

# ATAM 评估文档

评估项目：《罗生蛋糕订购》

评估团队：《1230666 购票》

评估成员：

孙康 141250117

王嘉琛 141250137

## 目录

ATAM 评估文档 .....	1
ATAM 流程 .....	2
质量属性效应树 .....	4
架构方法和驱动对应列表 .....	4
架构方法分析 .....	5
敏感点列表 .....	18
权衡点列表 .....	19
有风险决策列表 .....	20
有风险决策分类列表 .....	20
非风险决策列表 .....	21
挑战和经验 .....	21
成员和分工 .....	22

# ATAM 流程

一、展示 ATAM 过程

二、展示业务驱动

- a) 重要功能需求
  - 订购与支付，订单信息查看，促销方案管理，取货柜信息同步，订单验证等核心功能的详细说明，场景描述，业务需求
- b) 经济约束
  - 系统的开发成本和硬件成本必须在三年后小于总收入并在以后每年不超过当年收入的 80%
- c) 系统特性
  - 单次请求计算量小
  - 请求频繁、随机、并发
  - 用户体验是业务发展的关键
  - 地域分布广泛
  - 不存在库存问题，先订购再制作
  - 读 > 写：读请求集中在对商品浏览过程，写请求集中在订购过程，写过程与读过程相关性较小，一致性要求较低
  - 结合硬件设备：一个人口集中区域一部冷藏箱，可凭借订单标识自助提取蛋糕
  - 取货方式发展变化：提取码、二维码、语音识别、人脸识别……
  - 蛋糕存储设备会不断进化……
  - 用户访问访问方式多样：网页、桌面端、移动端……
  - 促销方式经常变化：积分抵扣、折扣、满减、代金券……多种可能同时存在也可能先后发生
  - 业务变化、拓展可能性大：拓展团体购，联合企业作为季度奖励实施员工代金券……
  - 用户覆盖面大：不同年龄、健康状况、文化程度……
  - 采用第三方支付平台
  - 不涉及工作人员管理

- d) 架构驱动
  - 设计组向评审组就主要架构驱动做出解释说明

三、展示架构

- a) 技术约束
  - i. 使用 Linux 服务器
  - ii. 系统需要连接嵌入式硬件网络
- b) 交互系统
  - i. 第三方支付平台
- c) 架构方案

ASR	Tactics	
可用性	故障检测	监视、心跳、时间戳

	故障恢复	主动冗余、回滚、重试、重启
	故障预防	移除、事务
性能	节流	提高资源利用效率
	开源	增加资源、多份数据、多份计算单元、队列、资源调度
可拓展性	策略模式、工厂模式、分布式	
互操作性	定位	发现服务
	管理接口	安排、追踪服务
安全性	检测攻击	检测消息延迟
	抵御攻击	验证用户、限制访问、限制接口暴露、数据加密、实体分离

#### 四、识别架构方法

Catalog	Architecture Approaches
关注可用性	CDN
	埋点数据
	数据备份
	读写分离
	动态路由
	心跳
关注可用性和可拓展性	拆分服务
	分布式部署
关注可用性和安全性	监控
	日志
关注安全性	身份验证
	ssh密钥加密传输
	限制接口暴露
	检测异常熔断
关注互操作性	协调服务调用

# 质量属性效应树

## 质量属性效用树



# 架构方法和驱动对应列表

#	Architecture Approaches	Architecture Drivers
1	拆分服务	可用性场景 1 可用性场景 3 可拓展性场景 4
2	分布式部署	可用性场景 1 可拓展性场景 4
3	CDN	可用性场景 1
4	埋点数据	可用性场景 1

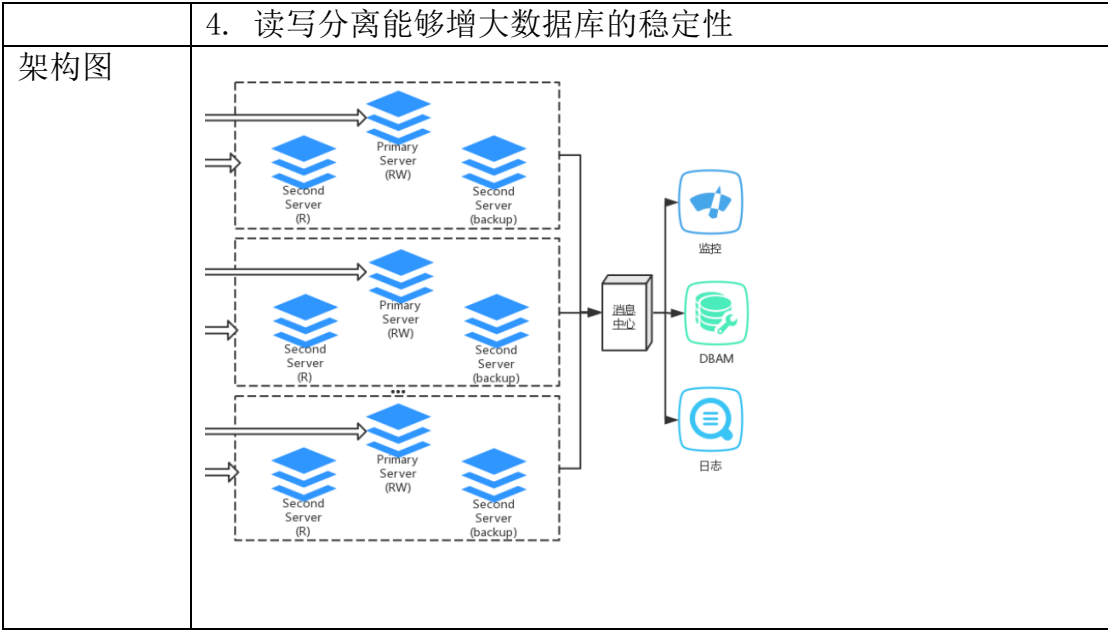
5	数据备份	可用性场景 2
6	监控	可用性场景 2 可用性场景 3 安全性场景 12
7	日志	可用性场景 2 安全性场景 12
8	读写分离	可用性场景 2
9	动态路由	可用性场景 3
10	心跳	可用性场景 3
11	身份验证	安全性场景 10 网络约束
12	ssh 秘钥加密传输	安全性场景 11
13	限制接口暴露	安全性场景 11
14	检测异常熔断	安全性场景 12
15	协调服务调用	互操作性场景 13

## 架构方法分析

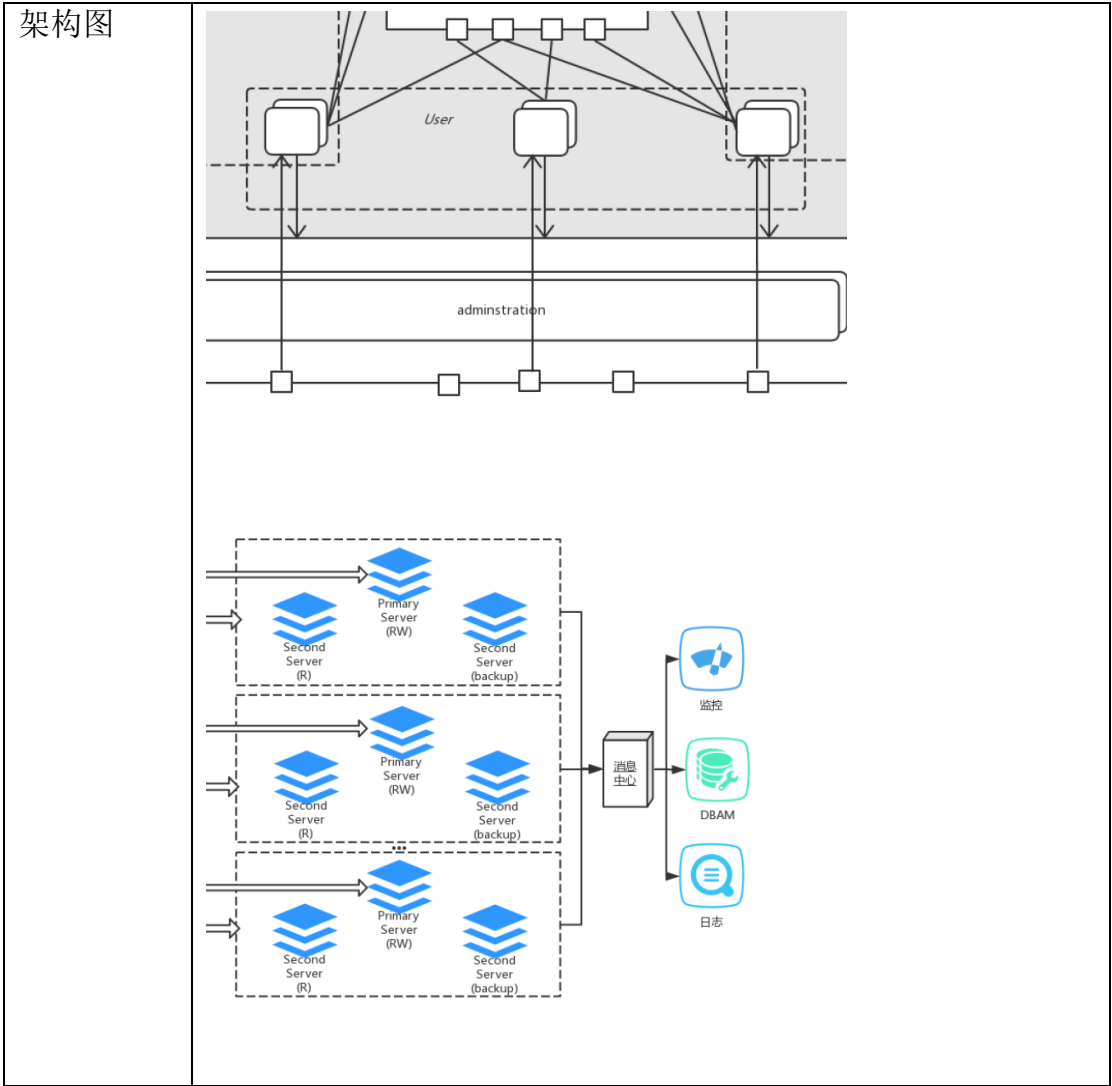
场景：A1	场景：服务器故障或者需要对系统进行人为干预。			
属性	可用性			
环境	正常模式或者安全模式			
刺激	服务器故障或者对系统进行消除漏洞的维护和升级			
响应	在出现故障前检测出错误，分散漏洞修复和升级，系统恢复 1. 系统停机维护的时间间隔不超过 3 分钟 2. 系统始终有 60%以上的资源可用 3. 系统不停止服务就可以从错误中恢复的比例达到 90%			
架构决策	敏感点	权衡点	有风险决策	无风险决策
拆分服务	S1	T1		
分布式部署	S2	T2		
主动冗余	S3	T3	R1	
负载均衡	S4		R2	
CDN	S5			
埋点数据	S6	T4		
分析	1. 拆分服务可能会使得系统全部失效的可能性变小，但是可能增大了系统部分失效的可能性。服务过细，服务之间调用频繁，通信压力增大，同时降低了性能。 2. 分布式部署可能会使得系统全部失效的可能性变小，但是可能增大了系统部分失效的可能性。同时会降低可维护性 3. 主动冗余一定会降低系统的整体性能（Risk）。同时也会增加被安全攻击的范围，降低了安全性			

	<p>4. 负载均衡主要由托管层逻辑控制，控制设计集中，职能复杂，耦合严重。并且存在设计缺陷，和 CDN 以及分布式设计职能不明确，概念重叠</p> <p>5. CDN 可以根据 ip 位置将服务发送到附近的服务器群，里面有缓存好的图片、视频，但是比如登录这些交互功能还是要真正从服务器实现。</p> <p>6. 埋点数据可以有效即时发现系统中出现的问题并报警，然后报警，但是耗费系统资源较多</p>
架构图	

场景：A2	场景：数据库故障			
属性	可用性			
环境	系统数据库故障，主数据库数据丢失			
刺激	系统数据库故障，主数据库数据丢失			
响应	系统数据库故障，主数据库数据丢失后 1. 能在 5 分钟之内连接到备份数据库 2. 数据库备份数据能在 1 小时以内同步到主数据库中 3. 多个主数据库同时发生数据丢失的概率不超过 1%			
架构决策	敏感点	权衡点	有风险决策	无风险决策
数据备份	S7			
监控	S8	T5		
日志	S9			
读写分离	S10			
分析	1. 数据备份能有效防止数据丢失的情况，有效保证了可用性 2. 监控能及时发现数据的丢失情况，进行报警，有效但是耗费太多资源 3. 日志能够记录历史操作，有助于记录系统的状态变化。			

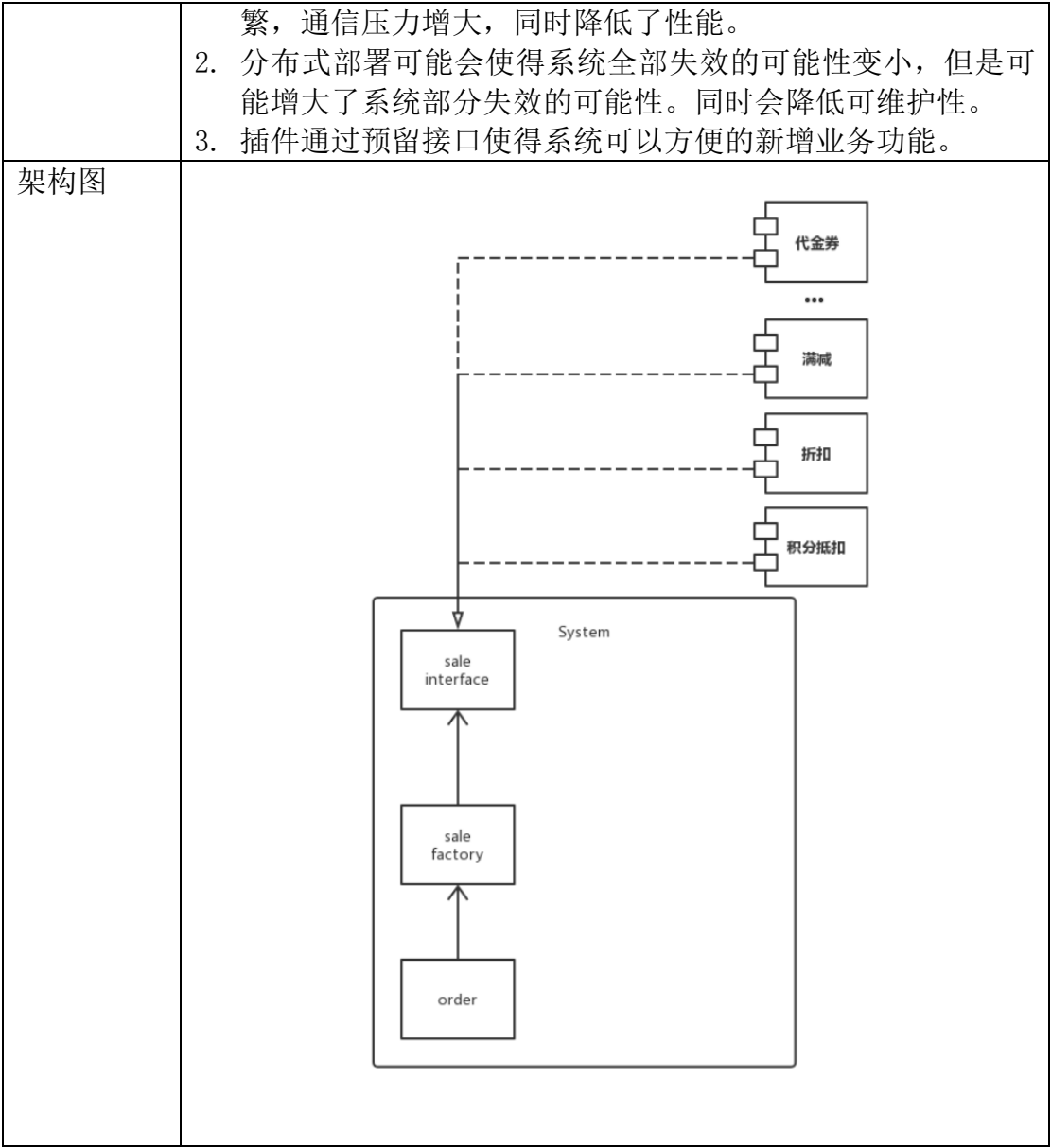


场景：A3	场景：无法连通取货柜			
属性	可用性			
环境	正常模式或者安全模式			
刺激	与取货柜网络通信不畅			
响应	尝试重连和更改路由重连，系统重新投入使用并且可以正常运行 1. 失联设备能在 30 分钟内连接到业务系统 2. 在故障发生前预期成功的概率达到 50%			
架构决策	敏感点	权衡点	有风险决策	无风险决策
动态路由	S11			
拆分服务	S1	T1		
监控	S8	T5		
心跳	S12			
分析	1. 动态路由因为频繁交换路由表，所以会增加系统的复杂度和实现难度 2. 拆分服务可能会使得系统全部失效的可能性变小，但是可能增大了系统部分失效的可能性。服务过细，服务之间调用频繁，通信压力增大，同时降低了性能。 3. 监控可以有效及时发现系统出现的问题，但是也耗费了大量的系统资源，降低了系统的性能。 4. 心跳可以有效的发现系统失效。 5. 监控与心跳职责重复，不需要两个都使用。			

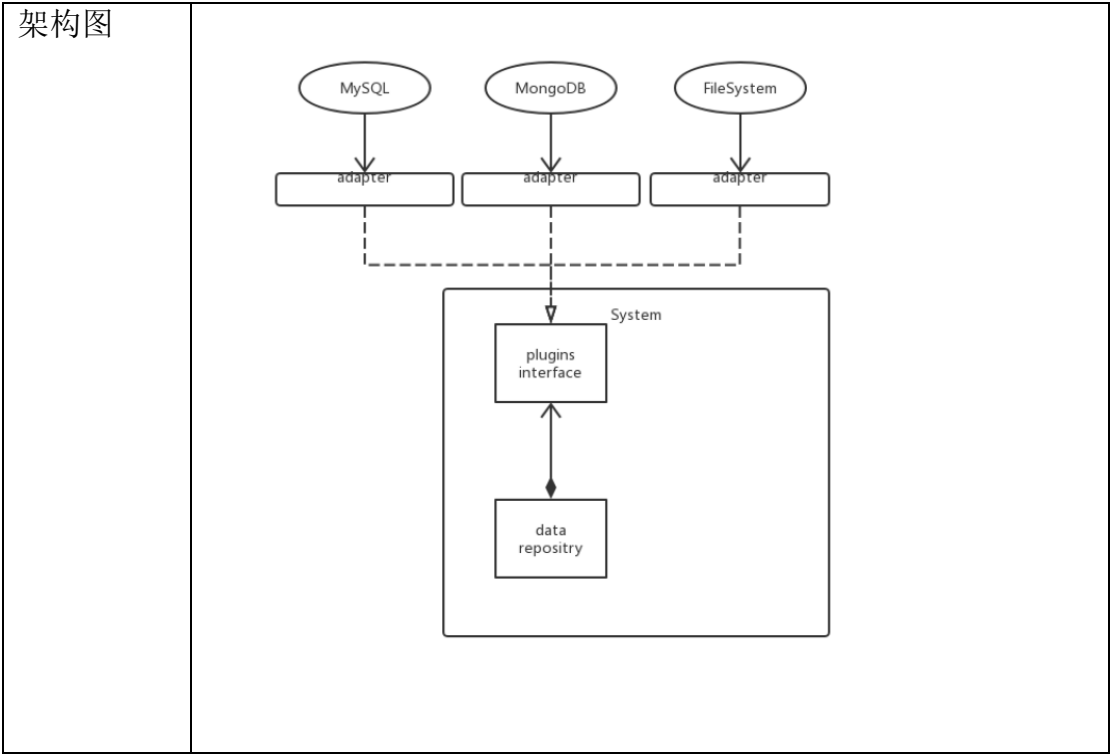


场景：A4	场景：新增业务功能			
属性	可拓展性			
环境	系统运行时			
刺激	系统需要新增业务功能			
响应	在 UI 层和业务逻辑层，可在不影响系统正常运行的情况下，增加与新的业务相关的用户界面和业务功能 1. 在运行成功时给系统添加功能成功次数与失败次数的比例 2. 新功能添加成功时对原来的代码所做的改动 3. 新功能对系统稳定性的影响			
架构决策	敏感点	权衡点	有风险决策	无风险决策
拆分服务	S1	T1		
分 布 式 部 署	S2	T2		
插件	S13	T6	R3	
分析	1. 拆分服务可能会使得系统全部失效的可能性变小，但是可能增大了系统部分失效的可能性。服务过细，服务之间调用频			

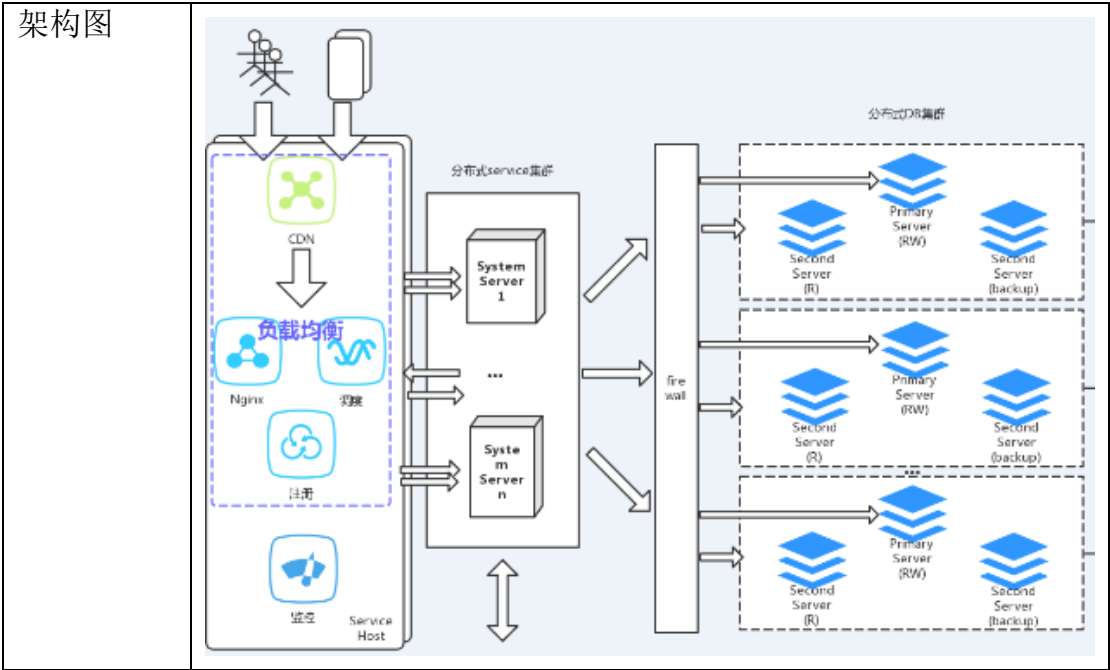




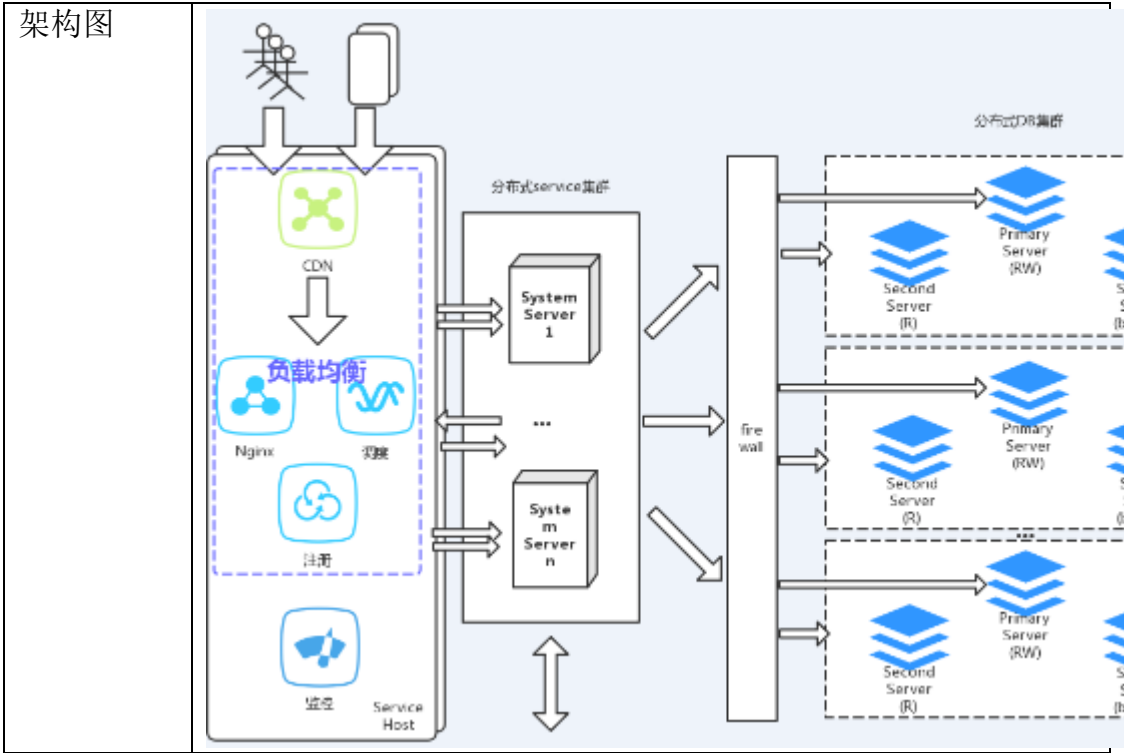
场景：A5	场景：新增或者改变数据库类型			
属性	可拓展性			
环境	系统运行时			
刺激	新增或改变数据库类型			
响应	在数据访问层，动态添加对新的数据库类型的支持 1. 数据库更改后系统正常运行的概率达到 99% 2. 更改数据库涉及到的代码改动不到 1%			
架构决策	敏感点	权衡点	有风险决策	无风险决策
插件	S13	T6	R3	
分析	1. 预留接口，可以方便进行拓展，只要调用原来的接口就可以与服务器连接。预留的接口可能会被作为攻击点。			



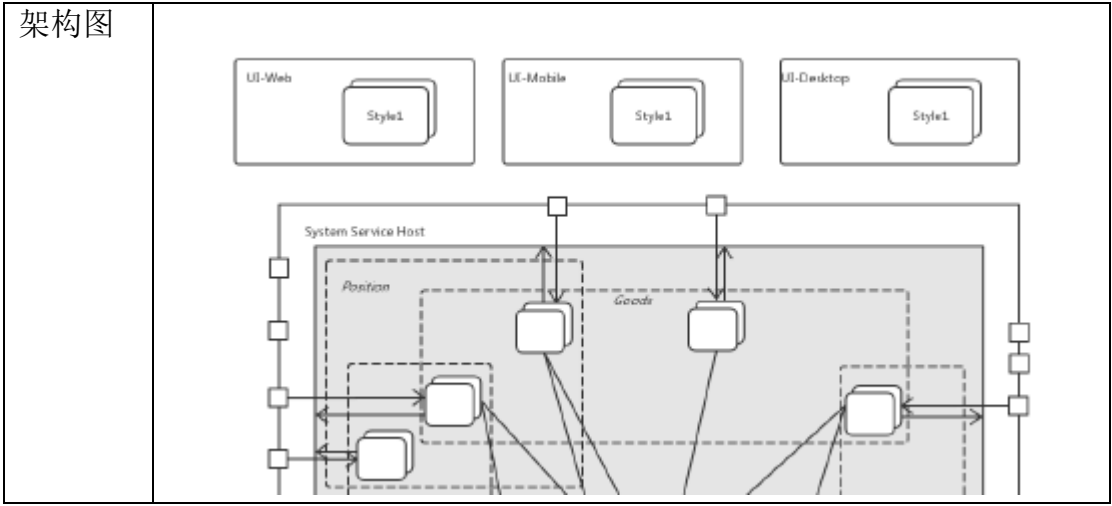
场景：A6	场景：用户访问量扩大			
属性	性能			
环境	系统部署、系统设计、系统运行			
刺激	业务扩展，用户大量增加，用户对系统的访问请求激增，系统无法承担大量访问需求			
响应	采取负载均衡措施，分发请求，减少单点的压力，避免单点故障造成瘫痪；关系型数据库进行分库/分表/分区等操作；数据库服务器主主互备，做到了访问量的压力分流，同时也解决了“单点故障”问题；在 Web 服务器和数据库之间建立缓存机制			
架构决策	敏感点	权衡点	有风险决策	无风险决策
分布式集群	S15	T2		N2
CDN	S5			
负载均衡	S4		R2	
缓存	S14			
分析	<div>1. 使用分布式集群的方式能够有效的提升系统性能但是同时造成系统成本和复杂性的增加，故而影响系统可维护性。</div> <div>2. 另外提升性能的方案是使用 CDN 和托管层的负载均衡，其中的负载均衡主要由托管层逻辑控制，控制设计集中，职能复杂，耦合严重。并且存在设计缺陷，和 CDN 以及分布式设计职能不明确，概念重叠。</div>			



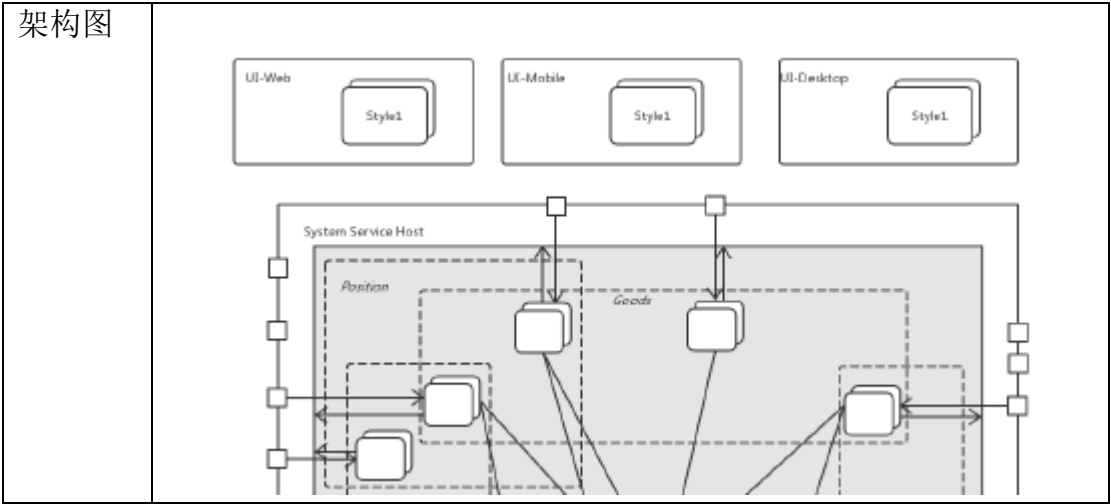
场景：A7	场景： 物联网设备和业务系统访问需求扩大			
属性	性能			
环境	系统部署、系统设计、系统运行			
刺激	用户增加、业务增加，物联网设备与系统相互发送的请求激增，系统无法承担大量访问需求			
响应	采取负载均衡措施，分发请求，减少单点的压力，避免单点故障造成瘫痪；关系型数据库进行分库/分表/分区等操作；数据库服务器主主互备，访问量压力分流			
架构决策	敏感点	权衡点	有风险决策	无风险决策
分布式集群	S15	T2		N1
CDN	S5			
负载均衡	S4		R2	
分析	1. 使用分布式集群的方式能够有效的提升系统性能但是同时造成系统成本和复杂性的增加，故而影响系统可维护性。 2. 另外提升性能的方案是使用 CDN 和托管层的负载均衡，其中的负载均衡主要由托管层逻辑控制，控制设计集中，职能复杂，耦合严重。并且存在设计缺陷，和 CDN 以及分布式设计职能不明确，概念重叠。			



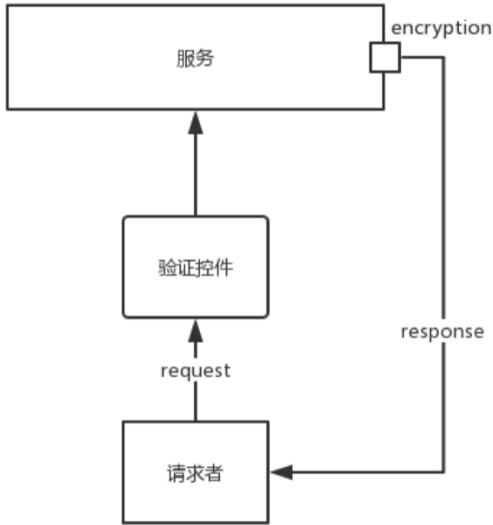
场景：A8	场景：前后端交流影响开发进度			
属性	可维护性			
环境	系统开发以及维护时			
刺激	前后端交流成本会影响开发进度			
响应	系统分层，通过接口通信，对接口只添加不修改，保持稳定；前后端分离，分开部署，以此减少前端后端人员的沟通成本；			
架构决策	敏感点	权衡点	有风险决策	无风险决策
分层	S16	T7		N2
低耦合 高内聚	S17			
接口稳定	S18			N3
前后端分离	S19			
分析				



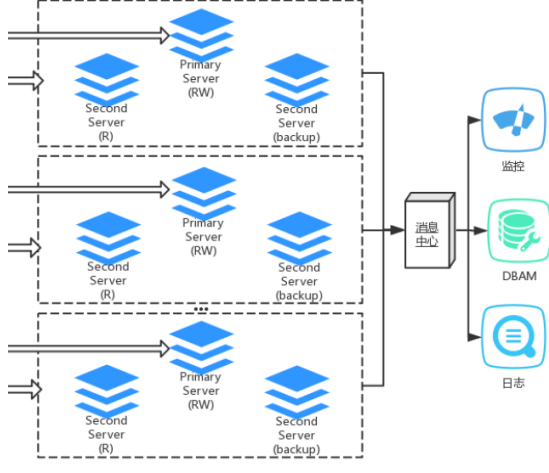
场景：A9	场景： 非开发人员参与系统维护			
属性	可维护性			
环境	系统开发以及维护时			
刺激	源码理解成本会影响开发进度			
响应	系统分层，通过接口通信，对接口只添加不修改，保持稳定；提高代码的内聚性，松散耦合，将变化控制在尽量小的区域			
架构决策	敏感点	权衡点	有风险决策	无风险决策
分层	S16	T7		N2
低耦合 高内聚	S17			
接口稳定	S18			N3
拆分服务	S1	T1	R4	
分析	<div>1. 分层的应用会导致控制逻辑的冗杂，产生性能瓶颈，但是对于系统的维护成本确实有正面影响</div> <div>2. 低耦合高内聚的设计有助于限制维护系统的目标在单个模块内，减少维护设计代码量低。</div> <div>3. 接口稳定，强调接口的设计，对于普遍功能具有积极影响，但对于规定文件格式的接口承载了更多了修改压力。所以对于本系统有利有弊，需要权衡。</div> <div>4. 拆分服务的决策使得系统更加易于理解,但是服务粒度变细必然导致服务调用次数的增加和服务件复杂的逻辑,可能产生性能缺陷。</div>			



场景：A10	场景：支付流程安全受到攻击			
属性	安全性			
环境	系统运行、用户支付			
刺激	支付过程泄露或篡改支付信息			
响应	支付过程采用安全证书，验证用户身份			
架构决策	敏感点	权衡点	有风险决策	无风险决策
加密	S20	T8	R5	
身份验证	S21			
分析	1. 数据加密必然有利于系统安全性，但是如果选择相对而言较为复杂的加密技术，会导致计算资源和存储资源的大量开销，故而不利于性能。 2. 身份验证的采用能够很好的解决安全问题，并且不会占用太多的系统资源。			
架构图	<pre> graph BT     A[支付请求] -- encrypt --&gt; B[用户验证]     B --&gt; C[支付控件]     C --&gt; D[第三方支付]           </pre> <p>The flowchart depicts the sequence of operations in a payment process. It starts with a box labeled '支付请求' (Payment Request) at the bottom. An arrow labeled 'encrypt' points from this box to a box labeled '用户验证' (User Verification). From '用户验证', an arrow points up to a box labeled '支付控件' (Payment Control). Finally, an arrow points from '支付控件' to the top box labeled '第三方支付' (Third-party Payment).</p>			

场景：A11	场景：安全性场景 11			
属性	安全性			
环境	系统运行、提取码发送			
刺激	系统对个人和物联网设备发送提取码遭拦截或篡改			
响应	发送提取码采用加密方式，提取码存在服务器端，获取提取码之前需要登录验证身份			
架构决策	敏感点	权衡点	有风险决策	无风险决策
ssh 秘钥加密传输	S22			
限制接口暴露	S23	T9		
分析	<p>1. 采用 ssh 秘钥加密传输的方式可以有效提高系统的安全性。在传输加密数据以及处理加密数据时会额外占用资源，影响系统的性能，但是性能这一架构需求对系统而言不是至关重要的。</p> <p>2. 本系统对安全性要求高，限制接口暴露能在软件设计上减少被攻击的可能性，同时也是软件设计的原则之一。采用限制接口暴露能够提高系统的安全性，虽然限制接口暴露会对系统的互操作性产生一定影响，但是互操作性对系统而言不是至关重要的。</p>			
架构图	 <pre> sequenceDiagram     participant Requester     participant Validator     participant Service     Requester-&gt;&gt;Validator: request     Validator-&gt;&gt;Service:      Service-&gt;&gt;Requester: response     Note over Service: encryption </pre>			

场景：A12	场景：安全性场景 12			
属性	安全性			
环境	系统运行			
刺激	物联网设备被控制对系统进行流量攻击			
响应	系统流量监控、日志记录，及早发现异常，流量异常激增时采取熔断机制，保护服务器的正常运行			

架构决策	敏感点	权衡点	有风险决策	无风险决策
监控	S8	T5		
日志	S9			
检测异常熔断	S24	T10		
分析	<ol style="list-style-type: none"> <li>1. 监控可以提高系统的安全性。通过监控异常流量，系统可以对可能的恶意访问做出反应，避免系统遭受攻击。虽然监控会消耗一定的系统资源，但是系统对性能的要求并不高，所以采用监控。</li> <li>2. 日志有助于系统定位安全隐患，虽然日志增大了系统的读写负担，影响性能，但是对本系统而言采用日志是无风险决策。</li> <li>3. 检测异常熔断可以防止对系统的恶意攻击，虽然检测需要占用额外的资源影响性能，但是系统对性能要求不高，可以采用检测异常熔断。</li> </ol>			
架构图				

场景：A13	场景：互操作性场景 13			
属性	互操作性			
环境	系统运行时			
刺激	用户支付			
响应	用户可以在多种支付方式中选择任意一种方式支付，系统与第三方支付平台正确通信，互留记录			
架构决策	敏感点	权衡点	有风险决策	无风险决策
协调服务调用	S25	T11		N4
接口管理	S26	T12	R6	
分析	<ol style="list-style-type: none"> <li>1. 协调服务调用通过控制机制来协调各个接口之间的协作，共同完成任务。协调服务调用是提高互操作性的一种方式，同时也有利于系统增加新的功能接口。虽然会在增加系统的复杂度，采用协调服务调用是无风险的决策。</li> <li>2. 接口管理，理解为除了 Orchestrate 之外的另一种接口管理</li> </ol>			



	方式 Tailor interface。它通过对现有的一个接口进行增删功能提高互操作性。但是这种方式对系统的可拓展性有影响，而可拓展性对系统而言至关重要，故不采用该方法。
架构图	<pre> graph BT     A[支付请求] -- encrypt --&gt; B[用户验证]     B --&gt; C[支付控件]     C --&gt; D[第三方支付] </pre>

场景：A14	场景：互操作性场景 14			
属性	互操作性			
环境	系统运行时			
刺激	业务系统与物联网设备通信			
响应	业务系统与物联网设备正常通信			
架构决策	敏感点	权衡点	有风险决策	无风险决策
接口管理	S26	T12	R6	
分析	1. 接口管理不利于系统的可拓展性，是有风险的决策。该场景未提供无风险的决策，列为该系统中的风险。			
架构图				

场景：A15	场景：用户进行下单支付			
属性	易用性			
环境	在系统运行时或配置时			
刺激	想要学习系统操作，有效使用系统所提供的功能进行下单、支付、取货			
响应	系统提供以下一个或多个响应来支持“学习系统特性”和“有效使用系统”：已经输入的数据和/或命令的重用；支持在界面中有效导航；具有一直操作的不同视图。；撤销；取消；从系统故障中恢复；识别并纠正工作人员错误；验证系统资源；显示系统状态			
架构决策	敏感点	权衡点	有风险决策	无风险决策
提供多种	S27			

方式完成一个操作				
分析	1. 提供多种方式完成一个操作，意味提供给客户的选择权利增大，系统更容易上手使用，十分便捷。			

场景：A16	场景：用户自助提取货物			
属性	易用性			
环境	系统运行时			
刺激	想要使用系统的验证方式自助取货			
响应	提供多种取货方式，自然人机交互。			
架构决策	敏感点	权衡点	有风险决策	无风险决策
暂停	S28			
撤销	S29			
分析	1. 暂停和撤销对于系统易用性有积极的影响。			

# 敏感点列表

编号	描述	有关质量属性
S1	拆分服务对系统的可用性产生影响。	可用性
S2	分布式部署对系统的可用性产生影响。	
S3	主动冗余对系统的可用性产生影响。	
S4	负载均衡对系统的可用性产生影响。	
S5	CDN 对系统的可用性产生影响。	
S6	埋点数据对系统的可用性产生影响。	
S7	数据备份对系统的可用性产生影响。	
S8	监控对系统的可用性产生影响。	
S9	日志对系统的可用性产生影响。	
S10	读写分离对系统的可用性产生影响。	
S11	动态路由对系统的可用性产生影响。	
S12	心跳策略对系统的可用性产生影响。	
S1	拆分服务对系统的可拓展性产生影响。	可拓展性
S2	分布式部署对系统的可拓展性产生影响。	
S13	使用插件策略对系统的可拓展性产生影响。	
S15	使用缓存策略对系统的性能产生影响。	性能
S15	使用分布式集群对系统的性能产生影响。	

S5	使用 CDN 策略对系统的性能产生影响。	
S4	使用负载均衡对系统的性能产生影响。	
S16	使用分层对系统的可维护性产生影响。	可维护性
S17	使用低耦合高内聚策略对系统的可维护性产生影响。	
S18	使用接口稳定对系统的可维护性产生影响。	
S19	使前后端分离策略对系统的可维护性产生影响。	
S20	使用加密策略对系统的安全性产生影响。	安全性
S21	使用身份验证对系统的安全性产生影响。	
S22	使用 ssh 密钥加密传输对系统的安全性产生影响。	
S23	限制接口暴露对系统的安全性产生影响。	
S8	采用监控对系统的安全性产生影响。	
S9	采用日志对系统的安全性产生影响。	
S24	采用检测异常熔断策略对系统的安全性产生影响。	
S25	采用协调服务调用对系统的互操作性产生影响。	互操作性
S26	采用接口管理对系统的互操作性产生影响。	
S27	提供多种方式完成一个操作对系统的易用性产生影响。	易用性
S28	采取暂停策略对系统的易用性产生影响。	
S29	采取撤销策略对系统的易用性产生影响	

## 权衡点列表

编号	描述
T1	拆分服务会提高系统的可用性、可拓展性、可维护性，但是会影响性能。
T2	分布式部署（集群）对提高系统的可用性、可拓展性、性能，但是会影响可维护性。

T3	主动冗余会提高系统的可用性,但是会影响系统的性能和安全性。
T4	埋点数据会提高系统的可用性, 但是会影响系统的性能。
T5	监控会提高系统的可用性、安全性, 但是会影响系统的性能。
T6	使用插件策略会提高系统的可拓展性,但是会影响系统的安全性。
T7	使用分层策略会提高系统的可维护性, 但是会影响系统的性能。
T8	使用加密粗略提高了系统的安全性, 但是会降低系统的性能。
T9	限制接口暴露提高了系统的安全性, 但是降低了系统的互操作性
T10	采用检测异常熔断策略提高了系统的安全性, 但是降低了系统的性能
T11	采用协调服务调用提高了系统的互操作性, 但是降低了系统的性能, 提高了复杂度。
T12	采用接口管理提高了系统的可操作性, 但是降低了系统的可拓展性。

## 有风险决策列表

编号	描述
R1	主动冗余会大大降低系统的性能,同时也会增加被安全攻击的范围,降低系统安全性。
R2	托管层控制集中, 职能十分复杂, 和其他组件的耦合程度极高
R3	插件这一策略中, 预留的接口可能会被作为攻击点导致系统崩溃
R4	拆分服务导致服务粒度过细, 服务之间的调用频繁, 通信压力大
R5	加密解密功能需要占用计算处理资源, 完成大量的计算任务将会影响性能。同时, 加密技术的错误选择直接影响系统运行安全
R6	接口管理会影响系统的可拓展性, 而可拓展性对系统而言至关重要, 所以接口管理是有风险的决策

## 有风险决策分类列表

编号	描述	风险分类
R1	主动冗余会大大降低系统的性能,同时也会增加被安全攻击的范围,降低系统安全性。	系统安全隐患
R3	插件这一策略中, 预留的接口可能会被作为攻击点导致系统崩溃	
R5	加密解密功能需要占用计算处理资源, 完成大量的计算任务将会影响性能。同时, 加密技术的错误选	

	择直接影响系统运行安全	
R1	主动冗余会大大降低系统的性能，同时也会增加被安全攻击的范围，降低系统安全性。	系统性能隐患
R4	拆分服务导致服务粒度过细，服务之间的调用频繁，通信压力大	
R5	加密解密功能需要占用计算处理资源，完成大量的计算任务将会影响性能。同时，加密技术的错误选择直接影响系统运行安全	
R6	接口管理会影响系统的可拓展性，而可拓展性对系统而言至关重要，所以接口管理是有风险的决策	系统开发维护难度大
R2	托管层控制集中，职能十分复杂，和其他组件的耦合程度极高	

## 非风险决策列表

编号	描述
N1	分布式部署中部署的工作难度较大，但是该技术相对成熟，不会成为开发流程中的瓶颈
N2	分层架构会产生复杂的控制流，但是本系统层数少并且职能明晰，故不会对性能有太大影响
N3	接口稳定要求标准格式，接口只增不改，但是本系统的数据标准改动可能不大，同时对于接口的改动可能也比较低
N4	协调服务调用虽然会影响系统的可维护性，但是系统对可维护性的要求并不高

## 挑战和经验

在这次 ATAM 评估流程中，评估小组和设计团队以及其他涉众进行了完整的详尽的评估工作。我们遇到的第一个麻烦是，在理解设计团队架构方案的过程中发现沟通理解方案会耗费很多时间，并且需要反复多次的交流。我们面对这样的困难，采用如下两个方案：

1. 沟通成果尽早文档化，根据标准统一的文档来进行沟通，使得沟通的成果明确化，使得评估效率大大提高。
2. 评估团队及时发问，在发现问题及时记录，准备好问题列表进行沟通交流

我们在这次实践中最大的收获是对 ATAM 评估流程有了更加深刻的了解。通过课程我们对 ATAM 有了概念性的了解，但是对于具体过程的实施和潜在可能

出现的问题并没有实践性的了解。在这次评估流程中，评估组成员体验了完整的 ATAM 流程，也在遇到问题解决问题的过程中，对 ATAM 有了深刻的理解，哪些环节容易出现意见分歧，哪些环节容易遗漏过程产物，哪些环节对于最后产出有着至关重要的影响，等等。

## 成员和分工

评审组成员分工：

姓名	学号	工作
孙康	141250117	评估小组成员，ATAM 全过程参与，评估产物生成以及文档化，进行 presentation 展示
王嘉琛	141250137	评估小组成员，ATAM 全过程参与，评估产物生成以及文档化
于凡	141250175	设计团队成员，主要架构展示人员
章琦	141250199	设计团队成员，架构展示人员之一
周聪	141250204	设计团队成员，架构展示人员之一
周赛	141250207	设计团队成员，架构展示人员之一
王卉	141250135	设计团队成员，架构展示人员之一
谭琼	141250122	设计团队成员，架构展示人员之一

本小组项目展示组成员分工：

姓名	学号	工作
张文玘	141250192	参与设计报告 PPT 编写，主要负责图表的完善
周小帆	141250209	参与设计报告 PPT 编写，主要负责 broker 架构的分析汇总
吴嘉荣	141250148	参与设计报告 PPT 编写，主要负责 SOA 架构的分析汇总
余旻晨	141250177	协助两位被外派的测评同学进行评估报告的编写
王梦麟	141250140	整理设计报告 PPT，进行本系统的设计报告展示