**南京大学软件学院研究生学位论文中期检查报告格式**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **南京大学软件工程硕士学位论文中期检查报告** | | | | | |
| 导师1姓名 | 王金庆 | 研究生姓名  （学号） | 金思晔  (MF1832073) | 方向 | 软件工程 |
| 导师2姓名 |  |
| 论文题目 | 饿了么热斧平台的设计与实现 | | | | |
| 论文选题来源及研究的目的和意义（500字左右）：  论文选题源自于本人实习时参与的“饿了么热斧平台”。热斧平台负责三部分功能：超时规则与处理策略的制定、判定订单是否超时以及超时之后的处理。  在电商平台中，超时场景是指订单或者运单未在指定时间内达到某种状态。对超时场景的处理是服务类平台不可或缺的能力，饿了么、美团、淘宝等都有对超时场景的判定和相应的补偿操作。对于不同的超时场景，有不同的处理策略，例如淘宝用户下单后三十分钟未支付，那么系统会取消这笔订单，并释放之前锁定的库存；饿了么用户点外卖后，骑手没有在规定时间内将外卖送到指定地点，那么用户就会收到系统赔付的优惠券等。  饿了么既有的超时场景判定分散在各个服务中，例如逆向系统、赔付系统、用户触达系统等，由此产生了诸多问题：  一是不仅工作量巨大，而且容易出错。每次修改超时规则，都需要各个系统依次同步修改，不仅增加了额外工作量，而且经常发生同步不及时导致的各种意想不到的问题。  二是无法快速响应业务需求。目前超时规则采用“硬编码”方式，嵌套于代码，如果需要修改规则，需要产品提紧急需求，开发人员出技术方案，各方评审方案，开发人员开发，测试人员测试，然后在发布窗口发布。对于开发人员来说，增加了高优先级的任务，需要优先完成紧急需求，增加了开发人员的工作量；对于业务方来说，需求得不到快速响应，延误了策略的实施。  三是规则颗粒度控制困难。上述问题，运营人员无法做到对规则的细粒度把控：例如，根据天气配置规则，根据城市配置规则，根据订单来源配置规则等，每次规则变动的需求都会造成牵一发而动全身的影响。  为了解决上述问题，饿了么需要将分散在各个系统中的超时场景判定逻辑抽离出来，集中在一起进行判定，同时需要有一个配置平台，能够对超时规则以及处理逻辑进行配置，以达到快速响应业务需求的目的。为此，本人参与设计并实现了“饿了么热斧平台”，以满足上述要求。 | | | | | |
| 该方向的研究现状或技术进展综述（2000字左右）  在面对多变的业务逻辑，将业务逻辑可配置化一种可靠的应对方法，规则引擎是解决此类问题的方法之一。  Drools是一款开源的通用规则引擎，应用场景较广泛，例如在征税系统中，由很多业务规则需要进行匹配，比如企业名称的长度，合伙纳税人比例情况，外资企业比重情况等，这些业务规则会随着国家税制的完善和改革出现变动，使用drools规则引擎就能够很好的解决；  在游戏行业，桌游三国杀就运用了drools，它选择游戏脚本的开发模式，将游戏的逻辑规则相关的部分放进脚本中，以减少游戏各个模块之间的耦合，方便游戏的测试、修改以及内容的升级更新等。drools使用的是RETE算法，RETE算法是一种前向规则快速匹配算法，其匹配速度与规则数目无关。Rete算法通过形成一个rete网络进行模式匹配，利用基于规则的系统的两个特征，即时间冗余性和结构相似性，提高系统模式匹配效率。drools有自己的语言，同时有自己的以drl结尾的文件格式，用户可以使用drools的语言，在drl文件中编写业务逻辑规则，通过修改drl文件的方式来修改业务逻辑。drools也存在如下的几个问题：规则的配置不够友好，对于不会编程的使用者来说，编写drl文件是一个较为困难的事情，上手难度较高，而且不能保证语法和逻辑的正确性；drools的处理速度相较于原生程序代码慢很多，在调研过程中，使用drools和原生代码开发同一段业务逻辑，drools需要的时间比原生代码高出一个数量级，在应对时效性强的需求的时候，drools的表现不够理想。drools更适用于逻辑判断复杂，时效性没那么高的场景  除了使用开源的规则引擎，在业务逻辑不复杂，但又需要能够进行配置的场景中，将执行逻辑封装成一个执行单元，并以树的形式构建执行单元也是一种常用的方式。阿里巴巴零售通的营销平台就是通过这样的方式实现，营销平台需要进行优惠计算，平台实现了一些基本计算单元，例如逻辑与、逻辑或、逻辑非、算数加、算数乘等，同时实现了一些条件权益计算单元，例如amountAtCondition（满条件）、FreePostageRights（包邮权益）等，以树的方式进行组织，对树做后序遍历，即可以判定是否满足条件。通过前端页面配置的方式，可以轻松的更改判断条件，也就是更改树的节点，能够快速的应对需求的变更。  除此之外，还有一些商用规则引擎，例如国内的旗正规则引擎，国外的例如IBM的Ilog等。  目前来说，开源的规则引擎能够做到较好的通用性，但需要一定的学习成本，对于特殊的需求无法满足，同时存在对使用人员不友好，性能达不到要求的情况。商用的规则引擎需要支付高昂的使用费用，且存在数据泄露的问题。开源社区有类似的规则可配置化的软件，但这类软件都和具体的业务相关联，例如风控平台、营销平台等，无法直接使用，且二次开发成本不亚于开发一个新的软件，因此，需要开发一个贴合业务需求的规则配置与执行平台很有必要。 | | | | | |
| 论文的主要技术路线、研究思路和实现方法；相关项目应用前景：（重点说明变更部分）：  本文使用springboot作为基础开发框架，Mybasit作为数据库访问框架，msyql作为数据源，除此之外需要使用redis做缓存以加快访问速度，同时降低mysql的压力，系统与系统之间通信通过消息队列的方式进行，以及来减少系统之间的耦合。除此之外，服务治理也是一个很重要的事情，系统使用Haskar进行服务治理，Haskar是集服务发现，服务注册，配置中心为一体，大大减少了服务之间发现的成本。服务使用docker作为容器进行部署，使用k8s对docker服务进行管理，减少发布的难度。  应用前景：将多变的业务逻辑与代码分离是加快项目迭代，减少项目故障率的一种有效的方式，实现方式一种是通过规则引擎实现，例如使用开源的drools规则引擎或者使用商用规则引擎，另一种需要和当前业务进行结合，根据业务的繁杂程度与性能要求，进行定制。应用前景很广泛。 | | | | | |
| 本人在相关项目中的扮演的角色和承担的工作（重点说明变更部分）：  本人为项目的主要设计人员和开发人员，负责项目的总体设计和编码实现。 | | | | | |
| 论文的主要工作（500字左右）：  1 研究原有系统存在的不足，研究当前存在的解决方法，确定整体待解决问题  2 挖掘需求，明确具体需求，细化各个功能点  3 进行系统的总体设计，包括架构设计、部署设计、数据库设计等  4 系统的详细设计与实现，流程的设计，类图的设计等，进行编码  5 进行单元测试，集成测试，性能测试等。 | | | | | |
| 论文三级大纲：  [摘 要 V](#_Toc25947911)  [Abstract VI](#_Toc25947912)  [第一章 引言 8](#_Toc25947913)  [1.1 项目背景 8](#_Toc25947914)  [1.2 国内外发展现状 9](#_Toc25947915)  [1.3 本文主要工作 9](#_Toc25947916)  [1.4 本文的组织结构 9](#_Toc25947917)  [第二章 技术综述 9](#_Toc25947918)  [2.1 SpringBoot 9](#_Toc25947919)  [2.2 MaxQ 9](#_Toc25947920)  [2.3 Huskar 10](#_Toc25947921)  [2.4 WorkFlow 10](#_Toc25947922)  [2.5 Redis 10](#_Toc25947923)  [2.6 本章小结 10](#_Toc25947924)  [第三章 系统分析与设计 11](#_Toc25947925)  [3.1 系统总体规划 11](#_Toc25947926)  [3.2 系统需求分析 11](#_Toc25947927)  [3.2.1 规则管理模块 11](#_Toc25947928)  [3.2.2 审核流模块 12](#_Toc25947929)  [3.2.3 业务处理模块 13](#_Toc25947930)  [3.3 非功能性需求 15](#_Toc25947931)  [3.5 数据库设计 15](#_Toc25947932)  [3.6 本章小结 15](#_Toc25947933)  [第四章 系统实现 15](#_Toc25947934)  [4.1 规则配置模块 15](#_Toc25947935)  [4.2 审核流模块 16](#_Toc25947936)  [4.3 业务处理模块 17](#_Toc25947937)  [4.5 系统测试 17](#_Toc25947938)  [4.5.1 测试环境 17](#_Toc25947939)  [4.5.2 测试设计 17](#_Toc25947940)  [4.5.3 测试结果 17](#_Toc25947941)  [第五章 总结与展望 17](#_Toc25947942)  [5.1 总结 17](#_Toc25947943)  [5.2 工作展望 17](#_Toc25947944)  [参 考 文 献 18](#_Toc25947945)  [致 谢 18](#_Toc25947946) | | | | | |
| 论文和相关项目的当前进度：  目前项目已经实现完成，论文的编写已经过半。 | | | | | |
| 论文和相关项目进展过程中遇到的困难和问题，以及解决的措施：  问题：对于规则单元的粒度把控不好，规则单元的封装需要进一步的思考  措施：对比较粗粒度的封装以及较细粒度封装两者直接对开发、维护、拓展的影响 | | | | | |
| 主要参考文献：  [1]周里程,熊碧辉,裘瑞清,周后盘.Drools规则引擎的发展及应用[J].电子技术与软件工程,2017(21):62-63.  [2]陶晓峰. 基于规则引擎的业务高效处理研究和应用[C]. 中国电力科学研究院.2017智能电网发展研讨会论文集.中国电力科学研究院:北京市海淀区太极计算机培训中心,2017:364-367.  [3]解宁宇. 基于规则引擎的智慧家居网关设计与实现[D].重庆邮电大学,2017.  [4]舒琴. 一种适用于薪资计算的规则引擎的研究与实现[D].华中师范大学,2012.  [5]朱会兵. 基于Drools的信息管理与决策系统的研究与实现[D].武汉理工大学,2012.  [6]周旭东. 基于Redis分布式存储的负载平衡及性能优化研究[D].南京邮电大学,2019.  [7]唐赟. 基于Groovy的通用交易监控报警系统的设计与实现[D].南京大学,2015.  [8]余永城,翁秋华,段卿,袁伟.RabbitMQ在气象通信系统中的应用研究[J/OL].计算机技术与发展,2020(04):1-7[2020-03-11].http://kns.cnki.net/kcms/detail/61.1450.TP.20191218.1112.036.html.  [9]曾楚之.基于服务网格的微服务架构服务治理[J].电子技术与软件工程,2019(12):7.  [10]陶志,向忠清.微服务架构Service Mesh的设计与应用[J].自动化技术与应用,2020,39(01):49-53. | | | | | |
| 导师意见： | | | | | |
| 学院备案意见：        年 月 日 | | | | | |