**构架复习内容：**

**软件体系结构概念及ABC**

**SA的定义:**

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.

一个程序的软件体系结构或者一个计算机系统是一个构架或者是一个系统的构架。它包括软件元素、这些元素的外观，以及它们之间的关系。

体系结构 = 组件 + 连接件 + 约束

Architecture = Components + Connectors + Constrains

**SA的重要性：**

1. 构架是涉众之间进行交流的手段
2. 构架是早期设计决策的体现
3. 构架是可传递、可重用的系统抽象

**ABC:**

**The Architecture Business Cycle (ABC)**

ABC is cycle of influences, from the environment to the architecture and back to the environment.

ABC是一个来自环境对构架以及反馈回环境的影响的循环。

The business/organizational environment necessarily affects architectural design decisions.

事物/组织环境必定会影响构架设计决定。

The software architecture in turn changes the business environment.

Architectural design is part of every step of the development process.

These influences strongly affect the organization’s bottom line.

软件构架反过来改变事物环境。构架设计是开发过程的每一步的部分。这些会强烈影响组织的底线。

**什么是好的SA:**

**关于过程的建议**

1. 构架的设计应该由一位设计师或者在某设计师领导下的小组完成
2. 设计师应全面掌握系统的功能需求，并且应有一份所设计构架应满足的划分了优先级的质量属性列表。
3. 构架的文档应该完备。
4. 应该让各种涉众积极参与设计方案的评审。
5. 对构架人身进行分析，得出可应用的量化度量指标。
6. 构架的设计应有助于增量式实现。
7. 允许构架带来一定的资源争用，但应清楚地给出这些资源争用的解决方案。

**关于结构的建议：**

1. 构架应采用定义良好的模块。
2. 应该使用特定于每个属性的众所周知的构架战术来实现质量属性。
3. 构架不可以依赖于某个版本的商业产品或者工具。
4. 产生数据的模块与使用数据的模块分开。
5. 对于并行处理系统，应采取订立良好的进程或任务。
6. 每个任务或进程的编写都要考虑与特定处理器的关系。
7. 采用少量的、简单的交互模式。

**构架的重要性：**

1. 构架影响着开发组织的结构
2. 构架会影响开发组织的目标。
3. 构架可能会影响客户对下一个系统的要求。
4. 构建系统的过程丰富了整个开发团队的经验。
5. 一些典型的系统会影响并实际改变软件工程的发展。

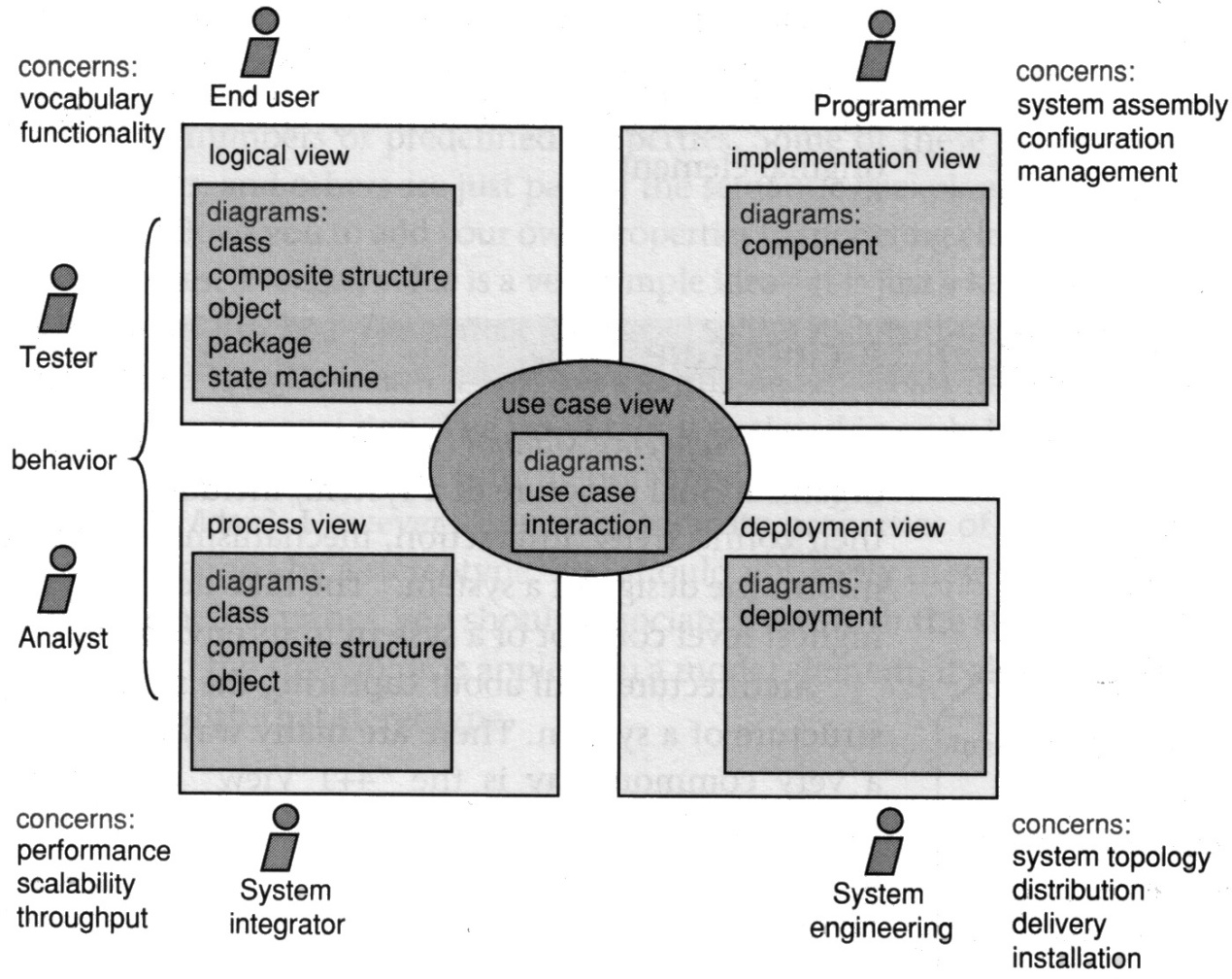
**构架的分类：**模块结构组件-连接器结构分配结构

常见软件体系结构风格及其特点－数据流、调用/返回、层次、仓库/黑板

SA风格的概念，常用SA风格特点，比较

**软件体系结构的视图和描述**

4+1视图理论，Architectural Structures（教材作者的多重结构理论）



**The design or logical view:** the classes, interfaces, and collaborations forming the vocabulary of the problem and its solution.

**The interaction or process view**: how control flows across the static classes at run time.

**The implementation view:** the organization of the software modules in the development environment.

**The deployment view:** A mapping of elements onto processing and communication nodes.

Uses cases cut across the views. They capture the system’s functional requirements for the system and help find conflicts.

**软件体系结构的质量属性及其战术（重点！）**

**常见质量属性:**

**质量属性场景**

刺激源、刺激、环境、制品、响应、响应度量

**保障质量属性的常用战术:**

**1.可用性战术**

错误检测

命令/响应、心跳、异常

错误恢复：

表决、主动冗余（仅用第一个反应）、被动冗余（主响应，其他更新）、备件、Shadow操作、状态再同步、检查点/回滚

错误预防：

从服务中删除、事务、进程监视器

**2.可修复性战术**

局部化变更

语义一致性、预期期望的变更、泛化模型、限制可能的选择

防止连锁反应

隐藏信息、维持现有的接口、限制通信路径、使用仲裁者

推迟绑定时间

运行时注册、配置文件、多态、组件更换、遵守已定义的协议

**3.性能战术**

资源需求

提高计算效率、减少计算开销、管理事件率、控制采样频率、限制执行时间、限制队列大小

资源管理

引入并发、维持多个副本、增加可用资源

资源仲裁

调度策略

**4.安全性战术**

抵抗攻击

对用户进行身份验证、对用户进行授权、维护数据的机密性、维护完整性、限制暴露的信息、限制访问

检测攻击

入侵检测

从攻击中恢复

恢复——查看可用性识别——审计追踪

**5.可测试性战术**

管理输入输出

记录回放 将接口与实现分离 特化访问路线接口

内部监视

内置监视器

**6.易用性战术**

分离用户接口

支持用户主动

取消

撤销

聚合

用户模型

用户模型

系统模型

任务模型

**软件体系结构设计的ADD方法**

**构架设计的重要性和时机：**

把构架作为软件开发过程基础的人和组织都需要理解构架在其生命中的位置。

何时开始设计：在生命期模型中，构架设计就是从初步的需求分析开始逐步进行迭代。

**ADD方法的步骤：**

1. 选择要分解的模块
2. 根据这些步骤对模块进行求精

2.1、从具体的质量场景和功能需求集合中选择构架驱动因素

2.2、选择满足构架驱动因素的构架模式。

2.3、实例化模块并根据用例分配功能。

2.4、定义子模块的接口。

2.5、验证用例和质量场景并对其求精。

3、对需求进一步分解的每一个模块重复上述步骤

**如何创建骨架系统：**

1. 首先实现处理构架组建的执行和交互的软件部分。这可能要求生成实时系统中的调度程序；实现规则引擎以控制在基于规则的系统中规则的激发；实现多进程系统中的进程同步机制或客户机。之后就拥有了一个运行系统。这为添加有用功能提供了基础。
2. 现在可以选择把提供功能的那些元素添加到该系统。这种选择可能是根据以下因素做出的：首先处理问题最多的部分来降低风险；或现在开发人员的类型水平；或尽可能快地将有用的产品推向市场。
3. 现在了提供下一个功能增量的元素后，就可以采用使用结构，以获知应该在系统中采用什么软件来支持该功能。
4. 随着过程的继续，系统的增量越来越大，直到软件开发完毕。

**软件体系结构评审及其常用方法（ATAM）**

**构架评审的重要性：**

All design involves tradeoff in system qualities（设计即折中）

System qualities are largely dependent on architectural decisions  
体系结构极大地影响系统质量

Promoting one quality often comes at the expense of another quality  
提高一个质量，经常会降低另一个质量

A software architecture is the earliest life-cycle artifact that embodies significant design decisions: choices and tradeoffs.  
“选择与折中”是设计中首要考虑的问题，软件体系结构是软件生命周期中最早一个遇到此问题的

Choices are easy to make, but hard to change once the system is implemented  
选择很容易做，但是一旦系统已经实现，就很难更改

**成本收益：**

Clarified quality attribute requirements

明确质量需求

Improved architecture documentation

提高体系结构文档质量

Documented basis for architectural decisions

文档化了的体系结构方案基础

Identified risks early in the life-cycle

及早发现风险

Increased communication among stakeholders

促进了角色之间的交流

The results are improved architectures.

结果是，体系结构得到改进

**时机：**

Academically, the time to use ATAM is right after the architecture has been specified when there is little or no code.

学术上说，应该在体系结构确定之后使用ATAM，此时没有或者仅有少量的代码

However, in practice, ATAM has been very effective in the following situations:

Evaluating alternative candidate architectures

评审候选的体系结构

Evaluating existing systems prior to committing to major upgrades

在升级之前，评审已有的系统

Deciding between upgrade or replace

无法决定是升级还是替换时

**前提条件：**

**Clients must have a Software Architecture**

**客户必须有一个软件构架**

Scope/scale must be manageable

其作用范围和程度必须可管理

ATAM will not work if the software architecture has not been created yet

如果体系结构还没有被建立，那么ATAM毫无用武之地

ATAM team members will review architectural artifacts, and may help refine documentation ATAM

组将评估体系结构，并帮助改善文档

Architect must prepare an architecture presentation

构架师必须准备一个体系结构讲解

**Clients must prepare a business/mission goals presentation**

**必须有一个商业/任务目标讲解**

**ATAM will review architecture artifacts, presentations, and read ahead material to become familiar with domain ATAM**

**要事先阅读一些材料来熟悉这个领域**

**人员组成：**

1. 评估小组
2. 项目决策者
3. 构架涉众

**过程（步骤）：**

1. 方法的表述
2. 商业动机的表述
3. 构架的表述
4. 对构架方法进行分类
5. 生成质量属性效用树
6. 分析构架方法
7. 集体讨论并确定场景的优先级
8. 分析构架方法
9. 结果的表述

**常用技术手段**

**评审结果：**

1. 一个简洁的构架表述。
2. 表述清楚的业务目标。
3. 用场景集合捕获的质量需求。
4. 所确定的敏感点和权衡点集合。
5. 有风险决策和无风险决策。
6. 风险主题集合。

**ATAM方法的这些方面知识**

考题类型

选择题（30%）

简答题（30%）

分析论述题（40%）

选择题

**All of the following roles are stakeholder except:**

客户、最终用户、开发人员、项目经理、维护人员、对系统进行市场营销活动的人

**All of the following are important system attributes except: attractivity**

Availability，Modifiability， Performance， Security， Testability， Usability

**Which of the following are elements of a SA: All**

特征损失模型（MODP）、回响模型（MODR）、噪音模型（MODN）、控制处理（CP）

**Which of the following is not a precondition for architectural review? finished design**

Review Preconditions

* Understand the context of the review 理解评审的背景
* Assemble the right people 集合正确的人
* Set organizational expectations and support 期望和支持
* Prepare for review 为评审做准备
* Obtain representation of the architecture 获得体系结构文档

**ATAM outputs include:**

* + - architectural approaches
    - utility tree
    - scenarios
    - risks and “non-risks”
    - sensitivity points and tradeoffs

简答题

**What is The Architecture Business Cycle (ABC)?**

ABC is cycle of influences, from the environment to the architecture and back to the environment

**List 5 architecture patterns/styles.**

数据流、调用/返回、层次、仓库/黑板，虚拟机

**How is an architectural pattern/style** **determined.**

a set of component types

a set of connector types/interaction mechanisms

a topological layout of these components

a set of constraints on topology and behavior

an informal description of the costs and benefits of the style

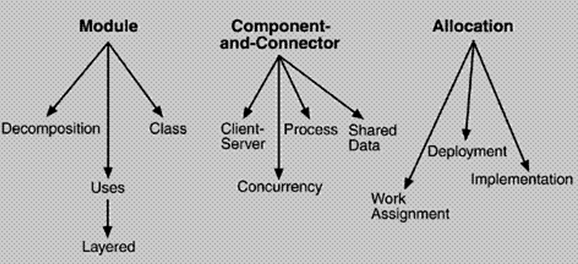
**Bass et al's classify all architecture structures into 3 main categories, what are them?**

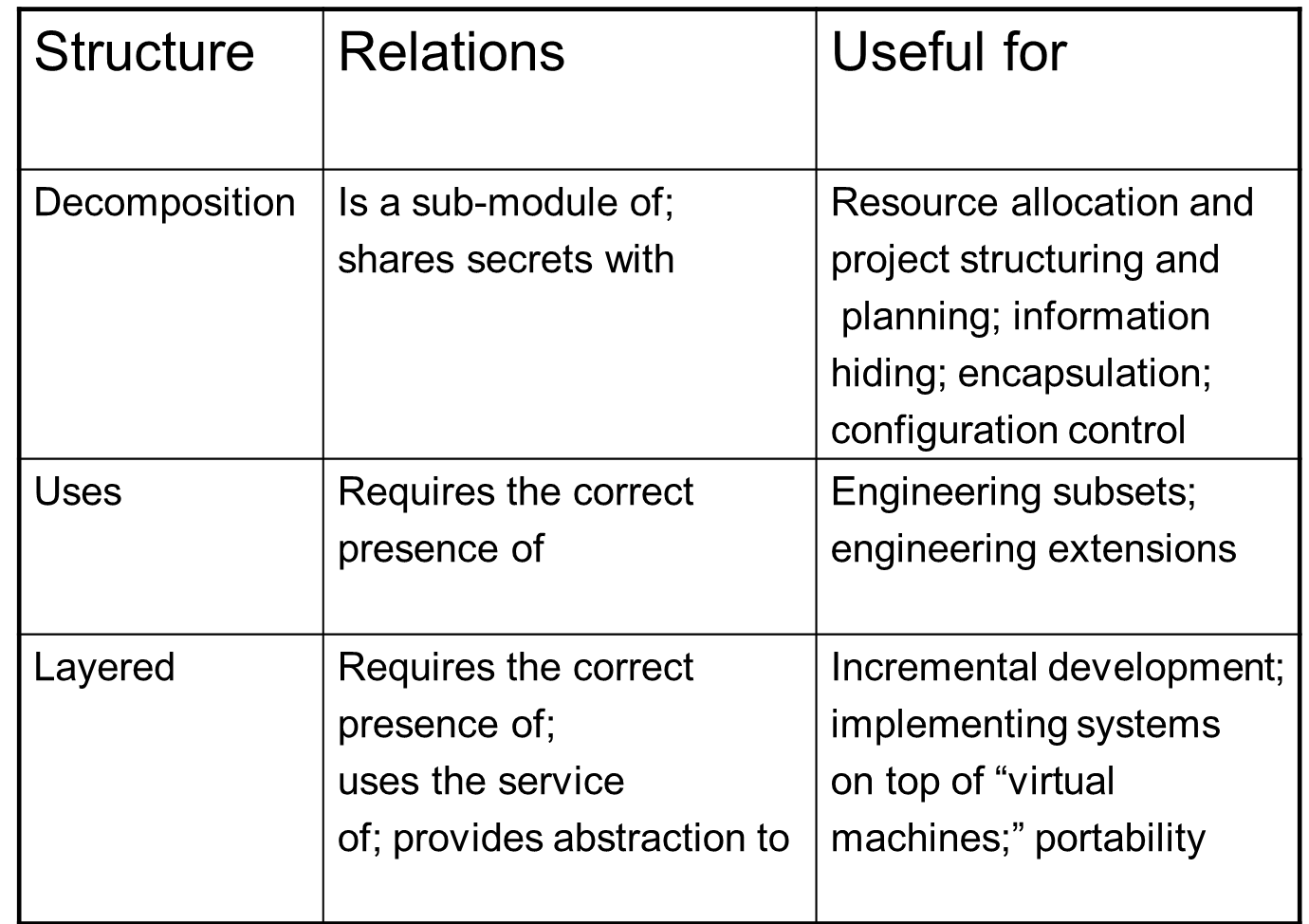
1.Module-based structures: the elements are software modules (the implementation units). Includes decomposition, uses, layered, class.

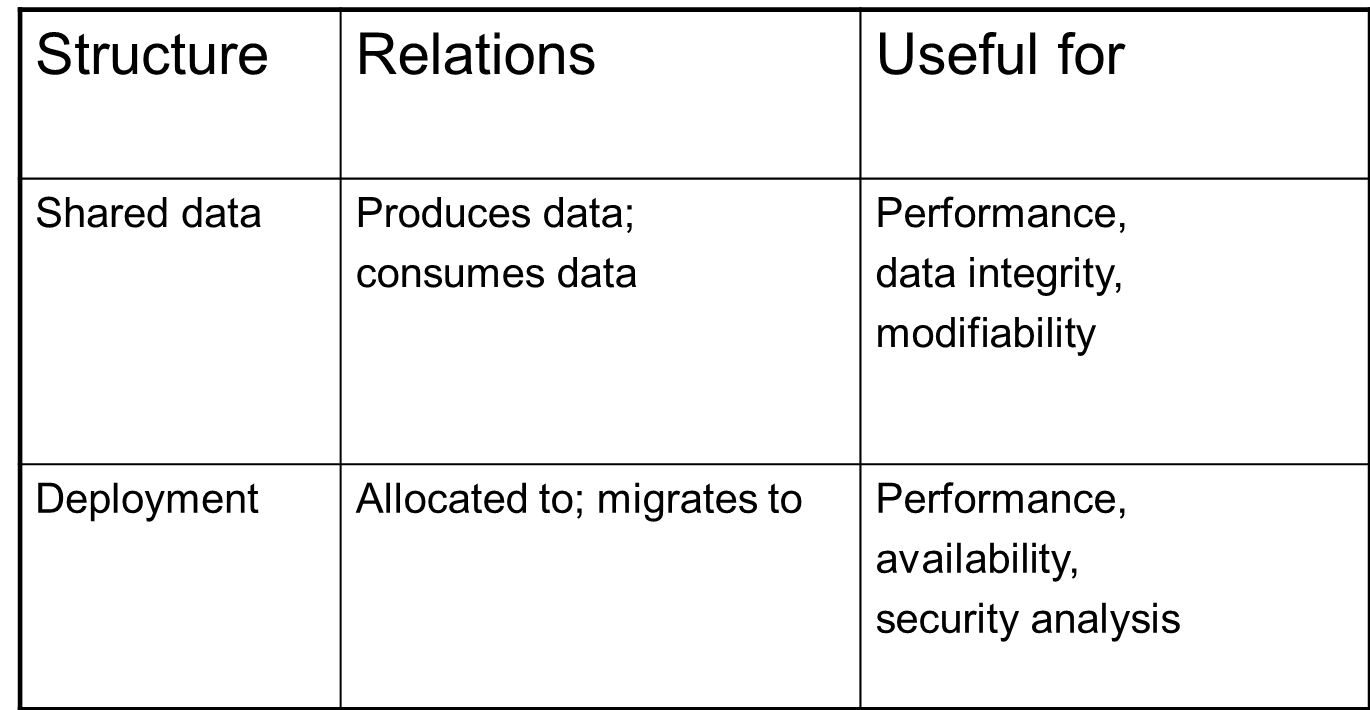
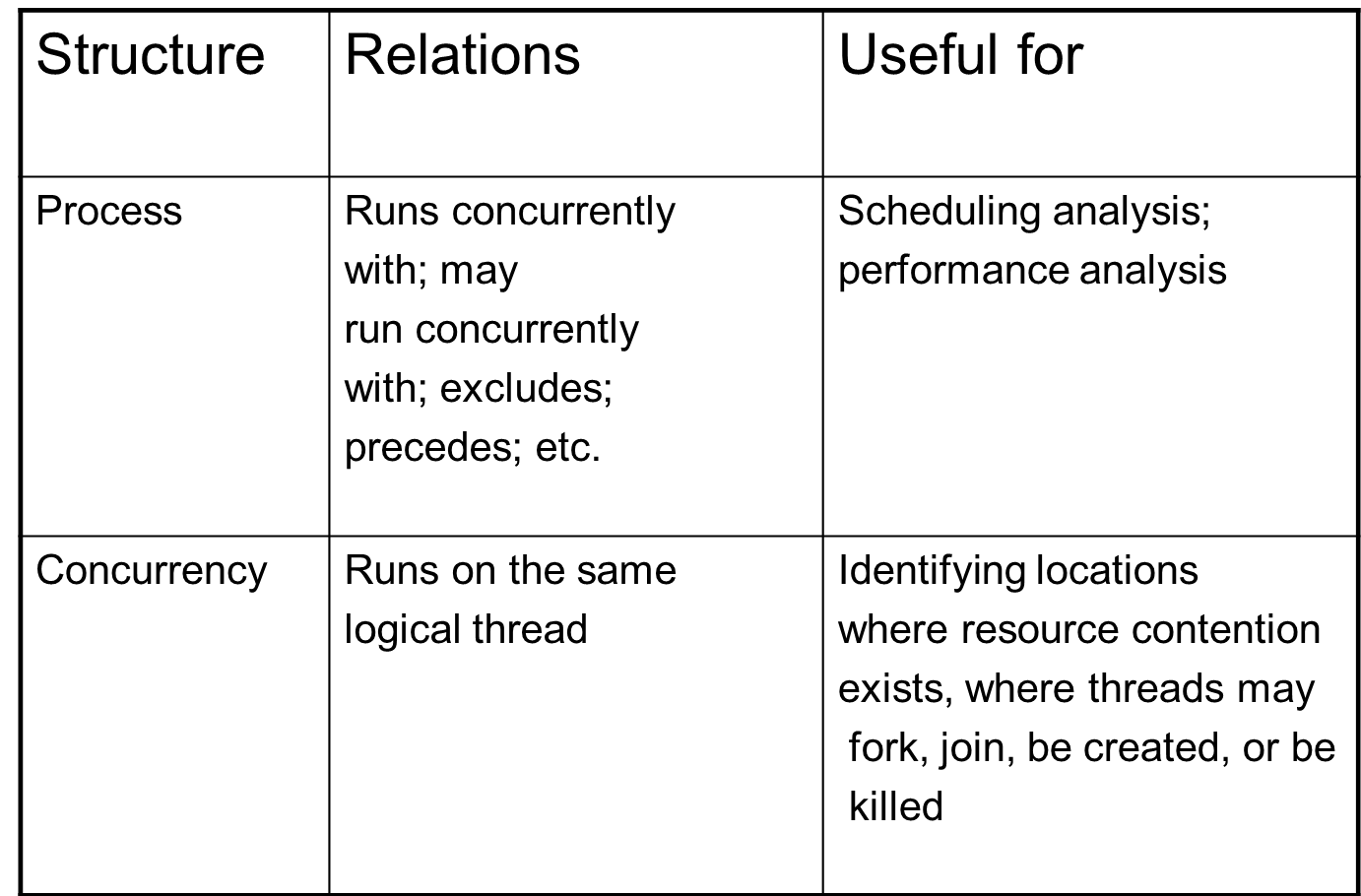
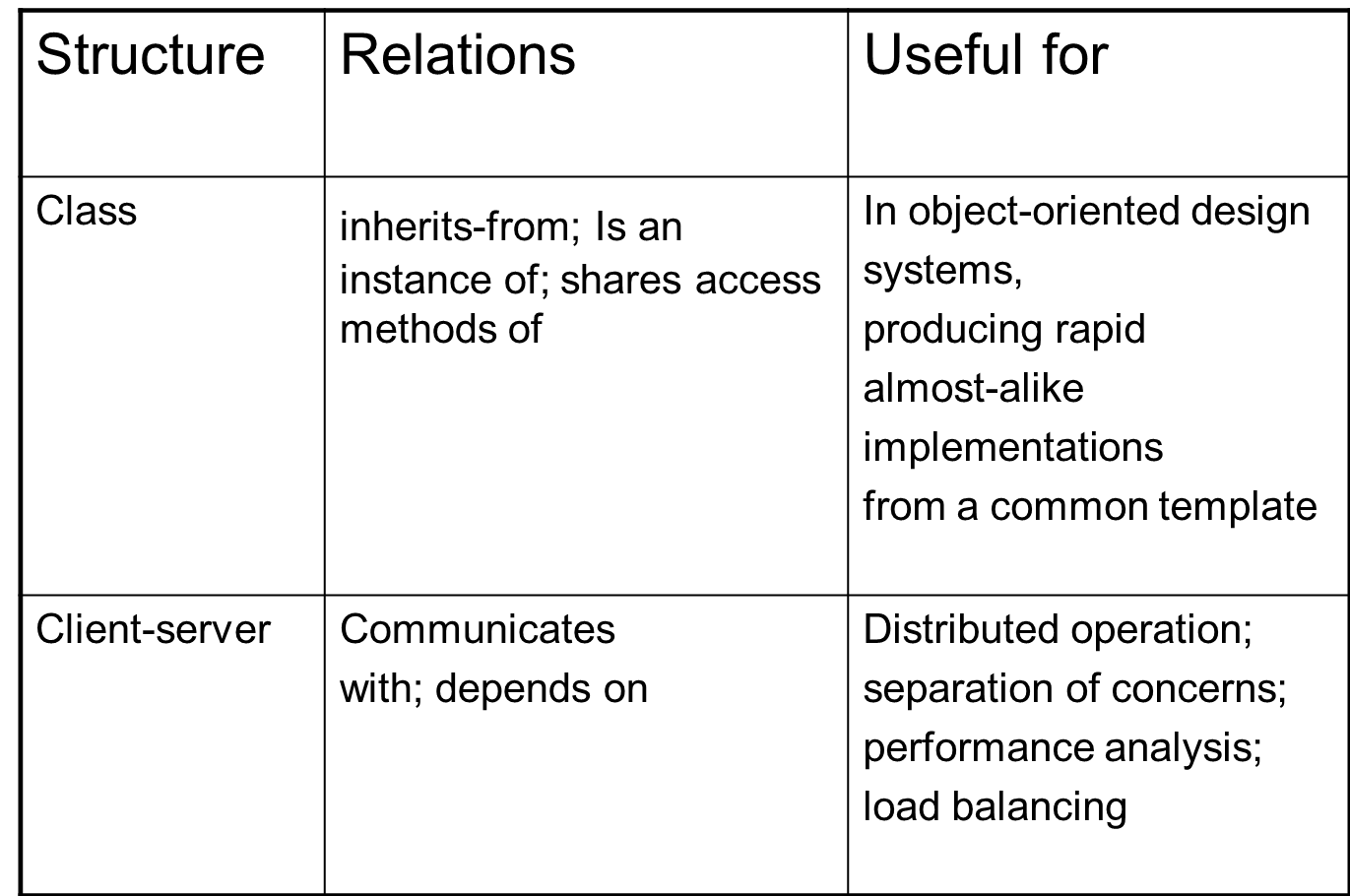
2.Component-and-connector structures: the elements are run-time components. Includes process communication, concurrency (parallelism), shared data production and consumption, and client-server communication.

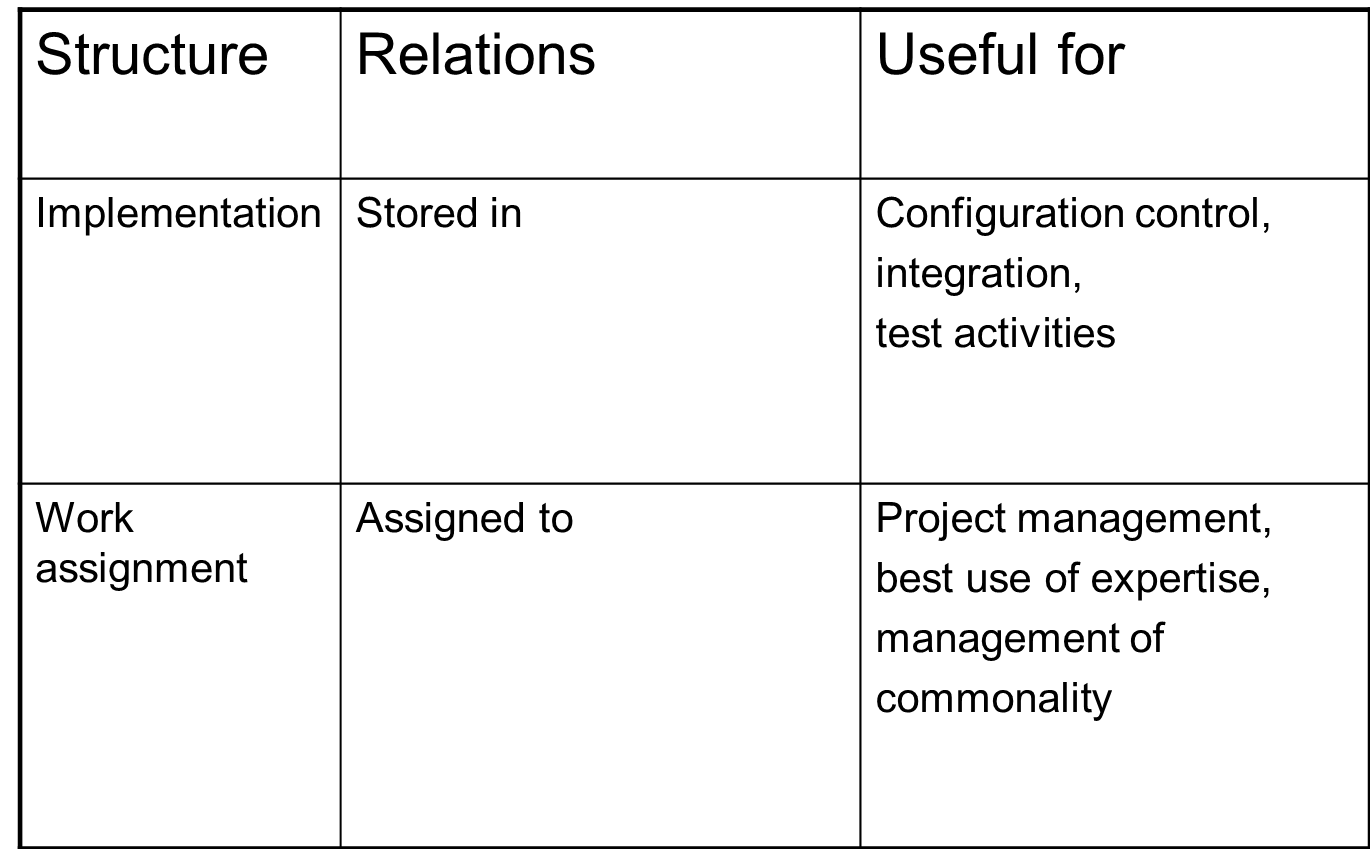
3.Allocation structures: the elements are deployment (onto hardware and communication components), implementation (in terms of the file system), and work assignment (breakdown according to development teams).

**List 5 architecture structures according to Bass et al.**





****

****

**What is a quality attribute scenario?**

a means to characterize system quality attributes, consists of 6 parts:

stimulus source

stimulus

environment

artifacts affected

system response

measurement of response

A unified way to express quality requirements

**Discuss the benefits of architectural reviews.**

1、financial

2、forces preparation for review

推动为评审做准备

3、captured rationale

4、early detection of problems

尽早地发现问题

5、validation of requirements

确认需求

6、improved architectures

提高体系结构质量

**When can architectural reviews begin?**

Early “architecture discovery review”

* + - is done after requirements are set, but before the architecture is firm  
      需求分析之后，体系结构还没有确定之前进行
    - is used to understand implications of requirements on architecture  
      用来理解需求在体系结构方面的隐含内容
    - checks for requirements feasibility（检查需求的可行性）
    - prioritizes architectural goals（为质量目标排序）

Full architectural review

* + - is done when architectural documentation is available  
      体系结构文档可用时进行
    - is used to evaluate qualities of proposed architecture  
      评价被评体系结构的质量

**What is an unplanned architectural review? why should the organization have it?**

计划外的评审是未曾预料到的，通常是因为项目存在严重的问题，需要采取极端的措施来补救以前的工作。

当管理层认为项目可能会失败，需要在开发过程中进行纠正时，进行计划外评审。

* Planned review
  + part of normal development activity 正常开发活动的一部分
  + non-adversarial（非敌对性的）
  + scheduled well in advance（事先安排好日程）
  + built into project’s schedule and budget 包含在项目日程和预算之内
* Unplanned
  + usually occurs when project is in trouble 通常在项目出现问题时采用
  + often devolves into finger-pointing 导致互相责备
  + can be painful for project already struggling

对于已经苦苦挣扎了很久的项目而言，有些痛苦

**What is brainstorming?**

大脑风暴法是将少数人召集在一起以会议的形式，对某一个问题进行自由地思考和联想，提出各自的设想和提案，所有参与者不准对其他人言论正确性或准确性进行 任何评价的一种讨论方法。它能将团队的智慧有效的结合利用起来，对于企业组织的决策具有重要意义，因而深受管理者的青睐。

**Explain risk points in architectural decision.**

**ATAM有风险决策**：根据所陈述的质量属性需求，可能导致不期望有的结果的构架决策。

* While risks are potentially problematic architectural decisions, …  
  风险是有潜在问题的体系结构

**Explain non-risk points in architectural decision.**

**ATAM 无风险决策**：根据分析被认为是安全的构架决策。

* Non-risks are good decisions relying on implicit assumptions.  
  非风险是在一个可信的假设之下的，好的方案

**Explain sensitivity points in architectural decision.**

**Sensitivity points**某决策很可能影响某些质量

* Sensitivity points are candidate risks and candidate tradeoff points.  
  敏感点是候选的风险和折中

**Explain trade-off points in architectural decision.**

**Tradeoffs** 某几种质量属性中的折中点

* Example risks
  + Rules for writing business logic tier of your 3-tier style are not clearly articulated.  
    三层构架下，商业逻辑层的规则还没有确定
  + There is no way of detecting the “live” failure of a critical component.  
    没有检测一个关键组件是否正常工作的机制
* Example non-risk
  + Assuming message arrival rates of once per second, a processing time of less than 30 ms, and the existence of one higher priority process, a 1 second soft deadline seems reasonable.  
    假定消息的到达速率是每秒一次，一次处理的时间小于30ms。如果对一个更高优先级的处理的响应时间要求是1秒钟，此系统可行（不会漏掉消息）
* Example Sensitivity
  + Changing the timing scheme from a harmonic framework to a non-harmonic framework would be easy, but due to implied timing dependencies, there would impact far reaching impacts to other modules.  
    把定时方法从一个谐波的框架移植到一个非谐波的框架可能很容易，但是因为各个模块对定时的依赖，可能会极大地影响它们的正常工作
* Example Tradeoffs
  + In order to achieve the required level of performance in the discrete event generation component, assembly language had to be used thereby reducing the portability of this component.  
    为了达到性能要求，不得不在离散的事件产生组件中使用汇编语言。此组件不再有移植性

**List ATAM steps.**

1. Present the ATAM 介绍ATAM
2. Present business drivers 讲解商业动力
3. Present architecture 讲解体系结构
4. Identify architectural approaches 明确体系结构方法
5. Generate quality attribute utility tree 生成有效树
6. Analyze architectural approaches 分析体系结构方法
7. Brainstorm and prioritize scenarios 自由讨论和为场景排序
8. Analyze architectural approaches 分析体系结构方法
9. Present results 讲解结论

**What is a utility tree?**

* + A utility tree is a top-down vehicle for characterizing the “driving” attribute-specific requirements 效用树是一个自顶向下的工具，用来刻画重要的需求
  + Select the most important quality goals to be the high-level nodes (typically performance, modifiability, security, and availability)  
    把最重要的质量目标放在高层节点（典型的有：性能、适应性、安全和可用性）
  + Scenarios are the leaves of the utility tree

**What is a product line?**

A product line is a group of products sharing a common, managed set of features that satisfy specific needs of a selected market, and that are developed from a common set of core assets in a prescribed way.

**生产线：**一组软件密集型系统，它们共享一个公共的、可管理的特性集，满足了某个特定市场或人物的具体需要，是以规定的方式用公共的核心资产集开发出来的。

**From a product line, how products can be produced?**

-从公共核心资产库(**core** **assets base**)中选取合适的构件

-使用预定义的变化性机制进行裁剪，如：参数化、继承

-必要时增加新的构件

-在整个产品线范围内共同的体系结构指导下，进行构件组装，形成系统

论述题

**How do the environments influences designs?**

Stakeholder with different, sometimes conflicting concerns;

Business Environment and Organizational Structure (organizational structure, resources, assets, business strategies);

The Architect's Background and Experience

**How do Architecture Design Change the Business Environment?**

They affect the structure of the organization.

They affect the business goals of the organization.

They affect customer requirements for new products.

They affect how future architectures are designed.

**What is availability? what tactics can be taken to use for availability?**

可用性与系统故障及其相关后果有关。当系统不再提供其规范中所说明的服务时，就出现了系统故障。

可用性战术

错误检测：

命令/响应、心跳、异常

错误恢复：

表决、主动冗余（仅用第一个反应）、被动冗余（主响应，其他更新）、备件、Shadow 操作、状态再同步、检查点/回滚

错误预防：

从服务中删除、事务、进程监视器

**What is ADD? describe the steps to carry out ADD.**

ADD（属性驱动设计）是一种定义软件构架的方法，该方法将分解过程建立在软件必须满足的质量属性之上。它是一个递归的分解过程，其中在每个阶段都选择战术和构架模式来满足一组质量属性场景，然后对功能进行分配，以实例化由该模型所提供的模块类型。

1. 选择要分解的模块

最初是整个系统。该模块要求的所有输入（约束、功能需求、质量需求）都必须是可获得的。

1. 根据这些步骤对模块进行求精

2.1、从具体的质量场景和功能需求集合中选择构架驱动因素

2.2、选择满足构架驱动因素的构架模式。

2.3、实例化模块并根据用例分配功能。

2.4、定义子模块的接口。

2.5、验证用力和质量场景并对其求精。

3、对需要进一步分解的每一个模块重复上述步骤

**How to create a skeletal system?**

* + 首先，实现处理构架组件的执行和交互的软件部分。包括：实时系统中的调度程序，实现规则引擎（带有规则的原型集）以控制在基于规则的系统中规则的激发；实现多进程系统中的进程同步机制或客户/服务器系统中的客户机/服务器的协同。
  + 选择把提供功能的哪些元素添加到系统中。可能根据以下因素：首先处理问题最多的部分来降低风险；或现有开发人员的类型和水平；或尽可能快地将有用的产品推向市场。
  + 选择了提供下一个功能增量的元素后，就可以采用使用结构，以获知应该在系统中采用什么软件来支持该功能。
  + 随着该过程的继续，系统的增量越来越大，直到软件开发完毕。（任意时刻集成和测试任务都不会很多，在每个增量中都很容易找到最近引入的错误源）

**What are the benefits of a product line?**

Defect reduction: defect fixed in one product automatically fixed for all future products

减少缺陷：在一个产品中修正的缺陷，自动在未来的产品中也被修正

Performance: performance issues addressed for all products (e.g., schedulability, deadlock, distributed system issues)

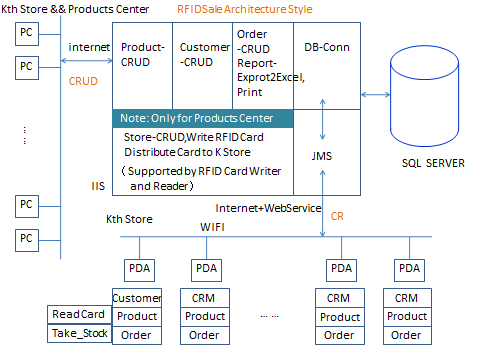
性能：性能问题在所有产品中都被解决

Planning: more accurate because all products are produced in the same way  
计划：更准确，因为所有的产品都用相同的方式生产

Reduction in

* + time to market
  + staffing

大题



1. 请根据所提供RFIDSale系统的构架模式和RFIDSale系统的需求规约简表RFIDSaleSRS，进行深入的分析，提出并明确系统的质量属性及其实现战术。
2. 在全面质量管理的思想指导下，根据系统的质量属性及其实现战术，提出系统的参考模型、参考的构架框架。
3. 在1,2的基础上，基于ADD方法，提出并详细描述目标系统RFIDSale的软件体系结构；
4. 基于 ATAM方法，提出目标系统RFIDSale的软件体系结构的项目组内部评估结果。
5. 各位把六个质量属性和它们对应的战术都和这个系统的需求结合起来，扯上去，就可以得分。

例如:

安全性：PDA只能下单。需要保持数据的安全性，机密性。然后结合具体把安全性战术写上去。

例如：

互操作性：PDA如何和系统交互：PDA和系统交互是通过JMS的（涉及到SOA的知识），把互操作性扯上去。

例如

性能：PC是通过Internet来和系统交互的。网络的带宽限制了性能。需要提高性能，这样可以使用.NET的缓冲池策略可以提高性能，（.net的缓冲池策略请参考南哥宝典442页上得PETshop）把想到的战术写上去。

等等。。。。自己想多点。。

1. 这个系统是是一个分布式-分层系统。PDA和服务器之间是采用了C/S的构架。PC和服务器是采用了B/S的构架。所以系统采用了C/S和B/S的混合构架模式。参考的构架框架采用了.net的MVC三层构架框架。
2. PDA和服务器之间是采用了C/S的构架。PC和服务器是采用了B/S的构架。结合ADD：先描述构架驱动因素，然后描述选择的构架。然后描述选择的构架，各位把上面说到的的C/S和B/S的优缺点，写上去，并且说出C/S和B/S混合后的优点，.NET框架的优缺点。（ C/S和B/S的优缺点请参考南哥宝典的140页之后描述的构架风格。），最后描述下分配的功能。

.net优缺点：

三层构架与具体语言没有关系  
广义的为数据层逻辑层显示层  
优点是代码复用性提高可维护性搞维护起来方便层与层之间分离  
缺点是如果你写的项目比较小那代码量可能比不用三层更多

1. 第四题的话。在下实在不知道如何表达。各位看着办吧。

上面的回答只是小女的愚见。不对之处请指点。南哥宝典貌似海东有群邮件过。。

考试点就在六大质量属性和战术，C/S,B/S优缺点和.NET的缓冲池策略，分布式-分层结构