学生期末考试成绩管理系统

体系结构设计报告

Version 1.0

2015-12-19

郭建敏 2013212138 魏祎 2013212143

软件工程导论

2015 秋季学期

# 变更记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本号 | 变更日期 | 修订人 | 变更原因 | 评审日期 | 评审人 |
| 1.0 | 18/12/2015-20/12/2015 | 郭建敏  魏祎 | 完善体系结构设计 | 20/12/2015 | 郭建敏  魏祎 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 文档审批

|  |  |  |  |
| --- | --- | --- | --- |
| 签名 | 姓名 | 标题 | 日期 |
|  | 魏祎 | 审核体系结构需求和质量评估 | 20/12/2015 |
|  | 郭建敏 | 审核体系结构设计方案和视图 | 20/12/2015 |

目录

[变更记录 2](#_Toc438412465)

[文档审批 2](#_Toc438412466)

[1. 文档简介 6](#_Toc438412467)

[1.1 目的 6](#_Toc438412468)

[1.2 范围 6](#_Toc438412469)

[1.3 定义、首字母缩写词和缩略语 7](#_Toc438412470)

[1.4参考资料 7](#_Toc438412471)

[1.5 概述 8](#_Toc438412472)

[2. 系统概述 8](#_Toc438412473)

[2.1系统简介 8](#_Toc438412474)

[3. 体系结构需求 9](#_Toc438412475)

[3.1 关键指标 9](#_Toc438412476)

[3.2 体系结构用例 9](#_Toc438412477)

[3.3各相关方对体系结构的要求 10](#_Toc438412478)

[3.3.1 客户相关方要求 10](#_Toc438412479)

[3.3.1 开发相关方要求 11](#_Toc438412480)

[3.4约束条件 12](#_Toc438412481)

[3.4.1 基于数据库的约束条件 12](#_Toc438412482)

[3.4.2 基于整体设计的约束条件 12](#_Toc438412483)

[3.5 非功能需求 12](#_Toc438412484)

[3.6 风险 14](#_Toc438412485)

[3.6.1 来自内部计划过程的风险 14](#_Toc438412486)

[3.6.2来自内部设计过程的风险 14](#_Toc438412487)

[3.6.3来自外部的风险 14](#_Toc438412488)

[4. 解决方案 15](#_Toc438412489)

[4.1 相关的体系结构模式 15](#_Toc438412490)

[4.2 体系结构概述 16](#_Toc438412491)

[4.2.1 概念级体系结构 17](#_Toc438412492)

[4.2.2 模块化体系结构 17](#_Toc438412493)

[4.2.3 代码体系结构 19](#_Toc438412494)

[4.2.4 运行时体系结构 21](#_Toc438412495)

[4.2.5 计算机硬件体系结构 22](#_Toc438412496)

[4.3 结构化视图 23](#_Toc438412497)

[4.3.1 用例视图 23](#_Toc438412498)

[4.3.2 逻辑视图 25](#_Toc438412499)

[4.3.3 其他视图说明 27](#_Toc438412500)

[5. 系统的质量分析和评价 27](#_Toc438412501)

[5.1 场景分析 27](#_Toc438412502)

[5.1.1用例场景 27](#_Toc438412503)

[5.1.2 增长性场景 28](#_Toc438412504)

[5.1.3探索性场景 28](#_Toc438412505)

[5.2 原型分析 28](#_Toc438412506)

**图表目录**

[图表 1定义、首字母缩写词和缩略语 7](#_Toc438412689)

[图表 2体系结构用例 10](#_Toc438412690)

[图表 3概念级体系结构 17](#_Toc438412691)

[图表 4模块化体系结构 19](#_Toc438412692)

[图表 5代码体系结构 20](#_Toc438412693)

[图表 6运行时体系结构 22](#_Toc438412694)

[图表 7计算机硬件体系结构 22](#_Toc438412695)

[图表 8学生用例 23](#_Toc438412696)

[图表 9教师用例 24](#_Toc438412697)

[图表 10教务员用例 24](#_Toc438412698)

[图表 11逻辑视图顶层包 25](#_Toc438412699)

[图表 12逻辑视图重要设计包Business 26](#_Toc438412700)

[图表 13效用树 29](#_Toc438412701)

# 文档简介

## 1.1 目的

1. 详细阐述学生期末考试成绩管理系统的体系结构设计，识别系统的体系结构需求、相关约束、非功能性需求，并给出体系结构设计的解决方案和相应的视图，最后进行体系结构的质量分析和评估。
2. 作为产品立项和产品开发的参考文档。给出各用户的详细功能要求，系统功能块的组成及联系，进程部署和硬件要求等，为开发方进行详细设计和编程提供基础，为测试人员验证测试系统提供依据。
3. 提高开发效率。详细周密的体系结构设计可以提高开发方的开发效率，并对其开发过程进行必要的限制，可减少事后重新设计、重新编码和重新测试的返工活动。便于在开发早期发现若干遗漏、错误的理解和不一致性，从而及时加以纠正。
4. 为系统成本计价和编制计划进度提供基础。
5. 作为系统不断改进提高的可靠基础。

本体系结构文档适用对象：开发小组成员

## 1.2 范围

1. 系统能够保持用户基本信息完整，学生查询成绩、教师录入成绩和教务员审定发布的成绩一致且无冗余。
2. 系统能保证具有密安性，不会泄露用户信息，也不会允许不具有特定权限的用户更改信息。
3. 系统推荐使用的DBMS：MySQL。其他大型DBMS也可以使用，但费用较高：如Oracle、DB2等。
4. 体系结构设计文档中的图片可以使用Report Generator软件输出并打印。

### 1.3 定义、首字母缩写词和缩略语

|  |  |
| --- | --- |
| 定义 | 说明 |
| 用例视图 | 描述系统中用户的使用场景 |
| 逻辑视图 | 描述系统中子系统的组成及联系，包与包、类与类的协作 |
| 进程视图 | 系统执行时的进程（包括线程、进程、进程组），以及通信的方式 |
| 部署视图 | 系统部署的硬件、支撑软件、网络环境。 |
| 实施视图 | 系统整体结构、层次结构、子系统，及第三方控件，自定义控件，及其接口 |

图表 1定义、首字母缩写词和缩略语

## 1.4参考资料

1. 百度百科， lamp （Web应用软件）[online]

Available at <http://baike.baidu.com/link?url=owa8zf0WJFRxkZZIirz7u89V9Ae1ZkxbDpBSJK6gDGE4Dp7HGbNMxPgVeShnnr4aFQNVG6y11h-L1yR9Y9BpWq0f3S0TDfvERqStQSzcFVa> (accessed 19 Dec, 2015)

1. 陈华，动态网站开发平台比较(ASP.NET, J2EE, LAMP) ，20 April, 2010 [online]

Available at

<http://www.chhua.com/web-note1900> (accesses 20 Dec, 2015)

## 1.5 概述

本文档从以下几个方面给出系统体系结构的设计。

1. 系统简介
2. 系统的体系结构需求。包括系统关键指标、体系结构用例、各相关方的要求、相关约束、非功能性需求和风险。
3. 系统体系结构设计的解决方案。包括相关的体系结构模式、各个级别的体系结构设计方案，相应的视图。
4. 系统体系结构设计的质量分析和评估。
5. 系统概述

## **2.1系统简介**

学生期末考试成绩信息管理是学校管理中一项常见、必需但又繁琐的工作。本系统为学生期末考试成绩信息管理提供了方便，实现学生期末考试成绩管理信息化，免去了人工统计学生考试信息的麻烦。主要涉及学生、教师、教务员等3个角色。各个角色的功能如下：

|  |  |
| --- | --- |
| 角色 | 功能 |
| 学生 | 查看基本信息 |
| 查询考试成绩 |
| 教师 | 查看基本信息 |
| 录入学生考试成绩 |
| 查询学生考试成绩 |
| 更改学生考试成绩 |
| 教务员 | 查看教师信息 |
| 审核学生考试成绩 |
| 发布学生考试成绩 |
| 授予老师修改学生成绩的权限 |

表格 1系统角色及功能

1. 体系结构需求
   1. 关键指标

系统投入使用后，需要满足各类学校师生的正常使用，在访问量和并发量迅速增加的情况下，也可在一定范围内正常使用，令众师生满意。系统对于用户量和课程量的指标最低要求如下:

|  |  |  |
| --- | --- | --- |
| 指标 | | 要求 |
| 用户 | 学生 | 30000 |
| 教师 | 2000 |
| 课程 | 课程量/每学期 | 500 |
| 学生容量/每门课 | 100 |

表格 2系统关键指标

## 体系结构用例

本系统有三个用户角色，学生，教师和教务员，另外还需要系统管理员/系统测试人员对系统进行日常维护和测试。其中，系统管理员/测试人员测试每个功能是否满足需求，并进行相应的日志记录，作为开发人员调试的参考依据。学生、教师、教务员在各自权限范围内可使用对应功能。

根据系统简介中各角色及功能说明，为系统绘制体系结构用例图，如下：



图表 2体系结构用例

## 3.3各相关方对体系结构的要求

### 3.3.1 客户相关方要求

本系统为学校设计，供学校中的师生们使用。学校除了要求满足关键指标（3.1）中指明的以外，还需要求工期、经费在一定范围内。图示如下：

|  |  |  |
| --- | --- | --- |
| 指标 | | 要求 |
| 工期 | 最佳 | 三个月 |
| 最多 | 六个月 |
| 经费 | 最佳 | 100000RMB |
| 最多 | 300000RMB |

表格 3客户相关方要求

### 3.3.1 开发相关方要求

开发方需要满足客户方提出的功能要求及性能、可靠性等非功能要求，并在客户方要求的工期内交付，经费控制在合理范围内。另外，从开发方的角度考虑，还需要满足一些别的要求：

1. 服务器最大负载量

开发方所设计的产品，其服务器的最大负载量必须高于客户方所要求的最大可容纳用户访问量，以保证产品在使用时的稳定性及密安性。

1. 数据库最大可容纳数据量

对于该学生期末考试成绩信息管理系统，其后台的数据库要求最大可容纳数据量要大于该产品所需最大可存储总信息量（最大可存储课程量+最大可存储用户量）。

1. 产品架构稳定性

数据库在进行海量数据处理时，若负载量较大，则很容易出现高并发死锁现象。因此需要使用服务器集群数据库集群进行负载均衡，以提高其架构稳定性。

1. 产品寿命

一个服务面向学校的大型综合系统，其产品寿命至少应在30~50年以上。并且系统内各模块的相对独立性较高以至于能保证每次系统更新换代的繁琐性可以被大大降低。

## 3.4约束条件

### 3.4.1 基于数据库的约束条件

1.主键约束（Primary Key Constraint）：唯一性，非空性

2.唯一约束（Unique Constraint）:唯一性，可以空，但只能有一个

3.检查约束（Check Constraint）:对该列数据的范围、格式的限制（如学号、性别等）

4.默认约束（Default Constraint）:该数据的默认值

5.外键约束（Foreign Key Constraint）:需要建立两表间的关系并引用主表的列

### 基于整体设计的约束条件

1. 网站开发可用的高级开发语言：J2EE 或 PHP
2. Web服务器部署于Linux系统之上以保证提供稳定可靠的服务
3. 为了系统将来的可扩展性，系统硬件使用国际通用的硬件，不应使用具有针对性的硬件。整个系统也应尽量减少各模块间的调用，尽量做到松耦合
4. 基于系统安全和保密性的考虑，系统的配置文件、数据存储文件等应进行加密处理，采用国际通用的加密算法，防止意外泄露或恶意攻击
5. 基于系统可靠性的考虑，系统的数据存储文件应进行冗余备份，比如磁盘冗余阵列存储RAID等

## 3.5 非功能需求

1.性能

该系统服务于学校内部，一般情况下较为稳定，且由于有严格的权限管理，所以不易出错。系统在运行过程中，可以负担得起大量（万人以上）的用户进行使用，且延迟较短，响应时间不会超过3秒。任何情形下进入该系统进行输入或访问数据库，延迟均不会超过3秒钟。

2.可依赖性

该学生期末考试成绩信息管理系统必须保证其高度的可信赖性，比如：内部数据库的管理、系统对于意外的错误处理等，保证其MTBF至少必须大于5万小时.

3.可用性

该学生期末考试成绩信息管理系统应保证高度的可用性，尤其是在用户使用的“高峰期”（大量用户同时登录该系统），要保证其可用性高于95%。

4.密安性

1)系统必须严格保护数据库中的学生成绩信息及各类用户信息不被丢失或发生错误，出错率应小于0.1%。

2)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.

3)系统应当能够保证不会因恶意攻击而崩溃，系统开发过程中不应存在明显漏洞.

4)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.

5.可维护性

1)易分析性：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.

2)易更改性：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.

6.可移植性

1)易安装性：系统应当能够保证在Windows、Linux等多平台上容易安装和部署.

2)共存性： 系统应当能够和其他软件共存于一个平台上，存在冲突的软件不超过.

3)易替换性： 系统应当能够容易地被卸载，也容易被更高版本的系统替换.

## 3.6 风险

### 3.6.1 来自内部计划过程的风险

计划是项目进度控制、协调的基础和依据，其质量的好坏对项目有序、高效的管理起到了至关重要的作用，只有在好的计划基础上，项目的进度管理才能事半功倍。与其它阶段的进度控制一样，设计阶段中进度控制的基础也是计划，计划的质量对后来的协调与控制有着重要的作用，计划的好坏为设计阶段的进度控制带来或大或小的风险。

### 3.6.2来自内部设计过程的风险

设计项目能否获得足够多的、优秀的设计人员是应对此类风险的重要条件，由于设计存在专业性，因此那些对后面工作影响大的工序的专业人才的需求更加主要。由于公司往往承接不止一个项目的，在多个项目并行时，不同项目的经理、负责人还需要对公司中有限的人力资源进行争夺，因此产生了新的风险。

### 3.6.3来自外部的风险

随着设计的进展，客户方对设计的具体情况有了进一步的了解，与开发方的想法结合起来，对设计项目组提出设计要求变更，其中有合理的，也有不合理的，但一般都会对设计进度造成影响。对于这种风险，一般可以通过合同、制度等进行控制，如在合同中对变更作出规定，将不必要的变更排除，减少变更对设计的不必要的干扰；也可以通过在项目预算和进度计划中预留一定的浮动时间，增加灵活性来进行消化与管理；还可以通过建立良好的沟通渠道，保持信息畅通，既减少了信息交流的时间与花在信息交流上的精力，又可通过与业主的及时协调，减少不必要的变更。

# 4. 解决方案

## 相关的体系结构模式

常见的信息管理系统采用的体系结构模式可以作为参考和借鉴。一般来说，市面上广泛流传的信息管理系统采用的是B/S模式。因此，建议采用B/S模式进行信息管理系统的开发。

B/S模式下，常用的体系结构模式有：

1. lamp：即 Linux+Apache+Mysql/MariaDB+Perl/PHP/Python,是一组常用来搭建动态网站或者服务器的开源软件，共同组成一个强大的Web应用程序平台。[1]
2. J2EE: 包含了很多容器和组件。其中，web开发组件是UNIX+Tomcat+Orecle+JSP的组合，主要是用JAVA编写程序。[2]
3. .Net : 基于Windows的开发框架。是Windows server +IIS+SQL Server+ASP的组合。[2]

对以上三种体系结构开发框架进行比较：

|  |  |  |  |
| --- | --- | --- | --- |
| 性能比较 | LAMP | J2EE | ASP.NET |
| 运行速度 | 较快 | 快 | 快 |
| 开发速度 | 快 | 慢 | 快 |
| 运行耗损 | 一般 | 较小 | 较大 |
| 难易程度 | 简单 | 难 | 简单 |
| 运行平台 | Linux/UINX/Windows平台 | 绝大多数平台均可 | Windows平台 |
| 扩展性 | 好 | 好 | 较差 |
| 安全性 | 好 | 好 | 较差 |
| 应用程度 | 较广 | 较广 | 较广 |
| 建设成本 | 非常低 | 非常高 | 高 |

表格 4动态网站相关体系模式之比较 [2]

体系结构模式的选择：LAMP模式或J2EE模式。若经费充足，工期较长，选择J2EE模式；若经费限制较大，工期较紧，且性能及维护性要求不十分严格，选择LAMP模式。理由如下：

1. ASP.NET 只能运行于Windows平台下，扩展性较差，系统的移植可能性不高，因此不作考虑。
2. 采用LAMP模式开发的软件项目投资成本较低，且从网站的流量上来说，70%以上的访问流量是LAMP提供的。[1]同时，根据比较结果来看，LAMP较为简单，开发速度也较快，扩展性和安全性也好，因此可以作为备选方案。
3. J2EE是企业级商业应用普遍采用的框架。采用该框架开发的软件运行速度快，运行耗损小，性能优良；同时，该框架较为完善，相对来说易于维护。但是开发难度较大，建设成本很高。因此，因此在经费和工期充足的条件下，首选J2EE模式。

## 体系结构概述

本系统采用J2EE体系结构模式作为基本构架。针对系统需求和功能进行具体设计。

### 概念级体系结构

本系统在概念体系结构上分为多个部件和连接器。部件：表现层、应用层、持久化层，连接器：各部件间的连接与交互。

其中，概念级体系结构中，表现层负责向用户展示系统的页面及功能，应用层负责处理用户的请求并返回相应数据，持久化层负责将用户的信息和数据保存在数据库中，并做必要的数据备份和恢复。图示如下：



图表 3概念级体系结构

### 模块化体系结构

本系统各个部件可以再细化为多个模块，模块间通过输入输出数据进行连接交互。其中，应用层可以划分为控制模块、业务逻辑模块。持久化层可划分为数据访问模块、数据存储模块。各层及模块间组成和关系如下：

表现层使用html标签语言和javascript、Ajax脚本语言向用户展示动态网页，用户浏览点击页面，从而表现层向应用层发起http请求。应用层控制模块接收请求，向业务逻辑模块发送命令，业务逻辑模块处理请求，并从持久化层获取数据，处理完毕后返回响应，最终将结果展示给用户。



图表 4模块化体系结构

### 代码体系结构

代码体系结构中，将各模块的代码文件及目录作了简单划分。图示如下：



图表 5代码体系结构

表示层Web页面文件由配置文件web.xml和用户登录注册页面、各用户主页面组成。应用层控制模块中展示了Servlet顶层基本类，业务逻辑模块表明，各个用户角色至少需要一个独立的类来实现其业务逻辑。持久化层数据访问模块表明需要实现访问数据库及各个实体的类。

其中，各层的代码文件可作为开发方的参考，具体实现可以有变化。

### 运行时体系结构

1. 每个用户均为一个进程，独立地访问服务器，发送请求并查看结果。
2. 服务器一般为分布式架构，例如分布式对象泛型或者RMI 泛型。可以有一台服务器作为Master服务器，管理Slave服务器的资源配置，负载均衡等。多台服务器作为Slave服务器，负责实际表现层、应用层、持久化层数据访问模块的实现。
3. 数据库服务层负责持久化层数据存储模块的实现，为各Slave服务器提供数据支持，各Slave服务器访问同一个数据库服务器，便于维护数据的一致性。

图示如下：



图表 6运行时体系结构

### 计算机硬件体系结构

1. 客户进程可基于PC端或移动终端，需要保证各终端上的网页显示均为正常且具有一定的美观性。
2. Master服务器和各Slave服务器位于共同的局域网内，网络带宽应为100M及以上，提供通畅的数据传输。为了保证数据处理及计算的高效，各服务器的处理器应为八核及以上，内存64G及以上，硬盘应为5T及以上。
3. 数据库服务器建议搭建在Linux平台上，并持续运行。为了保证高效运行，CPU应为八核及以上，内存16G及以上。为了数据的备份和恢复，可采用磁盘RAID阵列，进行可靠存储。

计算机硬件体系结构图示如下：



图表 7计算机硬件体系结构

## 结构化视图

### 用例视图

1. 学生用例视图：

学生使用账户名（推荐为学生学号）和密码登录系统后，可以查看个人信息和期末考试成绩，之后退出。同时还可以修改密码。图示如下：



图表 8学生用例

1. 教师用例视图

教师使用账户名（推荐为教工号）和密码登录系统后，可以查看个人信息、所教授学生的期末考试成绩、对应教务员的通知。如果通知需要录入学生的考试成绩，则录入学生的考试成绩；如果通知需要修改某学生的考试成绩，则修改相应学生的考试成绩，之后退出。同时还可以修改密码。图示如下：



图表 9教师用例

1. 教务员用例视图：

教务员通过帐户名（推荐为教工号）和密码登录系统后，可以查看所管理教师的信息。还可以审核学生的考试成绩，如果无最新考试的成绩信息，则给对应教师发布通知，通知其录入学生的考试成绩；如果审核发现某学生的考试成绩有误，通知对应教师修改该学生考试成绩，且赋予该教师修改成绩的权限。审核结束并且无误后，发布学生考试成绩，从而供学生和教师查看。之后退出系统。同时，还可以修改密码。图示如下：



图表 10教务员用例

### 逻辑视图

#### 概述

逻辑视图将详细阐述代码体系结构中的包和类的组成及关系。其中，系统的包有：Browser, XML documents, Client scripts, HTML pages, Business, Persistence。

Browser, XML documents, Client scripts, HTML pages为表现层实现所用包。

1. Browser即浏览器，维护用户访问cookies和DOM对象。
2. XML文件为配置信息，配置主HTML页面及必要的跳转。
3. HTML页面即为展现给用户的各个页面，包含各种静态文本和动态表单信息。
4. Client scripts为实现页面美观及动态验证等功能的包，包括各种脚本文件和样式表文件

Business为应用层实现包，包含实现各种业务逻辑的Java对象。

Persistence为持久化层实现包，包含访问数据库各个实体的Java对象。



图表 11逻辑视图顶层包

#### 影响软件体系结构的重要设计包

##### Business包

Business为应用层实现包，包含实现各种业务逻辑的Java对象。其中的类有：

Student, Teacher, ManageTeacher.

Student.java实现学生的各种业务逻辑，Teacher.java实现教师的各种业务逻辑，ManageTeacher.java 实现教务员的各种业务逻辑。

其中，一个教师教授多个学生，一个教务员管理多个教师。

下图给出了Business包中的UML类图：



图表 12逻辑视图重要设计包Business

### 其他视图说明

其他视图还包括进程视图、部署视图和实施视图。

1. 进程视图类似运行时体系结构图(图表6)，不再图示。每个用户是一个进程，向服务器发起请求。Master服务器中包含实现各Slave服务器负载均衡的进程，根据用户请求向Slave服务器发起远程过程/方法调用的进程。Slave服务器包含处理用户请求的线程，向数据库获取及更新数据的线程。
2. 部署视图类似计算机硬件体系结构图(图表7)，不再图示。
3. 实施视图是从开发的角度来描述系统架构，包括其整体结构、层次结构、子系统，以及要使用的第三方控件，自定义控件，以及它们之间的接口。其中，整体结构、层次结构、子系统等在其他体系结构和视图中有明确展示，且第三方控件暂不需要，因此不再赘述。

# 5. 系统的质量分析和评价

## 5.1 场景分析

### 5.1.1用例场景

描述系统的典型使用，可以启发出实际的应用情况。

1. 用户期望一个账号不被允许同时异地登录，该场景代表了用户期望系统具有高度的密安性。
2. 当出现异常数据时，系统应及时显示警告消息给管理员并在异常数据未被处理之前发送提醒消息给所有正在登录的用户，同时在屏幕上用红色字体显示出来。该场景代表了用户期望的可靠性。
3. 用户在操作系统时，其最大响应时间不超过3秒。该场景代表了系统性能要求。

### 5.1.2 增长性场景

预期未来系统修改时可能发生的场景。

1. 当用户量剧增，服务器负载过重时，可以考虑使用服务器集群，均衡单个服务器的负载量，从而降低远程登录用户操作该系统时的响应时间。
2. 通过扩充现有的数据库表的规模，把检索时间降低到平均1秒之内。

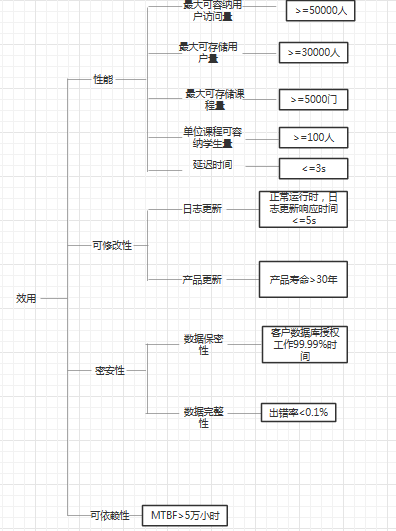
### 5.1.3探索性场景

可以涵盖极端的修改情况，期望对系统进行“压力”确认。

1. 改进系统的可使用性，从95%提升到99.999%。
2. 正常情况下，当一般服务器宕机时，不影响整个系统的可使用性。

## 5.2 原型分析

我们使用效用树法进行原型分析，直接和有效地将系统的业务具体分解到质量的属性场景。其中，性能是系统成功的核心，因为必须保证客户可以正常、稳定地使用该系统。可修改性、密安性、可依赖性，也是系统成功的关键所在，必须使系统能够保证客户数据的高度保密性以及能够响应市场竞争带来的不断变化。图示如下：



图表 13效用树