软件系统设计与体系结构

作业一

1．应用基于情景的（刺激 - 反应）分析方法

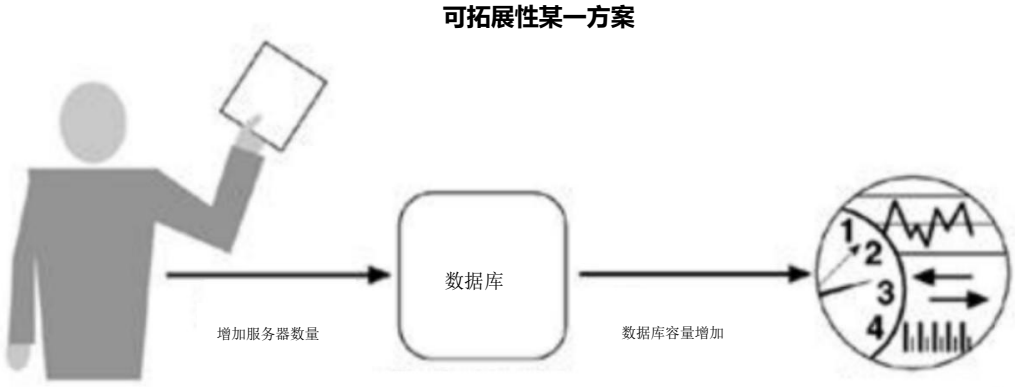
- extensibility vs. Scalability(可扩展性 vs 可延展性)

- 可扩展性（Scalability）是指系统/网络/软件能被快速方便应用到的不断增长的工作中的能力。例如，好的数据库系统可以通过增加硬件设施扩大数据库容量。

可扩展性一般情景生成

场景的一部分	可能的值
源	系统外部，系统内部
刺激	需要升级系统
制品	系统
环境	正常模式，维护模式
响应	数据库容量增加；功能增强；速度加快
响应度量	性能测试，用户数，运行速度

可拓展性某一方案



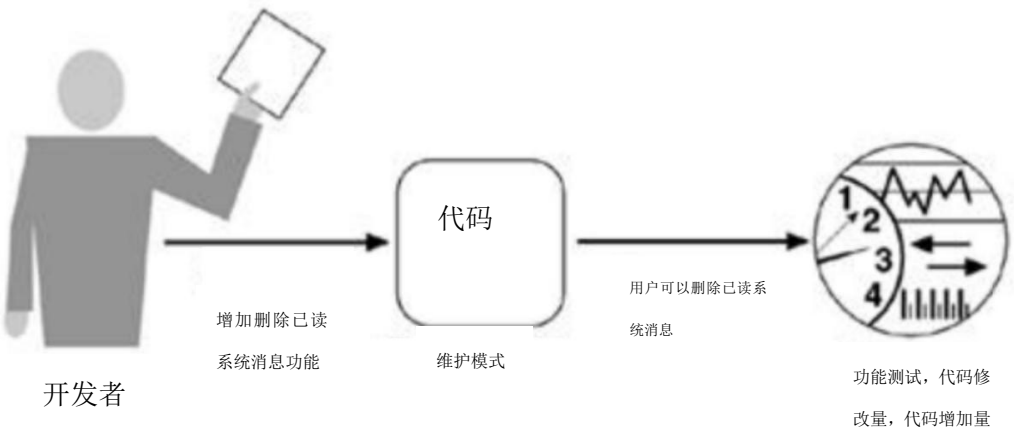
开发者　　维护模式　　数据库容量，查询速度，数据库修改速度

- 可延展性 （Extensibility）是指系统能够方便的增加组件、扩充功能的能力。例如，好的（松耦合）软件在增加/修改功能时不需要花很多时间去处理耦合关系，可以轻易的延展其功能。

可延展性一般情景生成

场景的一部分	可能的值
源	系统内部
刺激物	系统需要增加修改功能
制品	代码
环境	维护模式
响应	系统功能得到增加或修改
响应度量	代码修改效率，代码增加量，代码修改量，修改代码分布情况，功能测试

可延展性某一方案



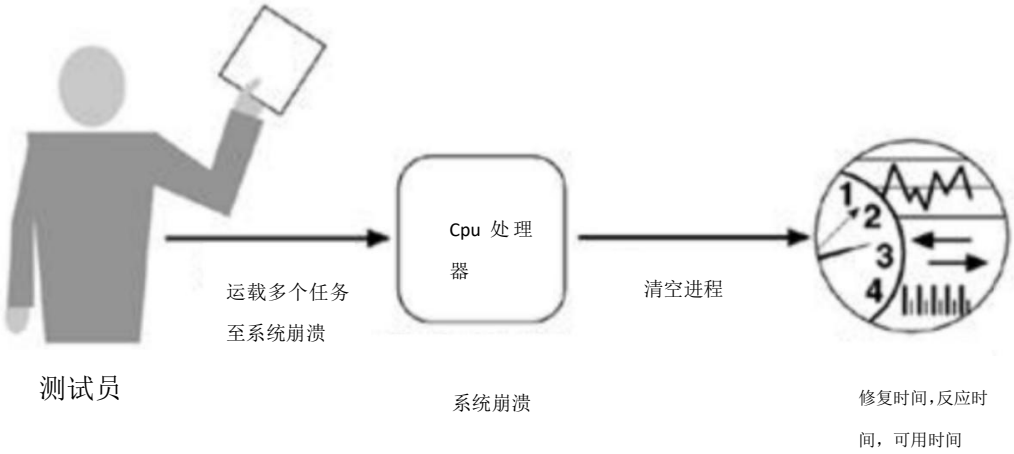
- stability vs. Reliability（稳定性 vs 可靠性）

■ 稳定性（Stability）：与修改所造成的未预料结果的风险有关的软件属性。

稳定性一般情景生成

场景的一部分	可能的值
源	系统内部，系统外部
刺激物	错误: 疏忽: 构件未能对某个输入作响应。 崩溃: 构件不断遭受疏忽的错误。 时间: 构件做出了响应，但响应时间太早或者太迟。 响应: 无响应.
制品	指定系统可靠性的资源:处理器，通信通道，进程，永久存储。
环境	系统崩溃，系统正常运行
响应	系统出现崩溃的时候，应该具备的反应，比如: 记录崩溃，通知适当的各方，禁止导致崩溃的事件源，在一段时间内不可用，在降级模式下运行等等。
响应度量	系统必须可用的时间间隔，可用时间，系统在降级模式下运行的时间间隔，修复时间等。

稳定性某一方案

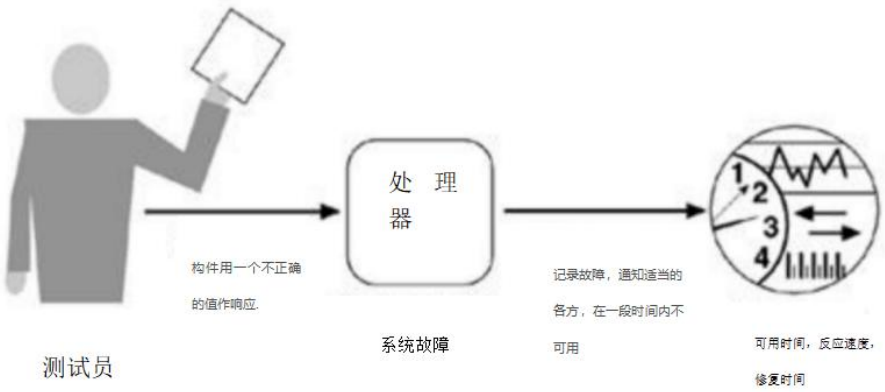


- 可靠性是指在规定运行条件下和规定时间周期内，与软件维护其性能级别的能力有关的一组属性。可靠性反映的是软件中存在的需求错误、设计错误和实现错误，而造成的失效情况。

可靠性一般情景生成

场景的一部分	可能的值
源	系统内部，系统外部
刺激物	错误: 疏忽: 构件未能对某个输入作响应。 崩溃: 构件不断遭受疏忽的错误。 时间: 构件做出了响应，但响应时间太早或者太迟。 响应: 构件用一个不正确的值作响应.
制品	指定系统可靠性的资源:处理器，通信通道，进程，永久存储。
环境	当出现错误或者故障的时候，系统状态的期望值。比如:如果看到了一系列的错误，可能希望关掉系统。 如果看到了第一个错误，可能希望只是对功能的降级。
响应	系统出现故障的时候，应该具备的反应，比如: 记录故障，通知适当的各方，禁止导致故障的事件源，在一段时间内不可用，在降级模式下运行等等。
响应度量	系统必须可用的时间间隔，可用时间，系统在降级模式下运行的时间间隔，修复时间等。

可靠性某一方案



2．根据您的研究（而不是讲义）分析每对中两个质量属性之间的关系并进行比较，并用适当的例子进行详细说明。

- extensibility vs. Scalability(可扩展性 vs 可延展性)

可扩展性是系统，网络或流程处理越来越多的工作的能力，或者是为了适应这种增长而扩大其潜力的能力。例如，如果系统在添加资源（通常是硬件）时能够在增加的负载下增加其总输出，则认为系统是可扩展的。当在经济背景下使用该词时，隐含着类似的含义，其中公司的可扩展性意味着基础商业模式为公司内部的经济增长提供了潜力。

可伸缩性作为系统的属性通常难以定义，并且在任何特定情况下，有必要在那些被认为重要的维度上定义可伸缩性的特定要求。这是电子系统，数据库，路由器和网络中非常重要的问题。在添加硬件后，与增加的容量成比例地改善性能的系统被称为***可扩展系统***。

的算法，设计，网络协议，程序，或其它系统被说成*比例*，如果它是适当有效的当应用于大的情况下（例如，大的输入数据集，有大量的输出或用户，或大量的和实用在分布式系统的情况下参与节点）。如果设计或系统在数量增加时失败，则*不会扩展*。在实践中，如果存在大量影响缩放的事物（n），那么资源需求（例如，算法时间复杂度）必须小于 n^2，因为 n 增加。一个例子是搜索引擎，它不仅可以扩展用户数量，还可以扩展其索引的对象数量。可扩展性是指随着需求的增加，网站的规模增加的能力。

可扩展性的概念在技术和业务设置中是期望的。基本概念是一致的 – 企业或技术接受增加的数量而不影响贡献边际（= 收入 – 可变成本）的能力。例如，给定的设备可能具有 1-1000 个用户的容量，而超过 1000 个用户需要额外的设备或性能将下降（可变成本将增加并且减少贡献边际）。

另一个例子是事件指挥系统（ICS），即美国各响应机构使用的应急管理系统。例如，ICS 可以将资源协调从单引擎路边火灾扩展到州际野火。现场的第一个资源建立了 IC，有权在控制范围内订购资源和委派责任

（管理五到七名官员，他们将再次委派多达七名，随着事件的发展而继续）。由于复杂性保证，高级官员在顶层担任命令。这个经过验证的系统非常简单，完全可扩展，近半个世纪以来一直在拯救生命和财产。

可延展性是一种软件工程和系统设计原则，其中实现考虑了未来的增长。术语可延展性也可视为延展系统能力的系统度量和实现延展所需的工作量。扩展可以通过添加新功能或通过修改现有功能来实现。中心主题是提供变更 - 通常是增强功能 - 同时最大限度地减少对现有系统功能的影响。

可延展系统的内部结构和数据流最小或不受新功能或修改功能的影响，例如，在创建者或其他程序员更改系统行为时，可能不需要重新编译或更改原始源代码。由于软件系统使用寿命长，并且会针对用户要求的新功能和附加功能进行修改，因此可延展性使开发人员能够延展或增加软件的功能并促进系统重用。它的一些方法包括允许插入用户自己的程序例程的工具，以及定义新数据类型以及定义新格式化标记标记的能力。

- stability vs. Reliability（稳定性 vs 可靠性）

  稳定性是：表征给定系统变化的敏感性，这是系统变化可能造成的负面影响。

  可靠性是一个主要特征，包含：

  - 成熟度：该子特征涉及软件故障的频率。
  - 容错：软件承受（和恢复）组件或环境故障的能力。
  - 可恢复性：能够将故障系统恢复到完全运行状态，包括数据和网络连接。

例子：
目标：编写一个程序来添加两个数字

可靠但不稳定：

```
add(a,b):
    if randomInt mod 5 == 0:
        throw exception
    else
        print a+b
```

稳定但不可靠：

```
add(a,b):
    if randomInt mod 5 == 0:
        print a+a
    else
        print a+b
```

3．讨论至少两种可能的策略来改善每个质量属性; 为每种可能的策略提出至少两种不同的策略，并确定它们对每对质量属性的潜在影响。 使用包含质量属性列和策略和策略行的表来描述每种策略可能带来的潜在影响的好处和惩罚。

- extensibility vs. Scalability(可扩展性 vs 可延展性)

质量属性	策略	好处	坏处
可扩展性	*负载可扩展性*：扩展和收缩其资源池，以适应更重或更轻的负载或输入数量。	可以容易地修改，添加或移除系统或组件，以适应变化的负载	对硬件要求较高
	地理可扩展性：从局部区域的集中扩展到更分散的地理模式	保持性能，实用性或可用性。	难以管理
可延展性	白盒：这种可扩展性的形式下，可以通过修改源代码来扩展软件系统	它是最灵活，限制最少的形式	工作量较大
	黑盒子：黑盒扩展通常通过系统配置应用程序或通过定义组件接口来使用特定于应用程序的脚本语言来实现。	方便，可以任意无关人员进行	没有关于系统实现的细节用于实现部署或扩展； 仅提供接口规范

- stability vs. Reliability（稳定性 vs 可靠性）

质量属性	策略	好处	坏处
稳定性	时间检测：系统一直长久运行，直到系统崩溃	成本低	花费时间长
	任务检测：过多任务加载，直到系统崩溃	应对反应速度快，直接了当	花费时间短，成本高
可靠性	多用户并发访问	有使用性	有一定偶然性，成本高
	错误信息发送测试	成本低，应对反应速度快，直接了当	测试可能不全面