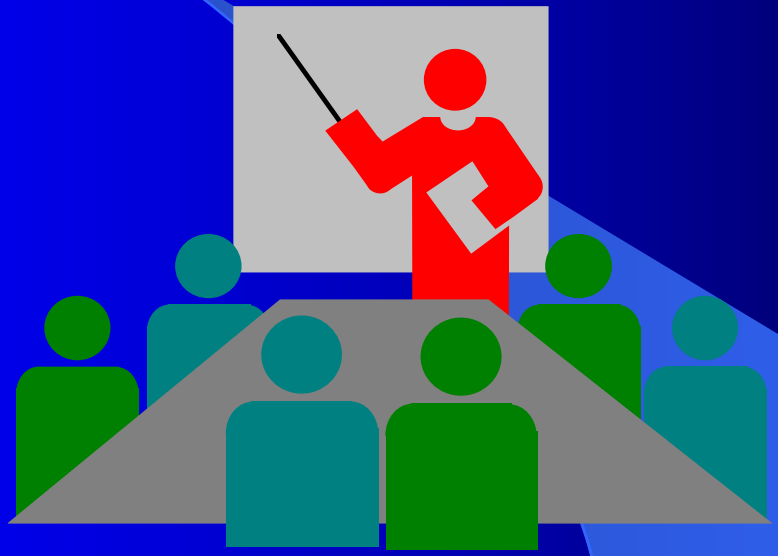
软件体系结构



二0一九年六月

第六章 软件体系结构分析评估

6.1 概述

6.2 软件体系结构评估方法

6.3 软件体系结构风险分析

6.1 概述

软件体系结构评估，是对系统某些属性(性能、可修改性、可靠性等)进行评价和判断，得到的评估结果可用于确认潜在的风险，并检查设计阶段所得到的系统需求质量，在系统被实际构造之前，预测其质量属性。

6.1 概述

1、体系结构质量要素

一个良性的软件体系结构应该有以下五点质量要素：

1）体系结构应是适宜的

它包括软件的体系结构是适宜用户需求，具有适应用户需求，及需求变化的能力。

它是适宜开发的，用现有的技术和技能完成软件的实现、维护和测试。

它是适宜项目管理，有利于对项目的资源(人力、资金、设备)的配置，减少项目开发的风险。

6.1 概述

1、体系结构质量要素

2）体系结构应是概念完整的

体系结构设计应该表现出整体的协调、一致和可预测性，也就是概念完整性。不论有多少人参与了设计与实施，系统结构应该反映为统一的设计思想并遵循统一的概念和表达。这里主要强调的是内在结构的统一完整性，也包括外在表现的统一的完整性。

3）体系结构应是易于维护和升级的

体系结构应是清晰的，由易于替换、自成体系的基本组件构成的，各基本组件间，有较少的内部依赖，体系结构支持可重构，可扩充。

6.1 概述

1、体系结构质量要素

4）体系结构应是便于移植的

体系结构的设计能够把开发成功的设计从一种环境移植到另一种环境下运行，实现体系结构的部分设计或整个设计可以在不同的环境下获得重用。

6.1 概述

1、体系结构质量要素

5）体系结构的理性化

设计者在实施体系结构时对设计的正确性就很有信心，那么开发活动受了理性化的控制；如果设计者仅仅依靠软件实现后的测试判定核心设计的正确性，那么开发过程就很难受到理性化控制。体系结构的理性化要求设计者对所采用的技术和设计的结构具有清晰的认识和理解。受到理性化控制的体系结构设计必然是结构清晰合理、关系明确可控、表达简捷正确、并易于理解的。一个理性化的体系结构有利于控制工程费用和进度，有利于质量和性能预测及后期维护升级。

6.1 概述

2、体系结构的评价指标体系

一个良性的体系结构体现了五个质量要素，下面分别从需求角度、开发角度和项目管理角度提出评价指标，形成了软件体系结构的评价指标体系。

（1）从需求角度

（2）从开发角度

（3）从项目管理角度

6.1 概述

2、体系结构的评价指标体系

（1）从需求角度

1）功能性

体系结构应该体现系统应该具有的功能，能完成所期望的工作的能力。一项任务的完成需要系统中许多或大多数构件的相互协作。体系结构应能反应系统需求的所有功能点。

我们可以通过体系结构的各个部分如何交互、如何相互协作完成系统功能来考察功能性。通过体系结构满足系统的状况，评价功能性的强弱。

6.1 概述

2、体系结构的评价指标体系

（1）从需求角度

2）性能

是指按该体系结构实施后的系统的响应能力，即要经过多长时间才能对某个事件做出响应，或者在某段事件内系统所能处理的事件的个数。影响性能的因素有系统吞吐量、数据延迟等。诸如计算在组件上怎样分配以及组件(内部进程和交互进程)间的通信模式等决策都影响性能的体系结构因素。体系结构可以帮助进行部分的性能特性分析，体系结构设计应该以低层实现设计为基础，规划体系结构层次模型的资源及管理调度和基础控制描述。

6.1 概述

2、体系结构的评价指标体系

（1）从需求角度

3）可靠性

可靠性是按该体系结构实施下的软件系统在应用或系统出错面前，在意外或错误使用的情况下采用体系结构策略维持软件系统功能特性的基本能力。通常用它衡量在规定的条件和时间内，软件完成规定功能的能力。可靠性可以分成以下两个方面。

6.1 概述

2、体系结构的评价指标体系

（1）从需求角度

3）可靠性

(a)容错

其目的是在错误发生时确保系统正确的行为，并进行内部“修复”。例如在一个分布式软件系统中失去了一个与远程构件的连接，接下来恢复了连接。在修复这样的错误之后，软件系统可以重新或重复执行进程间的操作直到错误再次发生。

6.1 概述

2、体系结构的评价指标体系

（1）从需求角度

3）可靠性

(b)健壮性

是指保护应用程序不受错误使用和错误输入的影响，在遇到意外错误事件时确保应用系统处于已经定义好的状态。例如：软件体系结构通过在应用程序内部增加检测错误输入的组件和消除错误影响的组件，来增强系统的健壮性。

6.1 概述

2、体系结构的评价指标体系

（1）从需求角度

4）可用性

是指按该体系结构实施的系统能够正常运行的时间比例。经常用两次故障之间的时间长度或在出现故障时系统能够恢复正常的速度来表示。体系结构通过增加冗余和故障检测技术构件，来提高系统的可用性。

6.1 概述

2、体系结构的评价指标体系

（1）从需求角度

5）安全性

是指按该体系结构实施的系统在向合法用户提供服务的同时能够阻止非授权用户使用的企图或拒绝服务的能力。安全性是根据系统可能受到的安全威胁的类型来分类的。安全性又可划分为机密性、完整性、不可否认性及可控性特性。其中，机密性保证信息不泄露给未授权的用户、实体或过程；完整性保证信息的完整和准确，防止信息被非法修改；可控性保证对信息的传播及内容具有控制的能力，防止为非法者所用。体系结构可通过对构件交互、通信分析系统的安全性，并可修改软件体系结构，使之具有更高的安全级别。

6.1 概述

2、体系结构的评价指标体系

（1）从需求角度

6）扩充性

系统的体系结构应满足系统生存期间额外的要求的能力。体系结构中允许增加新的构件或结构，以满足系统新的需求。

7）死锁性

系统在执行过程不会出现被“卡住”，不能继续向前执行的状态。在体系结构的设计中，在涉及到构件间的交互时，要保证交互的无死锁性。

6.1 概述

2、体系结构的评价指标体系

（1）从需求角度

8）适应性

体系结构是否有适应用户变化而变化的能力。它要求软件体系结构设计是基于一定表达和处理模型的，允许用户按照需求独立定义和生成所需数据和功能。

6.1 概述

2、体系结构的评价指标体系

（2）从开发角度

9）可维护性

按该体系结构实施的系统在错误发生后“修复”软件系统的能力。为可维护性做好准备的软件体系结构往往能做局部性的修改并能使对其他构件的负面影响最小化。

6.1 概述

2、体系结构的评价指标体系

（2）从开发角度

10）可扩展性

是指该体系结构支持用新特性来扩展软件系统，允许使用改进构件版本来替换旧构件，并删除不需要或不必要构件。为了实现可扩展性，软件系统需要松散耦合的构件。它反映了体系结构能使开发人员在不影响客户的情况下替换构件以及支持把新构件集成到现有的体系结构中的能力。

6.1 概述

2、体系结构的评价指标体系

（2）从开发角度

11）结构重组

是指该体系结构根据开发需要重新组织软件系统的构件及构件间的关系的能力。例如：体系结构允许将构件移动到一个不同的子系统而改变它的位置。为了支持结构重组，软件系统需要精心设计构件之间的关系。理想情况下，它们允许开发人员在不影响实现的主体部分的情况下灵活地配置构件。

6.1 概述

2、体系结构的评价指标体系

（2）从开发角度

12）移植性

是指该体系结构实施下软件系统适用于多种硬件平台、用户界面、操作系统、编程语言或编译器的能力。为了实现可移植性，需要按照环境无关的方式组织软件系统，这些环境可能是硬件或软件，也可能是两者的结合。在关于某个特定计算环境的所有假设都集中在一个构件中时，系统是可移植的。如果移植到新的系统需要做些更改，则可移植性就是一种特殊的可修改性。

6.1 概述

2、体系结构的评价指标体系

（2）从开发角度

13）可变性

是指体系结构经扩充或变更而成为新体系结构的能力。这种新体系结构应该符合预先定义的规则，在某些具体方面不同于原有的体系结构。当要将某个体系结构作为一系列相关产品的基础时，可变性是很重要的。

14）可集成性

是指按此体系结构构成的系统能与其他部件协作的能力。可集成性也反映部件间、子系统间的协作容易程度。

6.1 概述

2、体系结构的评价指标体系

（2）从开发角度

15）互操作性

是指该体系结构实施下的系统与其他系统或自身环境相互作用的能力。为了支持互操作性，软件体系结构必须为外部可视的功能特性和数据结构提供精心设计的软件入口。

16）可测试性

是指软件体系结构可以为错误探测和改正，以及调试代码和部件的临时集成给于支持的能力。

6.1 概述

2、体系结构的评价指标体系

（2）从开发角度

17）可理解性

是指软件体系结构的概念和描述能够全面表达和深刻理解的程度，反映了该体系结构支持软件设计者之间、设计者与用户之间可以快速方便地交流知识、经验和新设计思想的能力。

18）可重用性

是指体系结构的框架或映射到体系结构中的构件可以重用体系结构库的结构或构件库的构件的程度。可重用性的衡量与开发部门所拥有的体系结构资源和构件资源的积累有关。

6.1 概述

2、体系结构的评价指标体系

（2）从开发角度

19）效率性

是指按该体系结构实施的系统在运行时存储空间、运行时间、吞吐量等对系统的影响程度。体系结构虽然不能通过运行得到上述信息。但可以根据构件效率、构件交互通信效率进行分析和估计，来反映系统的效率性。效率性的目标是建立性能与消耗比值高的体系结构。

6.1 概述

2、体系结构的评价指标体系

（3）从项目管理角度

20）部件无关

是指体系结构中部件无需了解其它组件的存在就能独立工作的特性。部件无关为开发带来便利，因为部件之间没有复杂的关系，可以分别单独开发，不需协调。

21）风险性

按此体系结构实施的技术和技能的风险度是多少。

6.1 概述

2、体系结构的评价指标体系

（3）从项目管理角度

22）复用性

反映该体系结构的抽象性和通用性的程度，该结构设计或构件设计在以后的项目开发中是否可以被复用。提高体系结构的可重用性，要求系统的组成模块或者成分应该是结构化(高内聚低藕合)和参数化(参数化的部件接口机制)的，并且按照适当的方式存档，能够反映领域知识、领域模型。

6.1 概述

2、体系结构的评价指标体系

（3）从项目管理角度

23）可度量性

按该体系结构实施，系统的开发进度、人力、资金、资源调配可以估计的能力。

6.1 概述

2、体系结构的评价指标体系

（3）从项目管理角度

24）正交性

是指反映体系结构中同一层次的构件之间不存在相互调用的状况。由于体系结构的正交性，可以把开发人员分成若干个小组进行并行开发，视开发难度情况，每个小组负责一条或数条线索。由于各条线索之间没有相互调用，各小组工作不会相互牵制。正交性强的体系结构，可大大提高编程的效率，缩短开发周期。

6.2 软件体系结构评估方法

从目前已有的软件体系结构评估技术来看，基本可以归纳为三类主要的评估方式：基于调查问卷或检查表的方式、基于场景的方式和基于度量的方式。

6.2 软件体系结构评估方法

1、基于调查问卷或检查表的方式

调查问卷是一系列可以应用到各种体系结构评价的相关问题，其中有些问题可能涉及到体系结构的设计决策；有些问题涉及到体系结构的文档，比如体系结构的表示用的是何种ADL；有的问题针对体系结构描述本身的细节问题，如系统的核心功能是否与界面分开。

检验表中也包含一系列比调查问卷更细节和具体的问题，它们更趋向于考察某些关心的质量属性。例如，对实时信息系统的性能进行考察时，很可能问到系统是否反复多次地将同样的数据写入硬盘。

6.2 软件体系结构评估方法

1、基于调查问卷或检查表的方式

这一评价方式比较自由灵活，可评价多种质量属性，也可以在软件体系结构设计的多个阶段进行。但是由于评价的结果很大程度上来自于评价人的主观判断，因此不同的评价人可能会产生不同甚至截然相反的结果，而且评价人对领域的熟悉程度、是否有丰富的相关经验也成为评价结果是否准确的重要因素。尽管基于调查问卷与检查表的评价方式相对比较主观，但由于系统相关的人员的经验和知识是评价软件体系结构的重要信息来源，因而它仍然是进行软件体系结构质量评价的重要途径之一。

6.2 软件体系结构评估方法

2、基于场景的方式

“场景”(Scenarios)是一系列有序的使用或修改系统的步骤。基于“场景”的方式由SEI 首先提出并应用在SAAM和ATAM 中。这种软件体系结构质量评价方式分析软件体系结构对“场景 ”也就是对系统的使用或修改活动的支持程度，从而判断该体系结构对这一“场景”所代表的质量需求的满足程度。这一评价方式考虑到了包括系统的开发人员、维护人员、最终用户、管理人员、测试人员等等在内的所有和系统相关的人员对质量的要求。基于“场景”的评价方式涉及到的基本活动包括确定应用领域的功能和软件体系结构的结构之间的映射，设计用于体现待评估质量属性的“场景”以及分析软件体系结构对“场景”的支持程度。

6.2 软件体系结构评估方法

2、基于场景的方式

不同的应用系统对同一质量属性的理解可能不同，比如，对操作系统来说，可移植性被理解为系统可在不同的硬件平台上运行，而对于普通的应用系统而言，可移植性往往是指该系统可在不同的操作系统上运行。由于存在这种不一致性，对一个领域适合的“场景”设计在另一个领域内未必合适，因此基于“场景”的评价方式是特定于领域的。这一评价方式的实施者一方面需要有丰富的领域知识以对某一质量需求设计出合理的“场景”，另一方面必须对待评估的软件体系结构有一定的了解以准确判断它是否支持“场景”描述的一系列活动。

6.2 软件体系结构评估方法

3、基于度量的方式

度量是指为软件制品的某一属性所赋予的数值，如代码行数、方法调用层数、构件个数等。传统的度量研究主要针对代码，但近年来也出现了一些针对高层设计的度量，软件体系结构度量即是其中之一。代码度量和代码质量之间存在着重要的联系，类似地，软件体系结构度量应该也能够作为评判质量的重要依据。赫尔辛基大学提出的基于模式挖掘的面向对象软件体系结构度量技术、Karlskrona/Ronneby提出的基于面向对象度量的软件体系结构可维护性评价、西弗吉尼亚大学提出的软件体系结构度量方法等都在这方面进行了探索。

6.2 软件体系结构评估方法

3、基于度量的方式

上述基于度量的评价技术都涉及三个基本活动：首先需要建立质量属性和度量之间的映射原则，即确定怎样从度量结果推出系统具有什么样的质量属性；然后从软件体系结构文档中获取度量信息；最后根据映射原则分析推导出系统的某些质量属性。因此，这些评价技术被认为都采用了基于度量的评价方式。

6.2 软件体系结构评估方法

3、基于度量的方式

基于度量的评价方式提供更为客观和量化的质量评估。这一评价方式需要在软件体系结构的设计基本完成以后才能进行，而且需要评价人对待评估的体系结构十分了解，否则不能获取准确的度量。自动的软件体系结构度量获取工具能在一定程度上简化评价的难度，例如MAISA可从文本格式的UML图中抽取面向对象体系结构的度量。

6.2 软件体系结构评估方法

典型方法

6.2 软件体系结构评估方法

1、SAAM

卡耐基梅隆大学软件工程研究所的Kazman等人和Texas大学的研究人员于1993年提出基于场景的体系结构分析方法SAAM(scenario-based Architecture Analysis Method)。该方法是一种非功能质量属性的体系结构分析方法，是最早形成文档并得到广泛使用的软件体系结构分析方法。它最初是用来分析软件体系结构的可修改性，但实践证明也可用于许多其它质量属性(例如可移植性、可扩充性、可集成性等)及系统功能快速评估。

6.2 软件体系结构评估方法

1、SAAM

SAAM的一个假设前提是单个质量属性之间是独立不相干的，这样的假设当然可以简化分析过程，但实际上多个质量属性之间并非独立，而是相互影响。比如说，如果一个系统对安全性要求很高，它在性能方面就会有所损失。

6.2 软件体系结构评估方法

1、SAAM

总的来说，SAAM评估分六个步骤：

(1)场景开发

通过集体讨论，风险承担者提出反映自己需求的场景。

(2)SA描述

SAAM定义了功能性、结构和分配三个视角来描述SA。功能性指示系统做了些什么，结构由组件和组件间的连接组成，而从功能到结构的分配则描述了域上的功能性是如何在软件结构中实现的。场景的形成与SA的描述通常是相互促进的，并且需要重复地进行。

6.2 软件体系结构评估方法

1、SAAM

(3)场景的分类

在分析过程中需要确定一个场景是否需要修改该体系结构。不需要修改的场景称为直接场景，需要修改的场景则称为间接场景。另一方面需要对场景设置优先级，以保证在评估的有限时间内考虑最重要的场景。

6.2 软件体系结构评估方法

1、SAAM

(4)单个场景的评估

主要针对间接场景，列出为支持该场景所需要对体系结构做出的修改，并估计出这些修改的代价。而对于直接场景只需弄清体系结构是如何实现这些场景的。

6.2 软件体系结构评估方法

1、SAAM

(5)场景交互的评估

两个或多个间接场景要求更改体系结构的同一个组件就称为场景交互。对场景交互的评估，能够暴露设计中的功能分配。

(6)总体评估

按照相对重要性为每个场景及场景交互设置一个权值，根据权值得出总体评价。

6.2 软件体系结构评估方法

2、ATAM

卡耐基梅隆大学的Kazman和Barbacci等人于2000年提出了体系结构折中分析方法ATAM(Architecture Trade-off Analysis Method)。该方法适应于多质量属性情况下的体系结构质量模型、分析和权衡，是在SAAM的基础上发展起来的，针对性能、实用性(availability)、安全性和可修改性，在系统开发之前，对这些质量属性进行评价和折中。

6.2 软件体系结构评估方法

2、ATAM

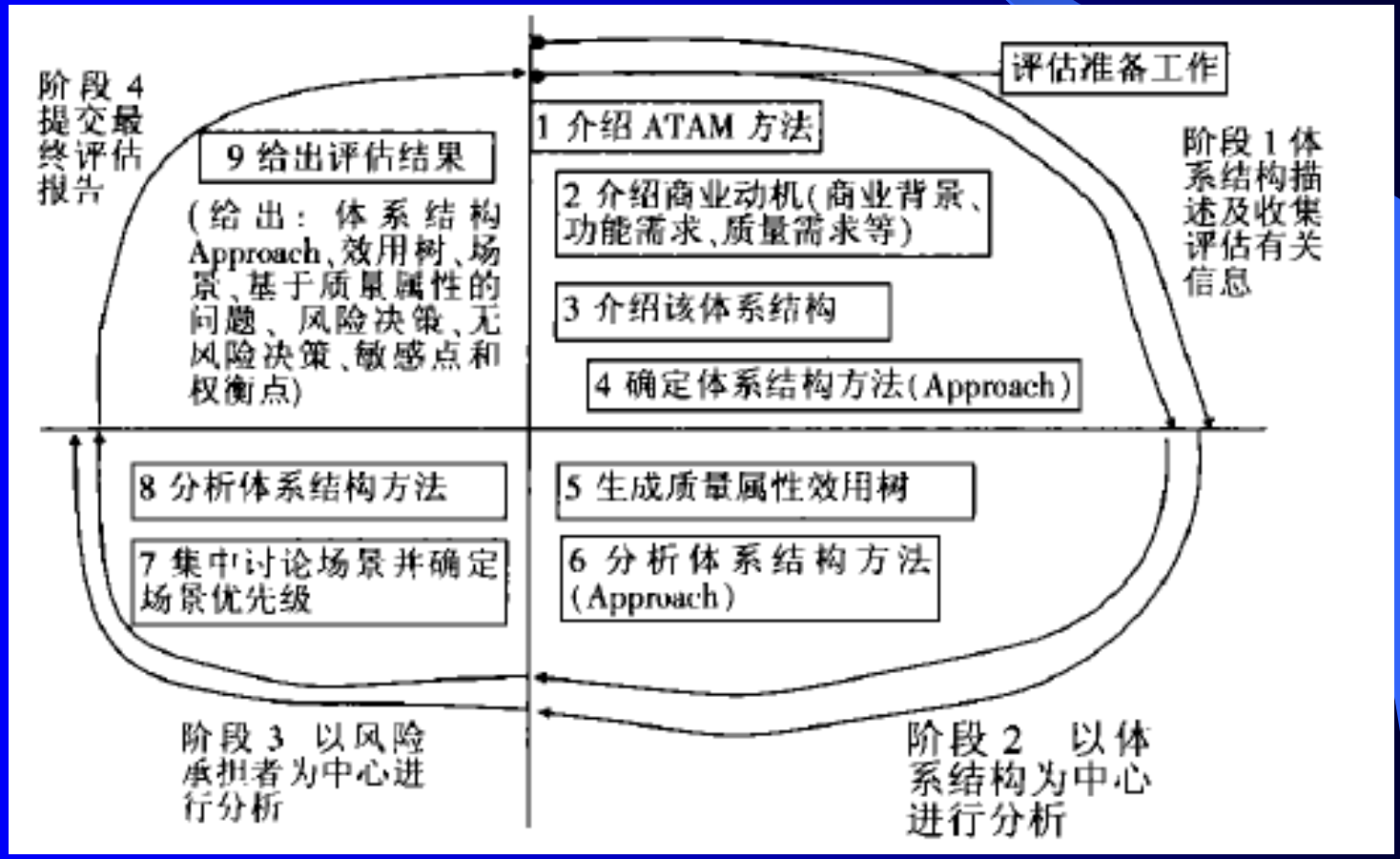
该方法与SAAM 最大的差别就在于考虑了各种质量属性之间的联系和冲突。

SAAM考察的是软件体系结构单独的质量属性，而ATAM提供从多个竞争的质量属性方面来理解软件体系结构的方法。使用ATAM不仅能看到体系结构对于特定质量目标的满足情况，还能认识到在多个质量目标间权衡的必要性。

6.2 软件体系结构评估方法

2、ATAM

ATAM的评估过程分为四个阶段，九个步骤。



6.2 软件体系结构评估方法

2、ATAM

1）介绍ATAM方法

评估小组负责人向参加会议的风险承担者介绍ATAM评估方法。

- ✓ ATAM方法步骤简介
- ✓ 获取和分析技术
- ✓ 评估结果

6.2 软件体系结构评估方法

2、ATAM

2）介绍商业动机

项目经理从商业角度介绍系统的概况。

- ✓ 系统最重要的功能需求
- ✓ 技术、管理、经济或政治方面的约束条件
- ✓ 商业目标和环境
- ✓ 主要的风险承担者
- ✓ 体系结构驱动因素（主要质量属性目标）

6.2 软件体系结构评估方法

2、ATAM

3）介绍体系结构

首席设计师或设计小组要对体系结构进行详细适当的介绍。

- ✓ 技术约束
- ✓ 要与本系统交互的其他系统
- ✓ 用以满足质量属性要求的体系结构方法

6.2 软件体系结构评估方法

2、ATAM

4）确定体系结构方法

由设计师确定体系结构方法，由分析小组捕获，但不进行分析。

5）生成质量属性效用树

评估小组、设计小组、管理人员和客户代表一起确定系统最重要的质量属性目标，并对这些质量目标设置优先级和细化。

6.2 软件体系结构评估方法

2、ATAM

6）分析体系结构方法

评估小组对每一种体系结构方法依据质量属性效用树完成相应质量属性的初步分析，确定它们的风险、敏感点和权衡点等。

- ✓ 理解体系结构方法
- ✓ 找出该方法的缺陷
- ✓ 找出该方法的敏感点
- ✓ 发现与其他方法的交互和权衡点

6.2 软件体系结构评估方法

2、ATAM

7）讨论和分级场景

风险承担者集体讨论用例场景和改变场景，改变场景又分成成长场景和考察场景。

场景设计完成后，风险承担者通过投票表决的方式确定其优先级。

8）分析体系结构方法

设计师把场景映射到所描述的体系结构中，分析体系结构如何实现该场景。

6.2 软件体系结构评估方法

2、ATAM

9）描述评估结果

对ATAM分析得到的各种信息进行归纳，并反馈给风险承担者。

- ✓ 已文档化的体系结构方法/风格
- ✓ 场景及优先级
- ✓ 基于属性的问题
- ✓ 效用树
- ✓ 所发现的风险决策
- ✓ 已文档化了的无风险决策
- ✓ 所发现的敏感点和权衡点

6.2 软件体系结构评估方法

3、SAEM

软件体系结构评估模型方法SAEM(Software Architecture Evaluation Model)是以准软件评估过程(ISO/IEC9216)为基础对质量模型进行选择，并且该方法提出用一个概念框架把质量需求、度量标准、体系结构的内部属性与最终的系统联系起来。

6.2 软件体系结构评估方法

4、SBAR

Bengtsson等提出的基于场景的体系结构再工程方法SBAR(Scenario-based Architecture Reengineering)。该方法不仅对体系结构设计起着重要作用，而且还可以对一个系统的详细体系结构进行基于场景的软件质量评估。

提供5种类型的体系结构转变：改变体系结构风格、应用体系结构模式、利用设计模式、将质量需求转变为功能性需求和将质量需求分类。SBAR方法有3个主要活动：将新的功能需求合并到体系结构中，软件质量评估和体系结构转变。

6.2 软件体系结构评估方法

5、ALPSM

Bengtsson和Bosch提出的体系结构层次的软件可维护性预测方法 ALPSM(Architecture Level Prediction of Software Maintenance)。该方法以体系结构考察场景的影响，并对系统的可维护性进行分析。它将可维护性的预测因子定义为场景变更的规模，然后通过场景变更所需的维护代价来分析体系结构可维护性。ALPSM方法结合设计经验和历史数据对可维护性框架进行验证，并且有效地引入预测、变更系统的可维护性，但该方法具有一些不确定性，例如如何验证可维护性框架具有代表性。

6.2 软件体系结构评估方法

5、ALPSM

ALPSM方法包括如下六个步骤：

(1)确认维护任务的分类。这种分类是基于应用或特定域的，因此并不抽象。

(2)合成场景。选择对于维护类别有代表性的场景，一般每一类选10个场景。这里的场景与通常讲的描述系统行为的用例场景不同，它描述的是与系统相关的可能发生的活动或活动的序列。一个变化场景则描述了系统的某个维护任务。

6.2 软件体系结构评估方法

5、ALPSM

(3)给每个场景分配一个权值。定义权值为在某个特定间隔时间内，这个场景导致一个维护任务的相对概率。产生场景的权值要么使用历史维护数据来推断，要么由体系结构设计师或域专家来估算。

(4)估算所有组件的大小。组件的大小影响在组件中实现一个改动所需的工作，因此通过估算组件的大小来估算维护工作。

6.2 软件体系结构评估方法

5、ALPSM

(5)分析场景。对于每个场景，评估在体系结构及其组件中实现该场景带来的影响，最终发现哪些场景受到影响及被改变到何种程度。

(6)计算所预计的维护工作。预测值是每个维护场景的工作的平均加权，体现了每个维护任务的平均工作量。

6.2 软件体系结构评估方法

6、ALMA

Bengtsson等人2004年提出的基于预测的软件体系可修改性的分析方法 ALMA(Architecture Level Modifiability Analysis)。该方法主要是基于风险评估和可维护性成本预测等度量指标，然后通过对变更场景的构建及评价进行可修改性的分析，并假设变更规模为最主要的可修改性成本因素，最后构造了一个修改性预测模型。它引入了定量的度量指标，支持从成本预测、风险评估、体系结构选择等多个角度评估体系结构的可修改性，并提供了场景构建的停止准则，但缺少了对结果准确性的判断和风险评估完整性的判断。

包括确定目标、体系结构描述、发现并选择变更场景、评价场景和得出结论5个主要步骤。

6.2 软件体系结构评估方法

7、ALRPA

Yacoub于2002年提出的一种软件体系结构层次的可靠性风险分析方法ALRRA(Architecture-Level Reliability Risk Analysis)。该方法主要基于如下假设：构件执行的频率越高，失效的可能性越大。它首先利用ROOM(Realtime Object Oriented Modeling)对体系结构进行建模，通过对体系结构的模拟进行可靠性关键程度的分析，构造体系结构的 CDG(Component Dependency Graph)图，通过图形转换算法对体系结构的风险进行分析和评估。但是ALRRA方法在度量指标参数的不确定性对评价结果的影响，方法在大型系统分析中的应用等方面需要改进，另外它采用了模拟和场景结合的技术，所以成本比单独采用场景技术的方法也要高。

6.2 软件体系结构评估方法

8、国内

国内一些研究人员也致力于软件体系结构的研究，但对体系结构分析评价方面的研究比较少。主要有：随机进程代数理论的体系结构性能评价方法**ESPA-SAPE**和基于队列网络模型的性能评价方法**QNM-SAPE**。

6.2 软件体系结构评估方法

8、国内

ESPA-SAPE是根据体系结构的特点，对随机进程代数PEPA进行扩展，然后再建立体系结构的ESPA模型，用以求解具有Markov或者semi-Markov特性的转换矩阵，进而求解稳定状态概率分析，计算系统的性能指标。

6.2 软件体系结构评估方法

8、国内

QNM-SAPE方法是基于队列网络模型(Queuing Network Model，QNM)的，对体系结构采用化学抽象机（chemical abstract machine，CHAM)进行形式化描述，然后导出体系结构性能评价模型，用于多个候选体系结构的选择。

6.3 软件体系结构风险分析方法

美国原子能委员会早在六十年代就提出了用风险来评估安全性。在ISO 8402(1994年)对安全性的定义为：将伤害(对人)或损坏(对物)的风险限制在可接受水平的状态。美国原子能委员会曾提出一个计算风险的公式，如公式所示：

风险（损失程度/单位时间）=频率（危险事件/单位时间）*严重性（损失程度/危险事件）

目前国际上都用风险来定义安全性，我国在1990年制定的GJB 900《系统安全性通用大纲》中对风险的定义为：危险事件的风险就是该事件的发生概率和损失程度的函数。

6.3 软件体系结构风险分析方法

现有的各种信息安全评估方法从不同的角度划分：有基于知识的风险分析方法、基于模型的风险分析方法、定性分析和定量分析。

6.3 软件体系结构风险分析方法

故障树分析法(FTA)是一种top-down方法，通过对不期望事件(系统的危险性故障，也叫顶事件)进行定性地分析，按树状结构，逐层细化故障；再在预测叶子结点的故障发生概率的基础上，通过逻辑关系最终得到所分析的系统故障事件的发生概率。进行故障树分析时，需要对整个系统进行全盘考虑。因此，在复杂系统中，这种方法将很复杂，因而在分析过程也很容易忽略某些故障而导致出错，而这些错误可能被忽略从而降低系统的安全性。

6.3 软件体系结构风险分析方法

危险性分析(Hazard Analyses)主要是识别环境危险在设计上的偏离，在使用上的潜在偏离及组件间接口的故障。

故障模式影响分析法(FMEA)主要是分析在系统关键组件可能的故障模式和每种故障模式对整个系统的影响。

6.3 软件体系结构风险分析方法

PSSA是一种早期系统安全性评估方法，它对一个已提出的体系结构进行系统的分析，并使用功能性危险分析FHA方法识别导致功能性危险的故障原因，然后还分析了FHA需求要怎么样才能得到满足。

Lisagor在此基础上又提出了一种轻量级PSSA方法用以对软件体系结构进行安全性分析。该方法以软件体系结构图、软件安全性方法和一种基于HAZOP的软件分析方法—SHARD为基础，通过构造构件依赖模型，演绎分析和回溯分析而进行。

6.3 软件体系结构风险分析方法

在计算机辅助安全性分析方面，国内外都包含了多种安全性分析工具，如FMECA、FTA、HAZOP、RiskVu等，但这些软件工具对于软件开发中的安全性支持力度有限，所支持的安全性分析领域比较固定。自20世纪80年代开始，国内也有一些单位开发了可靠性和安全性软件工具，如航天工业总公司一院12所、五院503所、北航等单位开发了FMECA软件，清华大学核能物理研究所开发了故障树分析软件THSFTA。受当时开发条件的限制，这些软件已经不能满足实际发展的需要。90年代，国防科技大学系统工程研究所开发了FTA、FMECA等软件工具，拥有数十家型号用户单位，在国内较有影响。

谢谢大家！

再见！