## 质量属性

- 不精确的需求会造成很多问题。

  “我的系统一定要是快速的/安全的/可扩展的”

- 系统的质量属性必须是精确的/可度量的:

  “它必须能够从最初100个地理上分散的节点扩展到1000个节点，而不会增加安装和配置的工作量和成本。

## 什么是质量属性?

质量属性种类:

- 性能 (Performance)
- 安全性(Security)
- 可用性（Availability）
- 可扩展性（Scalability）
- 易用性(Usability)
- 可靠性(Reliability)
- 可移植性(Portability)
- 可修改性(Modifiability)
- 可维护性(Maintainability)

## 架构与质量属性

质量属性的实现必须在设计、实现和部署的过程中考虑。

举例:

- 易用性可以从架构方面和非架构方面考虑:
  - 使用户界面易于使用是非架构方面
  - 为用户提供undo/cancel操作是架构方面

## 架构与质量属性

举例说明:

- 可修改性
  - 功能如何划分(架构方面)
  - 模块内编程技术(非架构方面).
- 性能
  - 组件间的通信量&共享资源如何分配 (架构方面)
  - 算法的选择&算法如何实现 (非架构方面)

## 质量属性冲突

系统的质量属性经常会互相影响

影响有时是正面的，有时是负面的

譬如:

- 组件粒度大会提高性能，但会降低可维护性。
- 引入冗余数据可提高可用性，但会使安全性的保障更加困难。
- 将安全性相关的功能本地化通常意味着更多的通信以致降低性能。

## 质量属性场景

质量属性场景是一个质量属性特定的需求。由六部分组成:

- 激励源– 产生激励的实体。
- 激励 – 影响系统的事件。
- 环境– 激励发生的特定条件

## 质量属性场景

质量属性场景的六个部分

- Artifact(工件） – 接受激励的系统或系统部分
- Response(响应) –激励到达后发生的活动
- Response measure(响应度量)– 响应发生时，应当以某种方式进行度量以测试需求是否被满足。

## 质量属性场景的六个部分