ATAM架构评估方法

ATAM方法步骤：

共 4 大部分，9 个步骤

部分	步骤	责任者	活动	目的
1. 表述	1. ATAM 方法的表述;	评估负责人	向评估参与者介绍 ATAM 方法并回答问题。 a. 评估步骤介绍 b. 用于获取信息或分析的技巧：效用树的生成、基于架构方法的获取和分析、对场景的集体讨论及优先级的划分 c. 评估的结果：所得出的场景及其优先级，用以理解和评估架构的问题、描述架构的动机需求并给出带优先级的效用树、所确定的一级架构方法、所发现的有风险决策、无风险决策、敏感点和权衡点等	使参与者对该方法形成正确的预期
	2. 商业动机的表述;	项目发言人（项目经理或系统客户）	阐述系统的商业目标 a. 系统最重要的功能 b. 技术、管理、政治、经济方面的任何相关限制 c. 与项目相关的商业目标和上下文 d. 主要的风险承担者 e. 架构的驱动因素（即促使形成该架构的主要质量属性目标）	说明采用该架构的主要因素（如：高可用性，极高的安全性或推向市场的时机）
	3. 架构的表述;	架构设计师	对架构做出描述 a. 技术约束条件，诸如要使用的操作系统，硬件，中间件之类的约束 b. 该系统必须要与之交互的其他系统 c. 用以满足质量属性的架构方法 d. 对最重要的用例场景及生长场景的介绍	重点强调该架构是怎样适应商业动机的
2. 调查和分析	4. 确定架构方法	架构设计师	确定所用的架构方法,但不进行分析	
	5. 生成质量属性效用树		生成质量属性效用树,详细的 根结点为效用，一直细分到位于叶子节点的质量属性场景，质量属性场景的（优先级，实现难度）用高（H）、中（M）、低（L）描述；不必精确	得出构成系统效用的质量属性(性能、可用性、安全性、可修改性、使用性等)； 具体到场景--刺激--响应模式，并划分优先级
	6. 分析架构方法		根据上一步得到的高优先级场景，得出应对这一场景的架构方法并对其进行分析 要得到的结果包括：	确定架构上的有风险决策、无风险决策、敏感点、权

			a. 与效用树中每个高优先级的场景相关的架构方法或决策； b. 与每个架构方法相联系的待分析问题； c. 架构分析师对问题的解答； d. 有风险决策，无风险决策、敏感点和权衡点的确认。	衡点等
3. 测试	7. 集体讨论,确定场景优先级		根据所有风险承担者的意见形成更大的场景集合 场景分类： a. 用例场景： 描述风险承担者对系统使用情况的期望。 b. 生长场景： 描述期望架构能在较短时间内允许的扩充与更改。 c. 探察场景： 描述系统生长的极端情况，即架构在某些更改的重压的情况。 注：最初的效用树是由架构设计师和关键开发人员创建的。在对场景进行集体讨论的过程和设置优先级的过程中，有很多风险承担者参与其中，与最初的效用树相比，两者之间的不匹配可以揭露架构设计师未曾注意到的方面，从而使得我们发现架构中的重大风险。	由所有风险承担者通过表决确定这些场景的优先级
	8. 分析架构方法		对第 6 步重复，使用的是在第 7 步中得到的高优先级场景，这些场景被认为是迄今为止所做分析的测试案例	发现更多的架构方法，有风险决策、无风险决策、敏感点、权衡点等
4. 形成报告	9. 结果的表述	评估小组	根据在 ATAM 评估期间得到的信息（方法、场景、针对质量属性的问题、效用树、有风险决策、无风险决策、敏感点、权衡点等），向与会的风险承担者报告评估结果。 最重要的是如下 ATAM 结果： a. 已经编写了文档的架构方法； b. 若干场景及其优先级； c. 基于质量属性的若干问题； d. 效用树； e. 所发现的有风险决策； f. 已编写文档的无风险决策； e. 所发现的敏感点和权衡点。	

以上步骤并不是固定的，有时必须对评估规划做某些动态的更改，以容许人员或架构信息的改变。

ATAM评估方法的的阶段：

第 0 阶段	建立评估小组， 建立评估组织和待评估组织间的合作关系	
第 1 阶	以架构为中心，重点获取架构信息并对其进行分析。	评估阶段，上面的 9 个

段		步骤 在这时完成
第 2 阶段	以风险承担者中心，重点为获取风险承担者的观点，并对第 1 阶段的结果进行验证。	
第 3 阶段	后续阶段，形成最终报告，对后续活动做出规划，评估组织在此阶段实现文档和经验的更新。	

评估小组各成员的角色及其职责

角色	职责	理想的人员素质
评估小组负责人	准备评估；与评估客户协调；保证满足客户的需要；签署评估合同； 组建评估小组；负责检查最终报告的生成和提交；	善于协调、安排，有管理技巧。善于与客户交流；能按时完成任务。
评估负责人	负责评估工作；促进场景的得出；管理场景的选择及设置优先级的过程；促进对照架构的场景评估；为现场评估提供帮助	能在众人面前表现自如。善于指点迷津。对架构问题有深刻的理解，富有架构评估的实践经验。能够从冗长的讨论中得出有价值的发现，或能够判断出何时讨论已无意义、应进行调整。
场景书记员	在得到场景的过程中负责将场景写到活动挂图或白板上，务必用以达成一致的措辞来描述，未得到准确措辞就继续讨论	写一手好字，能够在未搞清楚某个问题之前坚持要求继续讨论，能够快速理解所讨论的问题并提取出其要点
进展书记员	以电子形式记录评估的进展情况。捕获原始场景。捕获促成场景的每个问题。捕获与场景对应的架构解决方案。打印出要分发给各参与人员所采用场景的列表	打字速度快，质量高。工作条理性好。从而能够快速查找信息。对架构问题理解透彻。能够融会贯通地快速搞清技术问题。勇于打断正在进行的讨论以验证对某个问题的理解，从而保证所获取信息的正确性
计时员	帮助评估负责人保证评估工作按进度进行。在评估阶段帮助控制用在每个场景上的时间	敢于不顾情面地中断讨论，宣布时间已到。
过程观察员	记录评估工作的哪些地方有待改进或偏离了原计划。通常不发表意见，也可能偶尔在评估过程中向评估负责人提出基于过程的建议。在评估完成后，负责汇报评估过程，指出应该吸取哪些教训，以便在未来的评估中加以改进。还负责向整个评估小组汇报某次评估的实践情况	善于观察和发现问题，熟悉评估过程，曾参加过采用该架构评估方法进行评估
过程监督者	帮助评估负责人记住并执行评估方法的每个步骤	对评估方法的各个步骤非常熟悉。愿意并能够以不连续的方式向评估负责人提供指导
提问者	提出风险承担者或许未曾想到的关于架构的问题	对架构和风险承担者的需求具有敏锐的观察力。了解同类系统。勇于提出可能有争议的问题，并能不懈地寻求其答案。熟悉相关的质量属性。

商业目标分析结果表：

目标	内容列表	详细备注
主要商业目标	为跨学科的地球科学研究提供支持	

	大批量数据的并发摄入、处理和分配	
次要商业目标	为外部开发的科学算法 / 应用程序提供支持	
	科学数据再处理	
其他商业目标	采用自动化操作，以尽可能减少操作成本	

系统质量属性列表：

质量属性目标	标识号（数字表示效用树中的子序号）	针对质量属性的要求
可维护性	M1	更改某个子系统时不要求改动其他子系统
	M2	把子系统的部署要求分别降到最低
	M3	把回归测试时间从 5 天减少到 1 天
可靠性	R1	不至于因为数据输入输出而导致某个系统资源崩溃或等待较长时间，如超过 10 分钟
	R2	请求（输入出）中的某一部分数据错误不会妨碍其他部分的正常使用
	R6	因系统崩溃或备份造成的无法提供服务的时间不能超过 1 小时
可操作性	O10	系统应该能在 20 分钟内根据用户类型、数据类型、介质类型、目的地或用户对 1000 项定单重新设置优先级
	O14	应该能在不需要操作干涉的情况下，通过 V0 网关为 1000 个并发请求提供服务
可扩展性	S2	能同时支持 50 个场站
	S3	能够支持来自 100 个数据源的输入
性能	P1	在某种查询算法下，对 Landsat L-7 查询可使响应速度能提高 5 倍

第一阶段获得的带优先级的效用树：

第 1 层： 效用

第 2 层：质量属性	第 3 层：质量属性求精	第 4 层：质量属性场景	重要性	难度	累加和
可维护性	M1：对一个子系统的更改不要求改动其他子系统	M1.1：部署下一个科学数据服务器版本，实现对当前版本的更新。升级应该在 8 小时以内完成，并且不应影响其他子系统及查找、浏览或预定功能的使用。	30	30	60
	M2：独立地回退子系统的部署	M2.1：使科学数据服务器从 M1 回退	20	20	40
可操作性	O10：系统应该能在 20 分钟内根据用户类型、数据类型、介质类型、目的地或用户对 1000 项定单重新设置优先级	O10.1：积压任务管理-- 在系统连续 24 小时不能正常工作后，应在 30 分钟内为所积压的任务设置优先级，以保证各任务能够按优先级得以处理且正常的操作能够得以维持，不会在恢复正常操作状态后	30	20	50

		降低吞吐量			
	O14：应该能在不需要操作干涉的情况下，通过 V0 网关为 1000 个并发请求提供服务	O14.1：MODAPS 连续 24 小时不能正常工作，恢复并请求 2 天的数据；按优先级进行处理	20	20	40
		O14.2：接收 100 个并发查询请求，不拒绝高优先级的请求，在性能允许的情况下完成处理，不使系统过载	20	20	40
可靠性／可用性	Ra1：失败的或挂起时间超过 10 分钟的数据输入输出不占用系统资源	Ra1.1：L-7 按预定用 FTP 方式把数据推到某个 FTP 服务器已经崩溃的节点，在第一个请求未能成功处理的 10 分钟内系统被挂起，在请求被挂起期间内所有资源可用，不影响其他节点的分配。	30	10	40
可扩展性	Sc2：系统能够支持 50 个节点	Sc2.1：跨节点订单可记录 50 个节点，跨 5 节点订单历时 2 分钟。 Sc2.2：跨节点用户注册在 24 个小时内经历 50 个节点。	20 20	30 30	50 50
性能	P1：把 Landsat L-7 搜索的响应速度提高 5 倍	P1.1：正常情况下 Landsat L-7 搜索 100 次命中的时间不超过 30 秒	30	20	50
		注：这些难度等级只是粗粒度的划分，不必过于认真			

场景分析表：

1. 场景描述：

场景号：M1.1	场景：（M1.1）部署下一个科学数据服务器版本，实现对当前版本的更新。升级应该在 8 小时以内完成，并且不应影响其他子系统及查找、浏览或预定功能的使用。
质量属性	可维护性
环境	常规维护
刺激	部署下一个科学数据服务器版本，实现对当前版本的更新，并添加对经纬度的支持。
响应	升级应该在 8 小时以内完成，并且不应影响其他子系统及查找、浏览或预定功能的使用。

架构决策	有风险决策	敏感点	权衡点	无风险决策
AD1：接口的向后兼容性	R1			
AD2：客户占位程序在服务器中的静态连接	R2			
AD3：关键运行数据库的单个副本	R3	S1	T1,T2	
AD4：关于分布在整个系统中的数据类型的信息	R4,R5,R6	S2		
AD5：独立于子系统的名称			T3	
AD6：采用稳定、简单 API 的分布式对象				NR1
AD7：源文件间未受控制的依赖性	R7			

推理：定性的或量化的关于为什么这组架构决策能够满足此场景所表达的每个质量属性要求的基本原理。
架构图：一个或多个表示架构视图的图形，标注出支持上述推理的架构信息，必要的可加上解释性文字描述。

ARID评估方法---- Active Reviews for Intermediate Design

ATAM 和 SAAM 都是适用于对软件的完整架构进行评估的方法。但是，架构经常是要经历很长时间，阶跃式地逐渐完善，而不是一开始就以最终确定的完美形式出现的。

在架构设计过程中，对已经完成的部分逐步进行评审，及时发现某些部分的错误，不一致性或者是考虑不周的地方，而且在大多数项目开发中经常需要对系统的每个大组件或子系统进行这种评审。这一阶段需要的是一种简单易用的评估方法，应该重点关注适宜性，以一种风险承担者能够接受的方式展示设计方案，并能在缺少详细文档的情况下进行架构评估，这时就要用到 ARID 方法，active reviews for Intermediate Design。ARID 最适合于对尚不完善的架构进行评估，在这一阶段，设计人员就是想搞清楚从要求使用该设计架构的其他部分的角度来看，所采用的设计方案是否合适。或许设计中的这个框架的潜在采用者 / 重用者想要搞清楚该怎样使用这一框架。

传统设计评审和积极设计评审中所用的指令对比

传统设计评审问题	积极设计评审中所用的指令
每个程序是否都定义了例外情况	写出每个程序可能出现的例外情况。
每个程序是否都定义了正确的例外情况？	写出每个参数合法值的范围或集合，写出在哪些状态下调用程序是非法的。
是否定义了数据类型？	对每个数据类型，请写出： 1. 该数据类型直接量的表示； 2. 该类型变量的声明； 3. 该数据类型的最大值和最小值。
这些程序是否充分？	编写按该方案实现某已定义任务的一小段伪代码。
是否对某个应用程序的性质都做了足够详细的说明？	对每个应用程序，写出其最长执行时间，并列出它可能消耗的共享资源。

ATAM，SAAM，ARID的比较

ITEM	ATAM	SAAM	ARID
涉及的	不面向任何具体的质量属性，但	主要是可修改性和功能。	设计方法和适宜性。

质量属性	据其历史，它更侧重于可修改性，安全性，可靠性和性能。		
分析的对象	架构方法或样式；阐述过程、数据流、使用、物理或模块视图的架构文档。	架构文档，特别是阐述逻辑或模块视图的部分。	组件的接口规范。
适用阶段	在架构设计方法已经选定之后。	在架构已经将功能分配到各个模块中以后。	在架构设计期间。
采用的方法	利用效用树和对场景的集体讨论来搞清楚质量属性需求。通过对架构方法的分析确定出敏感点、权衡点和风险。	利用对场景的集体讨论搞清楚质量属性需求。通过来验证功能或对更改成本作出估计。	积极评审设计，对场景进行集体讨论。
资源需求	一般用 3 天的时间，另外还有预先的准备时间和之后的总结时间。参评人员有客户、架构设计师、风险承担者和 4 人评估小组。	一般用 2 天时间，另外还有之后的总结时间，参评人员有客户、架构设计师、风险承担者和 3 人评估小组。	一般用 2 天时间，另外还有预先的准备时间和之后的总结时间。参评人员有架构设计师、设计人员、风险承担者和 2 人评估小组。

所有的评估方法至少要用下列两种技巧：

- 提问技巧。 用问卷、检查列表或场景来调查某个架构满足其质量属性需求的方式。架构评估中所用的提问技巧通常涉及到要做某些“思维实验”，以预期系统的表现，因为此时系统还不存在。
- 度量技巧。 要用某个工具对软件产品进行度量。度量技巧包括要运行所评估架构对应系统的模拟程序，以搞清系统的某些情况。要选择恰当的单位。对复杂性、耦合度和内聚性的度量通常用来得出可修改性的结论。对数据流的度量（即度量沿通信通道的数据量的大小及其频率）则可用来对系统性能或性能瓶颈做出预测。

应注意的危险信号：

- 架构必须与当前的组织结构相对应。因为有相应的组织而添加 不必要的部分。
- 最顶层的架构组件超过了 25 个。过于复杂，架构设计师可能 难以进行明智的控制，负责实现架构的设计人员那当然也无法做到这一点。
- 设计方案的剩余部分受某一项需求的左右。架构是以满足极高的可用性需求为目标的。如果降低这一需求，整个架构就会显得过于复杂。过分强调某一需求可能会使其他需求得不到重视。
- 架构信赖操作系统中的可变部分。这样使架构受到操作系统升级的影响；这一明显的设计缺陷在实际中经常出现，其频度令人惊讶。
- 在可用标准组件的地方却使用了专用组件，使整个架构信赖于某个供应商。

- 组件的定义是根据硬件划分确定的。硬件会随着系统的演进而变化，硬件组件可能会合并成更通用的处理器，或分解成专用的设备。如果软件独立于硬件结构，就可使其免受硬件变化的影响。
- 有超出可靠性要求之外的冗余。这表明设计人员意见不一，导致不必要的复杂性，也会使系统维护的难度加大。
- 设计方案受例外情况的左右；重点强调的是可扩充性而不是共核部分。
- 开发组织不能确定出谁是系统架构设计师。
- 构架设计师或项目经理在确定该架构的风险承担者时感到很困难，这可能意味着他们根本就没有考虑关于风险承担者的问题。
- 开发人员在对某两个组件的组件的交互进行编程时有过多的选择余地。出现这种情况的原因可能是架构上提供了太大的选择自由，或者没有考虑这一问题。意味着架构还要进一步完善。
- 当要求架构设计师提供文档时，除了类图外，拿不出任何其他材料。
- 当要求架构设计师提供文档时，拿出的是大量的由某一工具自动生成的文档，但这些文档根本没人看过。
- 所提供的架构文档陈旧，显然已经过时。
- 当要求对架构做出表述时，设计人员或编程人员或者不能做出表述，或者所做的表述与架构设计师所讲的内容存在很大的差别。
- there must be more as the process progressed.

对项目开发威胁最大的管理问题：

- 没有清楚地确定出风险承担者
- 项目开发中没有相关领域专家的参与
- 项目资金没有真正落实
- 没有指定项目经理或项目负责人
- 没有制定出项目计划或规划
- 部署日期不实际
- 没有明确的衡量优劣的标准
- 没有选定软件的架构
- 虽然在每个层面上都有一位架构设计师，但没有人对总体架构负责
- 没有编写总体架构计划
- 没有独立的负责编写需求的小组存在
- 没有硬件和安装规则

- 没有合适而独立的性能研究计划
- 没有合适的质量保证组织
- 没有制定出系统测试计划
- more

影响项目开发成败的性能方面的问题：

- 最终用户没有确定出性能需求
- 未收集与性能相关的数据
- 没有确定性能开销
- 所期望的通信速率未能得到证实
- 未采用任何用以度量处理时间的机制或处理数量未得到证实
- 未采用任何评估手段来保证能够处理所要求的吞吐量
- 未采用任何评估手段来保证硬件上能够满足处理的需要
- 没有性能模型
- more

风险承担者列表：

风险承担者	定义	所关心的问题
系统的生产者		
软件架构设计师	负责系统的架构以及在相互竞争的质量需求间进行权衡的人	对其他风险承担者提出的质量需求所要进行的解释和调停
开发人员	设计或编程人员	架构描述的清晰和完整、 各部分的内聚性和受限耦合、 清楚的交互机制
维护人员	系统初次布署完成后对系统进行更改的人	可维护性、确定某个更改发生后必须对系统中哪些地方进行改动的能力
集成人员	负责组件集成或者组装的人员	与开发人员相同
测试人员	负责系统测试的开发人员	集成、一致的错误处理协议； 受限的组件耦合、组件的高内聚性、概念完整性
标准专家	负责搞清所开发软件必须满足的标准（现有的或未来的）细节的开发人员	对所关心问题的分离、可修改性、互操作性
性能工程师	分析系统的工作产品以确定系统是否满足其性能及吞吐量需求的人员	易理解性、概念完整性、性能、可靠性
安全专家	负责保证系统满足其安全性需求的人士	安全性
项目经理	负责为各小组配置资源、保证开发进度、保证	架构层次上结构清楚，便于组建小组；任

	不超出预算的人士，负责与客户打交道	务划分结构、进度标志和最后期限等
产品线经理或“拥有重用权的人”	设想该架构和相关资产怎样在该组织的其他开发中得以重复利用的人	可重用性、灵活性
系统的消费者		
客户	系统的购买者	开发进度、总体预算、系统的有用性、满足用户（或市场）需求的情况
最终用户	所实现系统的使用者	功能性、可用性、没有易用性？
应用开发者（对产品架构而言）	利用该架构及其他已有可重用组件，通过将其实例化而构建产品的人	架构的清晰性、完整性、简单交互机制、简单剪裁机制
任务专家、任务规划者	知道系统将会怎样使用以实现战略目标的客户代表，视野比最终用户更为开阔	功能性、可用性、灵活性
系统服务人员		
系统管理员	负责系统运行的人（如果区别于用户的话）	容易找到可能出现问题的地方
网络管理员	管理网络的人员	网络性能、可预测性
服务代表	为系统在该领域中的使用和维护提供支持的人	使用性、可服务性、可剪裁性
接触系统或与系统交互的人		
该领域或团体的代表	类似系统或所考察系统将要在其中运行的系统的构建者或拥有者	可互操作性
系统架构设计师	整个系统的架构设计师；负责在软硬件之间进行权衡并选择硬件环境的人	可移植性、灵活性、性能和效率
设备专家	熟悉该软件必须与之交互的硬件的人；能够预测硬件技术的未来发展趋势的人	可维护性、性能
以上就是所有的风险承担者的角色及其关注的问题。 **软件架构评估的质量在很大程度上取决于为评估召集起来的风险承担者的素质。**		