# 架构评估报告

1. ATAM方法简介

### 1、概述

ATAM（Architecture Tradeoff Analysis Method）：SEI提出的一种软件构架评估方法。

### 2、主要目的

1> 提炼出软件质量属性需求的精确描述；

2> 提炼出构架设计决策的精确描述；

3> 评估这些构架设计决策，并判定其是否令人满意的实现了这些质量需求。

### 3、ATAM评估方法

并非把每个可以量化的质量属性都进行详尽的分析，而是使众多的风险承担者（包括经理、开发人员、测试人员、用户、客户等等）都参与进来，由此而达到上述目标的。

ATAM是一种挖掘潜在风险，降低或者缓和现有风险的软件构架评估方法。因此，以下三点是评估中要特别注重的：风险、敏感点和权衡点。

### 4、构架涉众

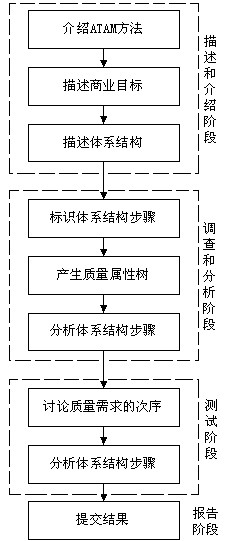
·普通用户

·用户管理员

·开发人员

·测试人员

### 5、评估步骤



1. 商业目标

**从开发组织角度：**开发一个模块性强、实时高效、界面良好、与外部其他系统兼容良好的系统，这使得开发组织能够把整个产品或某个模块卖给其他客户，同时由于良好的界面和业务处理效率而受市场欢迎。

**从客户角度：**系统容易操作，可维护性好、系统稳定、可以及时准确的处理用户的业务。

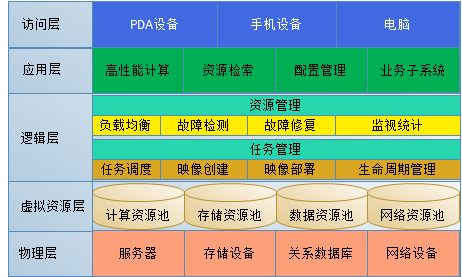
根据上述目标，质量属性可以划分为两类：

高优先级质量属性：性能、安全性、易用性、可用性、鲁棒性

重要但优先级较低的属性：可维护性、可修改性、可测试性

1. 描述架构

### 1、系统框架



### 2、质量属性及采取的战术

|  |  |  |
| --- | --- | --- |
| **目标** | **实现方式** | **所采取的战术** |
| **性能** | 用户访问的系统应该能在规定的时间内做出响应，如果系统由于网络或者数据库的原因不能在规定时间内作出反应，那么系统应该提出警告，不能出现让用户无故长时间等待的情况。 | 限制访问队列  缓冲池技术 |
| **可用性** | 当系统发生异常时，能在一定时间内恢复。 | 异常检测 |
| **安全性** | 当遭受攻击时，系统应该能提出警告并保证数据安全。 | 异常检测、降级 |
| **易用性** | 用户对系统的操作能得到正确及时的反馈。 | 支持用户主动 |
| 系统操作不应该太难。 | 分层设计 |
| **可维护性** | 系统运行有日志记录 | 日志记录工具 |
| **可修改性** | 在变更到达时，系统在时间和预算内完成所有测试和部署的变更 | 局部化修改  防止连锁反应  推迟绑定时间 |
| **可测试性** | 在完成系统开发的一个增量后，较轻松地对软件进行测试 | 输入/输出 |

1. 质量属性效用树

|  |  |  |  |
| --- | --- | --- | --- |
| **质量属性** | **属性求精** | **场景编号** | **场景** |
| **性能** | 响应时间 | XN01 | 在系统处于高峰期间，保证每个用户的操作能在5s内被响应。如需等待，则给出提示。（H,H） |
| 吞吐量 | XN02 | 系统可以保证同时响应2000个用户的操作。（H,M） |
| **可用性** | 容错性 | KY01 | 能够容忍用户在使用过程中发生的操作错误，并且能方便得从错误中恢复，保证系统不受或尽可能少受用户错误操作的影响。（M,M） |
| 备份与恢复 | KY02 | 备份时间应尽可能的短且在用户访问极少时进行。系统崩溃时能在1小时内恢复访问。（H,M） |
| 硬件更换 | KY03 | 硬件发生故障时，可以方便更换。（H,L） |
| **安全性** | 机密性 | AQ01 | 允许用户查看自己的信息，但不能查看他人信息。系统管理员不能随意查看用户隐私。（H,M） |
| 封闭性 | AQ02 | 对局域网外的用户来说，不能直接访问数据库，更不能随意对其进行修改。（H,M） |
| 防止恶意攻击 | AQ03 | 杜绝非法用户对系统进行攻击，使正常用户无法使用。（H,M） |
| 数据的完整性 | AQ04 | 在并发用户多的情况下，系统能保证数据的完整性。（H,L） |
| **易用性** | 界面友好，操作简单 | YY01 | 可以让用户迅速学会使用系统的方法（M,M） |
| 及时反馈性 | YY01 | 当系统发生错误，或程序运行时间过长时，用户界面应该为用户提供有意义的反馈信息。（H,M） |
| 界面一致性 | YY03 | 用户界面应该遵循一定的标准和常规，尽可能的在一个界面上完成，不要时常出现弹出框。（M,M） |
| **可维护性** | 易用维护 | KW01 | 系统有日志记录工具，能保存系统的运行记录。（M,L） |
| **可修改性** | 功能扩展 | KX01 | 增加其他功能能在一天内完成，并且不影响系统的其他功能（M,M） |
| 界面修改 | KX02 | 易于修改（M,M） |
| 文档完备 | KX03 | 系统必须提供详尽可靠的文档，以便于维护人员维护。（H,L） |
| 可配置性 | KX04 | 维护人员可以方便的配置系统参数、业务参数。（H,L） |
| 可升级性 | KX05 | 客户端发生缺陷后，可以自动更新。（H,L） |
| **可测试性** | 类的测试 | KC01 | 每个类和函数都应该单独测试，以验证其正确性。（M,L） |
| 系统功能模块测试 | KC02 | 对与系统功能相对应的模块进行测试，以保证业务的完备性。（M,M） |
| 系统的性能测试 | KC03 | 对系统进行压力测试，看能否达到设计时的访问量。（H,M） |
| Beta测试 | KC04 | 邀请用户代表进行Beta测试，体验界面的友好性和响应速度。（H,L） |

1. 质量场景的构架分析

在质量属性效用树中，我们对场景的优先级进行了划分，而同时由于分析时间宝贵，所以我们应该把宝贵的分析时间最先用于最重要且最难实现的场景上，即标注为 (H,H)的场景。在质量属性效用树的表格中，仅在性能这个质量属性下发现标注有(H,H)的场景，下面根据系统的体系结构和实现质量属性所采用的战术给出该场景的构架方法分析表格。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 场景号 | 场景：系统访问量达到高峰 | | | |
| 属性 | 性能 | | | |
| 环境 | 系统处于高峰访问时期 | | | |
| 刺激 | 用户请求访问 | | | |
| 响应 | 良好的响应请求 | | | |
| 构架决策 | 敏感点 | 权衡点 | 有风险决策 | 无风险决策 |
| 超出限制访问量的请求放在等待队列中 | S1 |  | R1 |  |
| 缓存 | S2 |  | R2 |  |
| 每个IP每次只允许同时发送一个请求 | S3 | T1 | R3 |  |
| 数据库连接池 | S4 |  |  | N1 |
| 推理 | 1. 由于在系统部署的时候用的是单应用服务器，而一台应用服务器可同时支持的并发用户数是很少的，在几千甚至上万的用户访问系统时，由于限制最大访问量，不能保证每个用户都能随时登陆，降低了用户的满意度。 2. 单服务器提供的缓存数有限。 3. 避免了非法用户的恶意攻击，但有可能降低系统的可用性 4. 减轻数据库的负担，提高系统的性能。 | | | |

1. 对系统架构的再分析

### 风险决策和敏感点

|  |  |  |
| --- | --- | --- |
| 采用战术 | 敏感点（S） | 有风险决策（R） |
| 超出限制访问量的请求放在等待队列中 | 提高了系统的稳定性和可用性，减少了崩溃的可能 | 会降低最大并发数目，使得用户等待时间过长，可能造成用户不满 |
| 缓存 | 提高系统的访问速度和性能 | 单服务器提供的缓存数目有限，并发用户数多的情况下，系统处理缓慢。 |
| 每个IP每次只允许同时发送一个请求 | 合理的要求，避免了非法用户的恶意攻击 | 可能将降低了易用性，但系统的安全性提高了 |
| 数据库连接池 | 数据库连接池允许应用程序重复使用一个现有的数据库连接，而不是重新建立一个，提高应用系统的性能 |  |
| 容错性 | 能够对用户出现的误操作进行检测和处理，并给出相应的处理信号，可以提高系统的可用性 |  |
| 系统备份与恢复 | 增强系统的容错能力 | 操作系统和数据库软件发生崩溃时，恢复时间较长 |

### 2、问题分析

在前面对系统结构的描述中，系统采用基于B/S的分层结构，系统部署在一台应用服务器上，这种结构有它独特的优点。但经过构架方法的分析，特别是对系统的关键质量属性和优先级最高的质量属性场景的分析，发现系统在上述场景下会出现如下的问题：

**性能方面**：在非常多的用户并发操作的情况下，单服务器系统将不能对用户的请求做出及时的响应，严重情况下服务器还会崩溃。

### (3)改进系统的构架

考虑到使用该系统的用户数目非常庞大，这样造成用户对系统的访问请求数目和对系统进行业务操作的请求数目也非常庞大，改进后的系统采用多层分布式结构，使用Web服务器集群和应用服务器集群来实现，这种集群机制支持动态负载平衡（Load Balance）和容错机制，可以将用户的请求以及对用户请求的处理分发到负载低的服务器中，非常适合具有并发用户数多，服务地点分散等这些特点，有较高的稳定性，能有效避免访问流量过多导致服务器瘫痪以及整个系统因为某台服务器崩溃而彻底瘫痪。

1. 评审结论

总体而言，通过对质量属性场景的分析，我们发现了最先提出的构架方案的不足，由此得出改进后的构架方案。采用改进后的构架方案可以获得良好的性能、易用性、安全性、可用性等等，达到了设计目的符合质量属性需求分析的要求！