来自林连南的第二次作业题目布置的回答情况：

答案均来自各位的回答，希望大家见怪不怪

## 第四章：

|  |  |
| --- | --- |
| **（1）原文** | What is the relationship between a use case and a quality attribute scenario? If you wanted to add quality attribute information to a use case, how would you do it? |
| **（2）翻译** | 用例和质量属性场景之间有什么关系？ 如果要将质量属性信息添加到用例中，您将如何进行？ |
| **（3）解答** | 用例和质量属性场景：相同事情的不同方法  在软件工程中，用例描述用户想要实现的功能，而质量属性方案包含功能在设计的应用程序中如何运行的描述。功能和质量属性是正交的，如果要将质量属性信息添加到用例，那就添加新的用例  质量属性场景在质量属性需求规范中的作用与用例在功能需求规范中所扮演的角色相同。  每个用例提供了一个或多个场景，该场景揭示了系统是如何同最终用户或其它系统交互的，从而获得一个明确的业务目标。用例要避免技术术语，取而代之的是最终用户或者领域。 |

|  |  |
| --- | --- |
| **（1）原文** | Do you suppose that the set of tactics for a quality attribute is finite or infinite? Why? |
| **（2）翻译** | 你认为质量属性的一套策略是有限的还是无限的？为什么呢？ |
| **（3）解答** | 经过我们讨论，是无限的，策略的重点是单一质量属性响应。在一种策略中，没有考虑权衡。必须由设计者明确考虑和控制权衡。策略将重叠，经常可以选择多种策略来改善特定的质量属性。选择使用哪种策略取决于诸如其他质量属性之间的权衡和实施成本等因素。这些考虑因素超越了对特定质量属性的策略的讨论。我们提出的策略可以而且应该得到改进。考虑性能：计划资源是一种常见的性能策略。但是，为了特定目的，需要将此策略细化为特定的调度策略，例如最短作业优先级，循环法等。使用中介是一种可修改的策略。但是有多种类型的中介（层，代理和代理，仅举几例）。因此，架构师将采用改进来制作每种具体的策略。 |

|  |  |
| --- | --- |
| **（1）原文** | Consider the choice between synchronous and asynchronius comunication(a choice in the coordination mechanism category) .What quality attribute requirement might lead you to choose one over the other? |
| **（2）翻译** | 请考虑在同步和异步通信机制之间进行选择（在协调机制下选择）。是什么质量属性要求导致你选择这一个而不是另一个？ |
| **（3）解答** | 同步通信效率更高，更加适合对速度要求高的传输，在性能的质量属性要求下我会选择同步通信；异步通信点对点，对时序要求低，在易用性质量属性要求下我选择异步通信。  同步是当用户发送一个请求，需要等待返回后，才能发送下一个请求。即，在此等待过程中不能关闭用户界面，用户会有相应的等待时间。  异步是增加了消息队列。将用户请求放入队列中，等待返回。即，在此过程中，用户不用 |

|  |  |
| --- | --- |
| **（1）原文** | Consider the choice between statfeful and stateless comunication(a choice in the coordination mechanism category) .What quality attribute requirement might lead you to choose one over the other? |
| **（2）翻译** | 请考虑在有状态和无状态通信机制之间进行选择（在协调机制下选择）。是什么质量属性要求导致你选择这一个而不是另一个？ |
| **（3）解答** | 有状态通信无需额外的调用，低延迟的优点，在性能和可用性的质量属性要求下我会选择有状态通信；无状态通信有防止数据丢失的优点，在安全性的质量属性要求下我会选择无状态通信。 |

|  |  |
| --- | --- |
| **（1）原文** | Think about the screens that your favorite automatic teller machine uses.What do those screens tell you about binding time decisions reflected in the architechture? |
| **（2）翻译** | 想想你最喜欢的自动柜员机使用的屏幕。这些屏幕所说的绑定时间决策如何反映在架构上？ |
| **（3）解答** | 通过推迟绑定时间，来使设计者（银行维护者）在维护方面来做到即使用户做出了修改，也不会影响到数据的改变。在屏幕上设置注册界面，使得用户有插入银行卡类似的注册动作。 |

|  |  |
| --- | --- |
| **（1）原文** | Most peer-to-peer atchitechture employs late binding of the topology.What quality attributes does this promote or inhibit? |
| **（2）翻译** | 大多数p2p架构使用后期绑定的拓扑，什么质量属性促进还是抑制 ？ |
| **（3）解答** | 对等网络，即对等计算机网络，是一种在对等者之间分配任务和工作负载的分布式应用架构，是对等计算模型在应用层形成的一种组网或网络形式。  其可以定义为：网络的参与者共享他们所拥有的一部分硬件资源（处理能力、存储能力、网络连接能力、打印机等），  这些共享资源通过网络提供服务和内容，能被其它对等节点直接访问而无需经过中间实体。  在此网络中的参与者既是资源、服务和内容的提供者，又是资源、服务和内容的获取者，也就是同一资源客户越多，共享就越快。 |

|  |  |
| --- | --- |
| **（1）原文** | Discuss the choice of programming language (an example of choice of technology) and its relation to architecture in general, and the design decisions in the other six categories? For instance, how can certain programming languages enable or inhibit the choice of particular coordination models? |
| **（2）翻译** | 讨论编程语言的选择（例如选择一门技术）及其与架构的大体关系，以及其他六个类别的设计决策？ 例如，某些编程语言如何允许或禁止选择特定的协作模型？ |
| **（3）解答** | 一个好的软件架构能够满足系统的品质，能够支持计划编制过程，对系统开发的指导性，能够有效的管理复杂性，为复用奠定了基础。绝大多数架构或者编程语言的产生都是来源于项目。比如php言语属于开源言语，利于学习，运用普遍。只能用来开发网站，无法用来开发软件或其他的一些应用，局限性比拟大。普通运用php开发的企业网站，通常会运用网上现成的开源CMS来搭建程序。 |

|  |  |
| --- | --- |
| **（1）原文** | We will be using the automatic teller machine as an example throughout the chapters on quality attributes. Enumerate the set of responsibilities that an automatic teller machine should support and propose an initial design to accommodate that set of responsibilities. Justify your proposal. |
| **（2）翻译** | 我们将在整个章节中使用ATM作为质量属性的示例。 列举自动柜员机应支持的一系列功能，并提出初始设计以适应这组功能。 证明你的提议是正确的。 |
| **（3）解答** | 刺激源 用户  刺激 办理业务  制品 系统  环境 正常运行  响应 请求被处理  响应度量 响应时间不超过5秒  4.安全性  刺激源 非授权用户  刺激 试图非法操作  制品 系统中的数据  环境 正常操作下  响应 对用户验证，阻止操作  响应度量 操作被拒绝  5.可测试性  刺激源 单元开发人员  刺激 已完成构架和子系统的集成  制品 代码段  环境 开发时  响应 准备集成环境  响应度量 执行测试的时间  6.易用性  刺激源 最终用户  刺激 办理业务  制品 系统  环境 在运行时  响应 根据操作处理相应业务  响应度量 95%的用户对结果满意 |

## 第五章

|  |  |
| --- | --- |
| **（1）原文** | 1. Write a set of concrete scenarios for availability using each of the possible responses in the general scenario.  2. Write a concrete availability scenario for the software for a (hypothetical) pilotless passenger aircraft. |
| **（2）翻译** | 1. 可用性的通用场景下有多个可能的回应，为每个回应写一组一般场景。  2. 为 (假设的) 无人驾驶客机编写具体的软件可用性场景。 |
| **（3）解答** | 1. A. Log the fault：把错误记录到日志里。比如硬盘检测到坏块时会把相关信息记录到日志里。  B. Notify appropriate entities (people or systems)：通知恰当的实体（人或系统）。如编译失败时编译器显示错误信息。  C. Disable source of events causing the fault：使导致错误的事件源失效。如过滤不正确的输入。  D. Be temporarily unavailable while repair is being effected：修复时暂停服务。操作系统重启时修复系统错误。  E. Fix or mask the fault/failure or contain the damage it causes：修复或屏蔽错误，或者控制损失。比如屏蔽发生错误的硬件。  F. Operate in a degraded mode while repair is being effected：修复时在降级模式下操作。如错误率过高时无线网卡降低传输速率。  2. A. Log the fault：把错误记录到日志里。如把速度传感器的错误信息记录到黑匣子中。  B. Notify appropriate entities (people or systems)：通知恰当的实体（人或系统）。如油料不足时通知地面控制中心。  C. Disable source of events causing the fault：使导致错误的事件源失效。如定时除冰使速度传感器不被冰覆盖。  D. Be temporarily unavailable while repair is being effected：修复时暂停服务。如发生机械故障时自动迫降。  E. Fix or mask the fault/failure or contain the damage it causes：修复或屏蔽错误，或者控制损失。如电路故障时停止照明。  F. Operate in a degraded mode while repair is being effected：修复时在降级模式下操作。如机舱泄露时到低空飞行，就近迫降。 |

|  |  |  |
| --- | --- | --- |
| **（1）原文** | 3. Write a concrete availability scenario for a program like Microsoft Word.  4. Redundancy is often cited as a key strategy for achieving high availability. Look at the tactics presented in this chapter and decide how many of them exploit some form of redundancy and how many do not. | |
| **（2）翻译** | 3.为类似于Microsoft Word的程序编写具体的可用性场景  4. 冗余通常被认为是实现高可用性的关键策略。 看看本章介绍的策略，并决定其中有多少采用某种形式的冗余，有多少则没有。 | |
| **（3）解答** | 3.A. Log the fault：把错误记录到日志里。记录使用word时可能产生的错误信息（无法打开文档的问题，格式转换的问题，图片显示问题，单词拼写问题，语法问题等）。  B. Notify appropriate entities (people or systems)：通知恰当的实体（人或系统）。如无法打开文档的问题会通知给开发者，单词拼写问题、语法问题会显示在屏幕上通知给使用者。  C. Disable source of events causing the fault：使导致错误的事件源失效。如单词拼写问题会用红线标注给与警示。  D. Be temporarily unavailable while repair is being effected：修复时暂停服务。如无法打开文档则会提醒错误并停止尝试打开。  E. Fix or mask the fault/failure or contain the damage it causes：修复或屏蔽错误，或者控制损失。如无法转换格式则会自动保存原格式。  F. Operate in a degraded mode while repair is being effected：修复时在降级模式下操作。如无法显示图片则会预留图片空间，待使用者解决。  4. 在Detect Faults（Ping/Echo, Monitor, Heartbeat, Timestamp, Sanity Checking, Condition Monitoring, Voting, Exception Detection, Self-Test）和Prevent Faults (Removal from Service, Transactions, Predictive Model, Exception Prevention, Increase Competence Set)的过程中没有采用冗余.  在Recover from Faults的过程中：  Active redundancy: 采取主动冗余  Passive redundancy: 采取被动冗余  Spare: 采取被动冗余  Exception Handing: 没有采用冗余  Rollback: 采取主动冗余和被动冗余结合  Software Upgrade: 采取主动冗余或者被动冗余二者都可  Retry: 没有采用冗余  Ignore Faulty Behavior: 没有采用冗余  Degradation: 没有采用冗余  Reconfiguration: 没有采用冗余  Shadow: 没有采用冗余  State Resynchronization: 没有采用冗余,但是工作的时候和主动冗余以及被动冗余合作  Escalating Restart: 采用了被动冗余(Level 0)  Non-Stop Forwarding (NSF): 采取主动冗余  **有一份不同的答案** | |
|  | Exploit some form of redundancy | Without any redundancy |
|  | Condition Monitoring (待定)  Voting  Active Redundancy  Passive Redundancy  Spare  Rollback  Software Upgrade  Shadow  State Resynchronization  Escalation Restart  Non-Stop Forwarding  Predictive Model  Exception Prevention | Ping/Echo  Monitor  Heartbeat  Timestamp  Sanity Checking  Exception Detection  Self-Test  Exception Handling  Retry  Ignore Faulty Behavior  Degradation Reconfiguration  Removal from Service  Transactions  Increase Competence Set |

|  |  |
| --- | --- |
| **（1）原文** | 5.How does availability trade off against modifiability ? How would you make a change to a system that is required to have “24/7” availability (no scheduled or unscheduled downtime , ever)? |
| **（2）翻译** | 5.可用性与可修改性如何进行权衡？当需要具有“24/7”可用性时应该如何更改系统（没有计划或非计划的停机时间）？ |
| **（3）解答** | 可用性指标是架构设计的重要指标，对外是服务承诺，对内是考核指标。将系统分层、分割的主要目的是增加系统可维护性，以及对后期发展提供更好的功能扩展能力、并发处理能力。在进行权衡时需要根据具体的使用场景以及需求来决定，比如在项目会有多次进行迭代的情况，而产品在迭代的过程中本身会暂时影响了正常使用，初期阶段重视可修改性会对项目进程有很好的帮助。而当项目趋向于稳定时更应该重视可用性以便提供更好的客户体验。  对于需要’24/7’可用的情况可以采用镜像的方式来确保业务的连续性。 |

|  |  |
| --- | --- |
| **（1）原文** | 6. Create a fault tree for an automatic teller machine. Include faults dealing with hardware component failure, communications failure, software failure, running out of supplies, user errors, and security attacks. How would you modify your automatic teller machine design to accommodate these faults? |
| **（2）翻译** | 6. 为自动出纳机构造故障树。包括处理硬件组件故障，通信故障，软件故障，资源贫乏，用户错误和安全攻击等故障。 您如何修改自动出纳机设计以适应这些故障？ |
| **（3）解答** | F:\最后的TIPS\1213246931\Image\C2C\AA10A7B3BF27CA0893E812B990EA41C3.jpg  作业第六题  在自动出纳机程序编写的时候增加相应的查错机制，同时删除一些长久不需要存储的、无关紧要的数据，对自动出纳机进行定期维修，检查软硬件使用情况。 |

|  |  |
| --- | --- |
| **（1）原文** | 7.Consider the fault detection tactics(ping/echo, heartbeat , system monitor, voting, and exception detection). What are the performance implications of using these tactics？ |
| **（2）翻译** | 7.考虑故障检测策略（响应机制，心跳机制，系统监控，投票和异常检测）。 使用这些策略有什么性能影响？ |
| **（3）解答** | Ping/echo（响应）  通过系统监视器发出Ping请求来探测服务的通信路径是不是通的，某个组件是不是还在工作。  监视(Monitor)  通过使用一个监视器的组件来检测其他组件的健康状况。  心跳(Heartbeat)  通过组建定区发出一个心跳信息，来主动的告诉系统它正在工作。  投票(Voting)  使用冗余的组件做同一件事情，以相同的输入，如果产生不同的输出，则忽略少数，采纳多数的结果。  性能：性能是指软件系统及时提供相应服务的能力。包括速度、吞吐量和持续高速性三方面的要求。  以上的可用性策略都需发出请求并通过返回的信息判断组件的可用性，在一定程度上会增加系统负载，提高系统响应时间，  但也提高了服务可用时间段，例如，一年内90%可用性的系统最多有36.5天的停机时间，使用可用性策略，可使系统可用性达到95%或以上  使用可用性策略或多或少都会影响系统的响应速度，即使它可以增加服务时间，你需要决定性能或可用性在当前环境下哪个更关键一点 |

## 第六章

|  |  |
| --- | --- |
| **（1）原文** | Find a web service mashup. Write several concrete interoperability scenarios for this system. |
| **（2）翻译** | 找一个web 服务的复合应用程序实例，并写出几个这个系统的具体的互操作场景。 |
| **（3）解答** | 使用外部系统登录   |  |  | | --- | --- | | 互操作性 | | | 刺激源（Source of stimulus） | 用户 | | 刺激（Stimulus） | 使用外部系统如QQ、微信等方式登录本系统 | | 制品（Artifact） | 使用外部系统登录后的系统 | | 环境（Environment） | 正常操作下 | | 响应（Response） | 登录成功 | | 响应度量（Response measure） | 99.9%登陆成功 |   导入facebook好友   |  |  | | --- | --- | | 互操作性 | | | 刺激源（Source of stimulus） | 用户 | | 刺激（Stimulus） | 点击导入facebook好友按钮 | | 制品（Artifact） | 显示好友照片、状态信息的页面 | | 环境（Environment） | 正常操作下 | | 响应（Response） | 导入成功 | | 响应度量（Response measure） | 99.9%导入成功 |   添加新浪微博数据源   |  |  | | --- | --- | | 互操作性 | | | 刺激源（Source of stimulus） | 用户 | | 刺激（Stimulus） | 添加第三方账户（如新浪微博账号等） | | 制品（Artifact） | 可使用新浪微博数据源的系统 | | 环境（Environment） | 网络连接正常下 | | 响应（Response） | 添加成功 | | 响应度量（Response measure） | 99.9%添加成功 | |

|  |  |
| --- | --- |
| **（1）原文** | What is the relationship between interoperability and the other quality attributes highlighted in this book? For example, if two systems fail to exchange information properly, could a security flaw result? What other quality attributes seem strongly related (at least potentially) to interoperability? |
| **（2）翻译** | 互操作性和这本书里面强调的其他的质量属性的关系怎么样？例如，如果两个系统不能正常地交换信息，可能会导致一个安全瑕疵（隐患）吗？哪些别的质量属性看上去和互操作性有紧密的关系（至少有可能是）？ |
| **（3）解答** | 互操作性指的是系统内或者系统之间不同的组件可以有效地进行信息交换，通常是以服务（Service）的形式来进行的。互操作性的关键因素包括通信协议，接口定义，数据格式的定义等等，而标准化是实现互操作性的重要手段。互操作性与其他质量属性关系如下：   1. 可用性Availability   互操作性对可用性具有良好的保证，因为对于可用性来说，用户对于操作可能会出现各种的问题，而互操作性会在一些服务场景上降低错误的发生，在很大的程度上增加了补救措施的种类。   1. 可修改性Modifiability   在使用具有互操作性强度高的系统时，系统的可修改性会提高很多。在进行本系统的更新时，系统自身的可修改性会因互操作性的提高而降低开发成本。   1. 性能Performance   由于互操作性的介入，系统的性能可能会随着功能的完善，导致性能的提高。由于互操作性的介入会使很多板块的灵活性增加，届时使得整个系统更加具备高内聚、低耦合的特征，同时使得整个系统可以更加有效的得到优化，提高运行效率。   1. 安全性Security   安全性可能会降低，比如一个论坛如果安全性较低，那么和他具有良好互操作性的一些外部系统的安全性将会受到极大的不确定性保证。   1. 可测试性Testability   互操作性较高的情况下，可测试性将会降低。介于安全性的考虑因素，某些外部系统对于测试会有高度敏感性。因此具有良好互操作性的系统在可测试性上会具有相对不好的表现。   1. 易用性Usability   在一定应用场景下，高度互操作性的软件会使得软件的易用性提高。具备不同的系统之间的交互，在一定情况下会降低用户的学习成本和使用成本，因为用户可以借由其他系统的信息进行合适的了解性操作。例如我们使用QQ登录一些论坛时，就降低了我们的注册成本或者是输入账号密码的登录成本。  如果两个系统不能正常的交换信息，可能会是一个安全隐患导致的。安全性，可用性，可修改性，看起来和互操作性关系很紧密。 |

|  |  |
| --- | --- |
| **（1）原文** | 3.Is a service-oriented system a system of system? If so, describe a serviceoriented system that is directed, one that is acknowledged, one that is collaborative, and one that is virtual. |
| **（2）翻译** | 面向服务的系统是否为一个由很多系统组成的更大系统？如果是的话，请分别描述一个具有以下性质的面向服务系统： |
| **（3）解答** | 当一个面向服务的系统是由许多各自提供服务的“小型”系统组成，以提供具有一致目的完整服务的更庞大系统时，可以认为该面向服务系统是一个“系统的系统”（System of System，缩写为 SoS）。这里的“小型”系统是针对这个面向服务的“大型”系统来说的，并不是指系统的规模很小。教材中关于SoS的定义为：  If you have a group of systems that are interoperating to achieve a joint purpose, you have what is called a system of system (SoS). An SoS is an arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities.  大致翻译一下，就是：SoS是由一组集成的系统组成，以实现一个联合目的。SoS是一个对各个系统的  管理和组织，使得各个独立的系统能够被集成到一个更大的系统，来实现特定的功能。按照组织和管理  性质，SoS系统可以划分为Directed, Acknowledged, Collaborative, Virtual：     Directed（直接性）：这些“小型”系统具有组成一个SoS的一致目的，有着中心化管理、经费运作和权力管理的SoS组织。这些“小型”系统是SoS的下属。   Acknowledge（公认性）：这些各自独立的系统和SoS属于同一层次的级别，各自发展、管理、运营，同时受到同个级别的SoS的中心化管理、协同。   Collaborative（协作性）：这些独立的系统没有受到一致的组织和中心化管理，各自独立存在。   系统们自愿一起协作，以达到共同的利益   Virtual（虚拟性）：和Collaborative类似，但是这些独立的系统之间甚至不清楚与自己协作的系统的身份。  具有以上性质的面向服务系统  1.属于“直接性”（Directed）类别的面向服务系统：互联网后端开发的微服务架构  在类似Spring Cloud这样的微服务架构开发中，一个完整的系统后端由许多个提供独立功能的小型后端系统组成，这些小型系统被称为“微服务”，它们各自具有完整、独立的后端组件，能够提供特定的独立的服务，但是单一的微服务并不足以提供构成用户需要的、有着完整业务逻辑的服务。整个微服务系统的统一管理和协调多个微服务，并添加了网关（gateway）、服务发现组件（service discovery）等来提高、保证微服务之间的互操作性。    【图】一个我自己写的基于Spring Cloud的小型电影微服务项目的架构图  2.属于“公认性”（Acknowledged）类别的面向服务系统：简易消息聚合  简易消息聚合（RSS, Really Simple Syndication）是一种基于XML标准，在互联网上被广泛采用的内容包装和投递协议。以新闻订阅为例子，在一些新闻门户网站，聚合了许多来源于其他媒体发布的新闻信息，以提供多种多样的新闻消息，方便用户根据阅读兴趣订阅。各个媒体独立发展、运营，提供RSS新闻消息供门户网站使用，虽然这些媒体提供的服务受到门户网站一定程度的管理和约束，但是这些媒体和门户网站之间是同一级别的平等关系，因此这样新闻RSS的系统属于“公认性”类别的面向服务系统。  3.属于“协作性”（Collaborative）类别的面向服务系统：  教材中给出的相关例子是：谷歌发布并维护谷歌地图的服务，其它需要地图服务来实现自己服务的各个系统调用谷歌地图的接口。每个系统各自独立发展，并没有一个统一的系统来管理它们，这些系统调用谷歌地图的服务，相互协作使得系统正常运行。（其实我觉得这个例子讲得不是很恰当和清晰。。  4.属于“虚拟性”（Virtual）类别的面向服务系统：区块链网络  在一些匿名区块链网络中，各个独立存在的系统只要调用区块链网络提供的服务来实现自己相关的功能，并不需要公开自己的身份。在这种场景下，这些系统依赖于该区块链网络的服务，但是又不受严格的约束，只要遵循一些特定的接口规范即可。这些系统独立性很强，组成了一个属于”虚拟性“类别的面向服务系统。 |

|  |  |
| --- | --- |
| **（1）原文** | 5.Why has the importance of orchestration grown in recent years? |
| **（2）翻译** | 为什么最近这些年来协调的重要性一直在增长 |
| **（3）解答** | “Orchestrate is a tactic that uses a control mechanism to coordinate and manage and sequence the invocation of particular services (which could be ignorant of each other).”这是课本中对Orchestrate 的定义。Ochestration 本意为管弦乐，此处是编排的意思。互操作性中有两个策略，分别为定位接口以及管理接口。而Orchestrate 便是管理接口的两个方法之一。这些年来orchestrate策略被人们越来越重视的原因，我觉得应该是和基于部件的技术的发展有关。软件部件是指在软件系统设计中能够重复使用的建筑模块，它包装了一系列互相关联的操作和服务。当各个部件之间需要用一种非常复杂的形式来完成一个很复杂的任务时，编排这项策略便会被使用，它负责描绘这些交互的过程的蓝图。这有点像mediator,这个设计模式。按我的理解的话，就像是虚拟化出一份现实中不存在的部件，它充当媒介，对外，与其他部件的接口进行交互，在内部则实现复杂的逻辑功能。这个与高内聚低耦合的软件设计模式也是十分吻合的。所以Orchestrate这项策略的重要性也就不言而喻的，善用Orchestrate能够使我们的系统更加的简洁，可复用性及可操作性提高，这也是软件工程这些年的目标吧，所以说，这些便是Orchestrate这项策略这些年来越来越受重视的原因。 |

|  |  |
| --- | --- |
| **（1）原文** | If you are a technology producer, what are the advantages and disadvantages of adhering to interoperability standards? Why would a producer not adhere to a standard? |
| **（2）翻译** | 如果你是一个技术生产者，坚持互操作性准则的优缺点有哪些？为什么生产者不会遵守一个准则？ |
| **（3）解答** | 优点:提供已有功能给更多应用，获取其他功能，更好的可用性，可修改性  缺点:降低产品竞争力，快速发展的技术打破操作闭环，安全性降低  理由:产品独立性，保持闭环，避免竞争，难以平衡多个准则 |

|  |  |
| --- | --- |
| **（1）原文** | 1. Universal Description, Disco, and Integration(UDDI) was touted as a discovery service, but commercial support for UDDI is being withdrawn. Why do you suppose this is? Does it have anything to do with the quality attributes delivered or not delivered by UDDI solutions? |
| **（2）翻译** | 统一描述、发现和集成（UDDI）被吹捧为发现服务，但UDDI的商业支持正在被撤销。 你认为这是为什么？ 是否与UDDI解决方案提供或未提供的质量属性有关？ |
| **（3）解答** | UDDI的衰落是必然的。  UDDI在提出时听起来是个好主意，他是一种基于[XML](https://zh.wikipedia.org/wiki/XML" \o "XML)的跨平台的描述规范，可以使世界范围内的企业在[互联网](https://zh.wikipedia.org/wiki/%E4%BA%92%E8%81%94%E7%BD%91" \o "互联网)上发布自己所提供的服务。具有如下好处：   * 强移植性，任何规模的行业或企业都能得益于 UDDI。 * 高性能，在 UDDI 之前，还不存在一种 Internet 标准，可以供企业为它们的企业和伙伴提供有关其产品和服务的信息。也不存在一种方法，来集成到彼此的系统和进程中。 * 强扩展性，定义一旦首选的企业被发现后如何启动商业 * 扩展新客户并增加对目前客户的访问 * 满足用户驱动的需要，为在全球 Internet 经济中快速合作的促进来清除障碍   UDDI的出现为web service提供了良好的理论基础和技术抽象，但是在实际使用中，它缺失了某些重要的软件质量属性：   * 过于复杂，低易用性。大型组织应该拥有其Web服务的目录，但这可以像维基页面一样简单。 * 忽略安全性。如果UDDI代理为用户动态选择服务提供商，用户就没有机会对服务的安全性进行任何尽职调查。 * 管理和收集小额支付的困难仍然存在。 * Web服务通常用于防火墙后面，用于SOA，将应用程序与业务伙伴集成，以及调用众所周知的API。对于这些目的，UDDI是完全多余的。 |

|  |  |
| --- | --- |
| **（1）原文** | With what other systems will an automatic teller machine need to interoperate? How would you change your automatic teller system design to accomodate these other systems? |
| **（2）翻译** | 自动柜员机需要和其他哪些系统进行交互操作？你会如何改进你的自动柜员机系统的设计来使它和其他系统和适应？ |
| **（3）解答** | （1）自动柜员机需要用到REST技术，对数据进行CURD操作，所以我认为它需要和本地或者远程数据库系统进行交互，这就要求能正确、准时、安全的完成相应的操作；（2）它是一种终端设备，所以要和人进行交互（如果把人看作一个系统的话），这要求它能提供友好的见面，傻瓜式的操作方法，简单易用；（3）它可能与电子货币（虚拟货币、银行卡、二维码、指纹、人脸等）有关，所以要和支付系统、安全保障系统进行交互（这里的支付系统、安全系统可能与之前的数据库系统有交集，但不完全相同），这就需要它可靠、稳定，出错率极低，出故障后恢复快等等。  综上，自动柜员机需要和数据库系统、支付系统、安全保障系统以及人进行交互。  改进措施：1. 准确而快速的连接数据库。可以提高数据存储设备的性能、容量，优化存储方式，及时清理垃圾数据，减少冗余，优化查询机制等等。这同时考虑到了互操作性质量属性中Discovery、Handing of the response两个重要的方面； 2. 简化操作，美化界面，以便用户能快速流畅的使用这个自动柜员机。要做好这方面，必须理解用户需求，做好调研、市场走向，做好充分的测试，将bug最小化。这主要考虑到了互操作性的Discovery方面；3. 力求搭建一条快速又安全的绿色支付通道。要充分与第三方（如果存在）进行沟通，与相关系统有一个完美的衔接过程，使得每一笔交易都能走好、走准属于它们的那一条路，并完善故障机制处理，增强系统的健壮性。这主要考虑到了互操作性的响应处理机制方面。 |

## 第七章

|  |  |
| --- | --- |
| **（1）原文** | 3.In a certain metropolitan subway system, the ticket machine accept cash but do not give change. These is a separate machine that dispenses change but does not sell tickets. In an average station there are six or eight ticket machines for every change machine. What modifiability tactics do you see at work in this arrangement? What can you say about availability? |
| **（2）翻译** | 在某大都会地铁系统中，售票机接受现金但不会找零钱。有一台独立的机器，它提供零钱但不卖票。平均每个车站有六台或八台售票机搭配一台换置零钱的机器。在这种安排下，你在工作中看到什么样的可适应性策略？关于可用性你能说些什么？ |
| **（3）解答** | * 我看到的策略主要有1）Reduce the Size of Module(减小模块规模)，卖票和找零属于不同的操作，将这两个模块分离开，方便了未来对系统的修改2）Increase Cohesion(增加内聚)，每一个模块只负责一个功能，不夹杂其他操作，实现了模块化，有利于整个系统的开发和运作，也方便调试系统，未来增加模块直接增加即可，不需要修改之前的模块 * Availability（可用性）是指能够满足用户的需求，在这里面由于模块化做的很好，这大大降低了系统出现问题的可能性，找零系统跟售票系统不属于同一个模块，当其他模块出问题的时候剩余模块还可以继续工作，这个系统属于一个High Availability的系统。 |

|  |  |
| --- | --- |
| **（1）原文** | 4. For the subway system in the previous question, describe the specific form of modifiability (using a modifiability scenario) that seems to be the aim of arranging the ticket and change machines as described. |
| **（2）翻译** | 对于前面问题中的地铁系统，描述可修改性的具体形式（使用可修改性场景）,该可修改性似乎是按照描述的方式安排票和兑换机的目的 |
| **（3）解答** | * 如上所述，在这里的可修改性具体体现形式在于分割（split module）模块和增加内聚（increase cohesion）。在此系统中，有六到八台售票机器搭配一台找零机器，这两种机器的要完成的功能不一样，售票机只售票，找零机只找零，根据他们不同的职责就可以将他们划分为两个模块，在划分模块的过程中既实现了分割模块的目的又实现了增加内聚的目的 * 场景 |

|  |  |
| --- | --- |
| **（1）原文** | 5.A wrapper is a common aid to modifiability. A wrapper for a component is the only element allowed to use that component; every other piece of software uses the component’s services by going through the wrapper. The wrapper transforms the data or control information fro the component it wraps. For example, a component may expert input using English measures but find itself in a system in which all of the other components produce metric measures. A wrapper could by employed to translate. What modifiability tactics does a wrapper embody? |
| **（2）翻译** | 5.包装是可修改性的常见辅助。组件的包装器是允许使用该组件的唯一元素; 每个其他软件都通过包装器使用组件的服务。包装器将数据或控制信息转换为它包装的组件。例如，组件可以使用英语测量的专家输入，但发现自己处于所有其他组件产生度量测量的系统中。可以使用包装器进行翻译。 包装器体现了什么可修改性策略？ |
| **（3）解答** | Wrapper embody 体现了cohesion 和 coupling 两个策略。首先原本系统中的每个组件都是相互独立的个体，因为包装器是唯一可以使用组件的元素，所以通过包装器将各个部件连接在了一起，这里体现了cohesion策略。其次，组件期望用英语度量作为输入，这里包装器就可以充当英语翻译的中介，让所有组件的输出都由包装器进行翻译，这样子，就不用每个组件自己进行翻译，减少了组件的coupling. |

|  |  |
| --- | --- |
| **（1）原文** | 6. Once an intermediary has been introduced into an architecture, some modules may attempt to circumvent it, either inadvertently (because they are not aware of the intermediary) or intentionally (for performance, for convenience, or out of habit). Discuss some architectural means to prevent inadvertent circumvention of an intermediary. |
| **（2）翻译** | 6.一旦将中介引入到体系结构中，某些模块可能会无意中（因为他们不了解中间人）或故意（为了方便，为了方便或出于习惯）而试图绕过它。 讨论一些建筑手段，以防止无意中规避中间人。 |
| **（3）解答** | 1.increase cohesion and reduce coupling ,通过这样使各个模块成为耦合度较低的的各部分，然后就要形成整体运作的时候就必须通过中介的作用将他们cohesion在一起。  2. split module, encapsulate, abstract common services等方法都可以。 |

|  |  |
| --- | --- |
| **（1）原文** | 7. In some project, deployability is an important quality attribute that measures how easy it is to get a new version of the system into the hands of its users. This might mean a trip to your auto dealer or transmitting updates it arrives. In project that measures deployability separately, should the cost of a modification stop when the new version is ready to ship？justify your answer. |
| **（2）翻译** | 在某些项目中，可部署性是一个重要的质量属性，用于衡量将新版本的系统交到用户手中的难易程度。 这可能意味着您的汽车经销商旅行或传输它到达的更新。在独立测量可部署性的项目中，如果新版本准备好发布，用来修改的开销是否应该停止？ |
| **（3）解答** | Answer: 是，在一个独立测量软件系统的可部署性系统，软件的基本功能能够完成即可，对于用户获取软件的便利程度才是最重要。在新版本发布之际，此时的修改会带来极大的不确定性，所以此时应尽量减少修改的开销，尽量不修改。 |

|  |  |
| --- | --- |
| **（1）原文** | 8. The abstract common services tactic is intended to reduce coupling, but it also might reduce cohesion. Discuss. |
| **（2）翻译** | 抽象公共服务策略旨在减少耦合度，但也可能降低内聚度，请讨论。 |
| **（3）解答** | Answer: 软件设计中通常用耦合度和内聚度作为衡量模块独立程度的标准。划分摸块的一个准则就是高内聚低耦合。耦合度从以下几个方面衡量：  （1）一个模块对另一个模块的调用；  （2）一个模块向另一个模块传递的数据量；  （3）一个模块施加到另一个模块的控制的多少；  （4）模块之间接口的复杂程度。  抽象公共服务是将两个模块中类似功能进行抽象统一接口，通过不同参数类型，提供不同服务。通过这种策略，代码量直接减少，模块间共用代码，直接降低耦合度。但是这也意味着多了一层抽象，致于类之间的关系变得模糊，引来了内聚度降低的问题。 |

|  |  |
| --- | --- |
| **（1）原文** | Modifiability comes in many flavors and is known by many names. Find one of the IEEE or ISO standards dealing with quality attributes and compile a list of quality attributes that refer to some form of modifiability. Discuss the differences. |
| **（2）翻译** | 可修改性有多种风格，并且有许多名称。找到处理质量属性的IEEE或ISO标准之一，并编制一系列质量属性，这些属性涉及某种形式的可修改性。并讨论它们的差异。 |
| **（3）解答** | 在课本194页图12.1展示了一个ISO标准( The ISO/IEC FCD 25010 product quality standard )。  涉及可修改性的质量属性：可用性(Availability), 可测试性(Testability)，性能效率(Performance efficiency),安全性(Security)  可用性：系统在考察时间内是否能够正常运行。  可测试性：发现软件系统错误，并且设计测试、执行测试的能力。  性能：一般指的是系统对于用户操作的响应时间的快慢。  安全性：系统保护数据和信息不被未授权访问所获取，而且提供数据信息给已授权的访问操作。 |

|  |  |
| --- | --- |
| **（1）原文** | For each quality attribute that you discovered as a result of the previous question, write a modifiability scenario that expresses it. |
| **（2）翻译** | 对于您在上一个问题中发现的每个质量属性，请编写一个表达它的可修改场景。 |
| **（3）解答** | **Availability:**  Source: Heartbeat Monitor  Stimulus: Server Unresponsive  Environment: Normal Operation  Response: Inform Operator Continue to Operate  Response Measure: No Downtime  **Testability:**  Source: Unit Tester  Stimulus: Code Unit Completed  Environment: Development  Response: Results Captured  Response Measure: 85% path Coverage in Three Hours  **Performance:**  Source: Users  Stimulus: Initiate Transactions  Environment: Normal Operation  Response: Transactions Are Processed  Response Measure: Average Latency of Two Seconds  **Security:**  Source: Disgruntled Employee from Remote Location  Stimulus: Attempts to Modify Pay Rate  Environment: Normal Operations  Response: System Maintains Audit Trail  Response Measure: Correct Data is Restored within a Day and Source of Tampering Identified. |

|  |  |
| --- | --- |
| **（1）原文** | For the subway system in the previous question, describe the specific form of modifiability (using a modifiability scenario) that seems to be the aim of arranging the ticket and change machines as described. |
| **（2）翻译** | 对于上个问题中的地铁系统，描述可修改性的具体形式（使用可修改性的场景），使它实现如描述安排售票机和零钱机的目的。 |
| **（3）解答** | Source：终端用户，地铁系统管理员，开发人员  Stimulus：买票或找零  Artifact：数据，功能组件，resources，configuration  Environment：compile time，initiation time，design time  Response：出售车票；找回等价零钱  Response measure：maybe in 3 minutes；多台机器同时维护 |

|  |  |
| --- | --- |
| **（1）原文** | Identify particular change scenarios for an automatic teller machine. What modifications would you make to your automatic teller machine design to accommodate these changes? |
| **（2）翻译** | 为自动取款机识别特定变化的情景。您对自动取款机的设计进行了哪些修改以适应这些变化？ |
| **（3）解答** | **Reduce size of a module:**  Split the automatic teller machine’s module as deposit module,withdrawal module,account management module,user login module and so on.  **Increase cohesion:**  To make the two module which have different responsibilities break,we can make a subdivision to some module and make reading bank card module,get cash module split.  **Reduce coupling:**  To accommodate the scenarios that providing service to different country’s people,we can split the language service by different languages.  **Defer binding:**  We can bind the ATM to more influencing factors and add more parameters. To make a better service for any clients,we can collect a few parameters,such as client’s age,nationality and using purpose then provide its financial service. |

## 第八章

|  |  |
| --- | --- |
| **（1）原文** | 1.“Every system has real-time performance constraints.” Discuss. Or provide a counterexample. |
| **（2）翻译** | “每个系统都会受到实时的性能约束”，请讨论这句话，或者给出一个反例。 |
| **（3）解答** | As is shown at the beginning of Chapter 8, performance is all about time and the software system’s ability to meet timing requirements. When events occur, like interrupts, messages, requests from users or other systems, or clock events marking the passage of time, the system, or some element of the system, must respond to them in time. For example, in complex measurement and control applications, real-time operating systems that require very high real-time performance must be used. Besides, performance depends on two aspects, the size of system and the ability of your system to compute. The size of system is fixed on development, and the ability to compute is fixed on deployment. In real-time running, every event will compete for the resources, or a interrupt occurred to occupy a portion of resources. The competition for resource is very common in a system. When the resources is occupied by some events, then other events can not fetch a resource, that is the bottleneck of a system. An extensively discussed example in the software design is the 12306 china railway system. The peak value of daily click in the system is about 157 billion times, occurred on January 24, 2018. However, 3 years ago, without a great design as now, facing to 36 billion peak value, about a quarter in 2018, the 12306 system are of high delay, hard to connect, showing inconsistent number of rest tickets and so on. This is the real-time performance. |

|  |  |
| --- | --- |
| **（1）原文** | 1. write a performance scenario that describes the average on-time flight arrival performance for an airline. |
| **（2）翻译** | 编写一个性能场景，描述航空公司平均情况下航班准时抵达的性能。 |
| **（3）解答** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Source: | stimulus: | artifact: | response: | response measure: | | Planes | Arrival or leaving | Schedule system | Schedule the flights | worst-case latency of 10min  average-case latency of 1min  Throughput of 5  Miss rate of 1% | |  |  | Environment: |  |  | |  |  | Normal operation |  |  |   In this scenarios, the planes arrival or leaving under normal operation. The schedule system schedules the flights with the worst-case latency of 10min, average-case latency of 1min, throughput of 5, miss rate of 1% |

|  |  |
| --- | --- |
| **（1）原文** | 1. Write several performance scenarios for an automatic teller machine. Think about whether your major concern is worst-case latency, average-case latency, throughput, or some other response measure. How would you modify your automatic teller machine design to accommodate these scenarios? |
| **（2）翻译** | 为ATM编写几种性能场景。考虑您的主要问题是最坏情况延迟，平均情况延迟，吞吐量或者其他一些响应措施。您如何修改ATM设计以适应这些情况？ |
| **（3）解答** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Source: | stimulus: | artifact: | response: | response measure: | | User | check balance  Transfer  Withdraw  Save | ATM system | Display the balance  Transfer are processed  Withdraw are processed  Save are processed | worst-case latency of 10s  average-case latency of 5s  Throughput of 1  jitter of 2s  Miss rate of 0 | |  |  | Environment: |  |  | |  |  | Normal operation |  |  |   In this scenarios, user ask for check balance, transfer or withdraw/save money under normal operation. The ATM system processes these requires with worst-case latency of 10s, average-case latency of 5s, Throughput of 1, jitter of 2s and Miss rate of 0.  The case for worst-case latency: improve the processing efficiency of the ATM  The case for average-case latency: use high efficiency algorithm  The case for miss rate: design the ATM to check the process result multiple times |

|  |  |
| --- | --- |
| **（1）原文** | 4.Web-based systems often use proxy servers, which are the first element of the system to receive a request from a client(such as your browser).Proxy servers are able to serve up often-requested web pages, such as a company’s home page, without bothering the real application servers that carry out transactions. There may be many proxy servers, and they are often located geographically close to large user communities, to decrease response time for routine requests. What performance tactics do you see at work here? |
| **（2）翻译** | 基于网络的系统经常使用代理服务器作为最先从客户端接受请求的构建。代理服务器可以在不增加处理业务的真实应用服务器负载的情况下，完成常用页面，例如公司主页的访问请求。许多代理服务器被部署在靠近用户社区的地方来降低访问延迟。在这个例子中用到了什么样的性能战术？ |
| **（3）解答** | 1. Reduce overhead tactic for control resource demand. For requesting the resources from the real serve, which needs a longer path also needs more network transmission overhead and time delay overhead. But if we set a neighbor server for the close user we can reduce these overhead and get the response faster. 2. Maintain multiple copies of data for manage resources. As a matter of fact, the resources maintained in the real server and the proxy server are the same, which is redundant. By using more memory space, we can reduce the requests of each resource and get our response faster. 3. Increase the resources. In this case we increase many proxy servers, which has the similar processing ability to the real server. So this tactic increase the resources. Different requests are distributed to different servers and the load of each server decrease. As the result, the total process time of the system decrease. |

|  |  |
| --- | --- |
| **（1）原文** | 5.A fundamental difference between coordination mechanisms is whether interaction is synchronous or asynchronous. Discuss the advantages and disadvantages of each with respect to each of the performance responses: latency, deadline, throughput, jitter, miss rate, data loss, or any other required performance-related response you may be used to. |
| **（2）翻译** | 协调机制之间的根本区别在于交互是同步还是异步。讨论这两种协调机制在以下几种性能响应方面的优缺点：延迟，截止时限，吞吐量，抖动，未命中率，数据丢失或您可能使用过的任何其他所需的性能相关响应。 |
| **（3）解答** | （1）Latency: Latency has more impact on synchronous interaction than asynchronous. For synchronous interaction, if the event couldn’t be processed by system, it will affect the operations which is waiting for the results of the process, while it has less impact on asynchronous interaction because the results of the process don’t have to be delivered back to where it comes from.  （2）Deadline: The impact resulted from deadline is similar to the one with latency. For synchronous interaction, it requires the system to response as soon as possible so as to improve the performance. For example, a client send a request about purchasing an item, if the server cannot respond in a limited deadline, then the client will probably get mad and cancel the transaction, which will also cause the merchants to loss. On the other hand, deadline has less impact on asynchronous interaction for the reason that the results may not that important with respect to time constrains.  （3）Throughput: Throughput is equally important to both types of interaction. Throughput is measured by the number of transactions the system can process in a unit of time. Either the synchronous interaction or the asynchronous interaction can improve performance by improving the throughput.  （4）Jitter: Jitter means the allowable variation in latency. Now that as is mentioned above about the latency, jitter has more impact on synchronous interaction than asynchronous. Asynchronous interaction allows a relatively larger range of time, while synchronous interaction won’t be likely to accept variation in latency.  （5）Miss rate: High miss rate is unacceptable to both synchronous interaction and asynchronous interaction. As a measure of performance response, the miss instead of right response will affect the performance, no matter what interaction it is.  （6）Data Loss: The situation about data loss is similar to miss rate. As a measure of performance response, the miss of arrival of events or responses from system will affect the performance, no matter what interaction it is. |

|  |  |
| --- | --- |
| **（1）原文** | 6.Find real-world (that is, non-software) examples of applying each of the manage-resources tactics. For example, suppose you were managing a brick-and-mortar big-box retail store. How would you get people through the checkout lines faster using these tactics? |
| **（2）翻译** | 请列举在非软件场景下应用每一个资源管理战术的例子。例如，假设你在管理大型实体零售店，你会如何使用这些战术让人们更快地埋单？ |
| **（3）解答** | 1. Increase resources. We will hire skilled cashiers rather than a green hand. For a skilled cashier, it need less time to server a consumer, so the line moves faster. 2. Introduce concurrency. Generally, in a brick-and mortar big-box retail store, there are many checkout counters. So we can checkout more consumers at the same time. For the consumers that wait in lines, the total time for waiting decrease. 3. Schedule resources. Make a suggestion for the consumers about which line they should wait in. We can estimate the waiting time and recommend the fastest line. 4. Bound queue sizes. If the store contains more people than the threshold, we should forbid the people coming continually to prove the serve quality. |

|  |  |
| --- | --- |
| **（1）原文** | User interface framework typically are single-threaded, Why is this so and what are the performance implication of this single threading |
| **（2）翻译** | 用户界面框架总是单线程的，为何如此？单线程的性能效果如何？ |
| **（3）解答** | 多线程的缺点：需要占用更多内存  程序更加复杂，出现BUG的概率会增大  有出现死锁的可能性  单线程在速度上不如多线程。  在实际使用中，用户即使运行多个程序，但同一时间也只会操作其中一个程序的界面；在操作界面时，人的反应速度远不如计算机的运算速度，单线程的运算速度足以应对数据的输入输出。 |

## 第九章

|  |  |
| --- | --- |
| **（1）原文** | Write a set of concrete scenarios for security for an automatic teller machine. How would you modify your design for the automatic teller machine to satisfy these scenarios? |
| **（2）翻译** | 请描述一组ATM（自动柜员机）的具体安全场景。并回答你将如何修改ATM的设计以满足这些安全需求？ |
| **（3）解答** | 见下表 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Source** | **Stimulus** | **Artifact** | **Environment** | **Response** | **Response Measure** | **Modify Plan** |
| Thief | Get money | Break the machine | Normal state | The bank will broadcast the alerts, and videotape the whole process | Guards get to the scene within 5 minutes | Lock the data in ATM when it is being damaged and Lock the bank’s door to avoid the criminal from escaping. |
| Thief | Get money | The card insert in the machine | When clients are drawing out money from ATM | System maintains the previous service | Drawback the card within 5 minutes | Add a detective sensor to distinguish foreign materials besides cards in the bayonet |
| Thief | Get money | The password of clients cards | When clients are drawing out money from ATM | / | / | Control unilateral visual angle within 30 degrees; Install a reflector on the ATM such that the clients know what’s happening behind them. |
| Thief | Get money | Get the money that clients have drawn out | When clients are drawing out money from ATM | / | / | Detect foreign materials in the banknote mouth. |
| Virus | Get access to and modify the data in the backend | Data within the system | When the machine is connected to the bank ‘s production network | System remains Audit Trail | Correct data is restored within a day and source of abnormal modification is detected | Add virus detection and port protection in the system; Control the execution of non-whitelist files to achieve security protection, while assisted by virus detection and killing, as well as virus traceability and site-specific clearance. |
| Virus | Change data | Data in the current clients’ account | Normal state | / | / | Add port protection and process protection in the system. |

|  |  |
| --- | --- |
| **（1）原文** | One of the most sophisticated attacks on record was carried out by a virus known as Stuxnet first appeared in 2009 but became widely known in 2011 when it was revealed that it had apparently severely damaged or incapacitated the high-speed centrifuges involved in Iran’s uranium enrichment program. Read about Stuxnet and see if you can devise a defense strategy against it based on the tactics in this chapter. |
| **（2）翻译** | 史上最复杂的攻击之一是由一种名为Stuxnet的病毒首次出现在2009年，但在2011年广为人知，当时它被证实显然严重损坏或使伊朗铀浓缩计划中涉及的高速离心机丧失能力。 阅读Stuxnet，看看你是否可以根据本章的策略设计防御策略。 |
| **（3）解答** | Stuxnet病毒与以往的病毒不同，其代码非常精密，可以通过网络传播，是首个针对工业控制系统编写的破坏性病毒，对大型工业、企业用户存在一定的风险。  针对Stuxnet病毒的特点，提供以下应对策略。   1. 检测攻击：当有设备进行服务器访问时，对签名进行检测，签名可以基于协议、TCP标志等。同时验证消息的完整性，避免未知数据进入。在打开附件或通过网络接收文件时，弹出安全警告或提示。 2. 抵抗攻击：当对设备或敏感数据进行操作时，识别并认证操作者，对不同的用户给予不同程度的访问限制。开启防火墙功能并下载最新补丁、更新病毒库。使用强口令。 3. 应对攻击：当检测到病毒已经入侵之后，及时撤销用户的访问权限并锁定电脑，通知操作员备份文件并查杀病毒。 4. 从攻击中恢复：利用备份的文件及日志恢复到病毒入侵之前的状态，同时可跟踪病毒，查找病毒源及其他受到或可能受到攻击的设备，进行应对处理。 |

|  |  |
| --- | --- |
| **（1）原文** | Some say that inserting security awareness into the software development life cycle is at least as important as designing software with security countermeasures. What are some examples of software development processes that can lead to more-secure systems? |
| **（2）翻译** | 有人说，将安全意识纳入软件开发生命周期的重要性不亚于将安全对策纳入软件设计中。请问有哪些软件开发过程可以带来更安全的系统？ |
| **（3）解答** | Software development processes with security is a software design process that targets secure software products, including secure schematic design or solution design, security coding, security testing, and security deployment.  Take Janusic SDL SaaS as an example. The simplest security development process includes three phases (development test phase, product release phase, operation and maintenance phase). The standard application development process includes project planning, requirements analysis, summary design, and development. Testing, product release, acceptance, operation and maintenance, etc.  Each stage has a corresponding quality control task, especially a safety task.  **Project planning stage, Demand analysis phase, Summary design phase, Development test phase, Product release phase, Acceptance phase, Operation and maintenance stage.** |

|  |  |
| --- | --- |
| **（1）原文** | Security and usability are often seen to be at odds with each other. Security often imposes procedures and processes that seem like needless overhead to the casual user. But some say that security and usability go (or should go) hand in hand and argue that making the system easy to use securely is the best way to promote security to the user. Discuss. |
| **（2）翻译** | 人们通常认为安全性和可用性是相互矛盾的。安全性通常会给临时用户增加一些程序或步骤，而这些程序或步骤往往被被看作不必要的开销。但是另有一些人说安全性和可用性是（或应该是）相互配合的，他们还认为让系统的安全使用轻松起来是促进用户安全的最佳方式。请就以上问题进行讨论。 |
| **（3）解答** | Software usability refers to the ability of software to perform specified functions within specified conditions and within specified time. Software usability, one of the inherent features of software products, demonstrates the ability of the system to perform its functions in accordance with user requirements and design goals. Among them, the specified conditions include the software operating environment and software operation profile. The operating environment refers to the hard and software environment in which the software runs, including the operating system, application, compilation system, database system, etc. of the software running; the hardware environment includes the CPU memory of the computer, input/output, and so on.  The main factors affecting software usability: (1) software scale (2) software internal structure (3) software development method (4) software development tools (5) software testing and investment  There are many expressions of understanding and definition of safety. For example, in ISO8402, the term Safety is defined as: limiting the risk of injury (to humans) or damage to an acceptable state. Ways to improve the security of the software are: (1) security management (2) software risk analysis (3) software requirements description and design verification (4) using fault-tolerant technology:  The relationship between usability and security is very tight, and the parameters that affect the relationship between them are mainly to change the deviation between the program state environment and the state. To this end, we can use the change of mutual relationship to further study the transition law between them, that is to say: use the changes of the environment and the deviation loss parameters to convert some usability problems into security problems, using security related Technology to deal with usability issues; it is also possible to translate some security issues into usability issues and deal with security issues with usability -related technologies. |

|  |  |
| --- | --- |
| **（1）原文** | List some examples of critical resources for security that might become exhausted. |
| **（2）翻译** | 有一些关键的安全资源可能被耗尽，请举出相关例子。 |
| **（3）解答** | Server resources are exhausted. There are generally several situations:  1） Suffering from CC attacks, people are maliciously using the "zombie machine" to access a large amount of instant, occupying your server resources, causing the CPU and memory to run out, causing the server to crash.  2）suffered from DDoS and other traffic attacks, resulting in exhausted server bandwidth resources, resulting in the website, applications can not open.  3）the server has some kind of self-replicating virus or Trojan, causing the hard disk resources and memory resources to be exhausted and causing the server to crash. |

|  |  |
| --- | --- |
| **（1）原文** | List an example of a mapping of architectural elements that has strong security implications. Hint: think of where data is stored. |
| **（2）翻译** | 绘制一个具有重大安全隐患的架构元素的制图  提示：考虑数据的存储位置。 |
| **（3）解答** | 数据库系统为后台服务提供数据的存储、查询功能。如果数据库系统产生安全问题，例如其访问端口暴露，并被人攻入，则后台服务的安全性也会遭受影响。反过来，如果后台服务出现安全问题，也会导致数据库系统的安全问题，例如通过SQL注入漏洞，可以获取或篡改数据库中的数据。 |

|  |  |
| --- | --- |
| **（1）原文** | Which of the tactics in our list will protect against an insider threat? Can you think of any that should be added? |
| **（2）翻译** | 我们列表中有哪些策略可以防范内部威胁？你有补充的策略吗？ |
| **（3）解答** | 书中，给行为人授权和加密数据可以用于降低内部威胁。另外，监控内部用户行为也是处理内部威胁的手段之一。 |

|  |  |
| --- | --- |
| **（1）原文** | In the United States, Facebook can account for more than 5 percent of all Internet traffic in a given week. How would you recognize a denial-of-service attack on Facebook.com? |
| **（2）翻译** | 在美国，脸书能占有一周英特网流量的5%以上。你如何识别一次对于脸书的拒绝服务攻击？ |
| **（3）解答** | 由于脸书的访问量巨大，其自身防火墙容量也维持在较高的水平，小型DoS攻击并不能造成多大影响。若脸书受到一次大规模DoS攻击，此时正常用户访问脸书应当会遇到可用性大幅度降低的问题，即频繁出错等。若自己或他人常常无法访问或访问出错，则可以考虑其正在受到攻击。 |

|  |  |
| --- | --- |
| **（1）原文** | The public disclosure of vulnerabilities in production system is a matter of controversy. Discuss why this is so and the pros and cons of public disclosure of vulnerabilities. |
| **（2）翻译** | 在生产环境下公开披露系统漏洞是有争议的行为。请讨论为什么这样是有争议的，并且分析一下这样做的利弊。 |
| **（3）解答** | 弊端：根据安全设计策略，一个安全的系统应该通过限制操作接口来减少可能存在的能被攻击者利用进行访问、修改数据等危险行为的系统漏洞。而在生产环境下公开暴露漏洞恰好是违背这个策略的，是很容易被攻击者利用的，是很危险的行为。  好处：让用户知晓这些漏洞可以避免用户因为这些漏洞而蒙受损失，减少因为漏洞而产生的错误。  因为这种行为既有好处又有弊端，因此这种行为广受争议。 |

## 第十章

|  |  |
| --- | --- |
| **（1）原文** | A testable system is one that gives up its faults easily. That is, if a system contains a fault, then it doesn’t take long or much effort to make that fault show up. On the other hand, fault tolerance is all about designing systems that jealously hide their faults; there, the whole idea is to make it very difficult for a system to reveal its faults. Is it possible to design a system that is both highly testable and highly fault tolerant, or are these two design goals inherently incompatible? Discuss. |
| **（2）翻译** | 可测试性系统可以很容易的在问题出现时中断，也就是说，当系统存在一个故障时，则不会花费很长时间或很多努力来显示故障。在另一方面，故障容错就是设计那些小心翼翼地隐藏故障的系统，从而使得整个的目标就是使系统很难记录自己的故障。那么设计一个有很高测试性又有很高容错性的系统可能吗？还是这两个设计目标固有就是相斥的？请讨论。 |
| **（3）解答** | 可测试性和容错性是可以兼容的。系统之所以会容错，是因为那些故障能被系统自行解决，或者这些无视这些故障，也不会很大程度上影响系统的运行。而可测试性的目的是寻找那些会使系统崩溃或者错误输出的故障，从而进行认为修改和完善。两者所针对的故障类型不同，为了达到的目的是相同的，反正应该能相得益彰。 |

|  |  |
| --- | --- |
| **（1）原文** | “Once my system is in routine use by end users, it should not be highly testable, because if it still contains faults-and all systems probably do-then I don’t want them to be easily revealed.” Discuss. |
| **（2）翻译** | “当我的系统在被终端用户日常使用时，系统不应该具有很高的可测试性，因为就像大多数系统一样，系统会存在故障，而我并不希望故障能被轻易的揭示。”请进行讨论。 |
| **（3）解答** | 系统在被终端用户使用时，如果具有很高的可测试性，则用户的反馈可以提高系统完善的速度，提高系统的质量；但与此同时，也给不法分子可乘之机，尤其是那些运用智商犯罪的人，尤为可怕，带来的后果也不堪设想。所以就需要设计商和用户两方都多留一个心眼，共同推动系统的进步。 |

|  |  |
| --- | --- |
| **（1）原文** | 3. Many of the tactics for testability are also useful for achieving modifiability. Why do you think that is? |
| **（2）翻译** | 3. 许多对于可测试性的策略同样适用于可修改性的实现。你认为为什么？ |
| **（3）解答** | 3、因为可测试性和可修改性在场景上有一定的共同之处，在source上，它们都可以由开发人员、用户发起，在environment上，都有编译时、设计时的情况，而且可修改性的场景往往会引出可测试性的场景。 |

|  |  |
| --- | --- |
| **（1）原文** | 4. Write some concrete testability scenarios for an automatic teller machine. How would you modify your design for the automatic teller machine to accommodate these scenarios? |
| **（2）翻译** | 4. 写一些具体的可测试性场景给一个自动柜员机。你将会怎样修改你的设计让自动柜员机可以适应这些场景？ |
| **（3）解答** | Testability scenario:   |  |  | | --- | --- | | Source | system testers | | stimulus | The complete implementation of the automatic teller machine | | environment | compile | | artifacts | The whole system | | response | Control and monitor the state of the system | | Response measures | effort to find a fault or class of faults,  probability of fault being revealed by the next test,  effort to detect faults,  reduction in risk exposure (size(loss)\*prob(loss)),  time to perform tests |   我会设计为主要检测修复故障，以及降低风险敞口，其次才是提高效率、注重执行时间。 |

|  |  |
| --- | --- |
| **（1）原文** | What other quality attributes do you think testability is most in conflict with? What other quality attributes do you think testability is most compatible with? |
| **（2）翻译** | 你认为有什么其他的质量属性是和测试属性最为冲突的吗？  你认为有什么其他的质量属性和测试属性是最为相容相似的吗？ |
| **（3）解答** | I think the most conflict attribute with testability is modifiability, because modifiability breaks the stabilization of the testability, and it means changing and extra work for testability and make it more difficult to realize, also, testability will require data and functions to be not “modifiable” because it will widen the range of the testability.  I think the most compatible attribute with testability is uasbility, because usability needs testability to provide and secure the usability for the user, also the testability aims at providing good usability for the users. |

|  |  |
| --- | --- |
| **（1）原文** | One of our tactics is to limit nondeterminism. One method is to use locking to enforce synchronization. What impact does the use of locks have on other quality attributes? |
| **（2）翻译** | 我们其中的一个策略就是限制不确定性。其中一个方法是使用锁来强制信息同步。使用锁会对其他质量属性有什么影响呢？ |
| **（3）解答** | I think locks will have impacts on these attributes: For Availability, it will reduce the errors due to the elimination of nondeterminism; For interoperability, locks will weaken it because of the forcing synchronization, all parts cannot work well together due to the forcing snchronization; for security, forcing synchronization eliminates the situation that changing after submission.  锁的使用会提高安全性。但是会降低性能，因为锁的使用会带来开销，降低性能。 |

|  |  |
| --- | --- |
| **（1）原文** | 7. Suppose you’re building the next great social networking system. You anticipate that within a month of your debt, you will have half a million users. You can’t pay half a million users to test your system, and yet it has to be robust and easy to use when all half a million are banging away at it. What should you do? What tactics will help you? What a testability scenario for the social networking system. |
| **（2）翻译** | 7. 假设你正在构建下一个伟大的社交网络系统。您预计在您的首次亮相的一个月内，您将拥有50万用户。你不可能支付50万用户来测试你的系统，但当所有的50万用户都在使用的，它必须是鲁棒的和易于使用的。你应该怎么做?什么战术对你有帮助?为社交网络系统编写一个可测试场景。 |
| **（3）解答** | 采用控制和观察系统状态(Control and Observe System State)，我们可以控制系统同时使用的人数，观察系统和服务器的性能，推算50万人同时使用时系统的性能。测试方法采用负载测试，通过测试系统在模拟环境资源超负荷情况下的表现，确保系统在超出最大预期工作量时仍能正常运行，同时评估响应时间、事务处理速率等。   |  |  | | --- | --- | | Portion of Scenario Possible Values |  | | Source System testers  Stimulus The complete implementation of the whole system  Environment Compile time |  | | Artifacts The whole system  Response Control and monitor the state of the system  Response Measure Effort to achieve 99% state space coverage |  | |

|  |  |
| --- | --- |
| **（1）原文** | Suppose you use executable assertions to improve testability. Make a case for, and then a case against, allowing the assertions to run in the production system as opposed to removing them after testing. |
| **（2）翻译** | 假定你使用了一个可执行的断言来提高可测试性。 请分别提出一个充分的理由来支持和反对，“允许断言运行在产品系统中，而不是在测试之后删除它们”。 |
| **（3）解答** | 断言不应该运行在一个发布的商业产品中，因为断言是处理来自内部的错误数据的，是处理程序员的错误的，且频繁的调用会极大的影响程序的性能，增加额外的开销而一个优秀的商业产品是不应该出现这样的问题的。再好的产品也是会有bug的，程序员总是会犯错的，让断言运行在部署代码中，并且在字段中打开以利于调试。 |

## 第十一章

|  |  |
| --- | --- |
| **（1）原文** | 1.Write a concrete usability scenario for your automobile that specifies how long it takes you to set your favorite radio stations? Now consider another part of the driver experience and create scenarios that test other aspects of the response measures from the general scenario table.  2. Write a concrete usability scenario for an automatic teller machine. How would you design be modified to satisfy these scenarios? |
| **（2）翻译** | 1.为汽车写一个具体的可用性场景并指出用户设置最喜欢的电台的耗时，现在考虑其他方面的驾驶体验，并创建一个场景来从一般方案表中响应度量的其他方面。  2.为自动售卖机写一个具体的可用性场景，你会怎样修改设计来满足这些情况。 |
| **（3）解答** |  |

|  |  |
| --- | --- |
| **（1）原文** | How might usability trade off against security? How might it trade off against performance? |
| **（2）翻译** | 实用性和安全性之间应该如何权衡？同样的，在实用性与使用性能之间又该如何权衡呢？ |
| **（3）解答** | 可用性是指系统能够提供给用户便捷的使用方法。安全性是指系统能够给系统数据和信息提供安全保护。可用性大多时候都与可变更性有着联系，一个用着高可用性的系统往往也用着高的可变更性以此来适应用户的需求。但可变更性对系统的安全性有着一定的影响。我觉得如何在可用性和安全性之间进行权衡时，应该考虑该系统的需求倾向，如果该系统是注重用户体验的可以保证基本的系统安全上更加注重可用性，如果该系统是涉及资金，隐私等重要信息的时候则应该更加着重安全性。在可用性和性能之间考虑的时候，也是根据系统的需求来进行权衡，如果系统是面对一些专业领域被要求有着齐全和强大的功能时，则可以牺牲一定的可用性来保证功能。 |

|  |  |
| --- | --- |
| **（1）原文** | Pick a few of your favorite web sites that do similar things,such as social networking or online shopping.Now pick one or two appropriate responses from the usability general scenario (such as “achieve the task at hand”) and a correspondingly appropriate response measure. Using the response and response measure you choose, compare the web sites’ usability |
| **（2）翻译** | 选择一些你最喜欢做类似事情的网站，比如社交网络或网上购物。现在从可用性一般情景中选择一两个适当的响应（例如“实现手头的任务”）和相对应的响应措施。 使用您选择的响应和响应措施，比较网站的可用性。 |
| **（3）解答** | 两个网站为：百度在线翻译和谷歌在线翻译。  可用性一般场景为：  源：用户  刺激因素：将一篇英文文章翻译成中文  环境：运行速度  制品：翻译窗口  响应：返回一篇中文文章  响应度量：完成翻译的时间和翻译错误的数量  通过比较，百度翻译完成需要2.6秒，而谷歌翻译需要1.2秒。  百度翻译的错误有2个，谷歌翻译的错误有2个。  所以从上面这个场景来看，谷歌在线翻译的可用性更高一些。 |

|  |  |
| --- | --- |
| **（1）原文** | Specify the data model for a four-function calculator that allows undo. |
| **（2）翻译** | 为四则运算计算器提出允许撤销操作的数据模型。 |
| **（3）解答** | 为该计算器设计一个栈用来存放用户的历史操作，用单链表来实现它。将用户的每一次运算都压入栈，当用户请求撤销时对最后一次运算进行出栈操作。计算器显示的是当前栈的内容。如下图栈的结构。   |  | | --- | |  | |  | | 7-8 | | 5/6 | | 3\*4 | | 1+2 |   可用单链表来实现，定义删除，增添操作。 |

|  |  |
| --- | --- |
| **（1）原文** | Why it is that in so many systems, the cancel button in a dialog box appears to be unresponsive? What architectural principles do you think were ignored in these systems? |
| **（2）翻译** | 为什么在如此多的系统中，对话框中的取消按钮似乎并不能很快地响应?你认为这些系统中哪些架构原则被忽略了？ |
| **（3）解答** | 当用户发出取消命令时，系统必须终止被取消的命令，释放已取消命令使用的任何资源，这一过程消耗较多时间，故很多系统在该过程结束时不向用户响应。这些系统应该对用户的请求给予响应，使用户更高效地使用系统，提高用户对系统的主动性和满意度。这些系统忽略了这些架构原则。 |

|  |  |
| --- | --- |
| **（1）原文** | Why do you think that progress bars frequently behave erratically, moving from 10 to 90 percent in one step and then getting stuck on 90 percent? |
| **（2）翻译** | 你认为为什么进度条经常表现得无规律性，比如进度一步就从10%到了90%，然后卡在了90%？ |
| **（3）解答** | 因为在大多数情况下，进度条并不能反映真实的任务处理进度，只是一个没有实际功能，负责提供心理安慰的“安慰剂按钮”而已。  假设有两个进度条，A和B，它们的完整时长都是100秒,进度条A先快后慢，只用了10秒就加载完前面的90%，之后的90秒都卡在90%；而进度条B接近匀速，10秒才加载了10%，20秒加载20%，直到90秒，才推进到90%。在这种情况下，相比10秒就跑到90%的A，进度条B更可能被关闭，而且更容易产生A程序加载速度比B要快的错觉。  而且从其他方面来看，能反映真实的任务处理进度的进度条也同样会卡在90%。这是因为这些进度条反映的是任务量，而非耗时量。所以如果计算机把耗时最长，的任务留到最后，这样子同样会让进度条卡在90%。  此外，多线程下载也是产生90%进度条的原因之一，这种情况进度条反应的是线程的完成数量，例如有90%的线程完成了工作，而剩下的10%线程未完成工作，但是这10%的线程处理的是最复杂的工作，在部分线程的耗时将会很大，这就使得进度条卡在了90%上。  所以，根据书中所写内容“Increasing confidence and satisfaction”可知，大多数进度条只是为了提高用户对系统的信心和满意度，故而它会在很短的时间内上升到90%并保持不动（但这并不表示了实际的耗时进度）。 |

|  |  |
| --- | --- |
| **（1）原文** | Research the crash of Air France Flight 296 into the forest at Habsheim, France, on June 26, 1988. The pilots said they were unable to read the digital display of the radio altimeter or hear its audible readout. If they could have, do you believe the crash would have been averted？In this context, discuss the relationship between usability and safety. |
| **（2）翻译** | 在对法国269航班坠毁在法国哈布斯海姆的森林的坠毁事故的调查中，飞行员表示他们无法读取无线电高度表的数字显示或听到其声音读数。 你认为如果他们可以（读取无线电高度表的数字显示或听到其声音读数）的话，坠毁会被避免吗? 在此背景下，讨论可用性与安全性之间的关系。 |
| **（3）解答** | 我认为坠毁会被避免；安全性是事物或者更准确的说是系统固有的一个属性，即是系统本应该有的一种属性，不安全的系统不应该被采用；而可用性是用户对系统的一种要求，系统为了满足用户的需求或者便于用户使用而具有的属性。  在某种时候，为了满足可用性会牺牲掉一定的安全性，例如系统满足用户多变需求的能力和系统架构的灵活性密切相关，架构越灵活，系统可用性越高，但是随着系统架构越来越复杂，那么系统的安全性将不可避免的受到影响。  但是在很多的情况下较好的可用性可以提高系统的安全性。如在上文的背景故事中，航班无线电高度表的数字显示或其声音读数若是可用，那么航班的安全性便得到了保证，即避免了坠毁。而安全性对于可用性而言有这样一个关系：假设一个系统的安全性为百分之百，这种时候，系统的可用性将得到提高；因为这样的一个系统，发生系统故障时对用户产生的影响很小，而系统采取的系统故障恢复措施也很容易执行。 |

## 第十二章

|  |  |
| --- | --- |
| **（1）原文** | 10. Relate monitorability to observability and controllability, the two parts of testability. Are they the same? If you want to make your system more of one, can you just optimize for the other? |
| **（2）翻译** | 10. 对比可监测性和可测试性的两个部分，可观察性和可控性。它们是一样的吗？如果你希望你的程序同时有这两个质量属性，你能否优化另一个质量属性？ |
| **（3）解答** | 10. 不相同，可检测性应用于程序执行的期间，以便让服务器和程序的运维人员能够通过程序的执行的一些反馈信息采取相应的措施。而可观察性和可控制性则在程序的测试期间，测试人员需要程序测试期间的大量信息，以便减少程序的错误，让程序能够发布使用。可以在软件测试期间同时设计一个监测应用程序，并在应用程序代码中加入反馈数据给服务器的代码，以便监测应用程序使用。  2. 可监测性。运维人员通过应用程序实时运行期间产生的错误进行收集，使开发组能开发更为优良的版本。应用程序收集用户数据，反馈给服务器，实现诸如个性化和推荐等功能。运维人员根据程序反馈，做出例如手动修改数据库，修改应用程序数据等操作。通用场景：刺激源：运维人员。刺激：应用程序返回数据。人工制品：返回的数据。环境：程序运行期间。反馈：程序行为可以被监测。应对措施：收集bug，开发新版本，能够即时处理的运维人员即时处理，个性化和推荐。 |

|  |  |
| --- | --- |
| **（1）原文** | 1. The kingdom of Bhutan measures the happiness of its population, and government policy is formulated to increase Bhutan’s GNH (gross national happiness). Go read about how the GNH is measured (try [www.grossnationalhappiness.com](http://www.grossnationalhappiness.com)) and then sketch a general scenario for the quality attribute of *happiness* that will let you express concrete happiness requirements for a software system. 2. Choose a quality attribute not described in Chapters 5–11. For that quality attribute, assemble a set of specific scenarios that describe what you mean by it. Use that set of scenarios to construct a general scenario for it.  3. For the QA you chose for discussion question 2, assemble a set of design approaches (patterns and tactics) that help you achieve it. |
| **（2）翻译** | 1. 不丹王国为衡量其人口的幸福，政府制定政策以提高不丹国民幸福总值。去了解GNH是如何测量的（请访问www.grossnational.com），然后勾画一个关于幸福质量的一般场景，能让你表达一个软件系统的具体幸福需求。  2. 选择第5章至第11章中未描述的质量属性。对于那个质量属性，组装一组特定的场景来描述你的意思。使用这组场景来构建它的一般场景。  3. 对于您为讨论问题2选择的QA，组装一组帮助您实现它的设计方法（模式和策略）。 |
| **（3）解答** | 1.  GNH（国民幸福指数）是指反映国民生活质量和幸福程度的指标。不丹的GNH（[国民幸福指数](https://www.baidu.com/s?wd=%E5%9B%BD%E6%B0%91%E5%B9%B8%E7%A6%8F%E6%8C%87%E6%95%B0&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)）测量，主要考察四个方面：可持续发展、环境保护、文化保护以及政府的有效管理。  幸福质量的一般场景：  场景的部分： 可能的值：  刺激源： 政府  刺激： 制定政策  环境： 不丹国家  制品: [国民幸福指数](https://www.baidu.com/s?wd=%E5%9B%BD%E6%B0%91%E5%B9%B8%E7%A6%8F%E6%8C%87%E6%95%B0&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)  响应： 政府指定政策并实施来提高国民幸福指数  响应度量： 政策执行后国民生产总值指数、社会健康指数、社会福利指数、社会文明指数和生态环境指数  2.  Scalability（可扩展性）  可扩展性可分为水平可扩展性和垂直可扩展性。  特定场景：可以是向服务器集群添加另一台服务器，或者向单个计算机添加更多的内存。这将出现如何有效地利用额外资源的问题。  一般场景：  场景的部分： 可能的值：  刺激源： 最终用户、开发人员  刺激： 添加或删除资源  环境： 运行时、构建时、设计时  制品: 系统用户平台、界面和系统服务器  响应： 处理添加或删除资源的影响  响应度量： 分配给现有和新资源的负载  3.  可扩展性是一种对软件系统计算处理能力的设计指标。可用Shared Space（共享空间）的设计模式，所有工作者线程通过共享空间共享数据，来协调各个线程的并发和数据分隔任务分隔问题。 |

|  |  |  |
| --- | --- | --- |
| **（1）原文** | For the QA you chose for discussion question 2, develop a design checklist for that quality attribute using the seven categories of guiding quality design decisions outlined in Chapter 4. | |
| **（2）翻译** | 对于你在讨论问题2中选择的QA，使用在第4章中概述的指导质量设计决策的七个类别，为该质量属性开发一个设计检查表。 | |
| **（3）解答** | Category | Checklist | |
|  | Allocation of responsibilities（职责分配） | 通过考虑技术、法律、社会用途和客户的变化，确定哪些扩展是可能发生的：  -确定扩展的纵向或者横向的职责  -确定扩展后的影响的职责  -确定模块的职责分配 | |
|  | Coordination model（协调模型） | 确定哪些功能或质量属性可以在运行时更改，以及如何影响协调，例如：在服务器集群中添加服务器会有什么变化，扩展要尽量影响少的模块  确定协调的设备、协议可能会扩展。 | |
|  | Data model（数据模型） | 确定对数据资源抽象、其操作或其属性可能发生的更改，还要确定对这些数据抽象的扩展或扩展类别将涉及到创建、初始化。  对每次的扩展，确定扩展是由最终用户、系统管理愿或开发人员进行。  对于每个潜在的变化：  -确定需要修改的数据抽象  -确定对哪些数据抽象的创建、初始化、持久性、操作、转换或销毁是否有更改  -确保数据抽象的分配  设计数据模型，以便分配给数据模型的每个元素的项可能一起更改 | |
|  | Management of resources（资源管理） | 确定职责或质量属性的修改将如何影响资源的使用  确定扩展对资源的添加或删除的影响，以及确保修改后的资源足以满足系统需求  封装所有资源管理器 | |
|  | Mapping among architectural elements（在架构元素映射） | 确定对函数或质量属性扩展的范围，如：  -执行依赖关系  -处理添加或删除资源的影响  确保这些扩展是使用机制执行的 | |
|  | Binding time decisions（绑定时间决定） | 决定最迟何时需要进行扩展  选择一种延迟绑定机制，在选择的时间交付适当的功能  确定这些扩展不会对系统造成不可接受的所害 | |
|  | Choice of technology（技术选择） | 确定技术选择使哪种扩展更容易或更困难  -技术选择是否有助于进行、测试和部署修改  -修改技术的选择的容易程度  选择支持最可能修改的技术，如Shared Space | |

|  |  |
| --- | --- |
| **（1）原文** | 5. What might cause you to add a tactic or pattern to the sets of quality attributes already described in Chapters 5-11 (or any other quality attribute, for that matter)? |
| **（2）翻译** | 什么可能会导致你将战术或模式添加到已经在第5-11章中描述过的质量属性集(或任何其他质量属性)? |
| **（3）解答** | 该质量属性在新的场景中不适用，没有紧凑的知识体系；质量属性运用到新的模型中；实现特定质量属性时。 |

|  |  |
| --- | --- |
| **（1）原文** | 6.According to slate.com and other sources, a teenage girl in Germany “went into hiding after she forgot to set her Facebook birthday invitation to private and accidentally invited the entire Internet. After 15,000 people confirmed they were coming, the girl’s parents canceled the party, notified police, and hired private security to guard their home.” Fifteen hundred people showed up anyway; several minor injuries ensued. Is Facebook “unsafe”? Discuss. |
| **（2）翻译** | 据slate.com和其他消息来源称，德国一名十几岁的女孩“忘记把自己的Facebook生日邀请设为私人活动，不小心邀请了整个互联网，然后躲了起来”。在15000人确认他们会来之后，女孩的父母取消了聚会，通知警察，并雇佣私人保安来保护他们的家。反正还是有一千五百人来了;随后发生了几起轻伤事件。Facebook“不安全”吗? 请讨论。 |
| **（3）解答** | 我们觉得Facebook就这个事件而言还是安全的，首先Facebook并无暴露用户信息，单纯是女孩错误操作而造成了一系列事故。就从发Facebook的角度来说，软件是无法准确识别操作者的主观意愿的，是否设置为私人并不在软件的安全范畴，纯属个人失误，不能怪这个软件不具备安全性。 |

|  |  |
| --- | --- |
| **（1）原文** | 7.Author James Gleick C"A Bug and a Crash," www.around.com/ariane. html) writes that "It took the European Space Agency 10 years and S7 billion to produce Ariane 5, a giant rocket capable of hurling a pair of three ton satellites into orbit with each launch. ... All it took to explode that rocket less than a minute into its maiden voyage ... was a small computer program trying to stuff a 64bit number into a 16-bit space. One bug. one crash. Of all the careless lines of code recorded in the annals of computer science,this one may stand as the most devastatingly efficient." Write a safety scenario that addresses the Ariane 5 disaster and discuss tactics that might have prevented it.   1. Discuss how you think development distributability tends to "trade of" against the quality attributes of performance, availability, modifability, and interoperability.   Extra Credit: Close your eyes and, without peeking, spell dstributability Bonus points for successfully saying development distributability" three times as fast as you can.  9. What is the relationship between mobility and security? |
| **（2）翻译** | 7.作者James Gleick 在《A Bug and a Crash》，www.around.com/ariane. html中写道：“欧洲航天局花了10年时间和7亿美元用于生产Ariane 5号，这是一种能够在每次发射时将一对3吨卫星投入轨道的巨型火箭......所有这一切都在火箭首航起飞不超过一分钟爆炸化为乌有......是一个小型计算机程序试图将64位数字填入16位空间。一个错误,引发了一个坠毁。在计算机科学史上记录的所有粗心的代码行中，这个可能会是最具破坏性的效率。“请写一个Ariane 5号灾难的关于安全的场景，并讨论可能阻止它的策略。  8.讨论您如何认为开发可分配性与质量属性性能，可用性，可修改性和互操作性的权衡。  9.移动性和安全性之间有什么关系？ |
| **（）解答** | 7、Ariane 5 的灾难是因为尝试用 16位的空间如存储64位的数据，因为类型或者说空间不匹配而导致了这次灾难。  Safety  **Scenario**: The Ariane 5 disaster  **Source**: developer  **Stimulus**: a data conversion from a64-bit floating-point number to a 16-bit signed integer value  Environment: after liftoff  **Artifacts**: system  **Response**: avoid the conversion  **Response measure**: avoid the bug and make the rocket work as we expected.  Tactics:  8、例如改变加密级别可能会对安全性和性能产生非常重要的影响。提高加密级别可以提高安全性，但可能要耗费更多的处理时间，影响系统性能。分布式软件开发需要分析多个质量属性间的关系，属性间可能存在冲突，需要权衡取舍。按照质量属性需求，评价体系结构设计。  9、安全性是使软件在受到恶意攻击的情形下依然能够继续正确运行及确保软件被在授权范围内合法使用。移动性处理平台的移动和可供性问题。  移动性和安全性之间相互制约。增强移动性会使软件环境更多样化增添软件接口，无形中增加软件漏洞，降低软件安全性。增强软件安全性会要求软件环境尽量减少，要求软件复杂度较低，使软件移动性不能太高。 |