1. 业务模型
   1. 开发背景

传统的学术会议和期刊都会采用邮寄和电子邮件两种。邮寄投稿方式需要很长时间的反馈，并有可能投递失误。近年来电子邮件投稿方式的弊端也逐渐凸显，一方面论文作者需要通过电话来咨询稿件的审稿情况，大量的稿件也需要编辑部逐一下载、登记、打字排版，这无疑增加编辑部的工作量；另一方面，接受的电子邮件稿件相关信息往往不是很完整，如果处理不及时，会延长稿件处理周期。

随着学术界的论文越来越多，论文种类越来越繁杂，学术界对效率要求越来越高，传统方法已经很难满足对大量稿件进行高效的处理；同时，在全球化的今天，世界各地的学术交流也越来也重要，传统方法无法实现即时沟通。因此，需要新的业务模型来解决这一问题。

* 1. 业务需求

在线投稿系统立足于网络投稿，旨在为用户提供一个友好的网络论文投稿平台。

1) 保证信息的准确性和时效性。随时通过计算机对系统进行管理，使得论文、作者等信息处于最新状态，保证统计数据和分析的准确。

2) 减轻劳动强度、提高工作效率。网络在线投稿系统可以发挥计算机的强大功能，让管理人员从大量繁琐的手工劳动中解放出来，全方位整合资源, 提高工作效率。

3) 吸引稿源。本系统旨在为作者提供一个方便的平台，作者可以及时获得稿件处理意见及相关进程，操作过程非常友好，加上友好的用户界面，因此作者也会更愿意在此平台投稿。

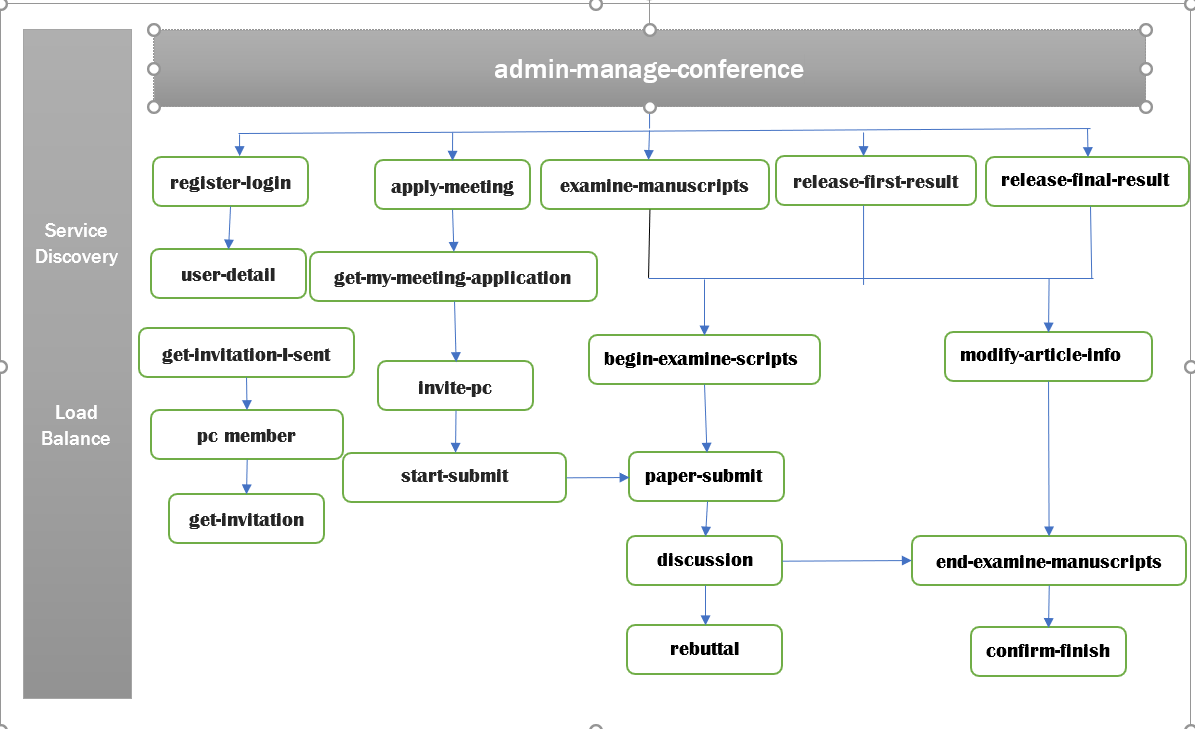
4) 增加论文管理的透明度，规范化管理。相较于邮寄和电子邮件，从投稿，审稿，编辑，发行等全过程的网络化管理更加规范，同时能即时发现学术不端等问题，论文管理更加透明，可以使学术更加公正。

1. 整体架构

本项目是基于微服务架构的会议管理系统，其中包含19种微服务。本项目没有前端界面，通过直接向后端的端口发送请求来调用api，所有api均为get或者post形式，post请求头规定为json。

项目所使用的编程语言为java，核心框架是Java-Spring Boot和Spring Cloud，而数据库使用了flask,pymongo,pytest,requests等python库。同时，使用redis作为session缓存，共享session，作为登录判断。

所有微服务的整体架构如下图所示：



1. 微服务简述

本部分将简述每一个微服务的功能。

Register-login：处理用户登录。

User-detail：查看用户信息。

Get-invitation-I-sent：查看发送的邀请。

Pc member：查看PC成员。

Get-invitation：查收邀请。

Apply-meeting：申请会议。

Get-my-meeting-application：查看会议申请。

Invite-pc：邀请pc。

Start-submit：开启稿件提交。

Examine-manuscripts：chair开启审稿

Begin-examine-scripts：pc开始审稿。

Paper-submit：提交稿件。

Discussion：讨论。

Rebuttal：稿件未录用时反驳。

Release-first-result：发表初审结果。

Release-final-result：发表终审结果。

Modify-article-info：修改稿件信息。

End-examnie-manuscripts：结束审稿。

Confirm-finish：确认结束审稿。