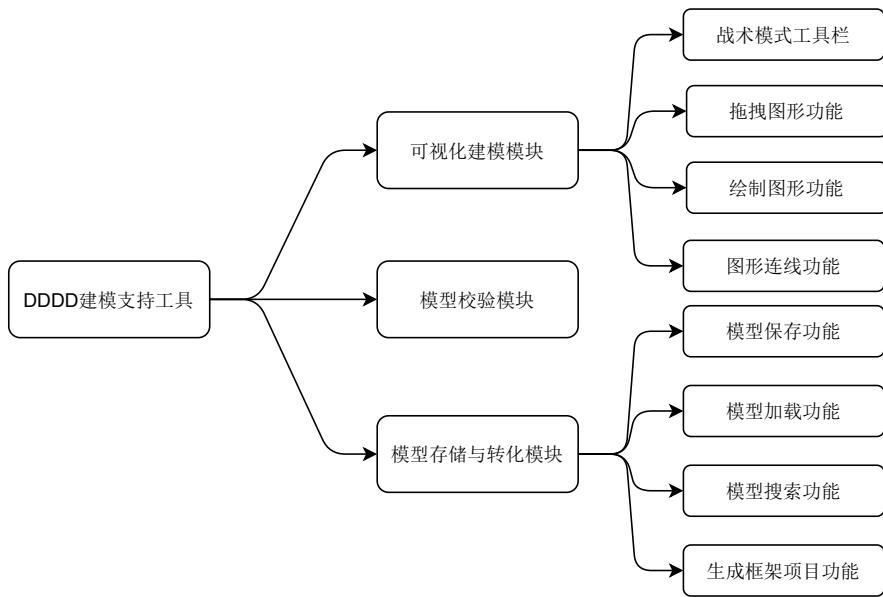


图 4-2: 工具总体功能

4.2.2 可视化建模模块需求分析

功能性需求

“可视化建模”模块是本建模支持工具的核心功能模块，如图 4-3 所示，开发人员或架构师可以通过“可视化建模”模块“选择战术建模模式”、“拖拽图形”、“绘制图形”以及进行“图形间连线”。

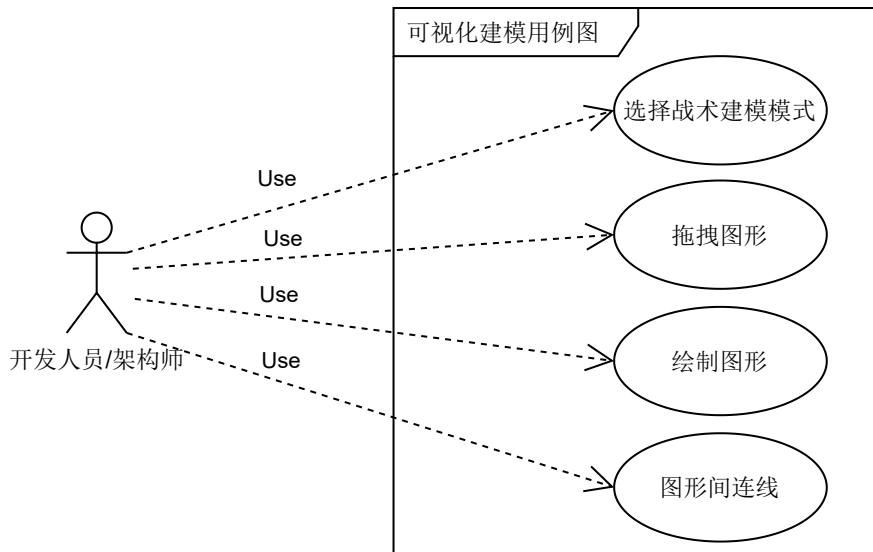


图 4-3: 可视化建模模块用例图

表 4-1 展示了“可视化建模”模块的用例描述，“可视化建模”为开发者提供灵

活方便的可拖拽式建模方式，可以自由选择需要使用的战术模式，拖拽到绘制面板中进行绘制，对绘制的模型对象可以双击修改文字信息，并将当前模型对象与其他对象进行连线，以表示它们之间的关系，在绘制面板上方还有以按钮形式提供的绘图操作菜单。

表 4-1: 可视化建模用例表

ID	UC1
名称	可视化建模用例
描述	开发者使用工具可以拖拽式地进行建模，选择战术模式相应图形化模型，拖拽到建模绘制面板中，并修改模型对象信息、对象间关系，达到灵活可视化建模。
触发条件	从主页点击“开始建模”按钮
前置条件	用户浏览器支持 JavaScript
后置条件	无
正常流程	(1) 用户点击“开始建模”按钮； (2) 用户点击展开战术模式列表，拖拽需要的模式进入绘图面板； (3) 松开鼠标，模型对象绘制完成，双击模型对象文字块，修改文字信息； (4) 当鼠标悬停在模型对象上且边框变绿，按住鼠标左键拖动绘制箭头，连接到其他模型对象上； (5) 点击绘图板上方按钮可对模型进行验证、组合、分解、删除、撤销以及缩放画布操作。
异常流程	无

非功能性需求

“可视化建模”模块的非功能性需求主要包括易用性、可靠性和可扩展性，具体需求如下：

- 易用性：可视化的建模过程易于学习和使用，用户仅需进行简单的拖拉拽操作即可完成图形化建模，速度快，图形变换灵活简单，建模结果清晰明确，符合用户建模的期望。
- 可靠性：在“可视化建模”过程中，绘图逻辑大多都在前端完成，即使网络连接断开也不影响建模过程，保障了建模过程的连贯性。
- 可扩展性：“可视化建模”的战术模式工具栏可以根据需求添加和删除战术模式，适应未来战术建模理论的更新和发展。

4.2.3 模型校验模块需求分析

功能性需求

“模型校验”模块用例图如图 4-4 所示，该模块帮助开发人员或架构师根据元模型和约束规则对建模结果进行验证和修改。

表 4-2 展示了“模型校验”功能模块的用例描述，“模型校验”根据本文定义的战术建模元模型约束规则和图形绘制规则对图形化的建模结果进行验证，对不符合规范和约束的模型进行警告和给出修改提示，对验证通过的模型进行命名并保存到数据库中。

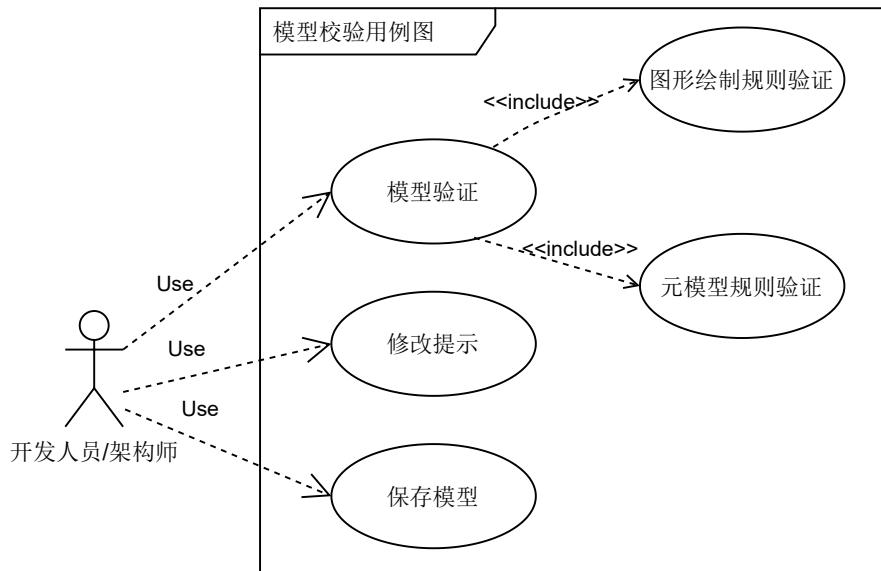


图 4-4: 模型校验模块用例图

表 4-2: 模型校验用例表

ID	UC2
名称	模型校验
描述	根据本文提出的战术建模元模型及约束规则对建模结果进行验证，并给出修改意见
触发条件	绘图界面点击“验证”按钮
前置条件	建模进行中或已经完成
后置条件	无
正常流程	(1) 当建模面板不为空时，开发者点击“验证”按钮； (2) 前端界面弹出验证通过或验证失败及修改提示信息； (3) 如果验证通过，将建模结果命名并自动保存在数据库中。
异常流程	保存建模结果时未填写名称，提示填写

非功能性需求

“模型校验”模块的非功能性需求主要包括易用性、健壮性和可靠性，具体

需求如下：

- 易用性：模型的验证结果以弹窗形式展示，指出建模不规范的地方并给出修改建议，让用户明确需要修改哪些建模结果。
- 健壮性：验证操作能够应对各种异常情况并作出响应。对于错误输入、非法字符和误操作等能够规避和纠正，有效防止建模过程中工具的崩溃。
- 可靠性：支持每次验证操作都将建模结果自动保存到数据库，保障了建模结果不丢失和及时更新持久化内容。

4.2.4 模型存储与转化模块需求分析

“模型存储与转化”模块用例图如图 4-5 所示，开发人员或架构师通过“模型存储与转化”模块对模型进行导入和导出，对数据库中存储的模型进行管理，即搜索和查看模型，还能根据模型生成框架项目。

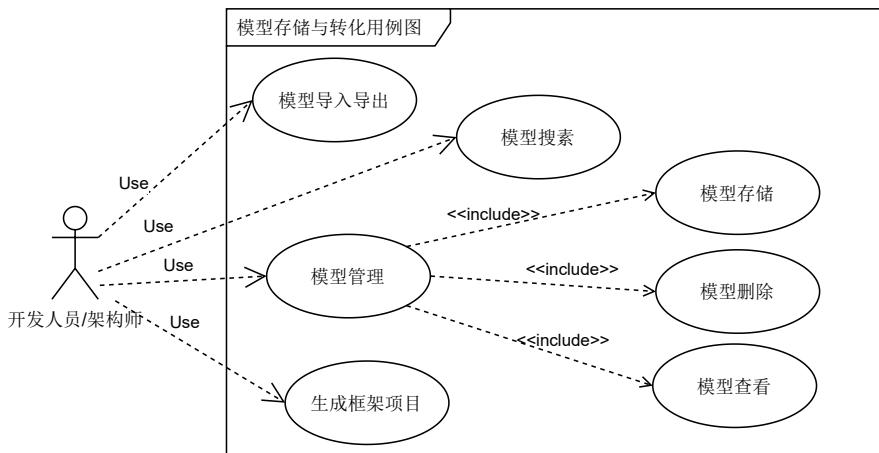


图 4-5: 模型存储与转化模块用例图

表 4-3 展示了“模型存储与转化”模块的用例描述，“模型管理”可以将通过验证的建模结果保存到数据库中，“模型导入导出”可以将模型以 XML、图片等文件格式导出到本地，也可以导入本地模型文件或搜索数据库中现有模型进行展示和修改，通过“生成框架项目”可以得到符合领域驱动设计规范的框架项目代码包。

表 4-3: 模型存储与转化用例表

ID	UC3
名称	模型存储与转化

描述	将图形化建模结果存储到数据库，或直接以 XML、图形等格式导出到本地，也可以从本地导入 XML 模型文件，根据完整的建模结果，自动生成符合领域驱动设计结构规范的框架项目代码
触发条件	绘图界面点击“导入”、“导出”、“验证”、“保存”或“生成项目”按钮
前置条件	建模进行中或已经完成
后置条件	无
正常流程	(1) 进入绘图界面时，可以选择导入本地模型文件或搜索数据库中模型文件进行继续建模； (2) 模型验证通过后，选择导出建模结果到本地或存储到数据库中； (3) 点击绘图面板上方“生成项目”按钮可以生成框架项目文件。
异常流程	保存建模结果时未填写名称，提示填写

非功能性需求

“模型存储与转化”模块的非功能性需求主要包括可扩展性、可靠性和性能，具体需求如下：

- 可扩展性：工具采用前后端分离的结构，可以对前端或后端组件进行替换或扩展。满足存储持久化设施多样性，为后续存储更多模型提供保障。
- 可靠性：工具支持本地文件系统的导入导出，不强依赖远程数据库，满足各种网络条件下的建模需求。
- 性能：对于数据存储和检索使用了 ElasticSearch，基于倒排索引能达到高效的全文搜索 [?]。满足搜索模型的即时性，让工具使用速度提升。

4.3 工具设计与实现

4.3.1 总体设计

本工具的架构根据典型的四层架构 [?] 进行设计和划分，整体架构图如图 4-6 所示。第一层表现层负责向用户展示业务流程和数据，包含可视化建模与验证修改模型等功能；第二层服务层，包含对模型的存储以及读取、对模型的二次校验和框架项目文件生成功能；第三层持久层主要保存了业务模型对象的数据，即对实体、值对象和领域服务等对象数据进行保存；第四层数据层主要负责建模结果的保存和读取。

- 表现层：该层为用户提供访问入口，通过该层用户可以直观地使用“可视化建模”、“模型校验”等工具核心功能。工具前端采用 Vue.js 框架结合

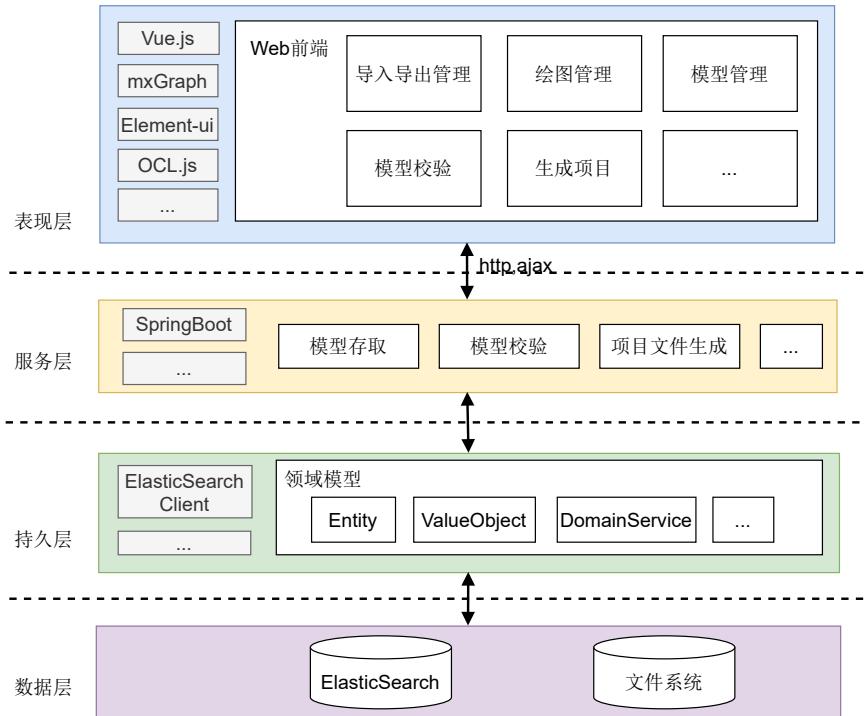


图 4-6: 支持工具整体架构图

ElementUI 组件进行开发，使整个页面整洁美观，交互良好；对于图形的拖拽和绘制，采用开源框架 mxGraph 作为底层支持，mxGraph 拥有优秀的交互和卓越的性能，已经在许多商业软件中被使用，能有效支持灵活的图形编辑；OCL.js 主要用来支持模型规范和标准的校验，充当对象约束语言和前端 JavaScript 之间的媒介，可以直接使用 JavaScript 完成 OCL 的功能。

- 服务层：该层主要提供后端接口服务和对建模逻辑的二次验证。后端整体采用 Spring Boot 框架进行开发，封装抽象了核心的业务逻辑模型（实体、值对象等），提供了模型存取与框架项目文件生成的业务逻辑。
- 持久层：该层主要充当数据层与服务层间交互的媒介，将业务逻辑模型对象转化为底层数据。持久化存储采用 ElasticSearch Client 与数据层进行通信，通过 RESTful 接口进行写入和读取。
- 数据层：该层是底层的存储数据依赖，模型底层存储格式为 XML 文件，文件篇幅较长，但对读写并发要求不高。采用两种数据基础设施进行实现，采用 ElasticSearch 支持大篇幅文档型数据存储，提高模型搜索的检索效率，也可根据用户需要直接存储到本地文件系统。

建模支持工具 DDDD 是一个前后端分离的 Web 应用。前端采用 mxGraph

结合 Vue.js 进行开发实现，后端采用 Spring Boot 框架，以 ElasticSearch 作为持久化机制进行实现，前后端采用 RESTful 风格进行交互 [?]。

由于建模过程的可视化描述和模型校验反馈是工具的核心内容，负责与用户交互和展示模型的 Web 前端的架构也有一定复杂度，故采用如图 4-7 所描述的整洁架构 [?] 对前端进行划分。整洁架构是一种与数据库、客户端接口以及框架都无关的架构，可以用来构建复杂的前端项目，将业务逻辑抽离，但又不依赖后端，保证项目的可扩展性。本工具使用整洁架构将前端划分为三层，包括表现层、领域层和数据层。表现层包含绘图、校验、导入导出以及菜单等与用户交互的模块；领域层包含展示模型和校验反馈的底层业务逻辑，如拖拽绘制、校验模型、导入导出模型、画布管理、路由管理以及生成项目等功能；数据层包含调用后端 API 连接的远程数据源和本地数据源，负责存储和获取模型。

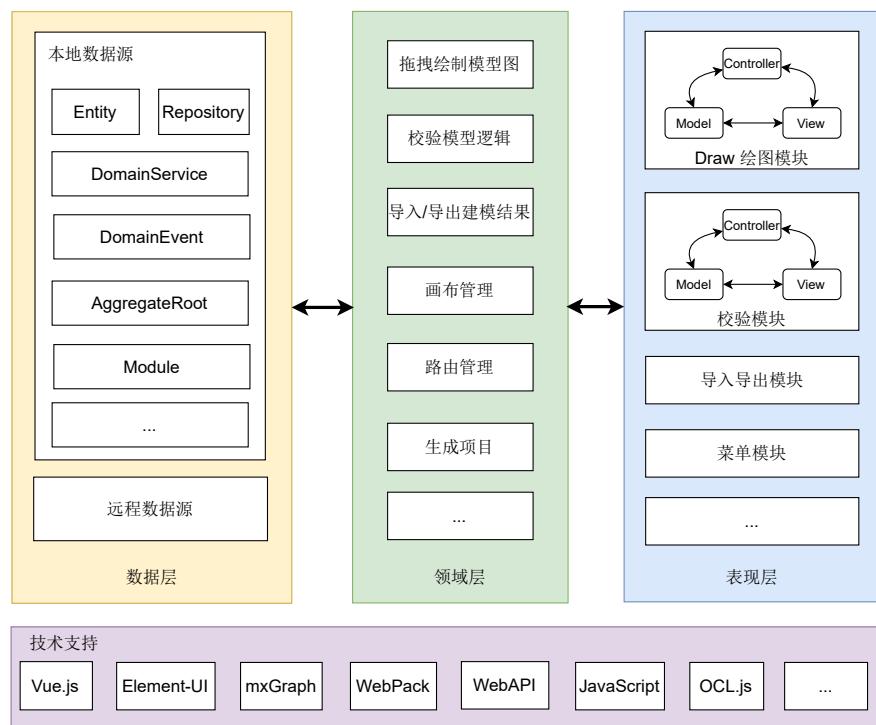


图 4-7: 支持工具前端整洁架构图

4.3.2 可视化建模模块详细设计与实现

本小节将介绍“可视化建模”模块的详细设计与实现。详细设计部分，通过类图展现“可视化建模”模块的静态详细设计，通过顺序图展现“可视化建模”模

块的动态业务流程详细设计。实现部分，通过部分核心功能的实现代码来进行展现。“可视化建模”模块具体的设计如下：

可视化建模类图

“可视化建模”模块类图如图 4-8 所示。本工具提供对建模过程可视化的支持，主要由 **Canvas** 类对 **mxGraph** 中封装好的画图工具类接口进行调用，并确保工具类是单例的，**Canvas** 类还负责工具运行时对画布的初始化，保证画图时的线条、箭头和颜色符合预期，最后，还包含绘制新图形时的判断逻辑。**Toolbar** 类主要提供了建模时对模式的选择支持，可以直接从中点击拖动实现选择战术模式。**Menu** 类包含了对画布操作的一系列接口。**Container** 类负责统一管理 **Toolbar**、**Canvas** 和 **Menu** 类，并收集反馈信息与用户进行交互。

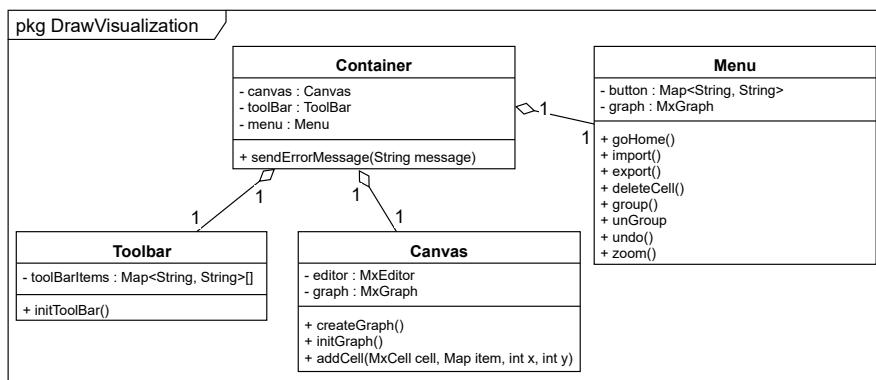


图 4-8: 可视化建模模块类图

可视化建模顺序图

“可视化建模”模块的示例过程如图 4-9 所示。首先开发人员或架构师通过使用 **Container** 类初始化画布、工具栏等可视化建模必要元素，通过 **Toolbar** 类实现拖拽图形化模式对象，并在合适的位置由 **Canvas** 类绘制，在建模过程中，通过菜单可以实现撤销、重做、组合、删除、缩放画布等一系列操作，辅助完成建模过程，在建模过程中画布会实时反馈信息给用户。

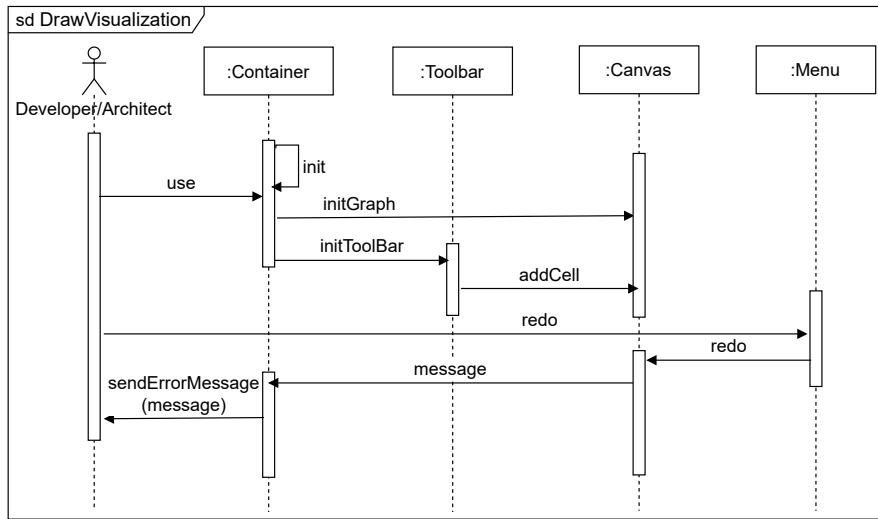


图 4-9: 可视化建模模块顺序图

可视化建模实现代码

如图 4-10 所示为“可视化建模”核心功能工具栏的初始化代码。代码描述了获取图形对象数组、拖动图形、绘制图形的逻辑。

4.3.3 模型校验模块详细设计与实现

本小节将介绍“模型校验”模块的详细设计与实现。详细设计部分，通过类图展现“模型校验”模块中各战术模式的静态详细设计；实现部分，通过校验算法展现“模型校验”模块的验证逻辑的设计，通过部分实现代码展示具体实现。“模型校验”模块具体的设计如下：

```
//核心方法，让工具箱项目可以被拖拽，放手时进行绘图
initToolbar() {
    //获取dom数组
    const domArray = this.$refs.toolbarItems;
    if (!(domArray instanceof Array) || domArray.length <= 0) {
        return;
    }

    domArray.forEach((dom, domIndex) => {
        //获取dom数组中每个item
        const toolbarItem = this.toolbarItems[domIndex];
        const {width, height} = toolbarItem;
        //释放对象图形时调用addCell函数
        const dropHandler = (graph, evt, cell, x, y) => {
            this.addCell(graph, toolbarItem, x, y);
        }
        //创建拖动时的预览图像
        const createDragPreview = () => {
            const elt = document.createElement('div');
            elt.style.border = '2px dotted black';
            elt.style.width = `${width}px`;
            elt.style.height = `${height}px`;
            return elt;
        }
        // 获取绘制cell位置的信息，如果释放位置有cell，判断该cell是否允许子元素加入
        const getDropTarget = (graph, x, y) => {
            const cell = graph.getCellAt(x, y);
            return this.R.propOr(null, 'dropable', cell) ? cell : null;
        }
        //使用mxUtils类使工具栏选项可以被拖动
        mxUtils.makeDraggable(dom, this.graph, dropHandler,
            createDragPreview(), 0, 0, false, true,
            true, getDropTarget);
    })
}
```

图 4-10: 初始化 ToolBar 代码

模型校验类图

“模型校验”模块类图如图 4-11 所示。本工具提供对建模结果的校验，战术模式统一继承自 Pattern 类，Pattern 类描述了模式的名称、类型、连接输入以及连接输出，并在 validation 方法中调用 OCLEngine 对象实例完成对象约束语言的验证。PatternData 用来记录所有 Pattern 子类实例的数据细节，辅助校验过程。

模型校验算法

“模型校验”功能的具体实现逻辑如算法 4.1 所示。该算法首先通过遍历所有图形化对象节点，建立两种连接关系双向映射，第一种描述源对象到目标对象映射关系，第二种描述目标对象到源对象映射关系；然后进行第二次遍

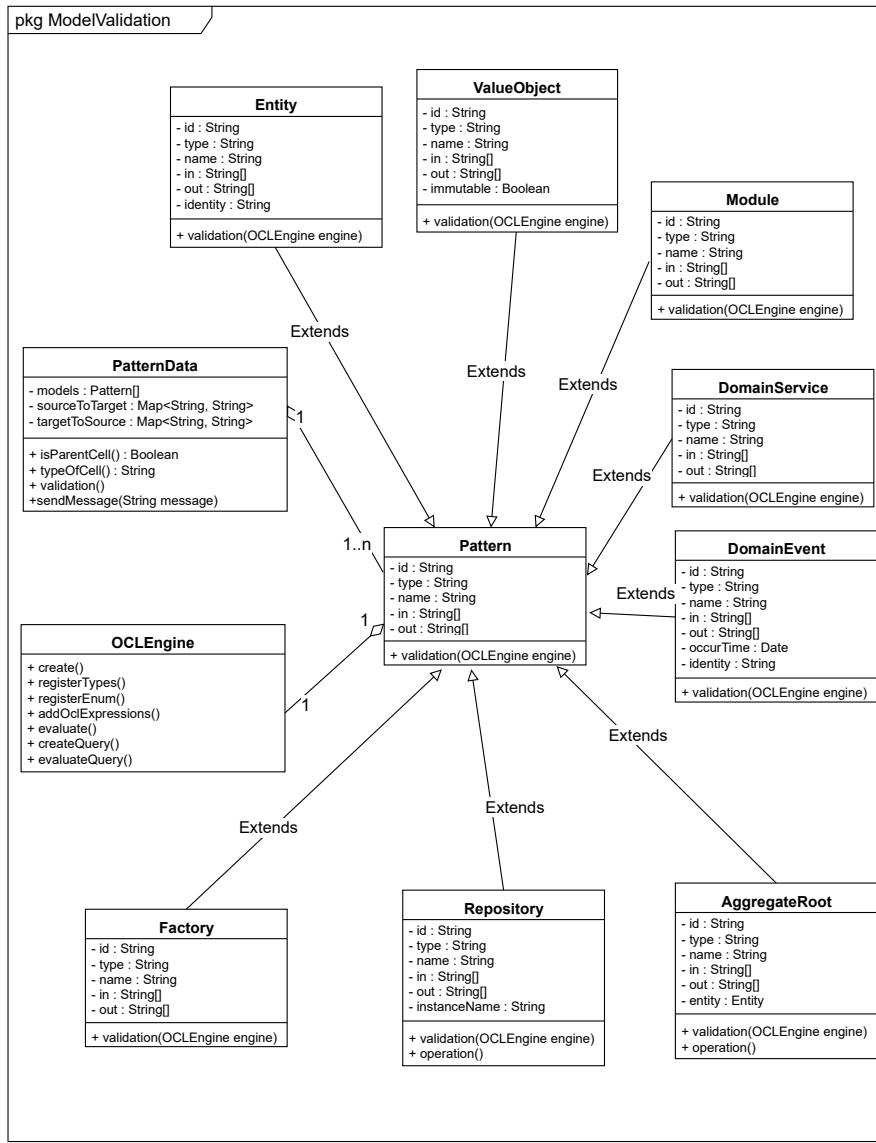


图 4-11: 模型校验模块类图

历，首先判断该对象节点是否为顶级节点，只有顶级节点才作为战术模式元素，如果为顶级节点，则调用 PatternData 类中的 validation 函数进行模式属性以及 OCL 约束验证，验证通过将该对象以具体模式（实体、值对象等）添加到 patternData 的 models 数据结构中，若验证失败，将布尔型成功标识变量标记为否，并退出循环体。无论验证是否通过，都将非顶级节点的子节点跳过，节省验证时间；如果不是顶级节点，则继续执行循环体；最终返回验证结果及具体建模结果。

模型校验实现代码

如图 4-12 所示是“模型校验”功能部分实现代码。主要描述 validation 方法中

算法 4.1 模型校验算法

```
Input: mxCells:HTMLCollectionOf<Element>
Input: patterns:PatternData
Output: models:Map<String, Pattern>
Boolean success = true;
for mxCell in mxCells do
    patterns.setSourceToTarget(mxCell.getId, mxCell.getTarget);
    patterns.setTargetToSource(mxCell.getSource, mxCell.getId);
end for
for mxCell in mxCells do
    if patterns.isParentCell(mxCell) then
        if patterns.validation(mxCell) then
            newPattern = new Pattern(mxCell);
            patterns.models.add(mxCell.getId, newPattern);
        else
            success = false;
            break;
        end if
        skip loopStep;
    else
        continue loop;
    end if
end for
if success then
    sendSuccessMessage();
    return patterns.models;
else
    sendErrorMessage();
end if
```

对 DomainService 对象的验证过程，包括从图形化对象转化为抽象模式类的过程以及对象约束语言验证与属性验证。

4.3.4 模型存储与转化详细设计与实现

本小节将介绍“模型存储与转化”模块的详细设计与实现。详细设计部分，通过类图展现“模型存储与转化”模块的静态详细设计，通过顺序图展现“模型存储与转化”模块的动态业务流程详细设计。实现部分，通过部分核心功能实现代码进行展现。“模型存储与转化”模块具体的设计如下：

模型存储与转化类图

“模型存储与转化”模块的类图如图 4-13 所示。本工具提供对建模结果的存储与转化，由 File 类负责收集和组织图形化模式的具体信息，可以直接在浏览器客户端导出建模结果的 XML 格式文件和图像格式文件，也可以导入模型的 XML 文件，直接将模型绘制在画布上继续进行编辑；当与后端进行通信时，会将模型信息抽象为 Model 类，由 ModelRepository 负责模型的存储、修改、删除以及检索，使用 ElasticSearch 提供的客户端进行持久化支持，可以极大提升

```

if(typeOfCell.replace(/\s*/g,'') == 'DomainService'){

    //从图形节点获取对象的各种参数
    let id = mxCells[i].getAttribute('id');
    let name = mxCells[i+1].getAttribute('value');
    let inId = mxCells[i+2].getAttribute('id');
    let outId = mxCells[i+3].getAttribute('id');

    //创建新 DomainService对象时，内部会进行OCL验证
    let domainService = new DomainService(typeOfCell, id, name);

    //获取 入参对应的对象Id 和 出参对应的对象Id
    let inObjectId = this.targetToSource.get(inId);
    let outObjectId = this.sourceToTarget.get(outId);
    if(inObjectId != null && outObjectId != null) {
        domainService.in.push(inObjectId);
        domainService.out.push(outObjectId);
        this.models.set(id, domainService);
    }else if(inObjectId == null) {
        success = false;
        this.sendErrorMessage("领域服务没有连接输入对象!");
    }else {
        success = false;
        this.sendErrorMessage("领域服务没有连接输出对象!");
    }
}
}

```

图 4-12: 模型校验部分实现代码

检索模型文件的效率。

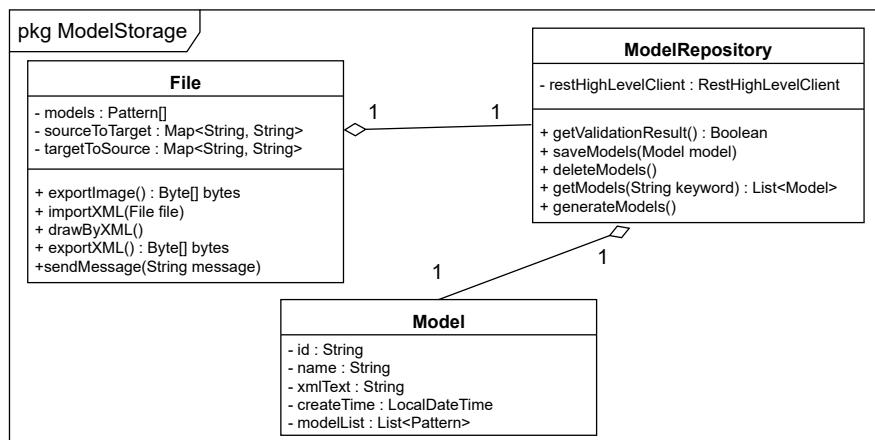


图 4-13: 模型存储与转化类图

模型存储与转化顺序图

“模型存储与转化”功能的示例过程如图 4-14 所示。开发人员或架构师使用浏览器客户端将 XML 模型文件导入画布，可以直接进行编辑，编辑完成后，可以选择导出为 XML 文件或图像格式，也可以将模型存储到数据库中并生成符合领域驱动设计规范的框架项目。

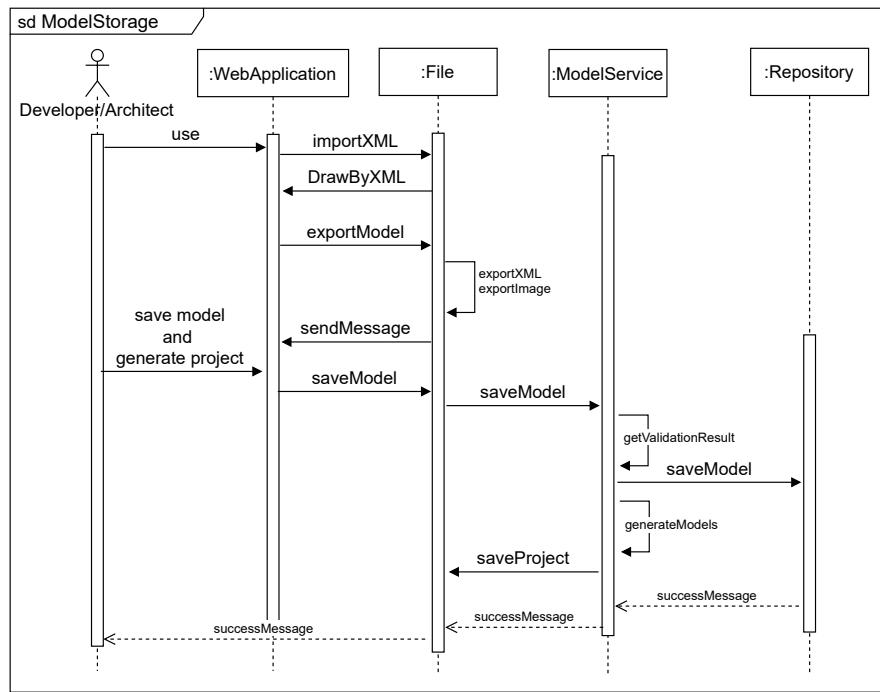


图 4-14: 模型存储与转化顺序图

模型存储部分实现代码

如图 4-15 所示是“模型存储”部分实现代码。主要描述将模型存储进 Elasticsearch 中的过程，包括对模型对象的构建、存储格式的转化、发起存储请求与接收响应。

```
public String saveModel(String name,
                        String xmlText,
                        List<PatternDTO> modelList) throws IOException {

    //日期格式化
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd hh:mm:ss");

    //构建model对象
    DDDModel model = DDDModel.builder().name(name)
        .xmlText(xmlText).modelList(modelList)
        .createTime(LocalDateTime.now())
        .format(formatter)
        .build();

    //创建请求
    IndexRequest request = new IndexRequest("dddmodels");

    //规则 PUT /index/_doc/
    request.timeout(TimeValue.timeValueSeconds(1));

    //数据放入请求
    request.source(JSON.toJSONString(model), XContentType.JSON);

    //客户端发送请求，获取响应的结果
    IndexResponse response = restHighLevelClient.index(request,
    RequestOptions.DEFAULT);

    return response.status().toString();
}
```

图 4-15: 模型存储部分实现代码

4.4 本章小结

本章介绍了建模支持工具的设计与实现。首先通过对建模场景中的用例分析，获取具体需求，包括“可视化建模”、“模型校验”以及“模型存储与转化”三个主要模块。然后对工具进行了功能性与非功能性需求的分析与展示，通过需求对工具进行总体设计与详细设计。最后通过各模块功能类图、顺序图、功能算法逻辑和实现代码来展示工具的具体实现过程。

第五章 建模支持工具测试与案例研究

本章将介绍建模支持工具 DDDD 的测试与案例研究工作。介绍了“可视化建模”、“模型校验”以及“模型存储与转化”三个模块的主要功能单元测试；还以案例的形式演示了工具的完整使用流程，验证了本文提出的建模支持工具满足开发人员和架构师的战术建模需求。

5.1 建模支持工具测试

本小节主要介绍对建模支持工具的功能测试和性能测试。主要对工具的三个模块进行测试，测试的功能主要包括：“可视化建模”、“建模约束校验”、“模型存储”以及“生成项目”。对“可视化建模”功能进行单元测试，验证建模支持工具是否支持灵活易用的建模过程；对“建模约束校验”功能进行单元测试，验证建模支持工具是否支持对建模结果正确性和规范性的保障；对“模型存储”和“生成项目”功能进行单元测试，验证建模支持工具是否支持模型的保存、复用和扩展使用，最后对支持主要功能的后端接口进行了性能测试。

进行单元测试时，主要针对实现功能的关键接口进行测试。涉及前后端交互的部分均以 Postman^①工具为辅助，采取 JSON 格式进行数据传输，代替实际项目中以抽象类封装的数据。进行性能测试时，以脚本形式对接口进行多次访问，统计接口的平均响应时间。

5.1.1 可视化建模测试

表 5-1 是对“可视化建模”进行测试的测试用例具体描述。通过输入一系列鼠标拖拽操作进行测试，测试的接口是实现绘制图形的核心接口“Draw.addCell”。测试目的是验证建模支持工具的“可视化建模”功能。单元

^①Postman 工具主页：<https://www.postman.com>

测试首先使用拖拽图形的方式进行“实体类”的绘制，并双击“实体类”修改名称；然后拖拽图形进行“值对象类”的绘制，并双击修改名称；最后，进行“实体类”和“值对象类”之间的连线绘制。

测试结果为建模结果的图像表示。对“可视化建模”进行测试的结果输出如图 5-1 所示，该图展示了成功绘制“实体类”和“值对象类”的图形建模结果，结果表明“可视化建模”可以支持正常建模。

表 5-1: 可视化建模测试用例

ID	TC1
测试目标	可视化建模实体与值对象，并建立连接
测试接口	Draw.addCell
前置条件	工具画布初始化正常
输入	(1) 拖拽“实体类”，绘制在画布中，并修改实体名为“Student”，唯一标识为“ID:String”； (2) 拖拽“值对象类”，绘制在画布中，并修改值对象名为“Book”； (3) 建立实体到值对象的连接。
预期输出	包含实体和值对象的图形化建模结果

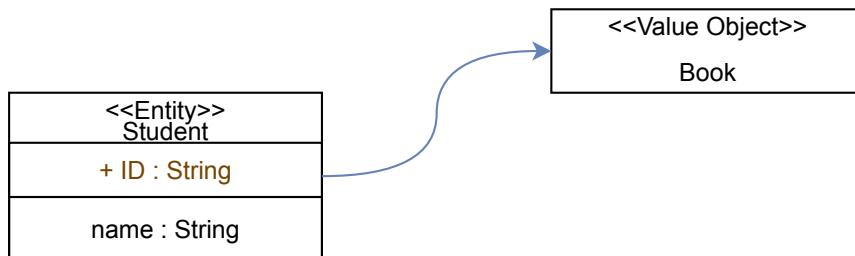


图 5-1: 图形化建模测试结果图

5.1.2 建模约束校验测试

表 5-2 是对“建模约束校验”进行测试的测试用例具体描述。输入为包含未连接任何对象的“资源库类”的建模结果，测试的接口是实现“建模约束校验”的核心接口“PatternData.validatio”。测试目的是验证对“资源库类”的“建模约束校验”功能。

由于表 ?? 中的约束 C12 规定，本次测试结果为校验失败，具体的结果如图 5-2 所示，以弹窗形式输出校验警告信息和修改提示。结果表明“建模约束校验”功能正常实现。

表 5-2: 建模约束校验测试用例

ID	TC2
测试目标	正确检测图形化建模约束问题，给出警告提示
测试接口	PatternData.validation
前置条件	工具画布初始化正常
输入	包含未连接任何对象的“资源库类”的建模结果
预期输出	校验失败信息“请给 Repository 连接一个它所存储的实体（或聚合根）”

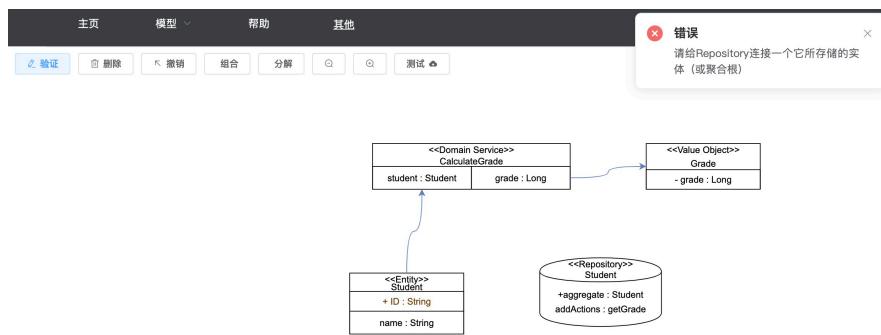


图 5-2: 建模约束校验测试结果图

5.1.3 模型存储测试

表 5-3 是对“模型存储”进行测试的测试用例具体描述。输入为模型名称以及模型 XML 格式文件，测试的接口是实现“模型存储”的核心接口“ModelRepository.saveModels”。测试目的为验证模型的“保存”功能。

数据库初始化正常且网络连接通畅，模型存储测试成功，并可以从数据库中检索出模型，绘制在画布上。结果表明“模型存储”功能实现正常。

表 5-3: 模型存储测试用例

ID	TC3
测试目标	正确存储模型，并能从模型数据库中检索重建模型
测试接口	ModelRepository.saveModels
前置条件	数据库初始化正常，网络连接通畅
输入	模型名称
预期输出	保存成功提示，成功从数据库中检索模型并重建模型

5.1.4 生成项目测试

表5-4是对“生成项目”进行测试的测试用例具体描述。输入为图形化建模结果，测试的接口是实现“生成框架项目”功能的核心接口“`ModelRepository.generateModels`”。测试目的为验证根据建模结果“生成框架项目”的功能。

测试结果为弹出生成项目成功的提示窗口，并可以在指定文件目录下访问项目文件，项目文件结构符合领域驱动设计战术建模规范。结果表明“生成项目”功能实现正常。

表 5-4: 生成项目测试用例

ID	TC4
测试目标	根据图形化建模结果生成框架项目
测试接口	<code>ModelRepository.generateModels</code>
前置条件	无
输入	图形化建模结果
预期输出	指定文件路径下的框架项目文件包

5.1.5 性能测试

性能测试部分主要针对“模型存储与转化”功能中的核心接口进行测试。测试环境如表5-5所示，描述了测试环境服务器的具体配置，将 `ElasticSearch` 部署至该测试环境中进行测试。

表 5-5: 测试环境

配置项	描述
操作系统	CentOS 7.6 64 位
CPU	Intel(R) Xeon(R) Platinum 8269CY, 4 核, 2.5 GHz/3.2 GHz
内存	8G
Java 版本	OpenJDK 1.8.0 181
ElasticSearch 版本	7.11.1

建模支持工具的所有模块中，“模型存储与转化”模块前后端交互频繁，需要经常请求持久化设施，而其他模块的前后端交互很少，几乎不需要考虑性能问题。故主要对“模型存储与转化”模块的接口进行性能测试。请求持久化设施会在网络上发生耗时的数据传输操作，所以响应时间（Response Time, RT）是衡量接口性能的核心指标，性能测试通过编写脚本对每个核心接口进行 50 次请求，并统计平均响应时间。具体测试结果如表5-6所示，展示了

测试接口的平均响应时间，测试的接口包括保存模型的“`saveModel`”接口，查询模型的“`getModels`”接口，删除模型的“`deleteModel`”接口，以及搜索模型的“`getModelByKeyword`”接口。表中平均响应时间说明后端接口反馈迅速，基本满足性能需求。

表 5-6: 接口性能测试

接口名	请求类型	平均响应时间
<code>saveModel</code>	PUT	50ms
<code>getModels</code>	GET	21ms
<code>deleteModel</code>	DELETE	13ms
<code>getModelByKeyword</code>	GET	18ms

5.2 案例研究

本小节对本文提出的战术建模支持方法及工具进行案例研究。通过案例演示使用支持工具进行建模的完整过程，来验证该建模支持方法和工具是否能够提供灵活易用的建模支持，提高建模效率；是否能够保证建模结果的标准化、规范化；以及是否能够满足建模结果的持久性、可复用性和扩展应用。

为验证本工具能够支持战术建模整个流程，本文以 Vaughn Vernon 的《实现领域驱动设计》[?] 中的企业协作软件系统 CollabOvation 为研究案例展开验证。

SaaSOvation 是一家企业办公软件提供商，CollabOvation 是由 SaaSOvation 公司开发的一款企业协作（Collaboration）软件，并且加入社交网络的功能。该产品的功能包括论坛、共享日历、博客、即时消息、wiki、留言板、文档管理、通知和提醒、活动跟踪和 RSS 等。所有协作工具都旨在满足企业服务的需求，帮助他们在项目中提高效率。CollabOvation 项目启动后，项目中的一些有经验的开发人员采用领域驱动设计将项目划分为协作上下文、身份与访问上下文和敏捷项目上下文，具体项目代码已发布到 GitHub^① 代码仓库。案例研究将 CollabOvation 的身份与访问上下文作为业务范围，进行战术建模。案例研究验证过程包括使用建模支持工具进行标准化战术建模的五个具体步骤，并给出平台截图展示。

^①IDDD-Samples 代码仓库地址：<https://github.com/VaughnVernon/IDDDSamples/>

5.2.1 验证步骤

本小节将对建模支持方法及工具进行验证，首先验证可视化建模能否支撑含有复杂设计的战术建模，满足图形化的建模基本需求；其次验证对精细化建模的支持程度，能否正确识别建模中不规范的操作，并给予相应的提示；最后验证建模结果的存储与转化使用，能否正确存储建模结果并生成框架项目指导编码工作。

Vaughn Vernon 在《实现领域驱动设计》[?] 中已经详细介绍了项目的背景和需求，但没有进一步细化划分出的限界上下文，以下的验证步骤的目的（也是本案例研究的目的）在于探究本文所提出的建模支持方法及工具能否支持对所选案例进行战术建模，具体验证工作步骤如下：

- 步骤一：**开发人员通过工具快速学习战术建模相关概念知识；
- 步骤二：**开发人员进入可视化建模页面，进行建模；
- 步骤三：**对于不符合战术建模规范的操作，没有通过战术模式约束校验的属性弹出警告提示；
- 步骤四：**建模结果校验通过，将模型导出为 XML 文件和图片，保存到数据库中，并生成框架项目文件；
- 步骤五：**读取保存的模型文件或数据库中存储的模型，进行二次建模。

5.2.2 验证过程

本小节将依据上一节所述的步骤对支持工具进行演示和验证。

步骤一：点击工具平台首页的“Can I DDD”按钮，进入展示战术建模相关概念知识的帮助页面，如图 5-3 所示为帮助页面。该页面帮助开发人员快速了解战术建模相关概念与使用场景，回顾战术建模知识，降低使用工具的门槛。



图 5-3: 工具帮助页面图

步骤二：开发人员或架构师进入可视化建模页面，根据项目背景需求进行战术建模，由于整个战术建模过程时间跨度长，模型复杂，故仅选择战术建模的一部分进行展示：

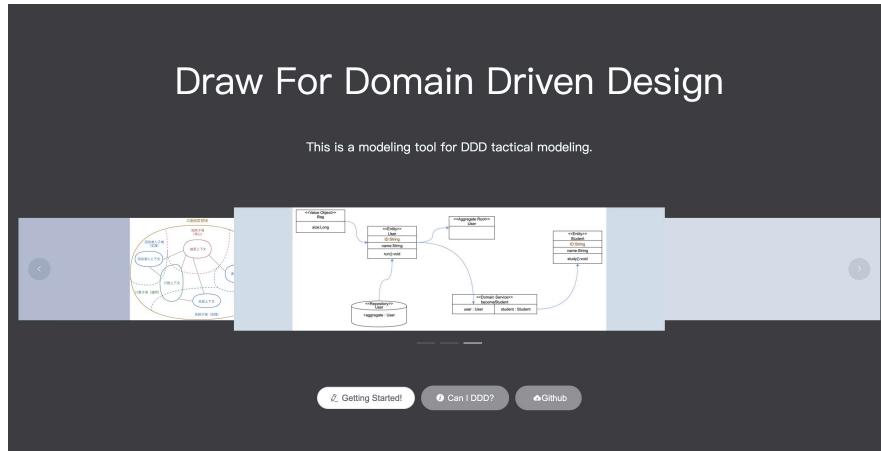


图 5-4: 工具主页

1. 工具主页如图 5-4 所示，点击“Getting Started!”按钮进入建模页面；
2. 进入建模页面后，拖动左侧工具栏实体、值对象以及领域事件模式图，放在右侧画布合适位置，进行绘制；
3. 双击需要修改的模式属性，输入文字进行修改；
4. 点击对象，按住拖动进行对象间连线；
5. 点击右侧画布菜单按钮，可以对建模操作进行撤销，对画布进行缩放，将建模对象进行组合和删除。

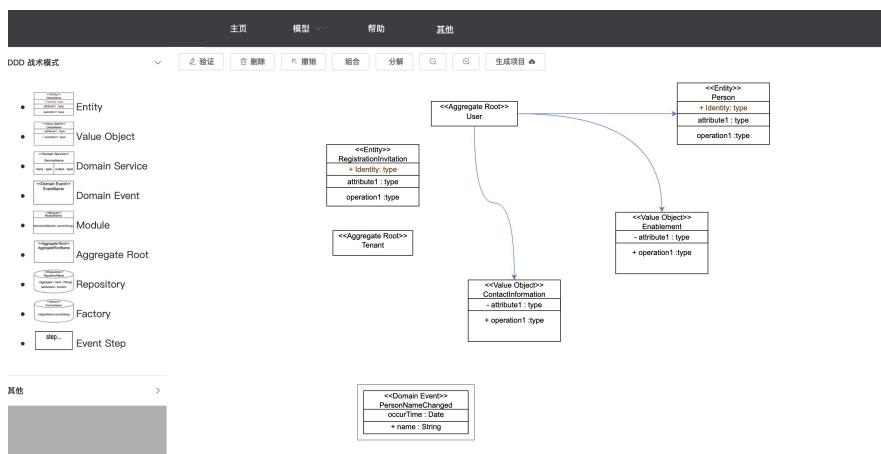


图 5-5: 可视化建模过程

如图 5-5 所示，展示了上述步骤二的建模过程最终结果。

步骤三：建模时可以对建模结果进行实时校验，校验规则严格按照本文提出的战术建模支持方法要求，并给出校验结果展示：

1. 进行可视化建模时，点击右侧画布菜单按钮“验证”；
2. 如图 5-6 所示，根据当前图形化建模结果，进行约束和规则校验，弹出不符合规范和约束的警告信息。

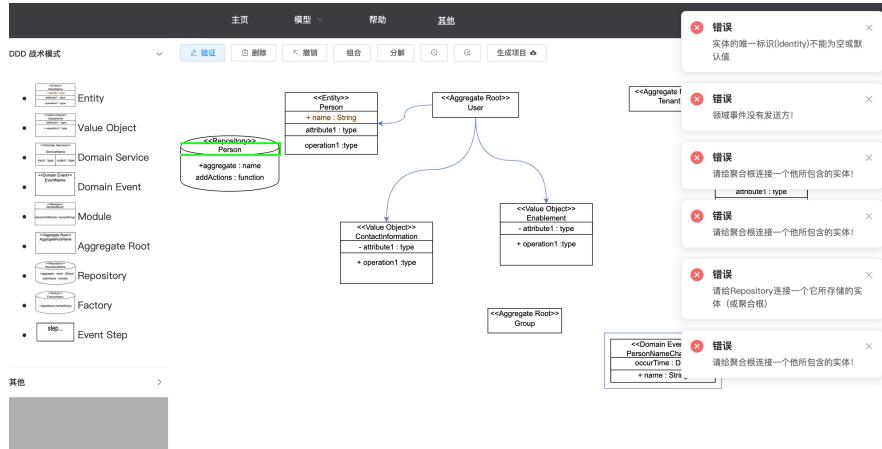


图 5-6: 模型校验

步骤四：选择菜单如图 5-7 所示，建模完成后，经过校验模型符合规范，可以将模型以 XML、图像形式导出，并存储到数据库中。

1. 建模完成，校验通过；
2. 在菜单中依次选择“模型”、“导出”、“XML 格式”，选择保存文件夹；
3. 在菜单中依次选择“模型”、“导出”、“图像格式”，选择保存文件夹；
4. 在菜单中依次选择“模型”、“保存模型到数据库”，将模型命名为“userAccess”，点击确定；
5. 在画布菜单中点击按钮“生成项目”，框架项目文件在指定文件目录中生成。



图 5-7: 保存导出模型

步骤五：再次使用工具时，可以使用“导入”和“加载历史模型”功能，来重现建模结果，并进行二次建模。

- 在菜单中依次选择“模型”、“加载历史模型”，持久化设施中保存的模型文件列表如图 5-8 所示，展示了模型名称、创建时间以及操作按钮；

历史模型		
模型名称	创建时间	操作
testModel3	2021-03-10 02:33:24	查看详情 加载模型 删除
userAccess	2021-03-12 06:49:13	查看详情 加载模型 删除
testModel1	2021-03-12 06:49:39	查看详情 加载模型 删除
student	2021-03-12 06:49:50	查看详情 加载模型 删除
car	2021-03-12 06:50:01	查看详情 加载模型 删除

图 5-8: 加载历史模型

- 选择想要加载的模型，点击“加载模型”按钮，画布中将绘制出模型；
- 加载的历史模型如图 5-9 所示，可以对历史模型进行新的操作。

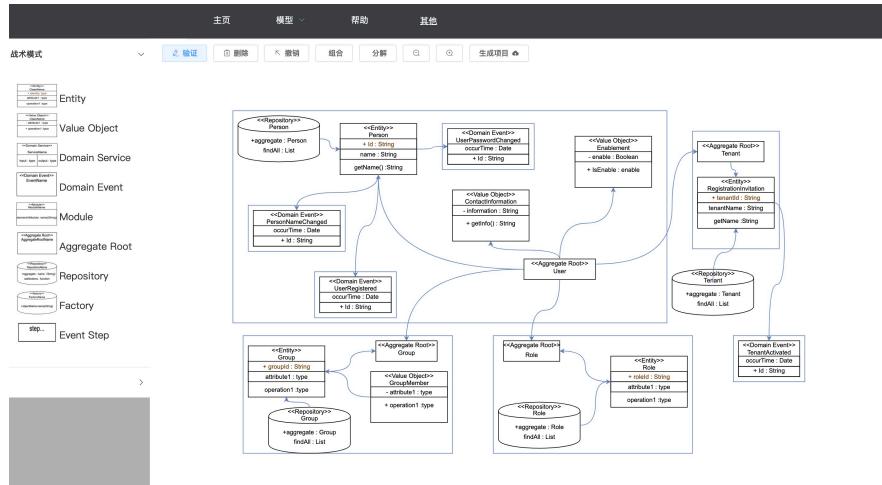


图 5-9: 建模结果图

通过上述五个步骤的验证，得到验证结果如表 5-7 所示，表格展示了每个步骤对应的功能需求和验证的结果。以上步骤和得到的结果表明，提出的建模

支持方法及工具达到了以下效果。第一，实现了可视化战术建模，使建模过程更加高效灵活；第二，对建模结果进行约束验证，来判断是否符合战术建模需要遵守的规则，解决了战术建模阶段流程化与标准化的规范性问题；第三，可以将建模结果进行存储，方便下次访问，并提供根据建模结果生成框架项目的扩展功能，提高了建模结果的可复用性和扩展性。

表 5-7：案例验证结果

步骤	需求	验证结果
步骤一	帮助开发者了解或回顾战术建模概念知识	将来自书籍和调研的经验理论总结成体系，供开发者学习，验证结果表明实现的工具满足开发者了解和回顾知识的需求
步骤二	通过拖拉拽图形和连线、双击修改文本，达到可视化建模	通过工具对“User”、“Person”和“Tenant”等领域概念进行建模，实现的工具达到了可视化建模的需求
步骤三	对建模结果进行校验，并给出修改提示	点击工具中“验证”按钮，对画布中现有的对象进行校验，弹出警告提示，实现的工具达到了验证建模结果标准性与规范性的需求
步骤四	将建模结果转化为多种形式保存到本地，同时将建模结果保存到远程数据库	通过工具保存了建模结果的 XML、图像和框架项目文件到本地，同时也保存了建模结果到远程数据库，实现的工具达到了存储模型的需求
步骤五	将建模结果再次从本地导入或从远程数据库加载	通过工具导入了本地 XML 格式的模型文件，也可以加载远程数据库的模型，实现的工具达到了再次编辑已有模型的需求

综上所述，本文提出的战术建模支持方法及工具满足战术建模的功能性需求，可以支撑战术建模全流程的实践。

5.3 本章小结

本章介绍了对战术建模支持方法及工具的测试和案例研究。首先设计测试用例对建模支持工具的“可视化建模”、“建模约束校验”、“模型存储”以及“生成项目”功能进行了测试，还通过在具体案例中执行建模的五个步骤对建模支持方法及工具进行了验证。

第六章 总结与展望

6.1 总结

领域驱动设计作为一种针对复杂业务流程的分析与建模方法，正在成为大型分布式系统设计与实现的最佳解决方案，其中的战术建模层次可以帮助更快地进行建模设计与实现代码的落地。然而，战术建模在应用时仍面临着许多挑战，由于对战术模式的规范和约束不明确，往往会导致战术建模结果不够标准和规范；开发人员和架构师对战术建模理解程度不同，无法统一和复用建模结果；领域建模过程缺少相应的支持平台和工具。为了解决这些挑战，对该领域的研究进展与工业界实践经验进行了调查与总结，提出了一套战术建模支持方法及工具。

具体来说，本文所提出的战术建模支持方法及工具包括以下三个具体贡献。其一，通过对领域驱动设计战术建模过程的理论调研和对工业界从业人员的访谈，本文总结得出一套战术建模指南，包括八种战术模式、这些模式的重要属性、使用时机以及实现技术，该建模指南经过工业界实践经验认证，可以作为战术建模实践时的指导。其二，通过开展焦点小组讨论，本文构建了一种战术建模语言，该战术建模语言作为使用战术建模支持方法的支撑，提升了建模的效率与准确性。以上两点构成了战术建模支持方法的主要内容。其三，以所提出的战术建模语言为基础，本文还实现了一个建模支持工具，实现了灵活易用的可视化建模过程，保证了扩展性与通用性，并避免了对特定平台的依赖。

本文提出的战术建模支持方法和工具，支持简单了解领域驱动设计战术建模基本概念，通过灵活的可视化建模来实现建模过程，并校验建模结果的标准性与规范性，为战术建模提供了一套可靠的流程。此外，还支持将建模结果以多种格式导出和存储，还能为实现阶段构建框架项目，提高了战术建模结果的复用程度。总体而言，本文提出的工具也降低了使用战术建模的最低要求，对领域驱动设计在实践中发展具有积极意义。

6.2 展望

本文提出的领域驱动设计战术建模支持方法及工具可以支持战术建模实践，但仍有很多方面可以进一步提升。首先，战术建模的属性和实现技术在实践时还应该做到根据业务特性进行变通，对不同开发团队的不同业务，战术建模工具应该有不同的个性化配置；其次，战术建模工具对于战术建模的结果利用程度还不够高，除了生成框架项目之外，未来该工具还将支持更多多种形式与更细粒度的扩展结果；最后，战术建模工具在多人协同建模方面支持度不够，仅能通过共享建模结果来进行协同，未来该工具还将支持通过在线协作实现即时协同建模，提升建模过程的体验并提高建模效率。

致 谢

在本篇论文最后，我想真诚地感谢在我撰写论文过程中指导和帮助我的老师、同学们，还有支持我、鼓励我的家人和朋友们。

首先，我要感谢在我毕设研究工作中，张贺教授对我的悉心指导和帮助，在跟随您的两年研究生学习工作中，您总是教导我们要多去实践，要以严谨认真的态度踏踏实实地完成科研学习工作，还要努力阅读大量学术文献，提高自己看问题的层次，还为我们提供了很多与工业界交流的机会，这些都为我的毕设工作和日常工作学习带来了很大的好处。

然后，我要感谢实验室的师兄师姐和同学们，感谢你们这两年中在学习、工作和生活中带给我的帮助与鼓励，特别是指导我毕设工作的师姐，对我的问题一一解答，对我的论文工作耐心指导，启发我进行深度的思考，反复与我评估和讨论论文研究的问题，得益于她的帮助，我的论文才能保质保量地顺利完成。

最后，还要感谢我的父母和家人，感谢你们在背后默默支持和陪伴我。很幸运，在研究生阶段遇到了相互扶持，互相鼓励的伴侣，在生活和学习上我们互相帮助，给予彼此很大的动力与希望。也要感谢我的舍友们，让我的生活中充满了乐趣。

在论文工作中，访谈和焦点小组中付出了大量的精力时间，实现工具时也遇到过困难与问题，但我依然坚持尽力做到最好，马上就要从学校步入社会，我将坚持“诚朴雄伟，励学敦行”的精神，不畏艰难，努力奋斗出属于自己的一片天地。

参考文献

- [1] 邵奇峰, 张召, 朱燕超, et al. 企业级区块链技术综述 [J]. 软件学报, 2019, 30(9) : 2571 – 2592.
- [2] ONIK M M H, MIRAZ M H. Performance analytical comparison of blockchain-as-a-service (baas) platforms[C] // International Conference for Emerging Technologies in Computing. 2019 : 3 – 18.
- [3] 刘宏宇, 梁秀波, 吴俊涵. 基于 Kubernetes 的 Fabric 链码管理及高可用技术 [J]. 计算机应用, 2021, 41(4) : 956 – 962.
- [4] GERRITS L, KILIMOU E, KROMES R, et al. A Blockchain cloud architecture deployment for an industrial IoT use case[C/OL] // 2021 IEEE International Conference on Omni-Layer Intelligent Systems, COINS 2021, Barcelona, Spain, August 23-25, 2021. [S.l.] : IEEE, 2021 : 1 – 6.
<https://doi.org/10.1109/COINS51742.2021.9524264>.
- [5] XIE H, ZHANG Z, ZHANG Q, et al. HBRSS: Providing high-secure data communication and manipulation in insecure cloud environments[J/OL]. Comput. Commun., 2021, 174 : 1 – 12.
<https://doi.org/10.1016/j.comcom.2021.03.018>.
- [6] SUN J, WU C, YE J. Blockchain-based Automated Container Cloud Security Enhancement System[C/OL] // IEEE International Conference on Smart Cloud, SmartCloud 2020, Washington, DC, USA, November 6-8, 2020. [S.l.] : IEEE, 2020 : 1 – 6.
<https://doi.org/10.1109/SmartCloud49737.2020.00010>.
- [7] TOSH D, SHETTY S, FOYTIK P, et al. CloudPoS: A Proof-of-Stake Consensus Design for Blockchain Integrated Cloud[C/OL] // 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). 2018 : 302 – 309.
<http://dx.doi.org/10.1109/CLOUD.2018.00045>.

- [8] LIANG X, ZHAO Q, ZHANG Y, et al. EduChain: A highly available education consortium blockchain platform based on Hyperledger Fabric[J]. Concurrency and Computation: Practice and Experience, : e6330.
- [9] WAN Z, CAI M, YANG J, et al. A novel blockchain as a service paradigm[C] // International Conference on Blockchain. 2018 : 267–273.
- [10] 才丽. 面向 BaaS 平台的资源调度算法研究与实现 [D]. [S.l.]: 浙江大学, 2018.
- [11] SHI Z, JIANG C, JIANG L, et al. HPKS: High Performance Kubernetes Scheduling for Dynamic Blockchain Workloads in Cloud Computing[C/OL] // 2021 IEEE 14th International Conference on Cloud Computing (CLOUD). 2021 : 456–466. <http://dx.doi.org/10.1109/CLOUD53861.2021.00060>.
- [12] 袁勇, 王飞跃. 区块链技术发展现状与展望 [J]. 自动化学报, 2016, 42(4) : 481–494.
- [13] 朱显锦, 姚建国, 管海兵. 区块链即服务: 下一个云服务前沿 [J]. 软件学报, 2020, 31(1) : 1–19.
- [14] ZHANG S, YAN H, CHEN X. Research on Key Technologies of Cloud Computing[J/OL]. Physics Procedia, 2012, 33 : 1791–1797. <https://www.sciencedirect.com/science/article/pii/S1875389212015994>.
- [15] 赵添喜, 王静, 陈天琪. 云原生技术剖析 [J]. 科学与信息化, 2021(1) : 1.
- [16] 刘博涵, 张贺, 董黎明. DevOps 中国调查研究 [J]. 软件学报, 2019, 30(10) : 3206–3226.
- [17] BHAGAVAN S, BALASUBRAMANIAN S, ANNEM P R, et al. Achieving Operational Scalability Using Razee Continuous Deployment Model and Kubernetes Operators[J]. arXiv preprint arXiv:2012.10526, 2020.
- [18] 王骏翔, 郭磊. 基于 Kubernetes 和 Docker 技术的企业级容器云平台解决方案 [J]. 上海船舶运输科学研究所学报, 2018, 41(3) : 7.
- [19] BEN-KIKI O, EVANS C, INGERSON B. Yaml ain't markup language (yamlTM) version 1.1[J]. Working Draft 2008-05, 2009, 11.

- [20] SPILLNER J. Quality assessment and improvement of helm charts for kubernetes-based cloud applications[J]. arXiv preprint arXiv:1901.00644, 2019.
- [21] CALCOTE L, BUTCHER Z. Istio: Up and running: Using a service mesh to connect, secure, control, and observe[M]. [S.l.] : O'Reilly Media, 2019.
- [22] LARSSON L, TÄRNEBERG W, KLEIN C, et al. Impact of etcd deployment on Kubernetes, Istio, and application performance[J]. Software: Practice and Experience, 2020, 50(10) : 1986–2007.
- [23] MCCORRY P, SHAHANDASHTI S F, HAO F. A smart contract for boardroom voting with maximum voter privacy[C] // International conference on financial cryptography and data security. 2017 : 357–375.
- [24] CHANG S E, CHEN Y. When blockchain meets supply chain: A systematic literature review on current development and potential applications[J]. IEEE Access, 2020, 8 : 62478–62494.
- [25] ZHANG Y, WEN J. The IoT electric business model: Using blockchain technology for the internet of things[J]. Peer-to-Peer Networking and Applications, 2017, 10(4) : 983–994.
- [26] LEKA E, SELIMI B, LAMANI L. Systematic literature review of blockchain applications: Smart contracts[C] // 2019 International Conference on Information Technologies (InfoTech). 2019 : 1–3.
- [27] CHRISTIDIS K, DEVETSIKIOTIS M. Blockchains and smart contracts for the internet of things[J]. Ieee Access, 2016, 4 : 2292–2303.
- [28] ZHAO X, CHEN Z, CHEN X, et al. The DAO attack paradoxes in propositional logic[C] // 2017 4th International Conference on Systems and Informatics (IC-SAI). 2017 : 1743–1746.
- [29] GAI K, GUO J, ZHU L, et al. Blockchain meets cloud computing: A survey[J]. IEEE Communications Surveys & Tutorials, 2020, 22(3) : 2009–2030.
- [30] 张富利, 侯培宇, 李杉杉, et al. 一种智能合约微服务化框架 [J]. 软件学报, 2021, 32(11) : 17.

-
- [31] YILMAZ O. Extending the Kubernetes API[M/OL] // Extending Kubernetes: Elevate Kubernetes with Extension Patterns, Operators, and Plugins. Berkeley, CA : Apress, 2021 : 99 – 141.
https://doi.org/10.1007/978-1-4842-7095-0_4.
 - [32] HENNING S, WETZEL B, HASSELBRING W. Reproducible Benchmarking of Cloud-Native Applications with the Kubernetes Operator Pattern[J], 2021.
 - [33] D'AQUINO A. Design, prototyping and validation of a Kubernetes operator for an IoT platform based on Red Hat OpenShift.[J], 2020.
 - [34] YU J, NAGANUMA Y, IDE T. System Operator: A Tool for System Management in Kubernetes Clusters[J]. CLOUD COMPUTING 2020, 2020 : 83.
 - [35] ZHOU N, GEORGIOU Y, POSPIESZNY M, et al. Container orchestration on HPC systems through Kubernetes[J]. Journal of Cloud Computing, 2021, 10(1) : 1 – 14.
 - [36] VUKASOVIć M, VESELINOVIC B, STANISAVLJEVIć . A development of a configurable system for handling X509 certificates[C/OL] // 2017 25th Telecommunication Forum (TELFOR). 2017 : 1 – 4.
<http://dx.doi.org/10.1109/TELFOR.2017.8249485>.
 - [37] SUKHIJA N, BAUTISTA E. Towards a framework for monitoring and analyzing high performance computing environments using kubernetes and prometheus[C] // 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCAL-COM/UIC/ATC/CBDCom/IOP/SCI). 2019 : 257 – 262.

简历与科研成果

基本信息

张富利，男，汉族，1997年8月出生，山东省临沂人。

教育背景

2015年9月 – 2019年6月 中国石油大学(华东) 计算机与通信工程学院 本科

攻读工学硕士学位期间完成的学术成果

1. Fuli Zhang, Peiyu Hou, “一种智能合约微服务化框架” 3篇论文！！ in *Proc. IEEE International Conference on Communications (ICC) 2010*, May. 2010.
2. 论文作者在攻读学位期间参与的科研课题的列表，按照日期从近到远排列。