华南理工大学
South China University of Technology

课程设计项目书

题目：银行运维与监控系统设计

学　院　经济与贸易学院
专　业　金融学(汇丰金融科技精英班)
学生姓名　王晓芙 向念 卢佑聪 范云杰 王阳妹
学生学号　201730711475
201730064168
201730272228
201730686216
201766031332
指导教师　左文明
课程名称　工程创新课程设计
起始日期　2020 年 5 月 1 日

目录

第一章 绪论

1.1 系统研究背景

大型银行有超过 1000 个不同的信息系统，同时支持成千上万的分行门店，还有过万的银行员工以及过百万的客户，同时，伴随着现代业务正逐步扩展，高科技技术被银行广泛运用，自助银行、网上银行、代理业务、电话银行、手机银行等都是银行发展出来的新的金融服务模式，快速解决客户业务故障、优化操作界面、保障网络信息安全、提取各类业务报表及数据、对银行业务人员进行全面的绩效考核及评估、保障 IT 设备的安全性等已成为商业银行面临的迫切问题。

在这种背景下，银行 IT 服务质量管理与监控系统以 IT 资源为管理对象，按照银行 IT 服务质量管理的实际要求，为银行业运营维护系统提供相关的流程化服务，包括监控面板管理、系统及网络拓扑管理、警告提示管理、资源管理，提供 IT 资源发现、面向业务的 IT 资源管理以及 IT 资源监控管理服务，来解决银行业各项迫切需求。但是 IT 系统的复杂性给银行信息管理系统的运营监控管理带来了巨大的困难，管理信息系统的稳定性要求对商业银行的 IT 部门压力也是越来越大，如何做好各系统以及实体资产间的运营以及监控，是下一步银行降低运营成本的关键所在。

正是由于银行网站和应用的增多及流量增大,银行需要持续增加服务器数量来维持服务性能,因此 IT 系统变得越来越复杂,使运维人员难以从容地处理各种各样的网络设备、中间件、业务系统等问题。尽管运维人员加班去部署和维护,也经常会由于服务器出现故障而导致业务无法继续,影响企业运行,从而带来巨大损失。出现这些问题的部分原因是银行缺少高效智能的服务器监控和诊断等 IT 运维工具,导致故障事件难以被高效、准确的处理。

根据银保监会发布的《中国银行业信息科技“十三五”发展规划监管指导意见》，监管部门对银行 IT 运维指出明确的发展方向 ：要进一步提升数据中心管理能力成熟度，强化运维管理体系建设，逐步实现自动化、智能化运维。企业要在流程、技术和人员管理三方面共同均衡发展，才能有效支撑数字业务目标。IT 运维需要进行实际转型，并引入 AIOps（智能运维） 作为支撑 IT 运维转型的核心。

1.2 银行运维体系现状

当前银行的运维方式存在一些局限性。

(1) 故障响应不及时。应用系统多，依靠传统的人工查看的方法，根本无法及时响应故障，往往是最终用户在使用中发现系统无法使用，才与管理员联系，管理员才开始着手解决故障问题。

(2) 排除故障耗时多。最终用户反映的故障问题，往往是针对具体应用系统而言，而产生故障的原因却是多种多样。

(3) 系统优化困难。没有一个强大的监控系统，系统优化只能依靠系统管理员的个人经验和猜测进行，而以这种方式进行优化，往往是很难成功的。

(4) IT 资产管理困难。过万的银行员工以及过百万的客户，每天不断产生的数据，

这些都需要技术精力去监控和管理。除了门店的服务器，还包括自助银行、网上银行、代理业务、电话银行、手机银行等发展出来的新的金融服务模式，还有众多的打印机等其他外设。

(5) IT 系统监管缺乏科学化以及流程化的管理，尚无统一界面；一般情况下，银行 IT 系统的管理人员需要来回在多个系统中进行切换，到目前为止还没有统一的管理步骤，从整体上看该系统的管理人员很难将 IT 系统的服务质量情况进行全面了解。

(6) IT 系统维护的工作成本巨大；高效管理方式的缺乏，必将引起 IT 系统维护成本的大量增加，这就导致系统维护费用经常超出预算范围。

针对以上当前运维的缺陷,需要有一种更高效,智能的运维技术。它可以大幅度提升运维效率和质量,包括监控的便捷部署,系统故障的及时发现和报警,还有故障根源的迅速定位,故障处理的智能决策。当今,人工智能技术已逐步成熟,并成功在各个领域中得到应用,在很大程度上节省了人力物力,提高了资源分配管控的效率,鉴于以上经验,可以把人工智能和 IT 运维结合起来,提高 IT 运维效率和质量,进一步减少运维中人为的操作,实现运维的无人化,完全自动化。

AIOps 又称为智能运维,它起源于 IT 运维，IT 运维分为三个阶段,分别为传统运维、自动化运维、智能运维。目前智能运维还处于初步探索阶段,近期在国外已经发起了面向智能运维的研究,但是在国内,大部分 IT 企业还处于自动化运维的阶段,一部分大型企业正在向智能运维的方向探索。

传统运维是 IT 运维的初级阶段,它是一种被动的,事后救援的方式。传统运维在系统出现问题后,相关部门联系运维工程师,运维工程师接收到信息后,利用 VPN 登录内网查看监控指标,利用自己的经验进行故障排查,花费大量时间定位故障,然后进行修复操作,最后故障恢复。传统运维方式存在以下问题:系统修复时间长;操作容易出错;随着业务增长,报警增多,无法及时处理,导致网站可用性下降。

自动化运维是 IT 运维的第二阶段,它建立在大量的自动化脚本产生的基础上，利用自动化脚本对系统和资源进行监测。自动化脚本通常设定一些阈值,当脚本检测到某个指标突破了设定的阈值,就会触发相关报警,接着自动化脚本会调用一些设定好的流程进行故障处理,如重启服务、重启主机等,以保证系统继续运行。自动化脚本也可以执行一些周期性工作,如病毒查杀、备份数据等,以减少人为操作环节,提高运维效率。尽管自动化运维相比传统运维减少了人为操作过程,在一定程度上提高了效率,但依然是一种不成熟的,智能化程度低的运维方式,如依然容易出现严重的漏报和误报,不能从根源上定位并解决故障,只能通过简单的重启方式恢复服务等。

智能运维的概念由 Gartner 在 2016 年提出,旨在使用大数据、机器学习等方法来提升运维能力,其目的是进一步降低自动化运维中人为干扰,最终实现运维无人化、完全自动化。随着现在人工智能技术的不断的发展,智能运维有望得以落

地,目前许多企业都在积极探索中。

1.3 市场分析

所以通过借鉴互联网的企业服务化理念，参考学习同业建设思路，基于大数据、人工智能、分布式数据库和图数据库等新兴技术，组织开展适合银行的智能运维体系研究，提出基于服务化的智能运维体系框架无疑是顺应市场和时代的需求的，因为智能运维有以下突出的优势：

1. 提升运维数据治理能力。运维大数据平台将原先分散在各监控管理平台的运维数据进行统一管理。在发生生产问题时便于进行不同维度运维数据之间的关联分析与排查，大大提高了问题解决的效率，减少了不同运维部门之间数据分散带来的时间成本消耗。

2. 提高生产系统的可用性。通过智能预警，数据中心运维保障人员对生产系统的运维模式由被动式故障抢修转变为主动式故障预判。对通过运维大数据平台发现的生产异常预警指标或者系统，相关运维人员可以提前介入进行重点监控、问题定位和故障排查，主动应对可能发生的生产事件。在一定程度上减少了生产问题实际发生的数量，提高可生产系统的可用性。

3. 显著提高数据中心的应急处理水平，降低问题的定位和处理时间。通过智能辅助定位功能，可以自动发现和定位问题根源，将原来问题的定位分析时间从数十分钟降低到数分钟。大大提高应急处理水平。

4. 显著降低人力成本。自动化运维可以提高数据中心的工作效率，而智能运维进一步提高了生产系统的预警能力和自动化的处理能力，从而进一步降低运维成本。尤其是随着分布式架构的推广，智能运维将使得数据中心人均运维的服务器数量大幅提高。

1.4 系统研究意义

1.业务发展的客观需要

为了充分发挥运维监控的预警作用，逐步提升各类运维监控指标的覆盖率和完备率，我们需要建立一套系统、规范、面向业务服务的运维监控指标体系。在管理层面，该指标体系旨在让企业管理者花更多的时间在决策上，而不是用于了解复杂、繁琐的 IT 细节上；从服务定义、服务水平管理、服务监 控、服务诊断的角度，让管理者一目了然；既满足企业要求的服务水平，确保最佳的业务系统表现，又辅助整个企业的业务运营与 IT 决策。

在技术层面，该体系既可以丰富开发新业务系统时的非业务功能需求，使开发团队在系统设计阶段，就把以后运维阶段需要关注的监控指标内嵌到应用系统中，起到 “未雨绸缪”的作用；又可以在老系统改造过程中增加指标的监控功能，起到“亡羊补牢”的效果；同时，该指标体系对于运维团队全面、有效地部署和配置各类运维工具也起到“有的放矢”的指导作用。

2.监管导向的客观需求

根据银监会《商业银行数据中心监管指引》第二十六条第八款“应集中监控重要信息系统和通信网络运行状态。采用运维监控工具，实时监控重要信息系统和通信网络的运行状况，通过监测、采集、分析和调优，提升生产系统运行的可靠性、稳定性和可用性。监控记录应满足故障定位、诊断及事后审计等要求。”为了满足上述 要求，迫切需要建立一套切实可行的运维监控指标体系来指导监控和分析工作，促进运维管理工作的系统化和规范化，降低运维风险。

3.智能化运维是运维工作发展的崭新阶段，它是大数据分析技术和人工智能等技术在运维工作中的具体应用。要持续做好智能运维，除了掌握金融科技新技术以外，培养良好的数据思维习惯是一个重要基础。从数据思维的角度，可以重新认知运维工作，理解现在为什么会发展到智能化运维阶段。业务领域的智能化经验和教训为运维领域提供了良好的借鉴。应用场景是智能运维的立足点和抓手，积极探索应用场景是研究和思考的方向。总而言之，金融科技为传统的运维工作注入了新活力，智能运维正当其时。

第二章 需求分析

金融产品的多元，银行市场的开放与业务服务的日益增长，不仅进一步增大了银行内系统的复杂程度，也对银行的风险控制提出了更高的要求。其中，系统风险的防范离不开银行内各系统的稳定运作与良好配合，现阶段对于各系统的运维监控，也将给银行科技部门带来更大的压力和挑战。

2.1 银行系统的现阶段特性

1. 银行系统体系庞大，依存度高。

首先，当今银行在规模化发展下会产生大量内部系统，如包含信贷操作与支付结算的核心业务系统；包含客户关系与人力资源信息的基础系统；包含柜面支付、网上银行与其他交易平台的渠道系统；以及与相关机构互为支持的外联系统（如反洗钱、征信系统等）……除此之外，根据银行 IT 架构的设计，还存在着如应用平台、数据中心、数据仓库等的支撑系统。

大量的银行系统不仅分别支撑着银行提供服务和内部管理的需要，也在很大程度上相互依存，互联互通。可以认为，银行系统的稳定是其服务可靠性的有力保障，因此作为银行的 IT 部门，在硬件层面对各系统进行全面把控无疑是至关重要的。

2. 银行系统标准不一，复杂度高。

其次，银行各系统往往有着不同的底层实现和组织架构：其硬件设备和网络服务可能来自不同的厂商；内部应用依赖于不同的操作系统；具体功能参照着不同的设计规范等等。对运维人员而言，这种不统一显然是巨大的负担，其表现可能为繁重的熟悉工作或更大的沟通成本。同时，这类现象的存在也不利于直观反映系统中的问题，因为标准的不明晰，相关部门可能难以及时发现可疑情况和故障。因此，为了降低银行系统整体的运维难度，减少运维成本，科技部门有必要实现各系统主要信息的统一化，运行状态的直观化和故障告警的自动化。

2.2 项目需求挖掘与描述

2.2.1 业务需求

通过对以上问题的分析，我们的项目需要实现一个能统一监控银行中所有系统运行状况的应用，同时该应用应满足下述需求：

1) 能实时监控所有系统的运行信息，并以统一的指标展示监控内容。
2) 监控的范畴涵盖服务器资源监控、网络环境监控、业务系统的中间件运行监控，其中具体监控指标见下述内容，详细介绍见【4. 功能介绍】。
3) 能够自动识别硬件故障和运行问题，并发送告警信息。

具体而言，该应用基于业务描述的需求如下：

1. 对各系统网络状况进行监控。

网络状况正常、网络性能良好是联网系统的运行基础与服务保障。银行的业务往往涉及交易记录、客户信息及大量机密内容，这一特性使其被要求具备高度

的数据传输可靠性与网络信息安全性。因此，网络自然会成为监控的重中之重。

该应用必须能稳定监控和精确反映所有系统相关设备的网络情况。目前初步考虑从网络设备、网络链路、网络流量与网络安全等多个方面展开，其相关指标都将在此应用中被统一检测。并且针对不同类型的设备，监控的形式也会有所侧重：对银行网点的各类自助设备，应做到对网络故障和网络性能层面的监控；对银行内部的较大型系统设施，还应在此基础上加以对网络配置和网络流量层面的监控；涉及到有较高网络安全性要求的系统，则必须重点关注网络安全方面的监控。除此之外，还应视严重程度与危害等级，制定不同级别的告警信号。

2. 对各系统服务器资源进行监控。

在系统运转与提供服务的过程中，服务器扮演着不可替代的角色。对于服务器，日常的运维既需要软件层面的及时维护，也需要硬件指标保持稳定正常；既要求机器内部的性能良好，也依赖外界环境提供一个可靠的工作保障。因此服务器的监控和维护应围绕硬件及操作系统各项指标与机房环境展开，相关指标大致如下：

- 监控指标：
  - ✓ CPU 监控；
  - ✓ 内存监控；
  - ✓ 磁盘监控；
  - ✓ 文件系统监控；
  - ✓ 进程监控；
  - ✓ 日志监控；
  - ✓ ……
- 机房环境:
  - ✓ 温度；
  - ✓ 湿度；
  - ✓ 电源；
  - ✓ 火警探测……

在银行中，不同系统的服务器选择的设备类型与供应商不尽相同，必然会导致人工监控和维护的难度极大。通过该监控系统，将所有的系统服务器关键指标进行整合，集中展示，更有利于 IT 人员的日常管理，从而保证银行的服务更优质，运营更高效。另一方面，这也对我们的监测应用提出了明确的兼容要求，来保证监控的准确和全面。

3. 对各系统中间件运行情况进行监控。

（https://blog.csdn.net/qq_38774360/article/details/84947760）
（https://www.docin.com/p-1149176872.html）

作为新层次的基础软件，中间件能够将不同的应用软件进行集成，并使其协同工作，而得以成为目前应用系统的有力支撑。在银行业务领域的整合异构平台、保障交易完整性等多个方面，中间件也有着出色表现，并被越来越多地应用于各系统中。

自然，我们的应用不能遗漏对于中间件的监控。在每个系统中，关于这部分的监控数据和展示将会放置在一个新的独立模块中，监控对象包括但不限于下述指标：

- JVM 监控
  - ✓ 堆使用
  - ✓ 垃圾收集
  - ✓ 死锁线程
  - ✓ 内存泄漏监测
- 线程池监控
  - ✓ 线程池使用
  - ✓ 超时线程或挂起线程
  - ✓ 线程池状态
- 数据库连接监控
  - ✓ 连接池使用
  - ✓ 连接失败
- 交易监控：

应用程序中的具体业务。一般需要评估 Java Transaction API 的健康状态并检查回滚的性质和出现率。

- ✓ JTA 状态
- ✓ 应用回滚
- ✓ 超时回滚

- JMS 监控：

Java 消息服务应用程序接口，属于面向消息中间件 API。两个应用需要通信时，可以利用 JMS 进行消息的转发实现应用间解耦。

- ✓ 等待字节
- ✓ 等待消息
- ✓ JMS 状态

- 应用程序监控
  - ✓ 部署应用状态
  - ✓ 应用请求异常
  - ✓ 应用请求超时
  - ✓ 请求响应时间
- EJB 监控
- ……

4. 自动告警需求。

该应用的研发旨在帮助 IT 部门更高效地对各系统硬件进行运维，但银行系统数以百计，每个系统所要关注的指标不一而足，如果只凭人工依靠监测面板去捕捉问题，这仍会是一个很大的工作量，而且难以获得良好的效果。

所以，自动告警功能的实现是刚需：该应用不仅需要监测所有系统的状况，还必须及时以警报形式向相关部门反映值得关注的问题——可能是故障、故障预警或异常的数据和用户操作。

通过自动告警，运维部门由被动响应变为主动防御，不但能极大地减轻人力负担，而且能有效降低故障率，减少维修成本，给银行的整体运营和对外服务带来提升。

此外，因为银行业务往往有很高的信息安全要求，所以自动告警的通知形式必须兼具时效性和保密性。普通警报的通知应做到渠道多样，响应迅速，具体而

言可以采用微信、钉钉和短信等移动端通讯进行；核心系统或严重警报的通知则尽量采用公司内部邮件+电话的形式进行，确保安全及时两不误。

2.2.2 功能需求

这一部分，我们将从功能性角度分解项目需求。该应用依据功能可以划分为三个模块：

1) 监控面板：提供数据可视化与故障用例报告等；
2) 资源管理：监控指标的分组和统一管理等；
3) 警告管理：各指标阈值设置，警告等级设置与通知方式设置等。

下面将一一介绍各部分的具体需求。

1. 监控面板管理

应用的监控面板将与操作人员产生直接交互。操作人员通过在监控面板进行选择和设置，可以随时查看任何系统的硬件参数、运行情况、告警历史和故障分析等。上述信息可以采用可视化或生成报表的方式直观展示。

2.资源管理

资源管理是对不同系统的各项指标进行有序，系统化的集中管理。在该功能的支持下，用户可以自行选择需要监控的指标，还可以设定**资源组**，对重要或相关联的数据集中监控，统一分析。详细介绍和功能示例见【功能分析】章节内容。

3. 警告管理

正如【业务需求】部分所介绍，应用在识别到可疑数据或故障时，应自动触发警报，从而实现监控的自动化，提高运维工作效率。警告管理功能的需求包括：合理的阈值设置、分级的警报信息以及恰当的通知方式。

阈值的设置直接关系预警功能的效果。我们需要应用在保证谨慎原则的基础上，尽可能减少误报的数量，提高预警正确率。具体的设置方法将在后续内容进行探究分析。

分级的警报有利于更细致地刻画警报信息，反映警告的严重程度和紧急性，从而做到更有效率地调配人员和资源进行处理。

通知的方式需在满足及时传达的前提下保障信息安全，杜绝一切隐私泄露的可能。具体操作见【业务需求】部分介绍，在此不做赘述。

4 巡检功能

想要提高运维工作的效率和质量，仅仅依靠对设备信息进行可视化和告警机制是不够的。如果运维人员在日常工作中缺乏对各系统、各设备较为全面的认知和了解，他将难以做到有针对性地监控和管理设备（因为设备数量大，要求长时间监控所有设备是不现实的）。为了让运维人员在监控过程中更具全局观和目的性——该应用需要为他们提供各系统/设备阶段性、总结性的数据汇总，即生成周期性的检查与报告。该功能在全自动的环境下实现，规律地为 IT 人员提供最新的、可靠的运维参考；也能为故障排除和修复提供对历史数据的回溯。

2.2.3 非功能需求

非功能性需求是从用户体验出发对应用需求的描述，也是在应用的开发中不可忽视的要素。该应用的非功能需求有如下几项：

1. 易用性

符合用户的习惯，适应用户的需要是对程序的易用性需求。因为该应用连接上百个系统，对应数十种硬件的管理，各种控件的排布和操作的流程会更为复杂。如果程序无法做到易学、易用，则难以实现真正的效率提升，甚至可能导致操作性风险。

大致的易用性要求包括：

1) 系统界面（如操作面板、资源管理面板）的风格统一，各界面入口和操作一致；
2) 提示信息与必要的操作说明到位；
3) 控件排列规律符合思维逻辑；
4) 数据支持定期保存与备份；
5) 有明显报警提示；
6) 支持自动软件升级；
7) ……

2. 可靠性

作为一款监控程序，其关乎整个银行的各系统运营保障，所以提出了很高的可靠性要求：即软件操作有高容错力；软件工作有高可靠度。具体而言，项目在实现过程中必须制定明确的质量标准和失效行为（根据具体的业务场景和需要进行设定），并通过划分开发过程、实施进度管理等办法对软件进行阶段性的评估。

大致的可靠性指标如下：

1) 有效性——衡量软件为用户提供服务的属性，如平均可用时间等；
2) 成熟型——衡量软件缺陷造成的故障等级与频度，如软件的系统缺陷率，平均故障时间 MTBF 等；
3) 故障承受能力——衡量故障对软件运行能力的影响；
4) 可恢复性——衡量遭遇故障后软件恢复工作性能的时长；
5) 可预测性——衡量软件的运行状态和行为在多大程度上可以被用户预测及把握，因为行为不可预测的软件意味着更大的操作难度和风险；
6) ……

软件的使命是为用户提供使用价值——这些价值应当是随时获得、稳定存在的。在规划时对软件可靠性充分地度量和考虑，在开发中始终坚持制定的标准和要求，如此方能实现软件的有用、好用、能用。

3. 性能与效率

软件的性能与处理效率、执行时间、响应速度等挂钩。以实时监测为核心功能，该应用显然是一个对时延和处理速度要求较高的软件，必须满足用户对性能和效率的需求。相关的考察指标如响应时间，相应速度，处理时间，延迟时间，吞吐量和负载等。

4. 安全性

安全性要求该应用能消除潜在风险并能够及时应对突发事件。对银行系统的监控软件而言，安全性漏洞无疑是致命的缺陷。因此在设计中应综合考虑内外部因素和各种异常条件，合理地设计、定义和分配各接口、各模块的功能，同时运用适当的网络安全技术，使突发事件出现时程序可以处于安全的状态，并最大程度地减小损失。

5. 保密性

在业务需求分析中，保密性已经被多次强调。银行作为信息安全高度敏感行业的代表，工作中的任一环节都应给予数据安全和隐私保护充分的重视。在应用的开发中，我们应从身份识别、身份验证、权限管理等多方面着手，最大限度保障保密性需求。

6. 可扩充性与可维护性

该应用在交付后，可能会面临功能的扩充或模块的调整，例如监控硬件设备的增加，新系统的开发与接入等情况，都要求开发成员在现有基础上对程序进行扩充。因此可扩充性保证了该应用对动态变化的需求的适应，以及新功能增加时带来的工作量和成本可以被接受。

类似的，应用的开发阶段结束后，不可避免会面对功能的优化或 bug 的处理，一款具有高度可维护性的软件能让开发者更便捷地测试和分析软件存在的问题，并得出改进的方案。相关属性包括可验证性（软件内部可视），可分析性（结构良好，代码可读），可变性（架构合理，耦合度低）等等。

可维护性与可扩展性都是支撑软件适应变化，生命长青的重要性能。事实上，它们不仅在产品的交付后才发挥作用，也会影响开发本身的效率和效果，因此也是非功能性需求的核心部分。

第三章 功能概述

3.1 资源管理功能：

一、资源信息管理：对主机、网络设备、应用、基础服务等资源进行监控。

1.资源发现：

资源发现可用于各系统中主机、网络设备、应用、基础服务等未被监控设备，进而可将发现的资源通过加入监控功能进而加入监控界面。

2.资源浏览：

资源浏览功能对所有监控对象实行统一管理，主要以列表的显示方式进行浏览，同时资源浏览功能中还可以进行资源检索，键入关键字快速定位于关键字相关的资源。其中列表的分类以设备类型为分类依据或以系统名称为分类依据，当纳入系统足够多时，以所处地为分类依据为佳。

3.增添资源:

添加监控对象主要是将监控代理收集到的监控信息，根据需要添加到资源监控对象列表。添加网络设备主要根据网络设备的监控采用 SNMP 协议，通过对该协议参数配置，将监控对象添加到资源监控列表。

4.加入监控：

该功能允许用户对资源监控对象列表中感兴趣的系统或设备资源等加入用户自身监控列表，使不同用户可以选择监控不同资源对象。当用户登录后可直接在监控面板中查看用户监控的所有对象信息。

5.取消监控：

该功能允许用户对自身监控列表中不再感兴趣的资源进行取消监控，取消监控后的资源在用户监控面板不再显示。

6.删除资源：

该功能允许拥有管理权限的用户删除资源列表中被停用、更换、淘汰的资源对象。

二、资源组管理：

该功能允许是用户可以自行选择一组资源组成资源组作为监控对象，对于用户来说，资源组是多种资源的集合，它们可以属于同一个系统或同一个分行。通过建立一个资源组，可以把与某一系统或某一分行相关的各种资源组合起来，统一管理，便于用户了解一组相关联的资源的运行状态。

1.新建资源组：

允许用户新建资源组，便于用户对感兴趣且相关联的资源整合起来统一管理，也便于用户将某一系统所涉及到的资源整合起来统一监控管理。

2.从资源组中移出资源

该功能允许用户对资源组列表中不再感兴趣的资源进行取消关联监控，从资源组中移出的资源在用户资源组监控面板中不再关联显示。

3.从资源组中添加资源

该功能允许用户向资源组列表中添加新的资源，当某一系统增添了新资源时，便于用于向已建好的资源组中添加该资源。

4.删除资源组：

允许用户删除资源组信息。

三、资源策略管理：

对资源策略和告警规则进行相应的维护，可以针对资源模型，控制资源模型的指标采集集合，同时控制指标的采集频度，控制性能指标出发报警的阈值，控制告警的发送、接收人及发送条件，控制告警升级的策略。

1.策略维护：

此功能用于用户自定义监控资源的警告触发策略，用户可以针对资源模型控制资源模型的指标采集集合，同时控制指标的采集频度。策略的制定需拥有相应权限。

2.阈值管理：

此功能用于修改触发策略的阈值、读取阈值等操作。当指标超过规定的阈值时触发告警。阈值的设置范围只能在该指标的数值范围内进行设置，阈值在设置时需要指定数值单位，防止数值因单位不同出现判断错误。阈值的规则制度需要由拥有相应权限的人员制定。

3.警告发送管理：

此功能用于当指标超过阈值或达到策略告警条件时，向用户发送告警，可以选择发送告警的方式、接收人及频率，同时用户可查看历史告警，处理后的告警能在历史告警中查看。

业务流程图：

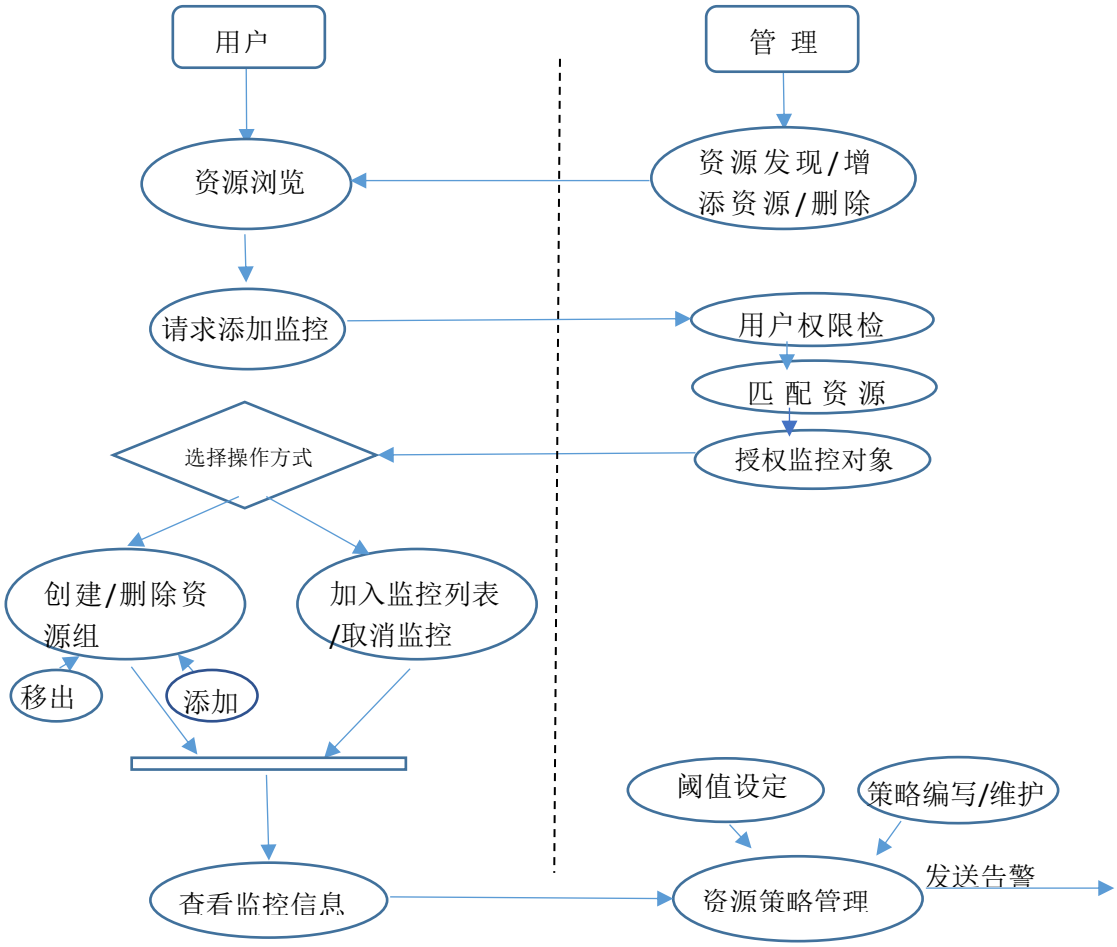


图 1 资源管理流程图

四．实现效果：

在资源管理页面可以查看已加入监控的设备，查看该这杯型号，地址。

然后可以将列表中的设备加入监控，则这些监控中的设备在监控面板可见且可以看到实时标签数据。

3.2 监控面板管理功能

一、查看监控参数

该功能是系统对一系列的监控对象设置的监控参数，包括但不限于监控各个服务器的硬盘剩余空间、CPU 利用率、内存利用率等信息。（此功能与资源管理中的加入监控相对应，用户可以查看加入监控的资源所拥有的各种指标信息：硬盘剩余空间、CPU 利用率、内存利用率、等）。对于需要实时更新的指标信息，需做到实时动态更新。

对监控指标信息，用户能进行如下操作：

1.新建报表：

允许用户新建资源报表，在用户监控对象中的指标单独或整合生成一份资源使用报表。允许用户在生成报表中，引入不同系统的同一资源指标信息进行横向对比，也可以进行同一资源指标信息在历史时间上的纵向对比。

2.保存报表：

允许用户将新建的资源报表保存，在需时可以登录后浏览

3.导出报表：

允许用户将新建或已有的报表信息导出图片或用户选择的相应文档

4.删除报表：

允许用户删除已存在的报表

二、自选资源

与资源管理中的加入监控相似，用户选择需要监控的资源，可以将用户重点关注的资源添加到自选资源列表，用户登录系统后就可以直接查看这些资源的运行情况，同时用户可以将监控对象的数据图表保存，然后在监控面板的已存图表中显示。

三、最近警告

用户可以在此功能块下查看监控系统中最近发出的警告信息（包括时间， 警告定义，警告来源，警告级别等）与资源策略功能相配合使用，当触发资源策略告警阈值时发送响应警告。

在此功能块下用户能进行以下操作:

1.解除警告：

对于发出的警告信息（与资源策略管理相对应），当用户处理警告信息后，可以在此手动解除警告，

2.浏览历史警告：

允许用户在此浏览发出过的所有警告及警告接收方等各项信息。必要时可以添加删除功能。

监控指标中，完整监控指标信息见项目需求部分-业务需求
其中部分指标如下：

1.对各系统网络状况进行监控

2. 对各系统服务器资源进行监控。

- ✓ CPU 监控；
- ✓ 内存监控；
- ✓ 磁盘监控；
- ✓ 文件系统监控；
- ✓ 进程监控；
- ✓ 日志监控；
- ✓ ……

3. 对各系统中间件运行情况进行监控。

- ✓ JVM 监控；
- ✓ 线程池监控；
- ✓ 数据库监控；

业务流程图

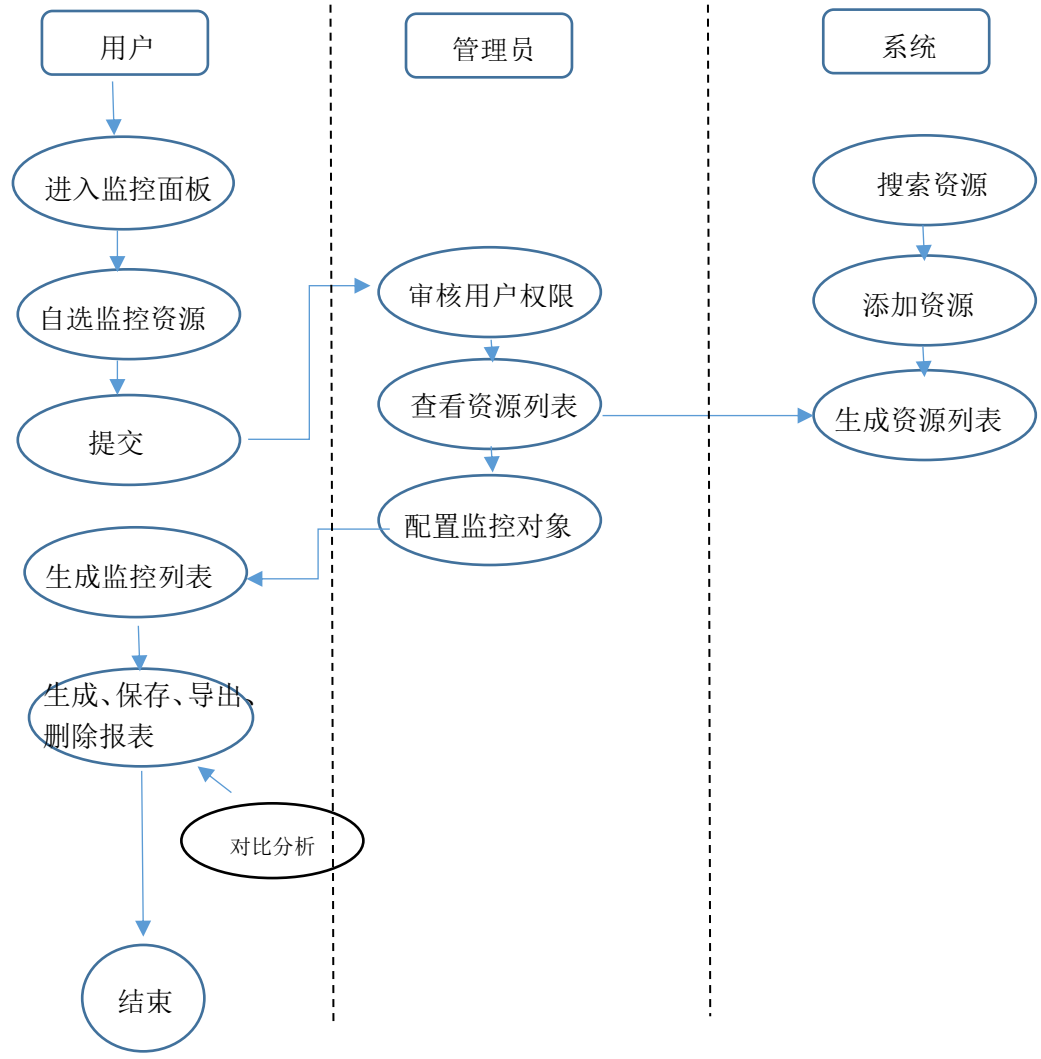
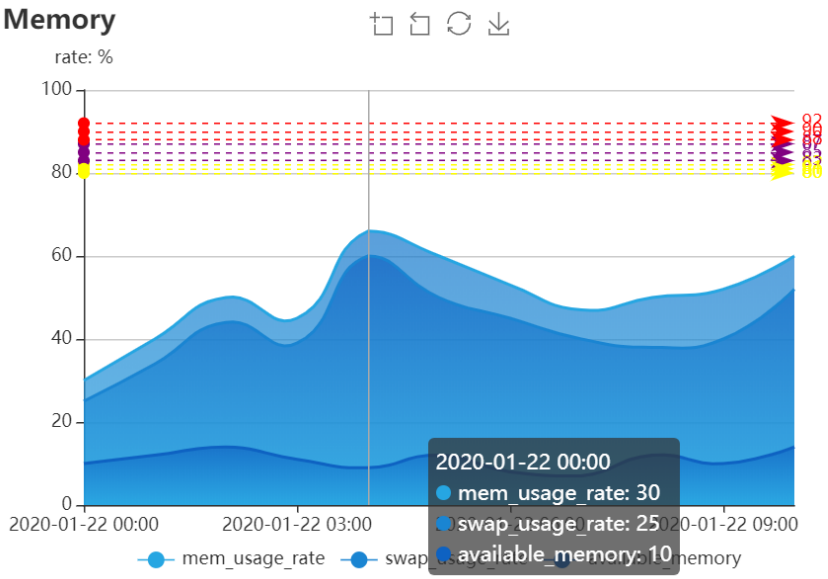


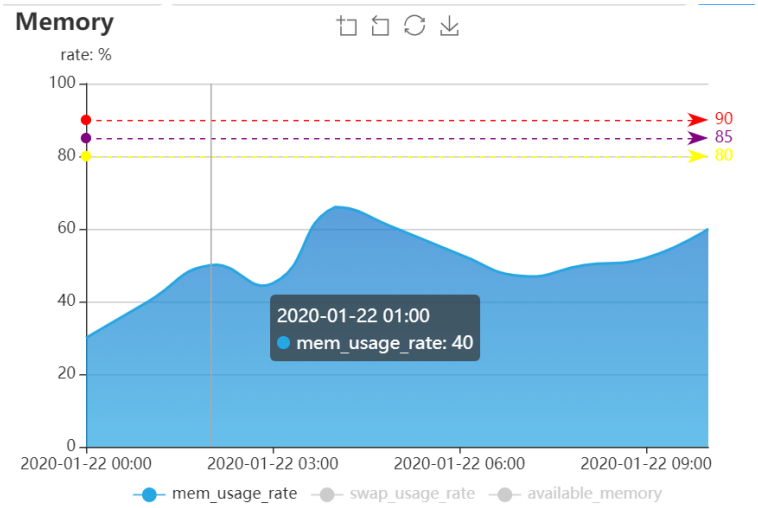
图 2 监控管理流程图

四．实现效果

用户通过选择 IP,和时间段展示设备相应的历史参数数据。
通过改变 IP 能直接查看相应设备的实时参数数据。

另外监控面板还可以看到已经设置的 IP 的对应指标的警告条件。不同等级的警告条件用不同颜色代表（红色：一级；紫色：二级；黄色：三级）




Memory
rate: %
100
80
60
40
20
0
90
85
80
2020-01-22 01:00
mem_usage_rate: 40
2020-01-22 00:00
2020-01-22 03:00
2020-01-22 06:00
2020-01-22 09:00
mem_usage_rate
swap_usage_rate
available_memory

3.3 警告管理功能：

警告功能的作用是银行对设备，网络的异常状态的警告进行定义，管理，处理。具体而言通过不同设备的日常工作状态进行监控，当设备参数超过某一设定的阈值则发出相应的级别的警告，之后用户要对警告进行相应的处理。

功能设计：

一. 警告定义：

包含警告发生条件，警告级别，警告信息的设置。警告发生条件取决于设备运行参数的阈值。当警告发生根据发生的条件发出相应级别的警告。

1. 警告发生条件：根据设备日常运行的参数数据以及异常发生时的数据设置异常发生的参数阈值作为异常警告发生条件。
2. 警告级别设置：根据不同大小的阈值将警告经行级别的区分，以区别资源异常的严重程度。
3. 警告信息设置：警告信息根据警告事件编辑而成，可以设置警告事件 ID 方便查询异常的详细信息。

二. **警告确认：**

当银行收到警告后，通知到相应的通知人员。通知人员收到警告后需要实地确认设备是否真实发生异常，若是则对该警告信息进行确认。

二. 警告处理：

当警告得到确认，相关警告处理人员对警告经行分析然后落实处理，处理后将状态信息反馈，确保系统知道警告已经被处理，不再重复发出警告。

业务流程图;

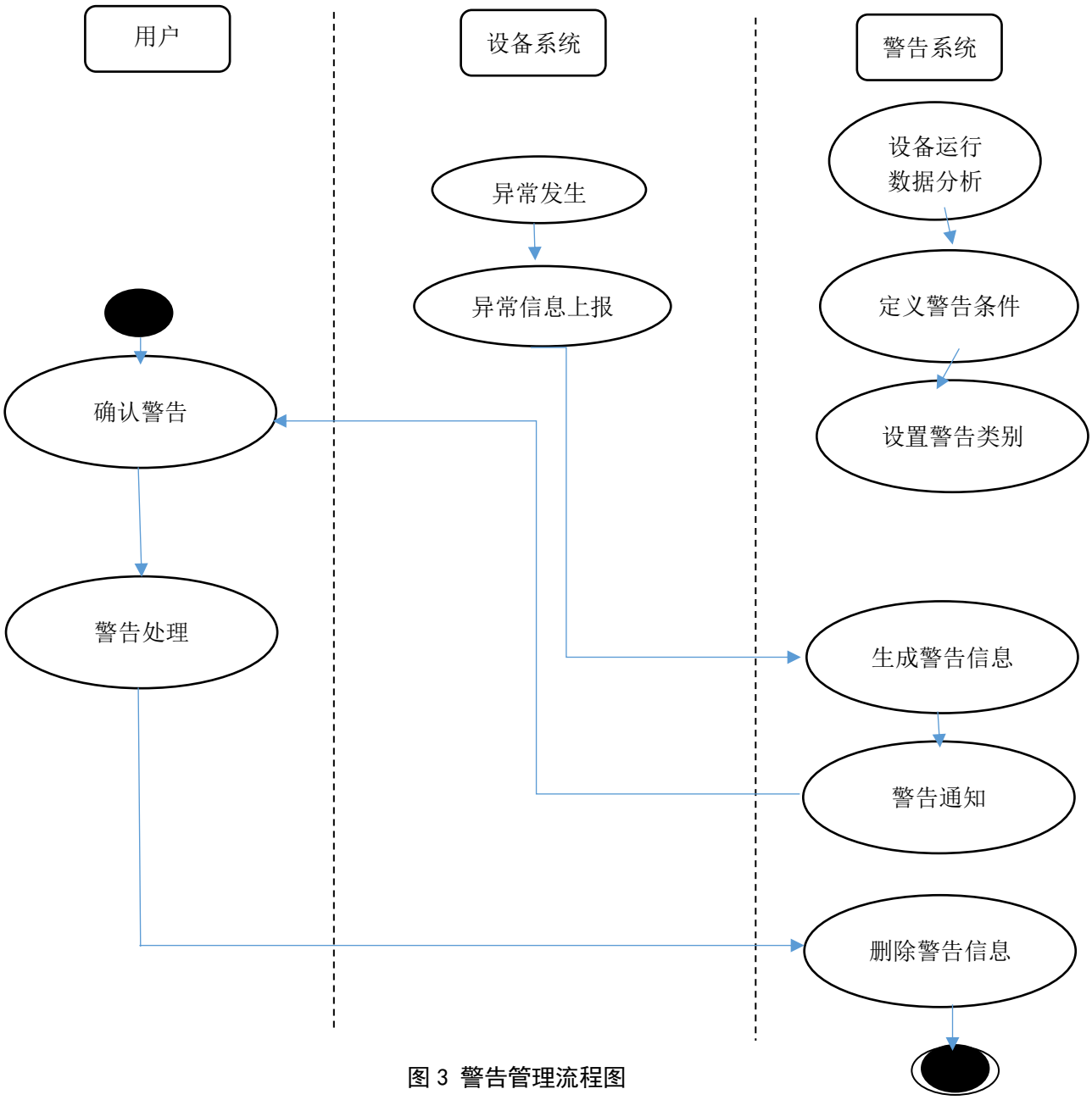
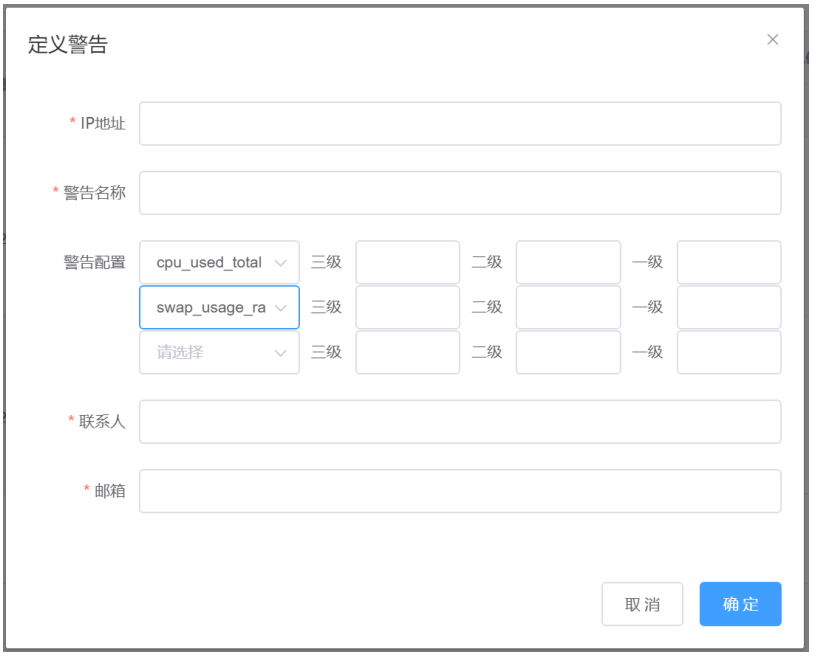


图 3 警告管理流程图

三．实现效果：

警告定义是根据现有设备，9 个指标各 3 级警告进行设置的。设置完后，效果可以在监控面板上查看。另外还可以对警告条件进行增删改的操作。

	警告名称	警告配置	接收人信息		操作
			姓名	邮箱	
1	警告定义1 ip地址：52.82.112.115	CPU_used_total: 三级警示线：73% 二级警示线：83% 一级警示线：93% mem_usage_rate: 三级警示线：74% 二级警示线：84% 一级警示线：94% dev_io_read_rate: 三级警示线：75% 二级警示线：85% 一级警示线：95%	张三	xxxxxx@xxx.com	
2	警告定义2 ip地址：52.82.136.163	CPU_used_system: 三级警示线：64% 二级警示线：74% 一级警示线：84% swap_usage_rate: 三级警示线：65% 二级警示线：75% 一级警示线：85% dev_io_write_rate: 三级警示线：66% 二级警示线：76% 一级警示线：86%	李四	xxxxxx@xxx.com	



3.4 巡检功能

设备的日常运行需要定时周期性地检查并作报告记录，方便对资源组进行升级。同时通过日常的检查可以及早发现问题，避免警告，异常的发生。另外统一的巡检报告管理可以方便发生问题时通过追查历史数据帮助解决问题。巡检功能包括巡检的具体设置，巡检报告的管理

功能设计：

（1）巡检报告管理：包含报告的录入，查看，导出和删除。允许有权限的用户新建巡检报告或者调出以前的历史报告进行查看和管理

（2）巡检模板设置：可以对巡检报告的名称，巡检的周期，巡检时间等进行规定。之后的用户巡检也只能用特定的报告模板记录信息。

业务流程图：

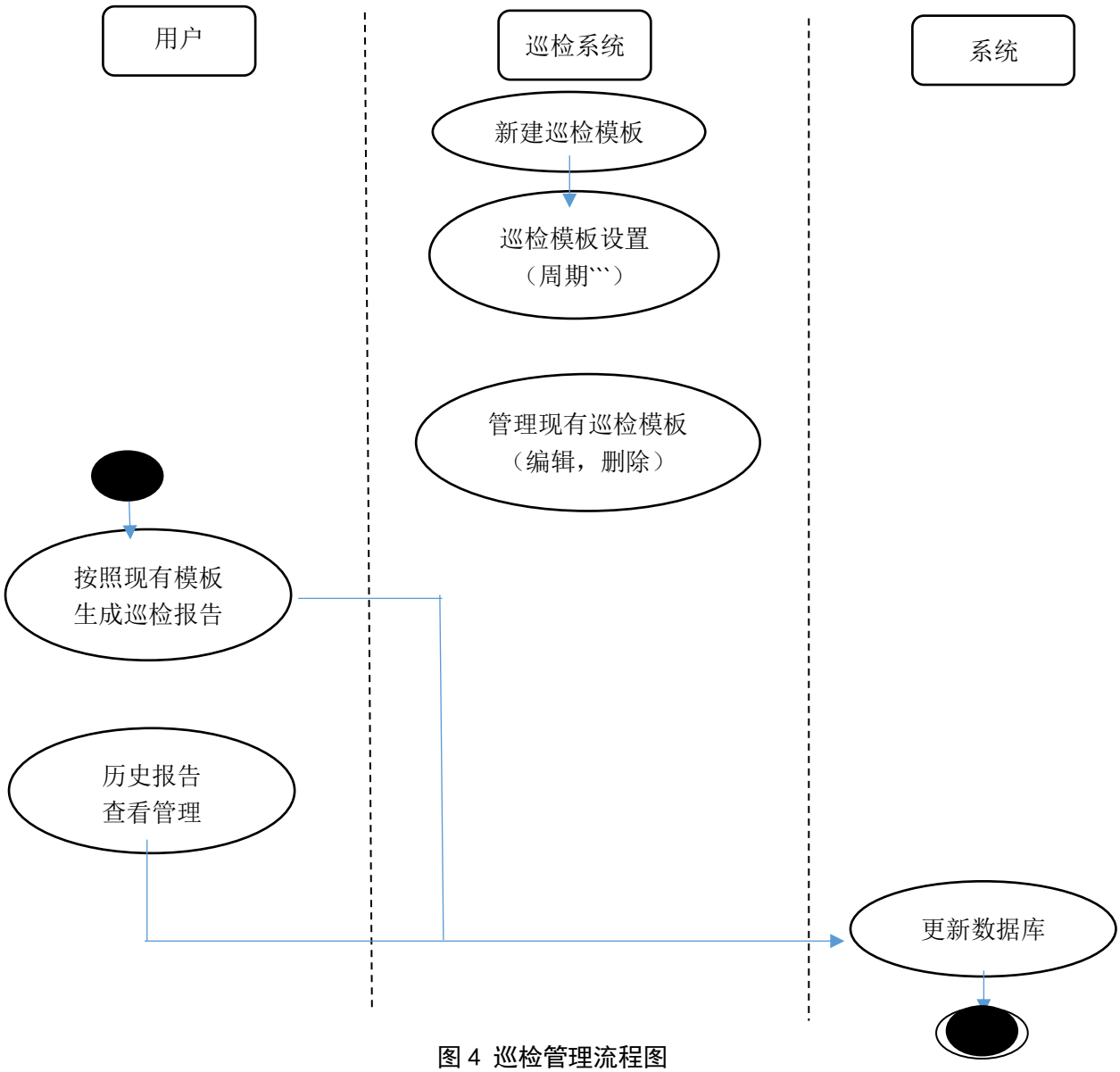


图 4 巡检管理流程图

3.5 系统权限管理：

系统不同的用户具有不同的权限，属于不同的角色。而系统权限管理可以对用户的访问权限进行设置，通过赋予不同用户不同的访问权限和角色以区分不同用户能使用的功能，从而保护系统的安全运行。

功能设计：

（1）系统用户管理：包含用户账号的创建，账号信息的编辑，账号的删除。

（2）权限管理：角色的创建，修改和删除。当确定哪些用户需要哪些权限后，将权限的集合赋予角色，然后将角色赋予对应的用户。或当某个用户需要特殊的权限时，可以单独地将某一权限赋予用户。

业务流程图：

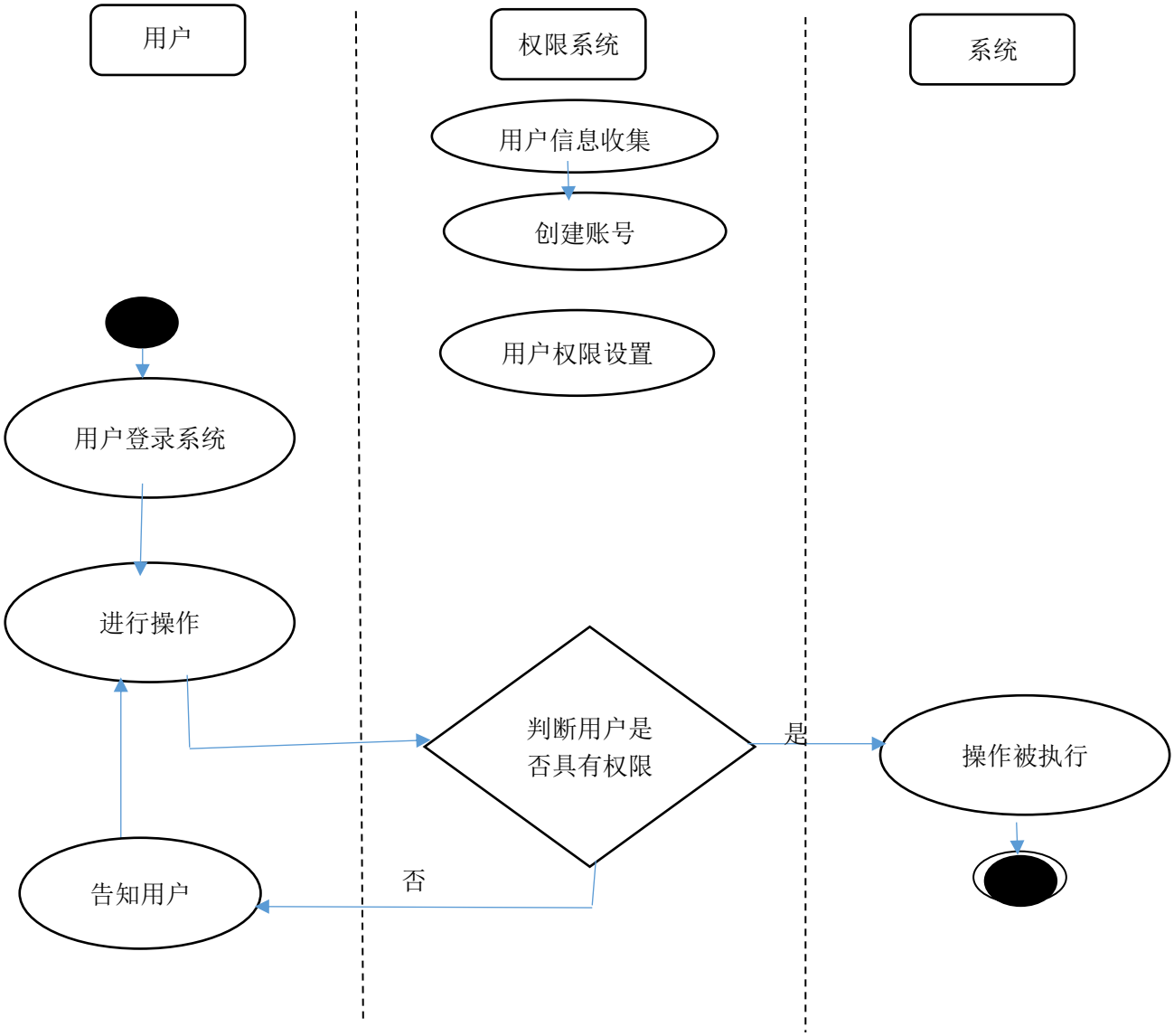


图 5 权限管理流程图

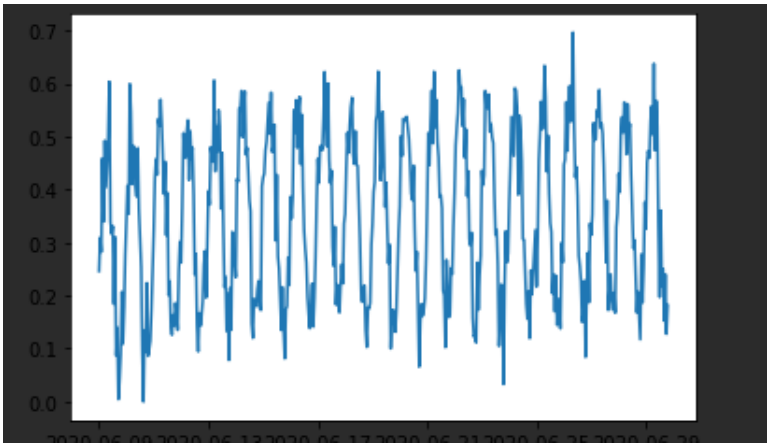
3.6 预测模块：

在机器学习模块中我们的目标是根据历史数据，预测出未来一段时间的 kpi 指标，同时配合资源管理与警告功能进行提前预警，以达到智能运维的目的，我们首先生成出 500 个含一定规律的数据，包括了趋势，周期性和噪声项，模拟 kpi 指标，用于训练预测模型。我们采用的是 facebook 的预言家库来进行拟合，模型包括了描述线性增长或逻辑增长的非周期性变化函数 g，描述周期性变化的因子 s，描述非规律性的节假日效应 h，和误差项 e，

原数据函数如下：包含噪声，趋势项，和波动性，模拟了某项 kpi 指标在一定条件下的规律性

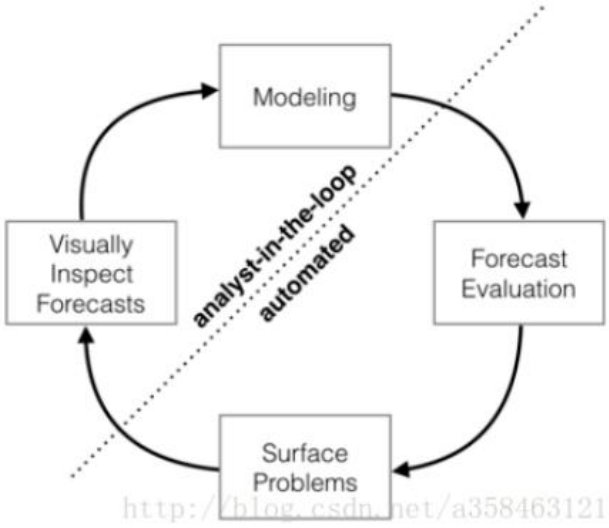
```
(0.2*math.sin(math.pi*(i/12))+np.random.normal(0.2,0.05)+math.log2(i+1)/50)
```

图像如下：



时间序列数据预测库：prophet：

整体框架：



上图是 prophet 的整体框架，整个过程分为四部分：Modeling、Forecast Evaluation、Surface Problems 以及 Visually Inspect Forecasts。从整体上看，这是一个循环结构，而这个结构又可以根据虚线分为分析师操纵部分与自动化部分，因此，整个过程就是分析师与自动化过程相结合的循环体系，也是一种将问题背景知识与统计分析融合起来的过程，这种结合大大的增加了模型的适用范围，提高了模型的准确性。按照上述的四个部分，prophet 的预测过程为：

1. Modeling：建立时间序列模型。分析师根据预测问题的背景选择一个合适的模型。
2. Forecast Evaluation：模型评估。根据模型对历史数据进行仿真，在模型的参数不确定的情况下，我们可以进行多种尝试，并根据对应的仿真效果评估哪种模型更适合。
3. Surface Problems：呈现问题。如果尝试了多种参数后，模型的整体表现依然不理想，这个时候可以将误差较大的潜在原因呈现给分析师。
4. Visually Inspect Forecasts：以可视化的方式反馈整个预测结果。当问题反馈给分析师后，分析师考虑是否进一步调整和构建模型。

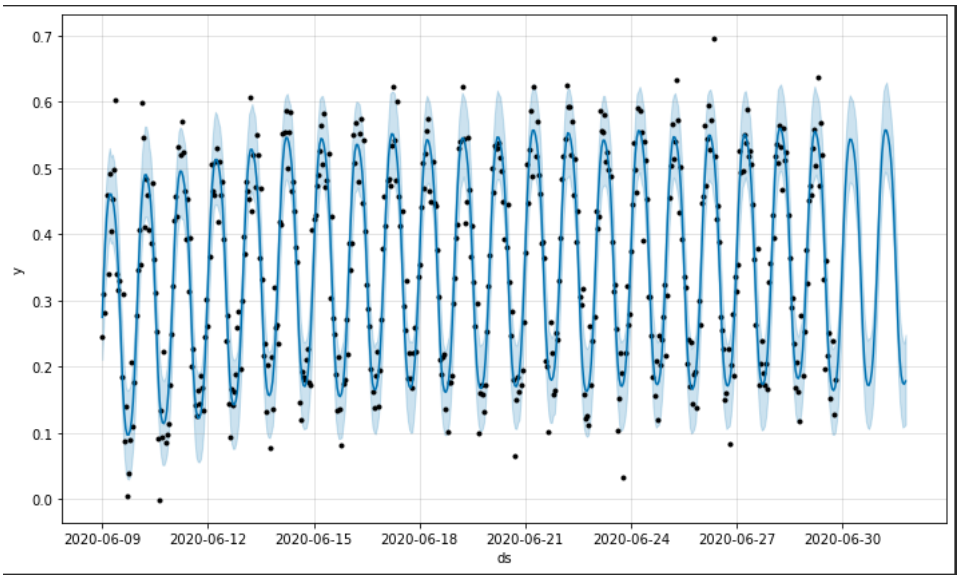
模型原理：

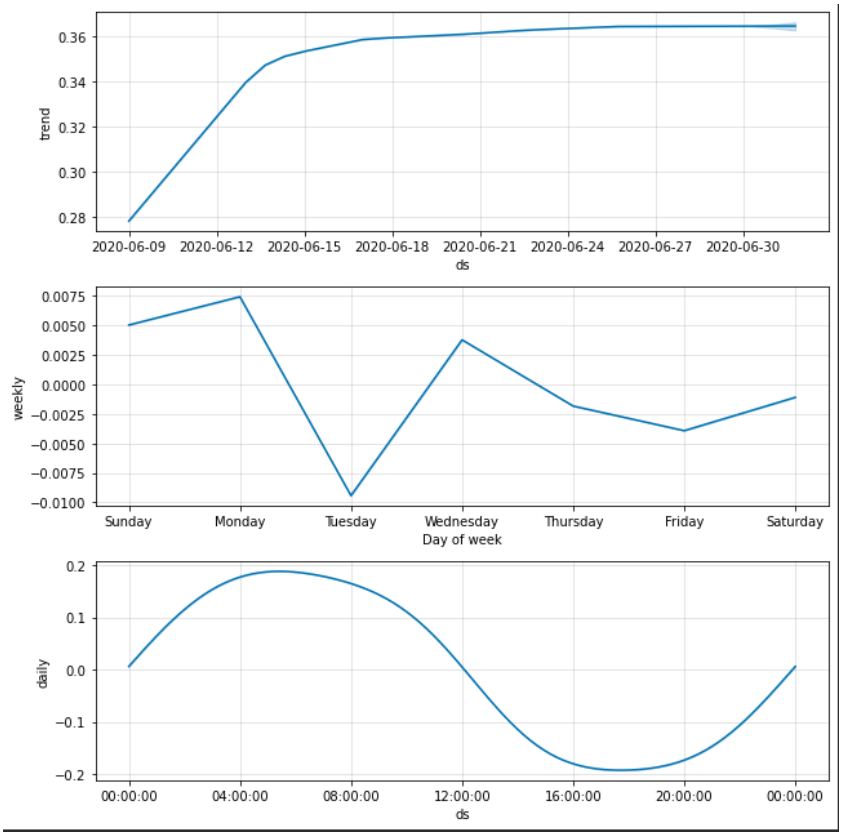
时间序列模型可分解为三个主要组成部分：趋势，季节性和节假日。它们按如下公式组合：

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

- **g(t)**：用于拟合时间序列中的分段线性增长或逻辑增长等非周期变化。
- **s(t)**：周期变化（如：每周/每年的季节性）。
- **h(t)**：非规律性的节假日效应（用户造成）。
- **et**：误差项用来反映未在模型中体现的异常变动。

预测结果：





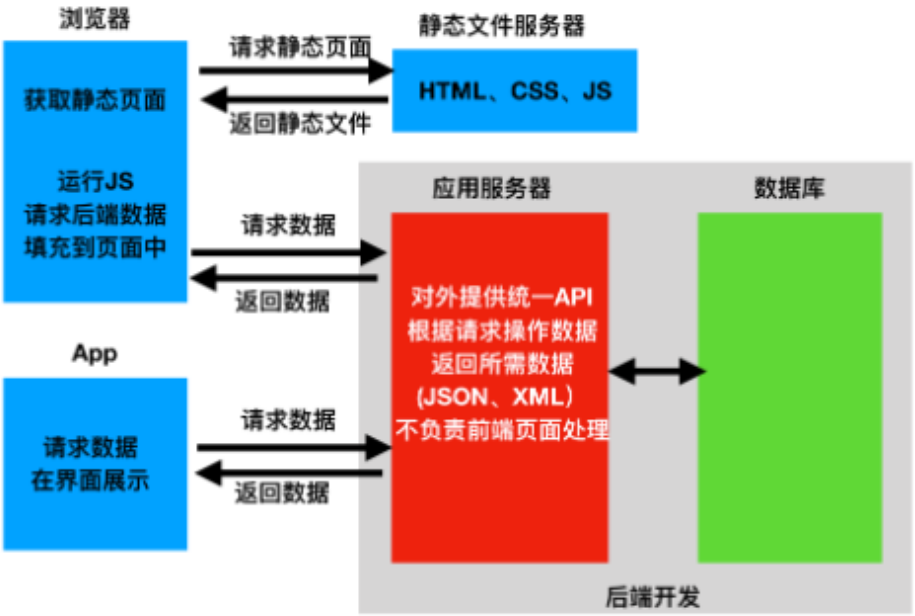
该模型功能强大，不需要提前告知拟合函数类型就能对时间序列数据预测，我们调好参数后对原数据进行训练，可以看到拟合的结果十分符合原数据曲线，并且在右边的成分图中，很好的描述了数据的趋势和周期性，有助于资源的提前规划和监控的提前预警

部分实现代码：

```
# print(dic)
df = pd.DataFrame(dic)
dic['ds']=df['ds'].values.tolist()
# 构建模型
m = Prophet(changepoint_prior_scale=0.3)  # 这个参数要调好
m.fit(df)

# 构建预测时间，这里的时间间隔设置为小时
future = m.make_future_dataframe(periods=48, freq='H')

# 使用模型进行推后48个小时的预测
forecast = m.predict(future)
# print(df)
# print(forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']])

predict={}
predict['ds'] = forecast['ds'].values.tolist()
predict['yhat'] = forecast['yhat'].tolist()

alldata['data'] = dic
alldata['predict']=predict
# 绘图查看结果
m.plot(forecast)
# 成分图
m.plot_components(forecast)
#print(alldata)
```

第四章 软件架构

基于 Django 和 HTML 的 Python WEB 框架:



前端框架：Vue.js（渐进式框架，自底向上增量开发，易与其它库或项目整合）

后端框架：Django（强大的数据库功能和后台功能）

数据库：SQLite3（性能稳定，小巧快速

4.1 MVC 和 MTV 模式

MVC 模式将应用程序分解为三个组成部分，其各自的功能如下：

M (Model 模型)——是应用程序中用于处理应用程序数据逻辑的部分。

通常模型对象负责在数据库中存取数据

V（View 视图）——是应用程序中处理数据显示的部分，通常视图是依据模型数 据创建的，负责把数据格式化后呈现给用户

C（Controller 控制器）——是应用程序中处理用户交互的部分，通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据。

在 Django 框架中，控制器（Controller）接受用户输入的部分由框架自行处理，在 Django 框架的模式中应用程序则被分为 MTV 三个部分：

M (Model 模型)——负责业务对象与数据库的对象(ORM)

T（Template 模板）——负责如何把页面展示给用户

V （View 视图）——负责业务逻辑，并在适当的时候调用 Model 和 Template

Django 的 MTV 模式本质上与 MVC 模式并没有什么区别，只是定义上由些许不同，主要目的都是为了各组件间的松耦合关系；MCV，MTV 模式的应用，使得软件有着耦合性地，重用性高，生命周期成本低以及部署迅速

等优点，十分适合银行运维监控系统。

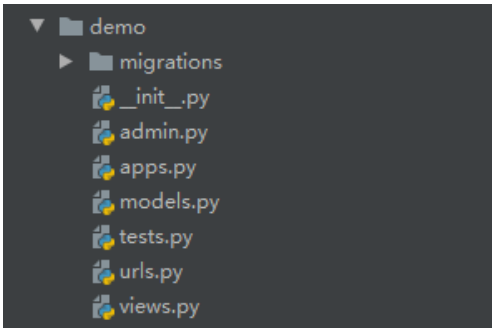
4.2 前后端分离架构

在开发架构上，本项目采用前端（HTML）+后端（Django）+数据库（MySQL）的模式，前端能处理大部分数据，在规模较大的银行系统数据处理时，减小对服务器的压力，同时后端能着重于处理业务逻辑，同时接口的复用率高，软件可扩展性强；在人员分工方面也能做到前后端专攻，提高软件质量。

4.3 Django 框架

Python 的 WEB 框架有 Django、Tornado、Flask 等多种，Django 相较与其他 WEB 框架其优势为：大而全，框架本身集成了 ORM、模型绑定、模板引擎、缓存、Session 等诸多功能。

Django 程序目录：



在配置文件方面，银行运维监控离不开数据库对大量数据的处理能力，Django 支持 SQLite 3（默认）、PostgreSQL 、MySQL、Oracle 数据库的操作：

```
#默认SQLit3配置
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

```
# MySQL的配置

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME':'dbname',
        'USER': 'xxx',
        'PASSWORD': 'xxx',
        'HOST': '',
        'PORT': '',
    }
}
```

Django 路由系统：URL 配置，本质是 URL 模式以及要为该 URL 模式调用的视图函数之间的映射表，在 urls.py 中对 URL 进行函数调用与管理：

```
from django.conf.urls import url
from . import views#导入views

urlpatterns = [
    #aaa$代表以aaa结束，views.index是关联对应views.py中的一个函数
    url(r'^aaa$', views.index, name='index'),
]
```

在 Django 框架中，通过 http 请求产生 HttpRequest 对象和 HttpResponse 对象，当请求一个页面时，Django 创建一个 HttpRequest 对象包含原数据的请求。然后 Django 加载适当的视图，通过 HttpRequest 作为视图函数的第一个参数。每个视图负责返回一个 HttpResponse 目标。

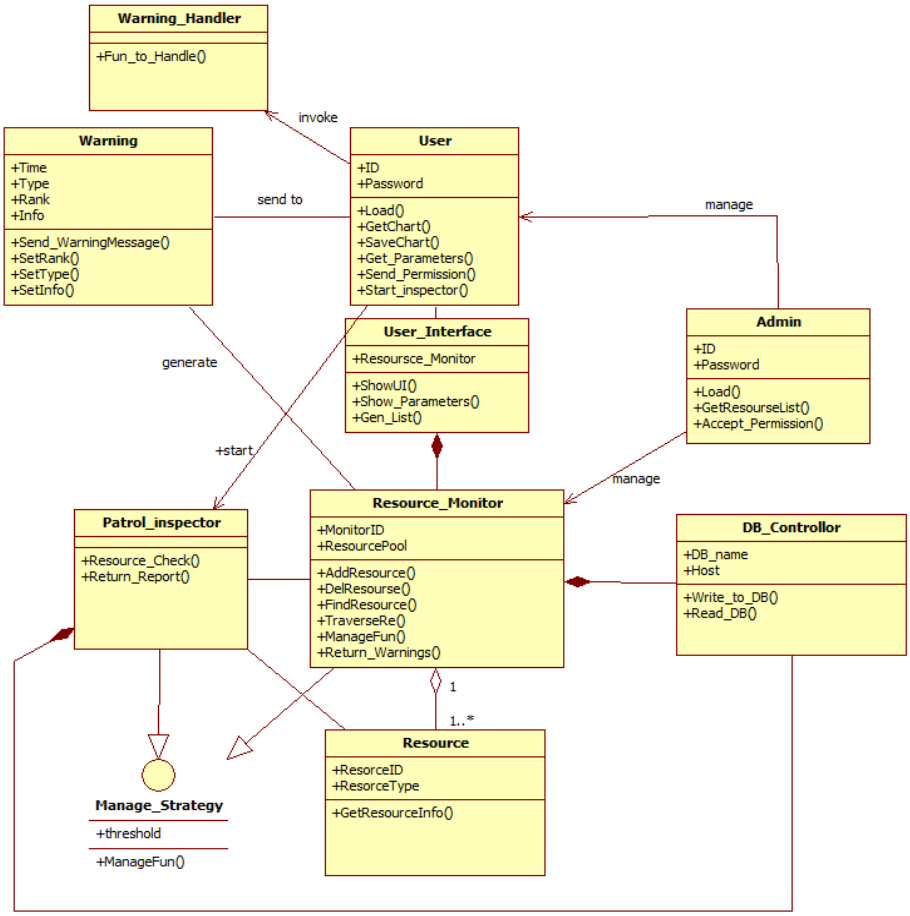
此外，Django 还有对模板（Template）的创建与执行，模型(Model)的建立与管理的过程，通过模型来构建和管理 WEB 应用程序中的数据，模板用于读取模型中获取的数据返回给用户。

模型创建一个表的基本结构：

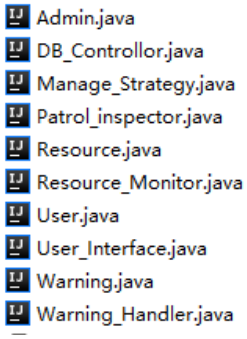
```
from django.db import models


class userinfo(models.Model):
    name = models.CharField(max_length=30)
    email = models.EmailField()
    memo = models.TextField()
```

模型主要负责与数据库进行交互，进行创建，修改等操作，Django 中遵循 Code Frist 的原则，即根据代码中定义的类来自动生成数据库表，除了上述的基本结构外，还有连表结构中一对多，多对多，一对一等数据库表类型，用于不同的应用场景，同时还可以对表进行增删改查等操作。

4.4 Uml 类图：



4.5 根据类图生成代码（部分）：



（生成的 java 类文件）

4.5.1 资源监控器：包含监控器 id 以及资源池，同时有对资源的添加，删除，查找，遍历等函数，每个资源监控器配备了数据库控制器，对监控信息进行读写，并继承了管理策略接口，使得调度方案更为灵活。

```java
//  Generated by StarUML(tm) Java Add-In
//
//  @ Project : Untitled
//  @ File Name : Resource_Monitor.java
//  @ Date : 2020/5/10
//  @ Author :
//
//
public class Resource_Monitor extends Manage_Strategy {
    public Object MonitorID;
    public Object ResourcePool;
    public void AddResource() {

    }

    public void DelResourse() {

    }

    public void FindResource() {

    }

    public void TraverseRe() {

    }

    public void ManageFun() {

    }

    public void Return_Warnings() {

    }
}
```

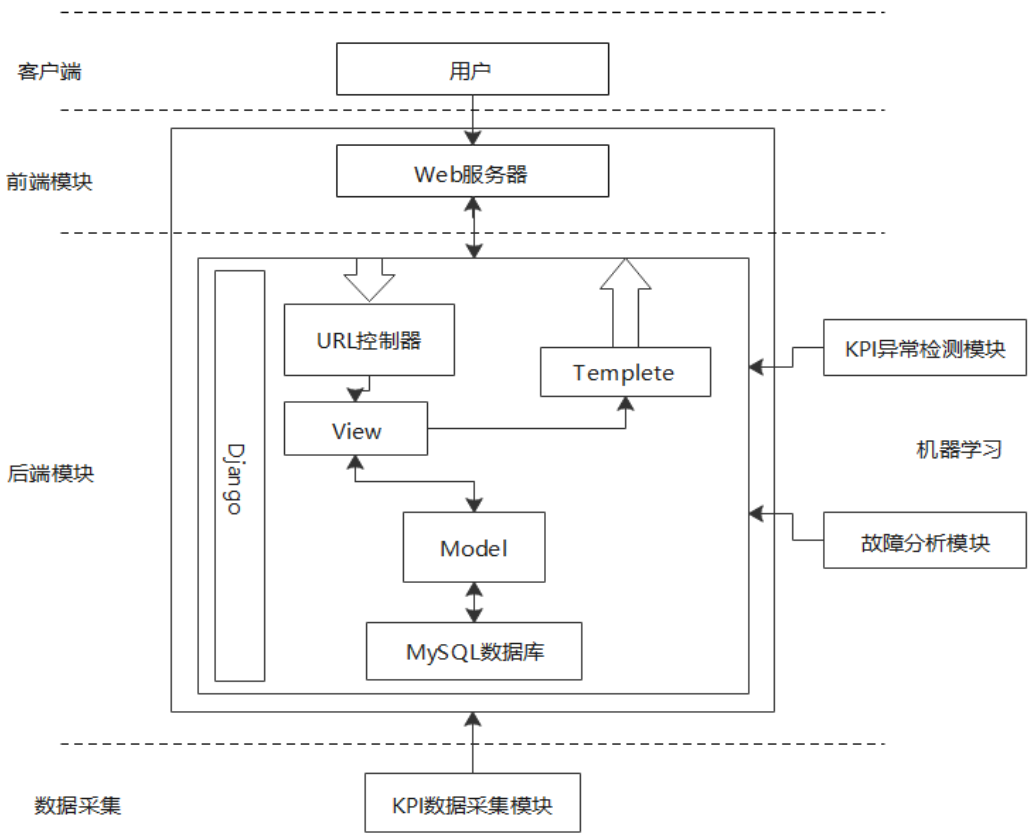
4.5.2 用户类：包含账户与密码，登陆时使用，同时定义了用户的基本操作如查看参数，生成与保存图表，设置巡检器等。

```java
//
//
public class User {
    public Object ID;
    public Object Password;
    public void Load() {

    }

    public void GetChart() {

    }

    public void SaveChart() {

    }

    public void Get_Parameters() {

    }

    public void Send_Permission() {

    }

    public void Start_inspector() {

    }
}
```

4.5.3 警告类：警告由资源的监控过程产生，基本参数包括了时间，警告类型，警告等级，警告信息，反馈给用户，由用户进行下一步的解决操作。

```
//  @ Author :
//
//


public class Warning {
    public Object Time;
    public Object Type;
    public Object Rank;
    public Object Info;
    public void Send_WarningMessage() {

    }

    public void SetRank() {

    }

    public void SetType() {

    }

    public void SetInfo() {

    }
}
```

整体架构：



4.6 数据库设计

数据库设计是在 DBMS 基础上针对具体的应用对象，构建合适的数据库模式，以满足用户的业务需求为目的，建立数据库和应用系统，有效的存储和存取数据。本系统利用 SQL 语言对系统数据进行操作，提供存储、维护、检索数据的功能，同时可以利用关系数据库的海量数据管理、事务处理、并发控制、记录锁定、数据仓库等功能。

4.6.1 数据模型

该监控系统实现对后台数据库的设计，数据模型是对系统实体类关系的映射，描述系统数据库设计的常用工具是 E_R 图，系统几个关键表关系如图所示。

4.6.2 数据库表结构设计

在关系数据库中，数据库表是一系列二维数组的集合，用来代表和储存数据对象之间的关系。数据库表的创建是数据库设计最基本的问题，数据库表作为存储信息系统数据的最佳方式，其设计要具有较好的使用性。设计出的表要以减少数据冗余，确保数据的准确性为宗旨。合理的数据库表设计对于提高数据库的性能有很大的作用。IT 设备运维监控系统部分数据库表结构设计如下。

表 3-1　监控设备类型表

字段描述	字段名	数据类型
服务器	Server	varchar
文件系统	File System	varchar

网络设备	Network DV	varchar
端口	Port	varchar

表 3-2　服务器监控信息表

字段描述	字段名	数据类型
服务器名	Sever Name	varchar
服务器 OS 类型	Server OS Type	varchar
IP 地址	IP address	varchar
默认网关	Default gateway	varchar
首选 DNS	Prime DNS	varchar
CPU 速度	CPU Speed	double
CPU 个数	CPU Num	Integer
内存大小	Memory size	Integer
文件系统个数	File system num	Integer
网卡个数	Nic num	Integer
描述	Description	varchar
CPU 利用率	CPU ratio	double
内存利用率	Memory ratio	double
空闲内存（KB）	Free memory	double
交换分区（KB）	Exchange partion	double
可用性	availability	varchar

表 3-3　文件系统监控信息表

字段描述	字段名	数据类型
文件系统类型	File System Type	varchar
容量	Capacity	Integer
剩余空间	Free Space	Integer
利用率	Use Ratio	double

表 3-4　网络设备

字段描述	字段名	数据类型
描述	Description	varchar
最大传输单位	Largest Transmission Unit	varchar
IP 地址	IP Address	varchar
广播地址	Broadcast Address	varchar
物理地址	Physical Address	varchar
网络掩码	Network Mask	varchar
接口标志	Port Flag	varchar
入流量	Import	Integer
出流量	Output	Integer
可用性	Availability	varchar

表 3-5　端口

字段描述	字段名	数据类型
描述	Description	Varchar
带宽	Bandwidth	Varchar
最大传输单位	Largest Transmission Unit	Varchar
IP 地址	IP Address	Varchar
物理地址	Physical Address	Varchar
网络掩码	Network Mask	Varchar
入流量	Import	Integer
出流量	Output	Integer
管理状态	Management	varchar
运行状态	Running State	Varchar
可用性	Availability	varchar

表 3-6　系统用户

字段描述	字段名	数据类型
用户 ID	User ID	varchar
用户姓名	User Name	varchar
性别	User Sex	varchar
民族	Nation	varchar
出生日期	Date of Birth	varchar
级别	Level	varchar

表 3-7　警告信息

字段描述	字段名	数据类型
警告 ID	Warning ID	varchar
警告名称	Warning Name	varchar
警告级别	Warning Level	varchar
报警方式	Warning Type	varchar
报警条件	Warning Condition	varchar
警告人	Warning People	varchar