**疫情期间人员活动上报分析系统**

**体系结构设计文档**

**<2.0>**

**<2020.12.15>**

**<Jayce>**

**<学号：2018333333>**

**<班号：201866666>**

**北京邮电大学软件学院**

**2020年秋季学期**

目录

[修订历史 3](#_Toc58433659)

[文档批准 3](#_Toc58433660)

[**1.** **介绍** 3](#_Toc58433661)

[**1.1** **目的** 3](#_Toc58433662)

[**1.2** **范围** 4](#_Toc58433663)

[**1.3** **功能简述** 4](#_Toc58433664)

[**1.4** **术语定义** 5](#_Toc58433665)

[**1.5** **参考文献** 5](#_Toc58433666)

[**2.** **体系结构需求** 5](#_Toc58433667)

[**2.1** **关键指标** 5](#_Toc58433669)

[**2.1.1体系结构** 5](#_Toc58433670)

[**2.1.2基于本系统** 6](#_Toc58433671)

[**2.2** **体系结构用例** 7](#_Toc58433672)

[**2.2.1用例说明** 7](#_Toc58433673)

[**2.2.2用户用例图** 8](#_Toc58433674)

[**2.2.3具体功能时序图** 10](#_Toc58433675)

[**2.3** **各相关方对体系结构的要求** 12](#_Toc58433676)

[**2.4** **约束条件** 13](#_Toc58433677)

[**2.4.1 实现约束** 13](#_Toc58433678)

[**2.3.2其他约束** 13](#_Toc58433679)

[**2.5** **非功能性需求** 14](#_Toc58433680)

[**2.5.1性能表现** 14](#_Toc58433681)

[**2.5.2可靠性** 14](#_Toc58433682)

[**2.5.3可用性** 15](#_Toc58433683)

[**2.5.4密安性** 15](#_Toc58433684)

[**2.5.4可维护性** 15](#_Toc58433685)

[**2.5.5可移植性** 16](#_Toc58433686)

[**2.6** **设计需遵循的标准** 16](#_Toc58433687)

[**2.6.1实用性原则** 16](#_Toc58433688)

[**2.6.2稳定性原则** 16](#_Toc58433689)

[**2.6.3复用性原则** 16](#_Toc58433690)

[**2.6.4扩展性原则** 17](#_Toc58433691)

[**2.6.5安全性原则** 17](#_Toc58433692)

[**3.** **体系架构实现方案** 17](#_Toc58433693)

[**3.1** **相关的体系结构模式** 17](#_Toc58433697)

[**3.1.1三次B/S体系架构简述** 17](#_Toc58433698)

[**3.1.2系统架构介绍** 18](#_Toc58433699)

[**3.2** **结构化视图** 19](#_Toc58433700)

[**3.2.1系统体系架构设计** 19](#_Toc58433701)

[**3.2.2负载均衡器(Nginx)** 20](#_Toc58433702)

[**3.2.2系统体系架构视图** 21](#_Toc58433703)

[**3.3.3模块级体系结构** 22](#_Toc58433704)

[**3.3** **行为视图** 23](#_Toc58433705)

[**3.3.1体系结构视角** 23](#_Toc58433706)

[**3.3.2逻辑视角** 23](#_Toc58433707)

[**3.3.3实现视角** 24](#_Toc58433708)

[**3.3.4部署视角** 25](#_Toc58433709)

[**3.3.4用例视角** 25](#_Toc58433710)

[**4.** **系统的质量分析和评价** 26](#_Toc58433711)

[**4.1** **场景分析** 26](#_Toc58433713)

[**4.1.1用例场景** 26](#_Toc58433714)

[**4.1.2增长性场景** 26](#_Toc58433715)

[**4.1.3探索性场景** 27](#_Toc58433716)

[**4.2** **风险** 27](#_Toc58433717)

[**4.2.1技术风险** 27](#_Toc58433718)

[**4.2.2进度风险** 27](#_Toc58433719)

[**4.2.3 质量风险** 28](#_Toc58433720)

# 修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **描述** | **作者** | **备注** |
| 2020/12/2 | 版本1.0 | 徐龙 | 体系结构文档框架设计 |
| 2020/12/9 | 版本2.0 | 徐龙 | 完成体系结构文档 |

# 文档批准

以下需求分析报告已经被以下机构人员批准并认可：

|  |  |  |  |
| --- | --- | --- | --- |
| **签字** | **打印姓名** | **标题** | **日期** |
|  | 徐龙 |  | 2020/12/2 |
|  | 徐龙 |  | 2020/12/9 |

1. **介绍**

在这一部分，主要介绍了本体系结构文档编写的目的、范围、相关概念定义、以及所使用的参考文献。软件体系结构设计属于高层设计文档，是符合现代软件工程要求的概要设计。

* 1. **目的**

本文档是对软件的体系结构进行多方面的设计之后编写的文档。该文档对系统的用例视图、逻辑视图、进程视图和体系结构视图进行了详细的设计。旨在为读者展示该系统的总体结构，包括逻辑设计、物理结构，分析系统的体系结构需求，包括约束条件、系统规模、性能指标、质量需求等，并使用多种视图给出体系结构设计的解决方案并分析建模，最后进行体系结构的质量分析。

此体系结构设计文档是进行软件项目设计开发的基础，也是编写测试用例和进行系统测试的主要依据，它对开发的后续阶段性工作起着指导作用。同时此文档也可作为软件用户、软件客户、开发人员等各方进行软件项目沟通的基础。方便了开发过程中不同人员间的交流，提高了软件系统开发的效率。

本文档的预期读者对象为：

1. **开发人员：**可以通过本文档了解该软件系统预期的功能以及相应的体系架构，并以此为指导对系统进行设计和开发。
2. **测试人员：**可以通过此文档选择与该系统体系结构相适应的测试模型与流程，针对性地进行系统测试。
3. **维护人员：**可根据本文档知道如何对本系统进行维护并在实际应用场景下，进行相关参数的调整。
4. **项目管理人员：**则能根据本文档评估软件系统开发大致的时间流程并制定详尽的开发计划。
   1. **范围**
5. **软件系统的名称：**疫情间人员活动上报分析系统（PARRS）
6. **文档内容：**本体系结构设计文档分析了PAARRS系统的在体系结构上的功能、阐述了系统的总体架构，说明了系统的总体设计策略，给出了体系结构设计的解决方案并分析建模，最后进行体系结构的质量分析和评估。
7. **应用范围：**本软件体系结构设计文档适合于PAARSE系统的总体应用结构，目的是满足系统的质量和可信赖性要求，以及此系统未来的维护、运行和升级改造等要求。
   1. **功能简述**
8. **市民：**
9. 市民可以登陆系统上报个人信息，例如过去24小时的行程信息、个人健康状况等；
10. 市民可以查询自己的核酸检测结果，历史上报情况，周围地区安全度、病例数量等信息。
11. 当检测结果为阳性时，系统主动提示市民防范措施，并通过网络开始24小时获取市民地理位置。
12. **医护人员：**
13. 医护人员可以登陆所属的医院系统，上传市民的核酸检测结果，医院内确诊病例信息。
14. **CDC人员：**
15. 能够查看医生和市民上报的数据，并根浴已有成熟的数据模型对疫情进行预测，提出不同的预报模型。
16. 通过医院上报的确诊病例信息，获取其在14天内的行程范围，计算出密切接触者名单，接触时长，以此为基础确定感染的概率，并打印出报表，构造传染范围预测图。
17. **决策者：**
18. 能够查看所有的疫情上报数据。
19. 根据CDC人员在系统中上传的预报模型和报表，对不同的区域做出应对的措施。
20. **系统管理人员：**
21. 可以管理用户的账号和密码，修改程序bug。
22. 管理员可以后台查看用户的个人信息，尤其是行程、个人健康状态等信息；
    1. **术语定义**

|  |  |
| --- | --- |
| **术语** | **定义** |
| PARRS | 疫情间人员上报分析系统 |
| CDC | 疾病控制中心 |
| C/S架构 | 客户端/服务器结构 |
| B/S架构 | 浏览器/服务器结构 |
| 三层B/S架构 | 三层B/S体系结构是在B/S结构的基础上，在服务器和客户端层增加了一层结构，称为中间件，使整个体系结构成为三层。 |
| 功能需求 | 功能需求是对未来软件的服务功能进行的描述。功能需求更关注系统的输入、加工（业务流程）直到输出的情况。 |
| 非功能需求 | 非功能需求是指功能以外的需求描述，主要是系统的一些关键的特性，例如可靠性、响应时间、存储空间、I/O 设备的吞吐量、接口的数据格式等。 |
| 用例图 | 用例图是指由参与者（Actor）、用例（Use Case）以及它们之间的关系构成的用于描述系统功能的视图。 |
| ATAM | （Architecture Tradeoff Analysis Method）体系结构折中分析方法。这是 SEI 开发的通过手工评估和确认体系结构的方法。 |

* 1. **参考文献**

1. 王安生.《软件工程化》[M]. 北京：清华大学出版社，2014
2. 李书杰,李志刚.B/S三层体系结构模式[J].河北理工学院学报,2002(S1)
3. 戴伟. 基于Nginx高性能Web服务器的理论研究与性能改进[D].南京邮电大学,2019.
4. 张丽,张艳.从C/S到B/S,再到三层(多层)结构——论体系结构的发展[J].河南师范大学学报(自然科学版),2002(03):24-27.
5. **体系结构需求**
   1. **关键指标**

**2.1.1体系结构**

软件体系结构为软件系统提供了一个结构、行为和属性的高级抽象，由构成系统的元素描述、元素相互作用、指导元素集成的模式以及这些模式的约束组成。一个良性的软件体系结构应该有以下五个质量要素：

1. 体系结构应是适宜的。
2. 体系结构应是概念完整的。
3. 体系结构应是易于维护和升级的。
4. 体系结构应是便于移植的。
5. 体系结构应是理性化的。

这五个质量要素体现了体系结构作为早期设计决策对系统需求的支持、实现的约束，管理的组织。

这五个质量要素是比较抽象的，我们从需求、开发、项目管理三个角度细化体系结构的指标，可以划分出以下的表格：

表2‑1 关键指标分类

|  |  |
| --- | --- |
| 角度 | 包含指标 |
| 需求角度 | 功能性  可靠性  可用性  安全性 |
| 开发角度 | 可维护性  可扩展性  可移植性  可测试性  可重用性 |
| 项目管理角度 | 部件无关  风险性  可度量性 |

**2.1.2基于本系统**

本系统主要面对市民、CDC人员、医护人员和决策者，致力于帮助政府快速获取确症病例信息，确定密切接触者及其活动轨迹，及时阻止疫情扩散，对数据进行分析，做出贴合实际的决策，降低整个社会的活动成本。

同时由于本系统将涉及大量个人数据，因此要对系统中的各类人员进行系统权限设置，例如市民之间不能相互查看其他人数据，系统管理人员在获取上级许可后才能修改数据，另一方面此系统面向范围为有千万人口的大城市，所以要求系统不仅要有足够高的安全性还要有强大的稳定性。由于涉及到政府对疫情的决策措施，所以要求对密切接触者确定计算、感染概率计算、行程图获取的误差一定要降到极低的水平，这样才能保证才能保证软件正常、快速、高效地使用，所以本系统在性能上应当具备以下关键指标：

1. **响应时间：**

在同时有100万名市民上报信息的情况下，系统响应时间不得大于2s。

在同时有100名CDC人员要查看的系统数据的情况下，系统响应时间不得大于3s。

在同时有 500名医护人员上传病人信息及核酸检测信息的情况下，系统响应时间不得大于2s。

1. **对处理数据量的要求**

对数据的处理量在TB级别时，系统的计算时间应当在20s之内。

所以本系统在可靠性上应当：

1. **出错率**

在累计1亿次服务请求的情况下，其出错率不得高于 10 次。

在累计20亿次服务请求的情况下，其出错率不得高于 30 次。

1. **稳定性**

在100万用户的并发操作下，连续24个小时工作的情况下，系统不会出现宕机的情况。

1. **易恢复性**

在停机进行恢复之后损失的数据不得大于1 MB。

在白天停机之后恢复工作的时长不得多于1H。

在晚上停机之后恢复工作的时长不得多于2H。

* 1. **体系结构用例**

**2.2.1用例说明**

本说明书的用例是从使用者的角度出发，描述用户所期望的与整个运行系统的交互。根据《软件需求分析说明书》对本系统用户需求的分析以及对用例角色的陈述，本体系结构说明书将用户的需求与系统进行交互，设计体系结构用例如表1

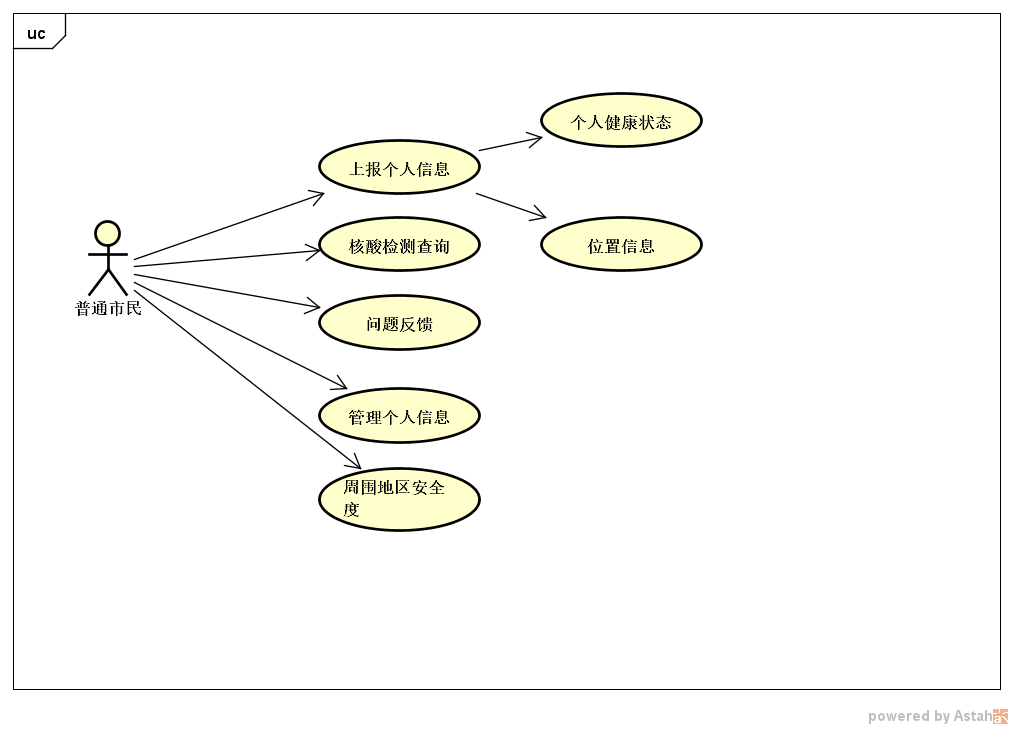
|  |  |
| --- | --- |
| 角色 | 用途 |
| 软件开发人员 | 1. 清楚知道软件的整体结构。 2. 清楚知道各模块间的接口，方便编写。 3. 清楚知道自己所要完成的功能。 |
| 产品经理 | 1. 知道软件系统的规模。 2. 根据体系结构安排工作给各开发小组。 3. 决定开发策略。 4. 规划工期进度安排 |
| 测试人员 | 1. 明确系统的性能指标。 2. 明确系统的组织结构。 3. 编写测试用例的依据。 |
| 甲方人员 | 1. 对该体系结构文档给出评价。 2. 验证其是否能满足自己的需求。 3. 项目验收的标准。 |
| 市民和其他用户 | 1. 对该体系结构文档提出修改建议。 |

表1-1体系结构表

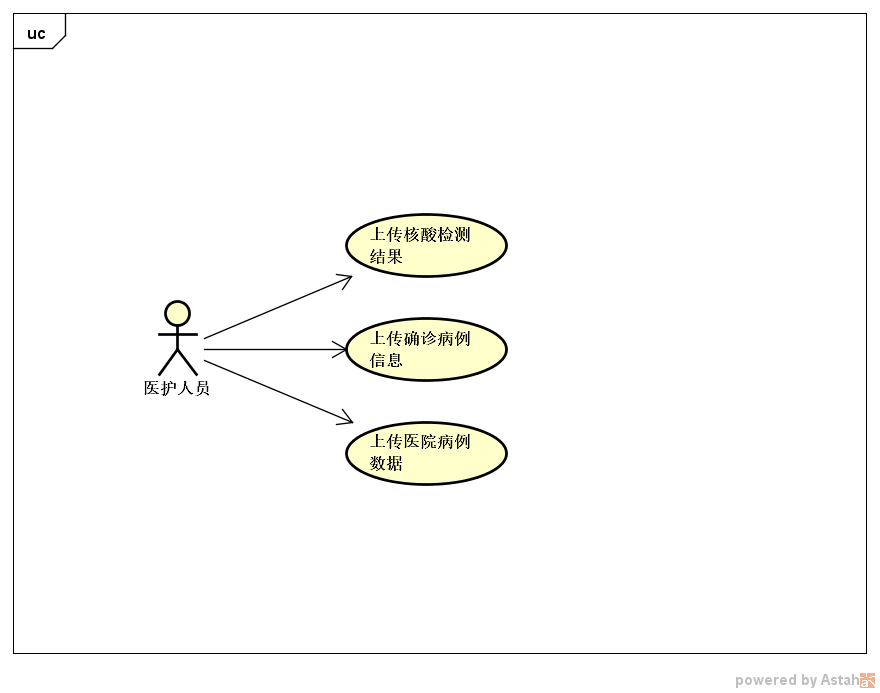
**2.2.2用户用例图**

PARRS系统共有五种用户角色：市民、医护人员、CDC人员、决策者、系统管理员，下面本文将针对五种用户画出它们的用例图。

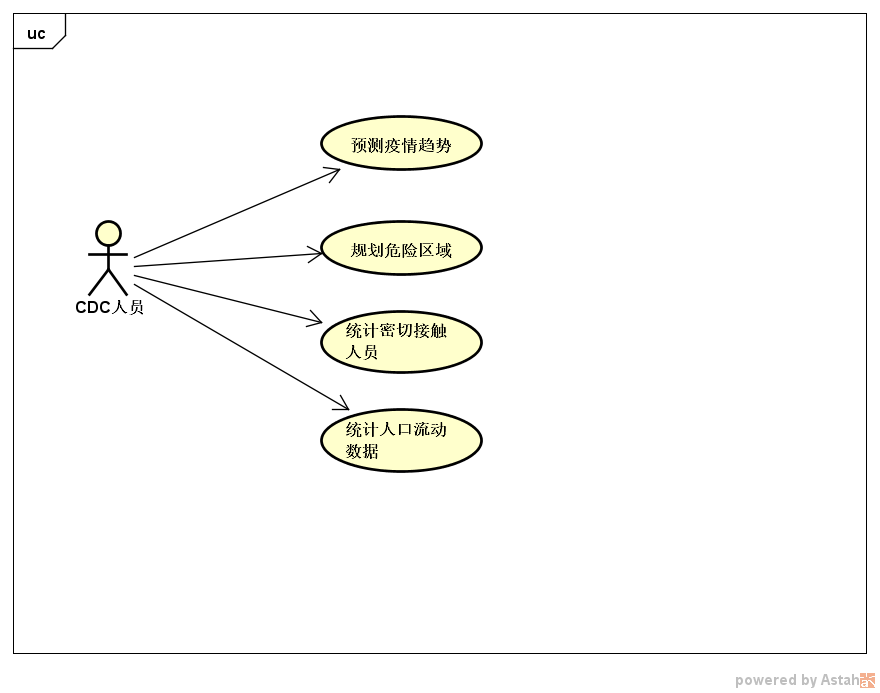
用例图-普通市民



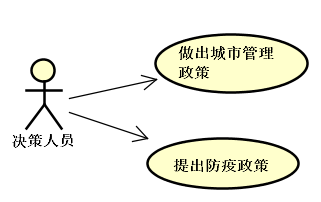
用例图-医护人员



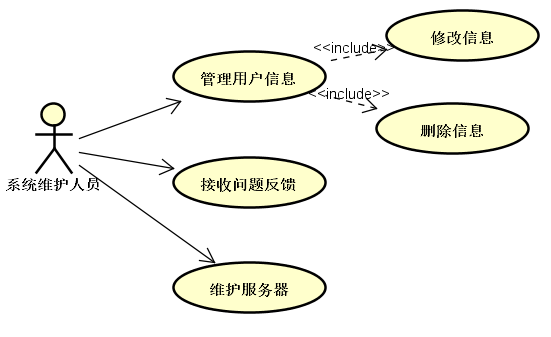
用例图-CDC人员



用例图-决策人员

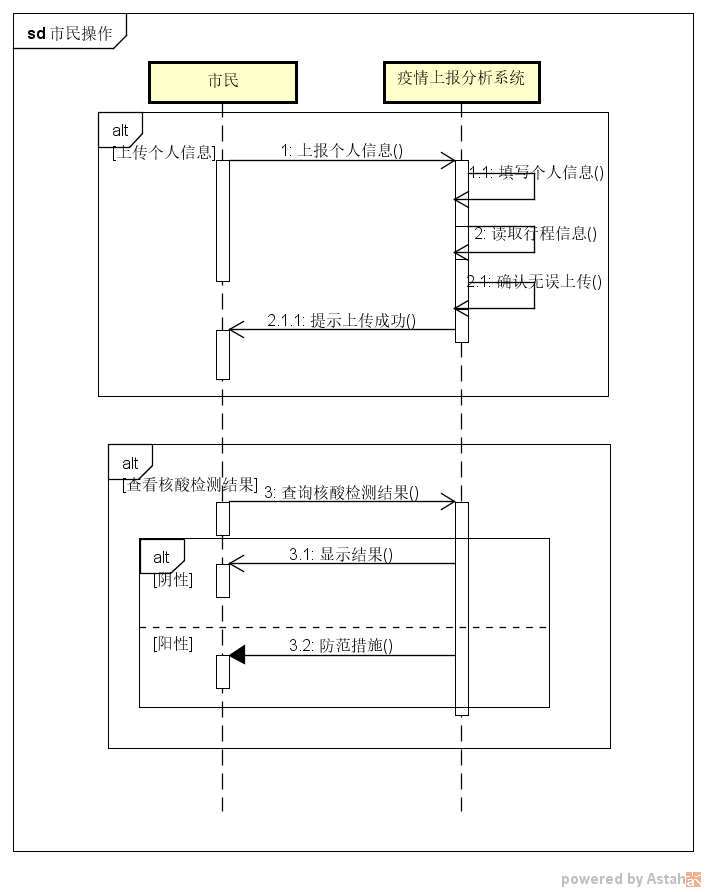


用例图-系统维护人员

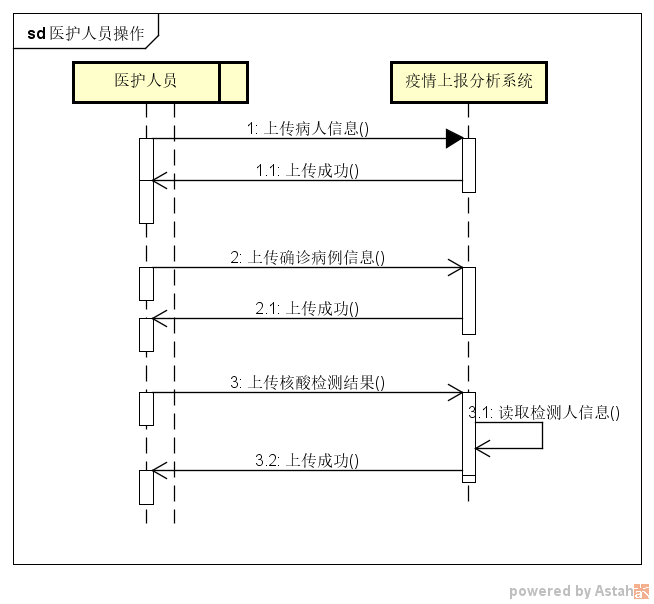


**2.2.3具体功能时序图**

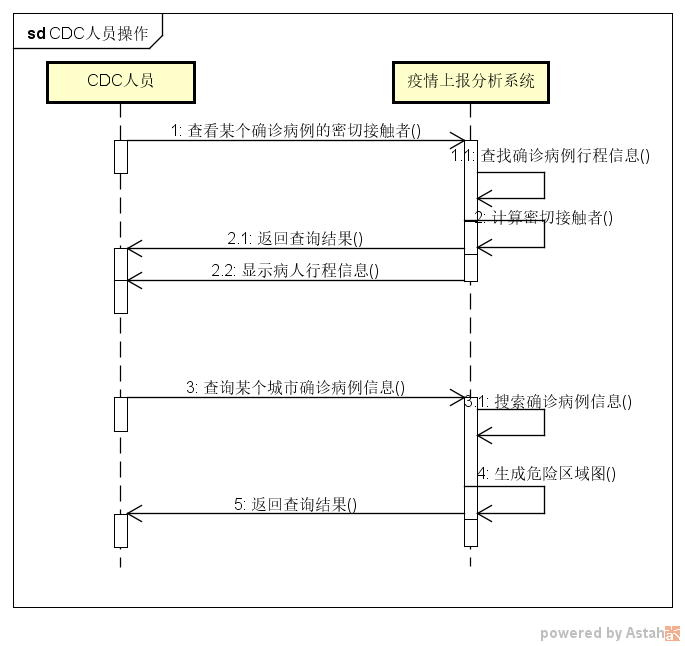
市民基本操作时序图



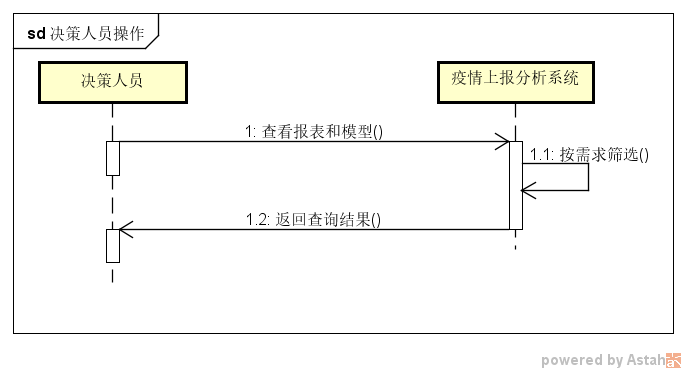
医护人员基本操作



CDC人员基本操作



决策人员基本操作



* 1. **各相关方对体系结构的要求**

1. 市民自己可以随时随地通过系统进行上报，并且这个过程应该是方便快速的。因此这就限制了体系结构应该是采用 Browser/Server 的结构，而不能是 Client/Server 的结构。
2. 程序为了适应未来在其他城市部署系统或者增添其他功能，这要求系统具有高可修改性、可移植性和可重用性，这要求体系结构要采用分层设计的思想。
3. 每一个市民只能看到自己的上报信息；医护人员只能通过自己所属医院上报平台上报病人信息或核酸检测结果；决策人员可以通过专属的页面查看所有上报信息。因此这需要体系结构可以满足很容易的修改一些用户界面的要求，比如 MVC 模式可以很好的满足这一要求。
4. 目前，随着大规模的网络技术迅速的发展使用，在人们日常的生活过程中，产生了海量的数据，为了能够有效的记录并利用这些数据，本地数据库己经无法满足人们的多种多样的需求，因此在工作、学习和生活过程中，必须开发基于网络的分布式应用系统，使用Hadoop、Mapreduce、云存储技术，来提高海量数据信息的使用效益。分布式体系架构是实现分布式应用系统的基本，为了能够使得分布式应用系统实现高性能访问和处理，诸多科学工作者对其进行了研究。
5. **系统开发、测试、维护人员要求**

体系结构的设计应与部件无关、低风险性、高可复用性高、可度量性、高可移植性、高可维护性。

* 1. **与部件无关：**是指体系结构的部件可相互独立地工作，无需了解其他部件的具体实现原理
  2. **低风险性：**按此体系结构实施的技术和技能的风险度较低。
  3. **可复用性：**体系结构设计抽象，且具有高通用性
  4. **可度量性高：**按该体系结构实施，系统的开发进度，人力、资 金、资源调配可以估计的能力。使项目管理人员能更好地进行项目进度的管理和安排
  5. **高移植性：**统软件的接口易改造，可以比较容易地转移到其他计算**机上使用；**同时系统使用了跨平台的 Java 语言来编写并且使用了开源库和复用了一些开源项目的代码，因此可以比较方便地移植到不同平台上。
  6. **高可维护性：**系统的构建结构合理，采用“高内聚、低耦合”的原则，可以比较方便地进行修改、升级、测试、维护。

1. **用户要求**

* 该具备容易操作的功能。

1. 提供针对不同用户的用户使用说明手册，方便用户学习使用，尤其要注意老人和小孩的使用。
2. 系统应提供在线帮助界面，方便用户学习操作。

* 由于操作该系统的人员有很多，且操作习惯、受教育程度、年龄阶段、接受 事物能力等都各不相同，这就要求系统具备良好的人机交互能力。系统提供的各 种功能便于用户理解，操作简单，用户很容易掌握。

1. 系统的界面设计应简洁明了，使用户能够自己学会使用本系统。
2. 系统应具有一定的美观性，可参考目前大部分网站的扁平化设计。
   1. **约束条件**

**2.4.1 实现约束**

本系统在进行代码编写的时候要求给出详细的代码注释，每一个功能函数都要给出函数的执行功能，输入和返回值，便于后期进行代码的重用。系统要求应用Java语言进行编写，并且所有变量的命名规范符合Java语言命名规范。系统所有网页HTML语言都要符合XHTML 5.0标准。系统要求对数据信息进行加密，要求使用当下市面上已有系统的数据加密算法。

**2.4.2其他约束**

1. 法律和政策方面的限制：开发此软件时，将严格按照有关法律和政策执行。
2. 系统要求在 2~3 个月内尽快上线，以满足疫情防控的紧急需要。
3. Web 服务器部署于 Linux 系统之上的系统一般运行稳定性高。
4. 服务器系统硬件的配置能支持服务器高效稳定的运行。
5. 为了系统将来的可扩展性，系统在硬件的使用上需尽可能减少针对性。整个系统也应尽量减少各模块间的调用，尽量做到松耦合。
6. 数据文件、系统配置文件应当安全可靠的存储。系统的数据需要具有足够的可靠性，才会保证系统正常运行
   1. **非功能性需求**

**2.5.1性能表现**

时间性能

1. 系统执行速度系统执行速度需要满足以下条件：系统响应快。数据 库单表操作时间不大于 0.5 秒。
2. 系统响应时间考虑到同时进行操作的用户人数，系统的响应时间应该达到 如下标准比较合理：

* 系统应当可以查看和修改用户的基本信息，一个流程的响应速度最佳为150 ms,平均为 200ms，不超过300ms，系统应当可以同时接受100万人同时操作。
* 系统应当能够准确计算出确诊病例的所有密切接触者，以及他们的感染概率。一个计算流程的响应时间不能超过 300ms。每个密切接触者计算时间不得超过400ms，系统应当可以同时接受1万人同时操作。
* 系统应当能够赋予决策者和CDC人员查看所有信息的权限，一个流程的响应速度不超过300ms。系统应当可以同时接受 1万人同时操作。
* 系统应当能够在500ms内画出某个市民的24小时行程路径图，并统计出每个时刻所在的位置。

空间性能：

1. 数据库容量能够存储至少10亿条市民行程数据记录。
2. 操作占用内存的限制

系统应当能够查看和修改四种用户信息，每次操作占用内存不超 过 10MB。

系统应当能够查询订单详细信息信息，每次操作占用内存不超过1MB

系统应当能够实时更新市民所在位置信息，每次操作占用内存不超过 20MB。

**2.5.2可靠性**

系统能较长时间下稳定运行，同时该系统需要具备一定的故障恢复能力，即有一定的容错能力。当用户的操作不当引起某些故障时，或者是由于操作系统或者 网络发生故障时，系统需要具备一定的故障恢复能力。选择数据库产品时，要考虑一定的数据负载能力。由于在处理员工信息、客 户信息、账户信息等信息时，系统需要做大量的数据统计和处理，因此要具备相 应的数据负载能力，详细分为如下：

1. 系统的可连续运行时间在一年以上，并且每次发生故障或者宕机后， 能够在半小时内恢复正常。
2. 容错能力。系统具有一定的容错和抗干扰能力，在非硬件故障或非通讯故障时，系统能够保证正常运行。如用户在系统中输入不规范时，不会引起系统的故障，且系统会给出提示信息，并限定不正确输入的次数。
3. 数据负载能力。选择数据库产品时，要考虑一定的数据负载能力。 由于在处理市民信息、医院信息、统计报表等信息时，系统需要做大量的数据统计和处理，因此要具备相应的数据负载能力。
4. 云服务的可靠性:

云服务的可靠性主要分为以下几个方面。

* 一是物理服务器的可靠性。服务器需要 7×24 小时不间断地运行， 因此需要完善的防灾措施，使其远离火灾、地震等灾害。
* 二是云服务逻辑上的可靠性。云服务的逻辑中不能存在死循环、 死锁、时间过长的响应等影响服务可靠性的因素。
* 三是网络的可靠性。与客户端相同，云服务端也需要一些工作来 保证网络请求和响应的正确性。例如客户端在收到服务器发送的响应 前断网了，应该在服务器缓存该响应。

**2.5.3可用性**

1. 该系统的可使用性体现在它可以支持多操作系统(至少包含Andorid, IOS, Windows, Unix, Linux)运行。同时该系统应该具备容 易操作的功能。

* 提供针对不同用户的用户使用说明手册，方便用户学习使用。
* 系统应提供在线帮助界面，方便用户学习操作。

1. 由于操作该系统的人员有很多，且操作习惯、受教育程度、年龄阶段、接受 事物能力等都各不相同，这就要求系统具备良好的人机交互能力。系统提供的各 种功能便于用户理解，操作简单，用户很容易掌握。

* 系统的界面设计应简洁明了，使用户能够自己学会使用本系统。
* 系统应具有一定的美观性，可参考目前大部分网站的扁平化设计。

**2.5.4密安性**

1. 系统应当能够保证用户信息不泄露，系统配置文件和数据库存储文件应该进行加密处理。
2. 通常来讲，实际使用的管理系统，必须具备相应的安全性能。该系统各级 用户有自的权限设置，例如用户之间不可以互相查看或修改其他人的个人信息。
3. 该系统应该通过设置防火墙确保数据传输的安全。使用可靠的操作系统 来保证系统的操作安全，确保系统在一个安全可靠的环境下运行。
4. 系统应保证用户信息不泄露，系统配置文件和数据库存储文件应 当进行加密处理。
5. 系统应保证不会因恶意攻击而崩溃，系统开发过程不存在明显漏洞。
6. 系统应当能够保证选取的开发方不存在商业竞争对手或类似的恶意对手。
7. 数据库的数据密安性：从数据库的角度来说，软件使用的过程中存在大量的数据库的增删改查操作，而每种用户删改查权限应该是不同的，应该对用户进行分组，每个用户组只能对特定的数据表特定的字段进行特定的操作。

**2.5.4可维护性**

1. 软件的可维护性是指改进软件的难易程度。系统部件遵守“高内聚，低耦合”原则。该系统的结构、接口、功能以及内部过程在开发以及跟踪阶段，容易被维护人员理解。
2. 系统应该能够容易诊断出存在的缺陷和失效原因，容易识别出待修改部分的可能性或能力。开发人员应当记录开发过程日志，以便备份追踪。
3. 系统应当能够保证开发过程的代码、设计和文档容易修改，代码应当结构清晰且有较详细的注释，设计文档详细明确
4. 从数据库的角度来说，本软件数据的可维护性包括以下几个方面。

* 是要求数据的物理存储是可以更换的。如果一台云数据库出现问题，可以及时地将该云数据库中地数据拷贝到另一台云数据库中
* 是要求 DBMS 是可以更换的，即数据地逻辑存储是可以更换的。
* 是要求数据的视图层是可以更换的。

**2.5.5可移植性**

1. 易安装性：该系统能够跨平台移动运行，包括 Windows 服务器平台以及 Linux 平台,客户端应该支持Android,IOS,鸿蒙系统等平台。
2. 共存性：系统应当能够和其他软件共存于一个平台上，不存在冲突 的软件。
3. 可替换性：系统可被容易地卸载，也易被高版本的系统替换。
4. 接口易改造：可方便地移植到不同的设备上。
5. 云服务应采用 MVC 的架构，模型层与数据库层直接联系，因此可用统一的方式编写，控制层在设计阶段就已确定，也可统一编写。如此， 视图层可以使用不同的框架进行编写，保证其可移植性
   1. **设计需遵循的标准**

该软件系统必须遵循以下设计原则：

**2.6.1实用性原则**

PARRS系统的设计过程中，要充分考虑各个地区的实际情况进行业务逻辑的设计，以保证最终开发出来的软件系统切实可用，可以良好地运行。

**2.6.2稳定性原则**

PARRS系统能 24 小时提供不间断的服务，要保证用户在任何时刻都能使用到我们的平台来他们所需的功能。因此，在设计时要考虑 到任何可能导致服务中断的原因，为用户提供稳定的服务。

系统设计时应有系统故障的应急预案，当主系统发生故障时，应保 证可从主系统平滑地切换到备份系统，尽可能使用户感知不到系统发 生的故障，把故障的损失降到最低。

**2.6.3复用性原则**

组件复用是PASSR系统设计中必须遵守的原则。对可重用的组件进行统一包装，包括系统级的应用组件和应用级的服务组件。其中，系统级的应用组件主要考虑应用服务器和数据库的广泛适用性，对各种企业级服务进行重新包装，以便于提高架构的平台独立性。以使后续能够为不同的高校开发出适合他们实际情况的PASSR系统。

**2.6.4扩展性原则**

PARRS系统必须具有良好地可扩展性。随着移动互联网的兴起， 越来越多的用户选择在通过小程序,快应用等新型模式使用互联网。因此，PARRS系统应能在扩展到支持小程序,快应用等平台，以满足用户的需求。

**2.6.5安全性原则**

PASSR架构的设计将在成熟稳定的硬件环境和应用软件基 础上，通过网络、系统、应用、备份恢复、安全控制机制、运行管理 监控等手段来保障系统的安全运行。

应充分、全面、完整地对系统的安全漏洞和安全威胁进行分析；

在网络发生被攻击、破坏事件的情况下，应尽可能地快速恢复网 络信息中心的服务，减少损失。应建立安全防护机制、安全检测机制。

1. 建立合理的实用安全性与用户需求评价与平衡体系。
2. 系统是一个庞大的系统工程，其安全体系的设计必须遵循一系列 的标准，这样才能确保各个分系统的一致性，使整个系统安全地互联互通、信息共享。
3. 技术与管理应相结合原则。应将各种安全技术与运行管理机制、人员思想教育与技术培训安全规章制度建设相结合。
4. 统筹规划，分步实施原则。
5. 等级性原则。将系统进行分为不同的等级，包括对信息保密程度 分级，对用户操作权限分级，对网络安全程度分级，对系统实现结构的分级，从而针对不同级别的安全对象，提供全面安全算法和安全体制。
6. **体系架构实现方案**

**3.1相关的体系结构模式**

**3.1.1三次B/S体系架构简述**

三层B/S体系架构包括表示层（用户交互界面)、业务逻辑层（起到数据交换中承上启下的作用）、数据访问层（负责数据持久化）各层之间采用接口相互访问，并通过对象模型的实体类（Model）作为数据传递的载体，不同的对象模型的实体类一般对应于数据库的不同表，实体类的属性与数据库表的字段名一致，该体系结构对于用户来说可以在任何地方进行操作而不用安装任何专门的软件,只要一台能上网的设备就能使用，客户端零维护。系统的扩展非常容易，只要能上网，然后用户在注册账号后就可以使用了；对于开发者来说各层互相独立，项目可以多人同时开发，开发人员可以只关注整个结构中的其中某一层，此外容易移植、维护。

主要架构图如下所示：

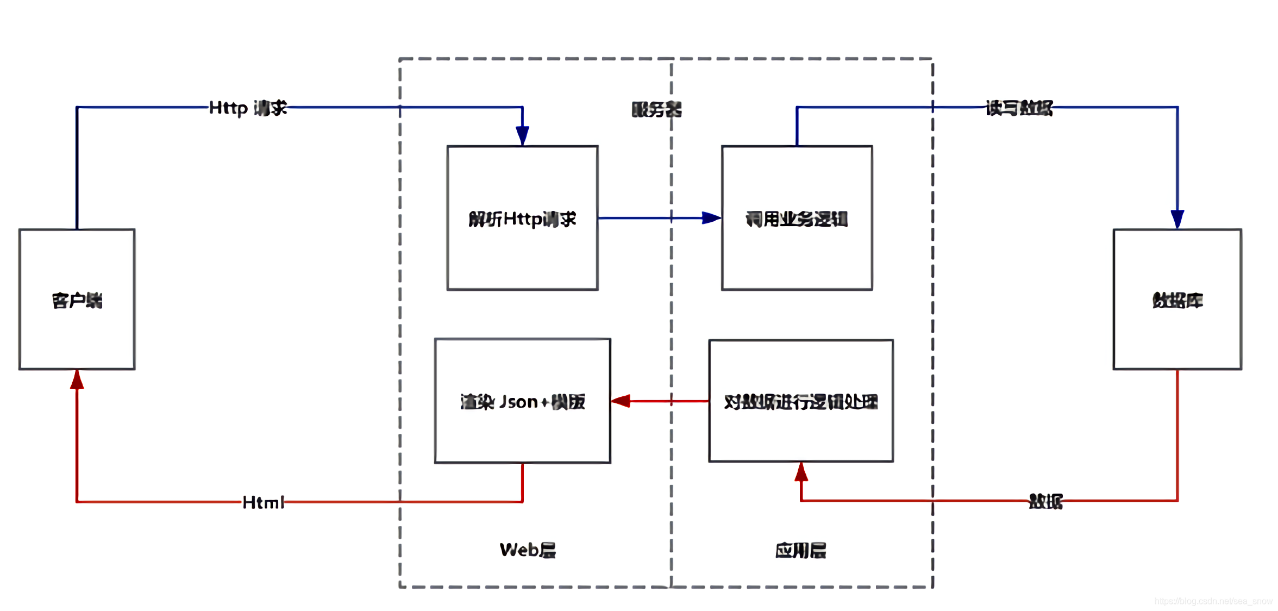


图3-1三层B/S架构图

**3.1.2系统架构介绍**

采用浏览器进行访问的时候不用考虑网络因素的限制，保证了信息的反馈速度。本系统结合B/S体系结构模式和MVC体系结构模式设计而成。

B/S架构主要体现在使用者借助浏览器或这微信小程序登录上报系统，开始进行相关数据的反馈，根据客户相关的业务逻辑，数据将会从数据库中取出同样显示在网页上面。MVC主要用于编写代码时的约束，对不同的用户提供不同的视图层，使得设计思路清晰，便于未来系统的修改和重新使用。

本系统采用B/S架构的原因如下：

1. B/S 最大的优点就是如前文所提到的，方便用户登陆操作以及系统的修改与维护，因此采用B/S架构进行系统的体系构建。
2. B/S架构建立在广域网之上的，不必是专门的网络硬件环境。例与电话上网，租用设备。信息自己管理，比C/S更强的适应范围，一般只要有操作系统和浏览器就行。因此本系统客户端
3. 系统维护方便，构件组成方便构件个别的更换，实现系统的无缝升级。系统维护开销减到最小，用户从网上自己下载安装就可以实现升级。
4. 建立在浏览器上，有更加丰富和生动的表现方式与用户交流，并且大部分难度减低,减低开发成本。
5. 对安全以及访问速度的多重的考虑，建立在需要更加优化的基础之上。比C/S有更高的要求，B/S结构的程序架构是发展的趋势，全面支持网络的构件搭建的系统，SUN 和IBM推的JavaBean 构件技术等使 B/S更加成熟。

**3.2结构化视图**

**3.2.1系统体系架构设计**

在目前用户需求的快速变化下，可能有多种方式访问应用的要求，而使用MVC系统体系结构可以满足这一需求，减少了代码的复制，即减少了代码的维护量，一旦模型发生改变，也易于维护。 其次，由于模型返回的数据不带任何显示格式，因而这些模型也可直接应用于接口的使用。

再次，一个应用的业务流程或者业务规则的改变只需改动MVC的模型层。 控制层的概念也很有效，由于它把不同的模型和不同的视图组合在一起完成不同的请求，因此，控制层可以说是包含了用户请求权限的概念。最后，它还有利于软件工程化管理。由于不同的层各司其职，每一层不同的应用具有某些相同的特征，有利于通过工程化、工具化产生管理程序代码。

MVC体系架构主要体现在PARRS系统上的，一共分为三层：表示层、应用层、服务层。这三层最终都是基于B/S架构的浏览器和服务器实现的。这三层主要用于编写代码时的约束，使得设计思路清晰，便于未来系统的修改和重新使用。

1. **表示层**

表示层位于用户客户端，该层的相关功能是接受用户从客户端浏览器发送的业务请求，将相关请求发送到逻辑业务处理层，并将逻辑业务处理层反馈的处理结果输出到浏览器上，供用户浏览。B/S 体系架构中的客户端使用人们常用的 Web 浏览器作为载体，输入系统服务器所在网址，自动弹出下载插件的通知，将其装在浏览器中即可供用户使用，无需下载安装程序，减少了客户端应用程序安装、运营更新的需求，大大的降低了网络管理员的工作量。

1. **逻辑业务处理层**

逻辑业务处理层位于 B/S 体系架构中的Web服务器上，也被很多场合成为 Web服务器层。逻辑业务处理层的相关功能是完成系统客户端发送过来的用户请求，由 Web 服务器分配处理任务，执行用户请求的逻辑业务，同时并根据用户的需求检测其是否需要数据处理，如果需要数据处理，则将处理结果的请求发送到数据库服务器上，由数据库服务器完成余下工作，同时将数据库服务器处理的结果反馈到客户端，供用户浏览。

1. **服务层**

数据层位于数据库服务器上，通常又被称为数据库服务器层，其相关功能是处理逻辑业务处理层发送过来的数据业务请求，该层可以使用数据库管理系统的相关操作，实现数据请求的添加、删除、修改等操作，同时将数据处理结果及时的反馈到逻辑业务处理层。

表3-1 MVC和B/S组织模型

|  |  |  |
| --- | --- | --- |
| MVC层次 | 理解性抽象 | 对应B/S架构 |
| 表示层(V) | 用户界面层 | 浏览器获取数据 |
| 应用层(C) | 用户身份验证层、信息上报和数据计算层 | 浏览器显示数据，服务器传送数据 |
| 服务层(M) | 云数据库层 | 服务器计算数据 |

用户界面层是和用户最直接接触的一个层次。这个层次通过网页界面或者微信小程序进行相应的展示，用户可以通过自己的微信或浏览器界面进行相应的访问和查看。

用户身份验证层是进行用户身份类型的检查。本系统面向出行市民用户、医护人员、CDC人员、在线客服等进行PARRS系统的管理和使用，对于不同身份的人系统提供的功能和界面是不同的。在实际中，采用识别出相应的用户ID来完成身份的验证，因此用户身份验证层是不可缺少的。

信息上报和数据计算层主要针对的是市民及医院信息上报、确诊病例信息的查询、计算密切接触以及人口流动图等功能、完成服务后与第三方接口实现。在这一层根据上一层获得的身份类型呈现不同的界面，比如普通用户只能在界面上报信息，查询或修改与自己有关的信息，而CDC人员可以查询所有疫情信息，使用系统计算出疫情总体态势。

云数据库层主要完成数据信息的存储和备份，根据上一层获取的条件返回相应的查询结果，并且对数据进行相应的权限管理，防止非管理人员进行相应的数据访问。此数据库主要采用基于Hadoop和Openstack云存储为大量数据访问计算提供坚实基础。

**3.2.2负载均衡器(Nginx)**

虽然本系统使用的MVC+B/S组织架构能够满足为不同用户提供不同功能，并方便用户随时登录系统，但是由于此架构是交互是请求-响应模式，通常动态刷新页面，客户界面框架内容、数据以及事件都必须依赖于网络传输，这种方式这会降低系统性能，导致系统响应用户时间可能存在过长的问题。

为了解决当百万级用户同时上传与访问数据，所带来的系统响应时间过慢的问题，PAARSE系统需要采用在原来的系统架构层级上加负载均衡层，即：客户端－负载均衡器(Nginx)－应用服务器－数据库。

**使用负载均衡器只要有以下优点：**

1. 能够使系统承受、高并发的大量的请求，然后将这些请求均匀的转发给内部的服务器，分摊压力.
2. 反向代理解决跨域引起的问题，因为Nginx应用服务器，云数据库都处于内网段中。
3. 由于Nginx非常擅长处理静态资源(img,css,js,video)，所以这降低外部对网页资源的访问，比如：前一个用户访问index.html, 经过Nginx－Node－应用服务器－数据库链路之后，Nginx会把index.html返回给用户，并且会把index.html缓存在Nginx上，下一个用户再想请求index.html的时候，请求Nginx服务器，Nginx发现有index.html的缓存，于是就不用去请求Node层了，会直接将缓存的页面(如果没过期的话)返回给用户。
4. Nginx可以通过端口检测到服务器内部的故障，比如根据服务器处理网页返回的状态码、超时等等，并且会把返回错误的请求重新提交到另一个节点，不过其中缺点就是不支持url来检测。比如用户正在上传一个文件，而处理该上传的节点刚好在上传过程中出现故障，Nginx会把上传切到另一台服务器重新处理。

**3.2.3系统体系架构视图**

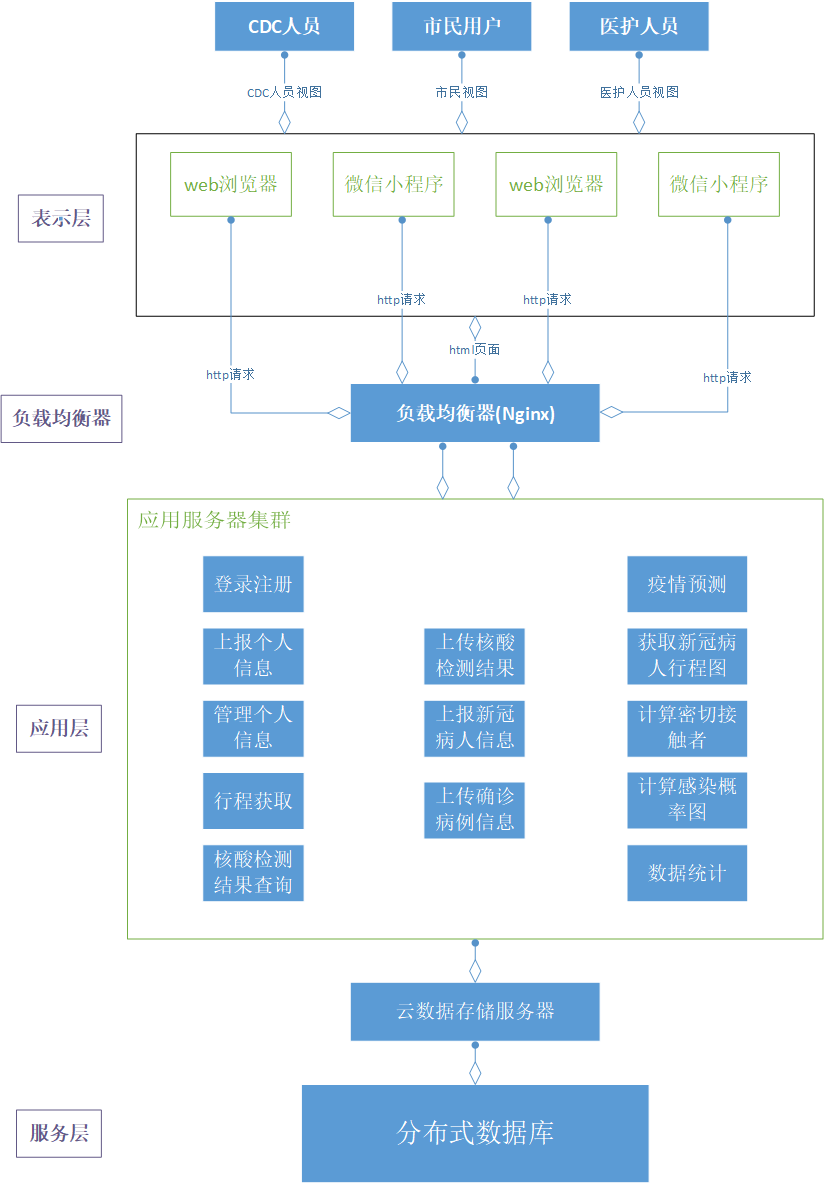
下图展示了详细的系统逻辑视图，包含四层架构设施。通过这样设计，系统的业务逻辑和数据操作均能够有序地进行处理。

图3-2 PARRS体系结构图

**3.3.4模块级体系结构**

系统的模块化设计是系统进行复用的关键。模块级体系结构反映了对软件代码实现时的期望，特别是对于程序规模较大的系统。模块化表达有两种方式；

1. 将系统按功能从逻辑上分解为系统、模块以及程序单元；
2. 按系统的层次进行划分。模块化分解方式很容易区分出概要设计和详细设计两个阶段。

本说明书将系统按功能从逻辑上分解，细化功能结构模块。

1. **登陆模块**

登陆系统主要为不同用户提供登陆功能以识别用户身份。用户在浏览器中输入账号密码时，客户端会读入用户信息，并将数据信息传送到服务器，服务器将输入数据与后台数据库内容进行比对来查看用户名与密码是否匹配，将结果传送给浏览器向用户输出。

1. **信息上传模块**

信息上传模块主要是为市民和医护人员提供信息上传途径。市民在浏览器界面写入自己的信息，系统通过网络获取其24小时的行程路径，点击上传后。服务器将数据存储到数据库中，并向用户界面显示上传成功提示，医护人员将核酸检测结果以及新冠肺炎感染者的信息上传到系统中。

1. **用户信息管理模块**

信息管理模块主要是市民管理自身个人信息。该模块的主要功能是实现用户信息以及权限的修改，查看其核酸检测结果，浏览器读入用户要修改的信息并将数据与请求发送给服务器，服务器根据浏览器传送的信息更新数据库，并返回结果。

1. **信息查询功能模块**

CDC人员通过该模块查询病例信息，及其密切接触者信息，将查询条件输入到浏览器页面请求服务器响应，服务器将处理数据，并将结果传送给客户端，浏览器页面通过该数据响应用户请求。

1. **密切接触者计算模块**

CDC人员通过该模块，获取该感染者的密切接触者信息及感染概率，通过输入感染姓名，系统响应请求，根据感染者的行动路径图计算结果，系统返回密切接触者名单及其感染概率，以及每个人的行程路径图等结果，浏览器将这些结果呈现给用户。

1. **疫情预测模块**

CDC人员通过该模块，计算出疫情未来的发展趋势图，系统通过之前的疫情数据，疫情发展趋势图，疫情爆发点及其原因，计算出详细的预测图标，将结果返回到浏览器页面。

**3.3行为视图**

**3.3.1体系结构视角**

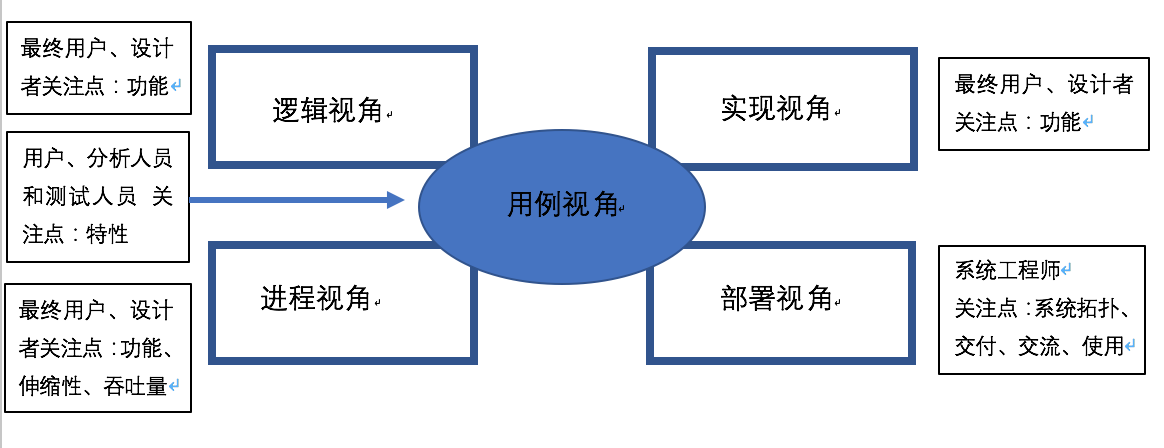
本说明书使用 Philippe Kruchten提出的从不同角度勾画系统的蓝图，即“4+1”视图模型。该模型从 4个角度(逻辑、实现、进程和部署)指出不同的相关利益方关心的事情，外加从使用者的角度对用例进行观察，分析其影响系统的上下文和商业目标情况

图3-3视图模型

**3.3.2逻辑视角**

本系统在软件逻辑架构的设计工作上采用的是轻量级 Struts 框架，该框架是基于MVC模式的一种架构。下图为本文系统的软件逻辑架构示意图

在Sturst框架技术中，Action和Actionservlet具有控制器的功能，而 JavaBean 则是用来充当系统的控制模型的角色。具体实现的过程为:当用户提出操作请求之后，第一步会由Nginx简单处理，之后将请求发送给服务器，服务器负责处理用户发出的操作请求的信息，在将此信息请求传送到控制器Actionservlet 中，接下来由 Action来负责接收操作指令，然后从云数据库中找到的对应信息，之后再将数据访问结果返回给视图呈现给用户。

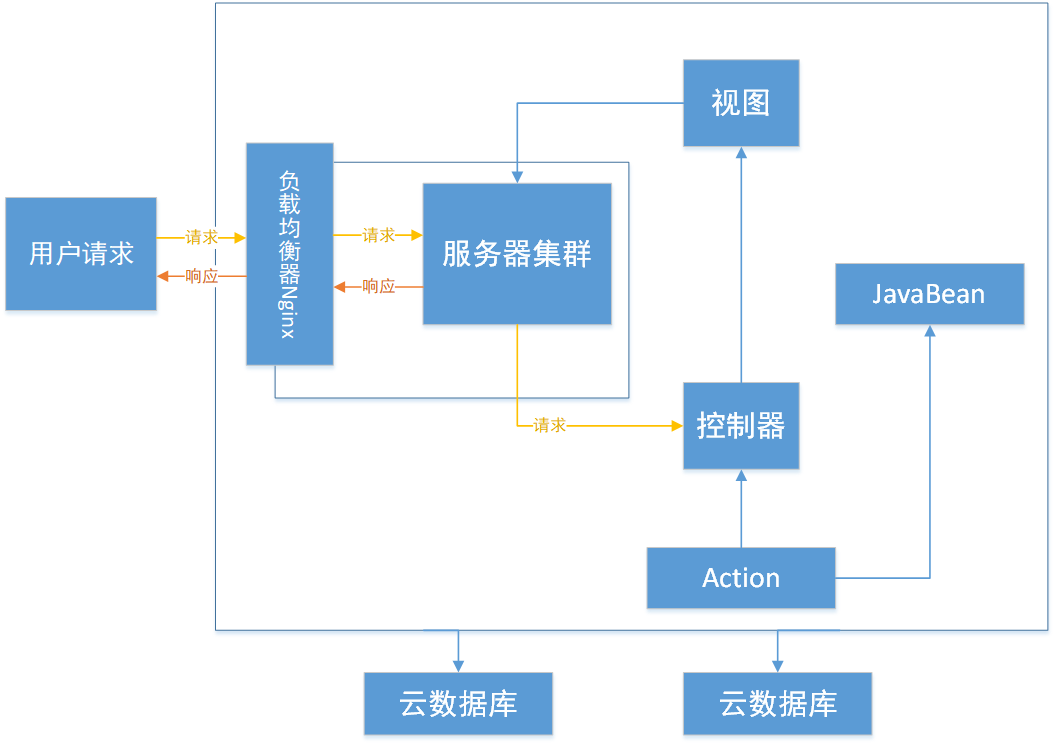
****

图3-4进程逻辑视图

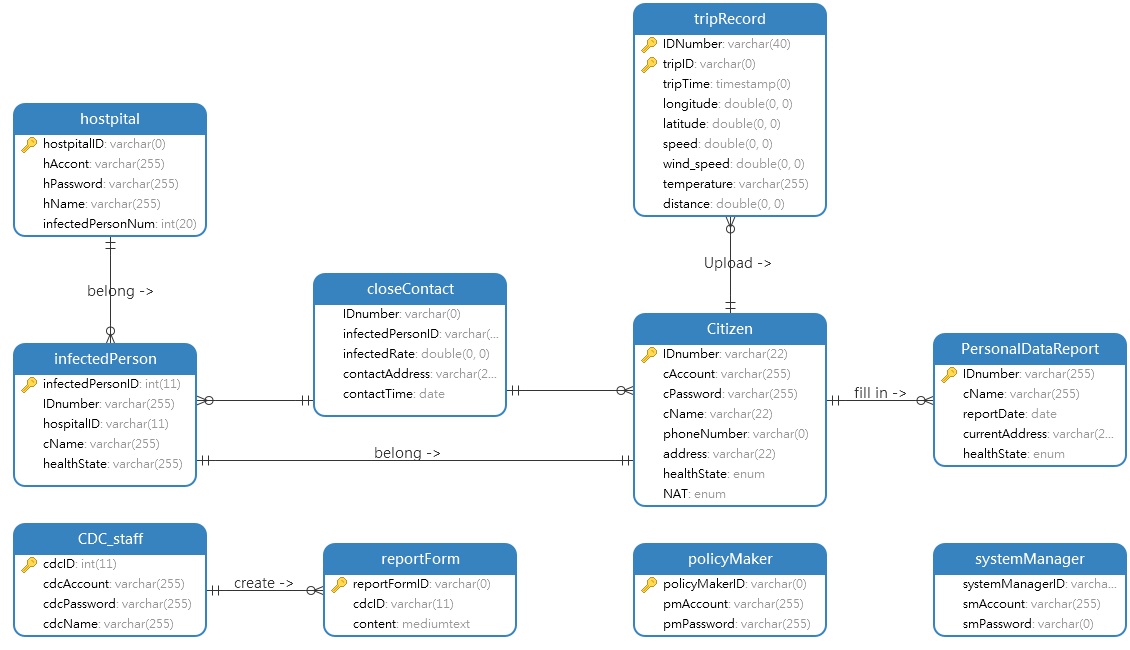
****

图3-5数据库逻辑视图

**3.3.3实现视角**

实现视角，又叫开发视图（Development View），描述了在开发环境中软件的静态组 织结构，即关注软件开发环境下实际模块的组织，服务于软件编程人员。将软件打包成小 的程序块（程序库或子系统），它们可以由一位或几位开发人员来开发。子系统可以组织 成分层结构，每个层为上一层提供良好定义的接口。因此系统的架构开发视图如 3-6。

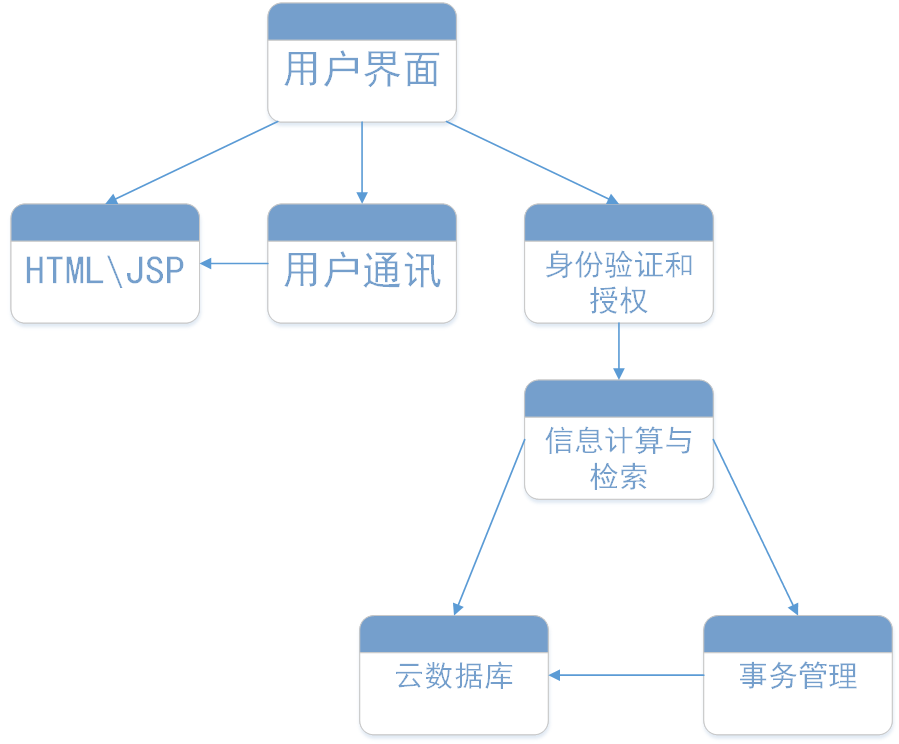


图3-6架构开发视图

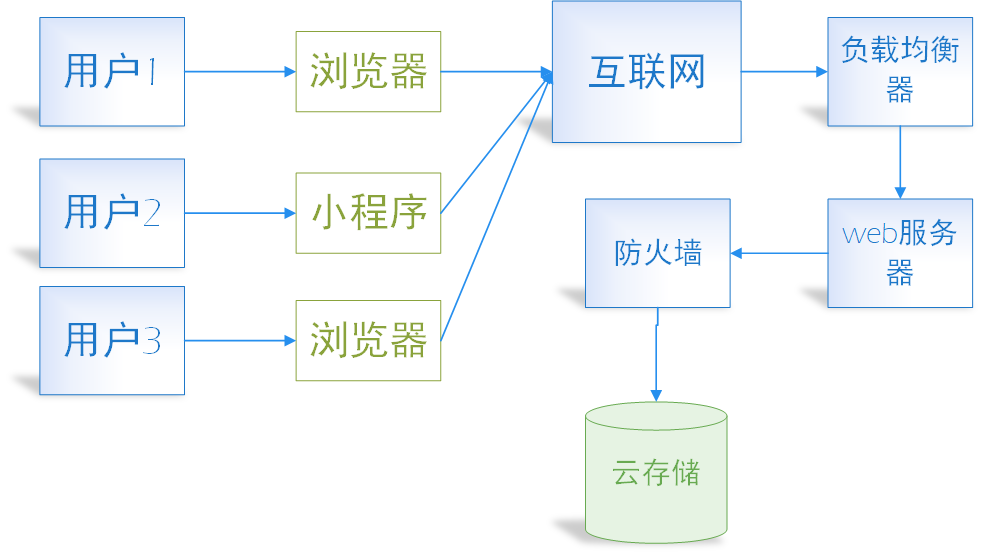
系统的实现视图用模块和子系统图来表达。完整的开发架构只有当所有软件元素被识 别后才能加以描述。但是，可以列出控制开发架构的规则：分块、分组和可见性。因此本说明书设计系统功能图如下



图3-7系统功能图

**3.3.4部署视角**

从系统软硬件物理配置的角度，描述系统的网络逻辑拓扑结构。模型包括各个物理节点的硬件与软件配置，网络的逻辑拓扑结构，节点间的交互和讯关系等。同时还表达了进 程视图中的各个进程具体分配到物理节点的映射关系。由于本系统采用的是多层B/S架构，结合现实的网络拓扑结构，本文设计部署视角如下图

.

**3.3.5用例视角**

用例视角，又称“场景视图”，它综合所有的视图。用于刻画构件之间的相互关系，将四个视图有机地联系起来。可以描述一个特定的视图内的构件关系，也可以描述不同视图间的构件关系。用例视角图见2.2.2用户用例图。

1. **系统的质量分析和评价**

ATAM是评价软件构架的一种综合全面的方法。这种方法不仅可以揭示出构架满足特定质量目标的情况，而且可以使我们更清楚地认识到质量目标之间的联系——即如何权衡诸多质量目标。

**4.1场景分析**

通过场景可以很好地评估体系结构。SEI开发了通过手工评估和确认体系结构的方法，称为体系结构折中分析方法。它是评价软件构架的一种综合全面的方法。这种方法不仅可以揭示出构架满足特定质量目标的情况，而且可以使我们更清楚地认识到质量目标之间的联系。本文档根据ATAM方法来对PARRS系统的体系架构进行分析和评估。

**4.1.1用例场景**

用例场景是从使用者的角度出发，描述用户期望的与整个运行系统的交互。

|  |  |
| --- | --- |
| 场景1 | 1. 系统能较长时间下稳定运行，并具有一定的容错能力和故障恢复能力，在发生硬件或者软件异常时，应该具有依然具有服务能力。服务能力虽然下降，但不会导致系统崩溃无法使用。 2. 本系统在发生停机故障之后，应该保证30分钟之内可以修复并且正常运行，并提示管理员。 3. 保证用户在正常使用本系统时，用户的操作或者误操作都不会导致数据丢失。 4. 系统对用户输入的信息有较为严格的验证，保证系统的可靠性。   该场景代表了系统可靠性要求。 |
| 场景2 | 1. 当用户发起信息上传请求时，系统应当在500ms内完成对于请求有效性的验证，并将结果返回给用户。 2. 系统中的数据得到更新后，系统的其他使用者页面的数据要在3秒内得到更新。 3. 100万个用户能够并发登陆系统，且此时系统产生的延迟不超过3秒。 4. 对用户的查询操作，系统的响应时间不超过3秒。   该场景代表了系统的性能要求。 |

表格 4-1用例场景

**4.1.2增长性场景**

增长性场景是指预期未来系统修改时可能发生的场景。

|  |  |
| --- | --- |
| 场景1 | 当用户使用数量上升时，当前的体系结构无法满足服务场景的需求，可增加新的数据库服务器、扩充现有的数据规模，降低远程用户的访问时间。 |
| 场景2 | 当用户访问量超过单一系统承受能力时，使用分布式集群进行计算，均衡单个服务器的负载量，从而降低用户使用该软件系统的响应时间。 |
| 场景3 | 当数据库查询效率无法达到系统要求时，通过扩充现有数据库表的规模，使用数据库索引功能，提高查询效率。 |

表格4-2增长性场景

**4.1.3探索性场景**

探索性场景是推动系统封装和降低工作压力的场景。场景的目标是解释当前设计的边界条件的限制，揭露出可能隐含着的假设条件。探索性场景在某种程度上可以为未来PASSR系统的修改给出更实际的需求。

|  |  |
| --- | --- |
| 场景1 | 优化算法，改进系统的性能，降低系统的响应时间。 |
| 场景3 | 正常情况下，当一半服务器宕机时，不影响整个系统的可使用性。 |
| 场景5 | 提到系统对高并发访问的处理能力。 |

表格 4-3探索性场景

**4.2风险**

在开发新的软件系统的过程中，由于存在许多不确定因素，软件开发失败的风险是客观存在的。在系统运行过程中，由于外部环境变化或者自身设计的不足，也可能产生各种风险。因此，风险分析对于软件项目管理是决定性的。

**4.2.1技术风险**

在软件项目开发和建设的过程中，战略管理技术因素是一个非常重要的因素。项目组一定要本着项目的实际要求，选用合适、成熟的技术，千万不要无视项目的实际情况而选用一些虽然先进但并非项目所必须且自己又不熟悉的技术。如果项目所要求的技术项目成员不具备或掌握不够，则需要重点关注该风险因素。重大的技术风险包括：软件结构体系存在问题，完成的软件产品未能实现项目预定目标;项目实施过程中才用全新技术，由于技术本身存在缺陷或对技术的在掌握不够深入，造成开发出的产品性能以及质量低劣。

预防这种风险的办法一般是经常和用户交流工作成果、品牌管理采用符合要求的开发流程、认真组织对产出物的检查和评审、计划和组织严格的独立测试等。软件质量的保证体系是软件开发成为可控制过程的基础，也是开发商和用户进行交流的基础和依据。所以制定卓有成效的软件质量监督体系，是任何软件开发组织必不可少的。

**4.2.2进度风险**

软件的工期常常是制约软件项目的主要因素。软件项目工期估算是软件项目初期最困难的工作之一。很多情况下，软件用户对软件的需求是出于实际情况的压力，希望项目承担方尽快开发出软件来。在软件招标时，开发方为了尽可能争取到项目，对项目的进度承诺出已远远超出实际能做到的项目进度，使项目在开始时就存在严重的时间问题。软件开发组织在工期的压力下，往往放弃文档的编写与更新，结果在软件项目的晚期大量需要通过文档进行协调时，却拖累软件进度越来越慢。此外，由于用户配合问题、资源调配等问题也可能使软件项目不能在预定的时间内完成任务。软件项目过程中有自身的客观规律性，用户对软件项目的进度要求不能与软件开发过程的时间需要相矛盾。

对于这种风险解决方案一般是分阶段交付产品、增加项目监控的频度和力度、多运用可行的办法保证工作质量避免返工。在项目实施的时间进度管理上，需要充分考虑各种潜在因素，适当留有余地;任务分解要详细，便于考核;在执行过程中，应该强调项目按照进度执行的重要项，再考虑任何问题时，都要保持进度作为先决条件;同时，合理利用赶工期及快速跟进等方法，充分利用资源。应该避免：某方面的人员没有到位，或者在多个项目的情况下某方面的人员中途被抽到其他项目，或身兼多个项目，或在别的项目中无法抽身投入本项目。为系统测试安排足够的时间，能使项目进度在改变之初就被发现，这对及时调整项目进度至关重要。在计划制定时就要确定项目总进度目标与分进度目标;在项目进展的全过程中，进行计划进度与实际进度的比较，及时发现偏离，及时采取措施纠正或者预防，协调项目参与人员之间的进度关系。

**4.2.3 质量风险**

任何软件项目实施过程中缺乏质量标准，或者忽略软件质量监督环节都将对软件的开发构成巨大的风险。有些项目，用户对软件质量有很高的要求，如果项目组成员同类型项目的开发经验不足，则需要密切关注项目的质量风险。矫正质量低下的不可接受的产品,需要比预期更多的测试、设计和实现工作。

预防这种风险的办法一般是经常和用户交流工作成果、品牌管理采用符合要求的开发流程、认真组织对产出物的检查和评审、计划和组织严格的独立测试等。软件质量的保证体系是软件开发成为可控制过程的基础，也是开发商和用户进行交流的基础和依据。所以制定卓有成效的软件质量监督体系，是任何软件开发组织必不可少的。