目录

第一章 3
1、互联网时代的软件开发有哪些挑战？ 3
2、从五个层面来阐述什么是 DevOps? 3
第二章 4
1. 云计算的大行其道对传统运维人员提出了新的要求， 传统运维人员应该做哪些改变才能从容应对？ 5
2. 云时代的运维与业界推崇的 devOps 有何联系？ 在互联网严峻的安全形势下，DevOps 过程该如何考虑信息安全的问题呢？ 5
3. 请描述 CMMI 模型的要点 5
4. 请描述 ITSS 模型的要点 6
5. 请描述 ITIL 模型的要点 7
6. 请描述 ISO20000 模型的要点 9
第三章 11
1.什么是一个好的软件架构 11
3.常见的软件架构的视图有哪几种？描述每种视图的作用和涉众 11
4.相比传统软件架构，互联网软件系统有什么特点 12
5.阐述单体架构的优缺点 12
6.阐述事件驱动架构中 Mediator 模式和 Broker 模式的区别 13
7.从运维角度，阐述微服务架构的优点 13
8，9，10-以 4+1 架构模型描述和演进教学支持系统 13
第四章 16
1、 在个体和小组两个层次上，有哪些典型的软件过程和方法？试论述每种方法的特点。 16
2、 软件过程改进的参考模型有哪些？ 16
3、 过程改进的元模型有哪些？ 18
4、 试着比较 Scrum 方法和 TSP 方法，两者有什么相似和相异之处？ 18
5、 PSP 估算方法是什么？这种方法有何特点？ 19
6、 PSP 有哪些原则？ 19
7、 PSP 的质量策略是什么？对实践有哪些启发？ 19
8、 CMM/CMMI 的五个成熟度级别的特征是什么？ 20
9、 试比较 CMMI 和 SPICE 两个模型的异同。 20
10、 DevOps 模型对现有的软件开发过程和方法可能带来哪些影响？ 20
第五章 21
1. 丰田生产方式（精益生产）的两大支柱是什么？并简述其作用意义 23
2. 《精益思想》一书中定义了精益的五大原则，它们分别是什么？ 24
3. 精益产品开发的目标是什么： 24
4. 支持精益产品开发目标的两大原则和支柱是什么 24
5. 精益产品开发相关的管理实践分为三大类，它们分别是哪三大类？ 24
6. 在丰田生产方式中，“看板”的原始含义是什么？ 24
7. 产品开发中的看板方法由哪五个核心实践构成？ 25
8. 看板方法中的可视化价值流动是最基础的实践，请简要描述要可视化哪三个方面 26
9. 显式化流程规则是看板方法的第二个核心实践，它是团队协作的基础，团队还应该

把它看做什么？ 26
10. 限制在制品数目是看板方法最为核心的实践，它除了加速价值的流动，还起到什么关键作用？ 26
11. 管理工作的流动这一实践，包括管理价值的输入、中间过程和输出，在看板方法中它分别对应什么活动？ 26
12. 反馈的目的是为了持续改善，看板方法的反馈可以分为两类，一类是关于流动是否顺畅的，另一类是什么？ 26
第六章 27
1. 什么是微服务架构？ 27
2. 比较微服务与单体应用架构、 SOA 架构的异同 27
3. 什么是 12 范式？ 28
4. 微服务架构的特征有哪些？ 28
5. 微服务的核心模式有哪些？ 28
第七章 29
1.Linux namespace 有几种，分别是什么？ 29
2.Dockerfile 的 CMD 与 Entrypoint 有什么异同？ 29
3.编写一个运行 Java 应用的 Dockfile 30
4.Docker 网络模式有几种，分别是什么 30
5.Doker Storage Driver 有几种？分别是什么？使用场景有哪些 31
6.Doker 编排引擎有几种，分别是什么？ 31
第八章 32
8.1 DevOps 流程涵盖哪些节点？ 32
8.2 容器技术对于 DevOps 实践有哪些帮助？ 32
8.3 什么是持续交付流水线？ 32
8.4 什么是持续集成？ 32
8.5 什么是持续部署？ 33
8.6 一个典型的 Java 应用的持续集成/持续部署包含哪些步骤？ 33
第九章 33
9.1 常见的 DevOps 工具集 33

第一章

1、互联网时代的软件开发有哪些挑战？

1. **业务不断创新的挑战**，IT 团队，应用和基础架构要能够灵活的支持业务的快速变革与尽快的切入市场。
2. **业务快速增长的挑战**，IT 系统要能够支撑用户的快速增长。
3. **业务连续性的挑战**， 用户会随时随地地使用企业提供的服务，因此需要保证企业 IT 系统的 24*7 可用

2、从五个层面来阐述什么是 DevOps?

价值观（DevOps 宣言）

个体和互动 高于 流程和工具
工作的系统 高于 详尽的文档
客户及程序员合作 高于 合同谈判
响应变化 高于 遵循计划

原则

1） 持续不断的及早交付功能（比“软件”更通用）
2） 软件功能只有在完整的系统交付给客户后才能实现。非功能性需求同样重要。（新增：为什么系统很重要）
3） 基础设施是代码，应该同样进行开发和管理。（新增。）
4） 欣然面对需求变化，及时在项目后期。为了客户的竞争优势，敏捷过程掌控变化。（相同）
5） 较短周期交付可工作的功能（软件->功能）
6） 业务人员，开发人员和运维人员必须相互合作（添加运维人员）
7） 激发个体的斗志，以他们为核心搭建项目。提供所需的环境和支援，辅以信任，从而达成目标（相同）
8） 不论团队内外，传递信息效果最好效率最高的方式是面对面交谈。（相同）
9） 可工作的软件并进行完整交付是进度的首要度量标准
10） 敏捷过程倡导可持续开发。责任人，开发人员，运维人员和用户要能够共同维持其步调稳定延续。（添加运维人员）
11） 坚持不懈的追求技术卓越和良好设计，敏捷能力由此增强。（相同）
12） 以简洁为本，他是极力减少不必要工作的艺术。（相同）
13） 最好的架构、需求和设计出自组织团队。（相同）
14） 团队定期烦死如何能提高成效，并依此调整自身的举止表现。（相同）
15） 此外，精益的原则也得到广泛认可。
    a) 消除浪费

b) 增强学习
c) 尽量延迟决策
d) 尽快交付
e) 赋予团队权利
f) 内建完整性
g) 全局优化

方法

常见的敏捷方法有 Scrum，XP，Kanban。
Scrum： 以经验过程控制为理论依据，采用迭代、增量的方法来提高产品开发的可预见性并控制风险。
极限编程： 把好的编程实践发挥到极致。
Kanban：主要规则有 可视化工作流； 限定 在制品； 衡量并管理周期时间。

实践

DEVOPS 中可以使用敏捷软件开发中常见的管理实现包括
迭代式计划、站立会议、回顾、评审、短周期迭代、团队估算等；
以及常见的技术实现包括
单元测试、持续交付、持续集成、编码标准、重构。

工具

源代码库
构建服务器
配置管理
虚拟基础架构和容器
测试自动化
管道编排

第二章

云计算

定义： 云计算是一种允许通过网络随时随地便捷地按需使用可配置的计算资源共享池的模式， 并且只需要投入很少的管理工作或与服务供应商进行很少的交互就能够快速地提供或释放这些计算资源， 具有按需自服务、 广泛的网络终端支持、 资源池、 快速弹性、 可度量的服务等五大关键特点。
分类：
按服务模式：
SaaS:软件即服务、 PaaS:平台即服务、 IaaS:基础设施即服务
按部署模式

私有云、 社区（行业） 云、 公有云、 混合云

思考题

1. 云计算的大行其道对传统运维人员提出了新的要求， 传统运维人员应该做哪些改变才能从容应对？

a. 实现自动化部署应用
b. 快速创建和复制资源模板
c. 动态扩容或缩小系统部署
d. 实时监控程序状态
e. 适应不同平台的差异

2. 云时代的运维与业界推崇的 devOps 有何联系？ 在互联网严峻的安全形势下， DevOps 过程该如何考虑信息安全的问题呢？

a. devOps 是云时代运维发展的一种趋势， 自动化运维是云时代运维的发展方向，devOps 是自动化运维的典型方法（这个是看书+上网以后 瞎猜的）

b. DevOps 考虑信息安全（这个从网上找的， 来自《the devOps Handbook》， 权当参考， 瞟一眼就好）

将安全融入到开发迭代演示中
确保安全工作包含于开发和运营的工作追踪系统中
将信息安全整合到事后剖析过程中
将预防性安全控制整合到共享源代码库和共享服务中
将安全整合到部署流水线中
保护部署流水线免受恶意代码影响
保护应用程序安全
保护软件供应链的安全
保护环境的安全
将信息安全整合到生产测量中

3. 请描述 CMMI 模型的要点

CMMI： (Capability Maturity Model Integration)能力成熟度模型集成
CMMI-SVC： (CMMI for Services 服务模型)
五个成熟度等级
1. 初始级： 个人贡献
2. 可重复级： 基础管理

3. 已定义级： 过程标准化
4. 量化管理级： 量化管理
5. 优化管理级： 持续过程改进

四类过程域:

过程管理、 项目管理、 服务建立和交付、 支持过程域

7 个与服务有关的特定过程域

服务交付(SD)、 战略服务管理 (STSM、 冲突解决与预防 (IRP) 、 服务系统转变 (SST) 、 服务系统开发(SSD)、 服务持续性 (SCON) 、 容量和可用性管理 (CAM)

CMMI 成熟度模型

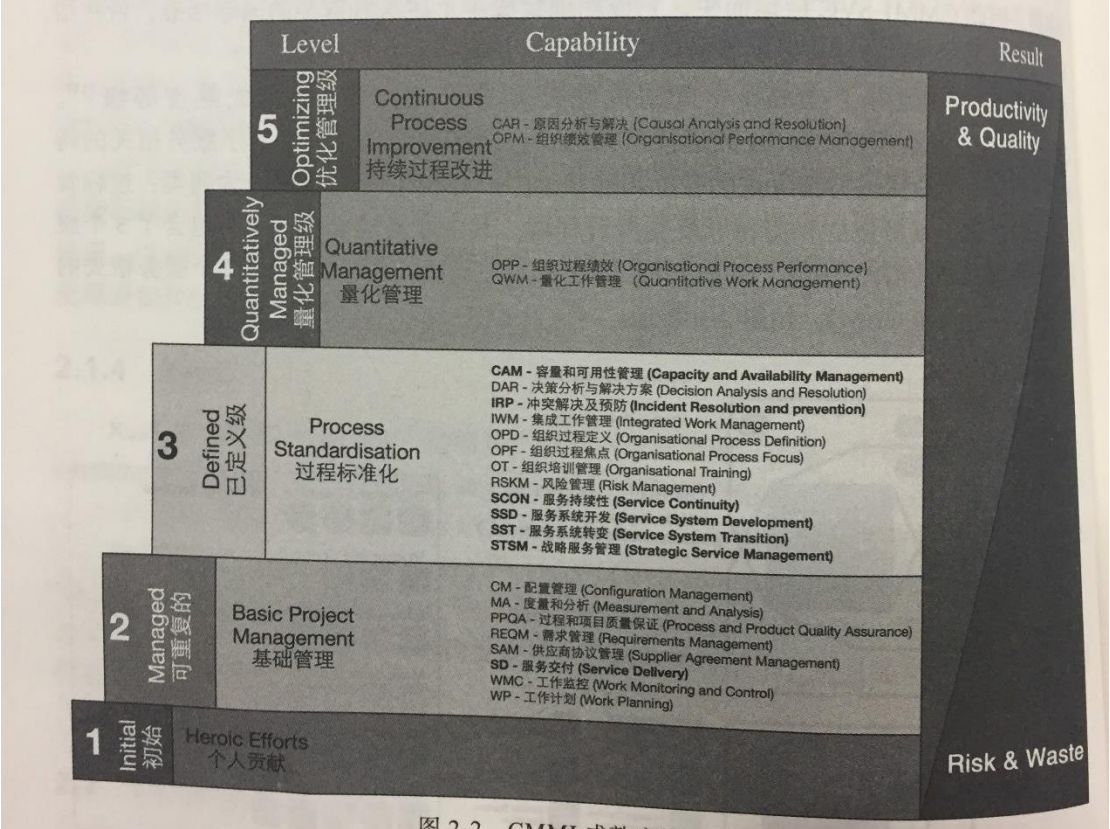


图 2-2 CMMI 成熟度模型 [13]

其中 7 个与服务

4. 请描述 ITSS 模型的要点

ITSS： Information Technology Service Standard 信息技术服务标准，是一套成体系和综合配套的信息技术服务标准库， 全面规范了信息技术服务产品及其组成要素， 用于指导实施标准化和可信赖的信息技术服务， 以保障其可信赖。

ITSS（IT 服务） 的生命周期——PIOIS

规划设计（Planning & Design）
部署实施（Implementing
服务运营（Operation）
持续改进（Improvement）
监督管理（Supervision）

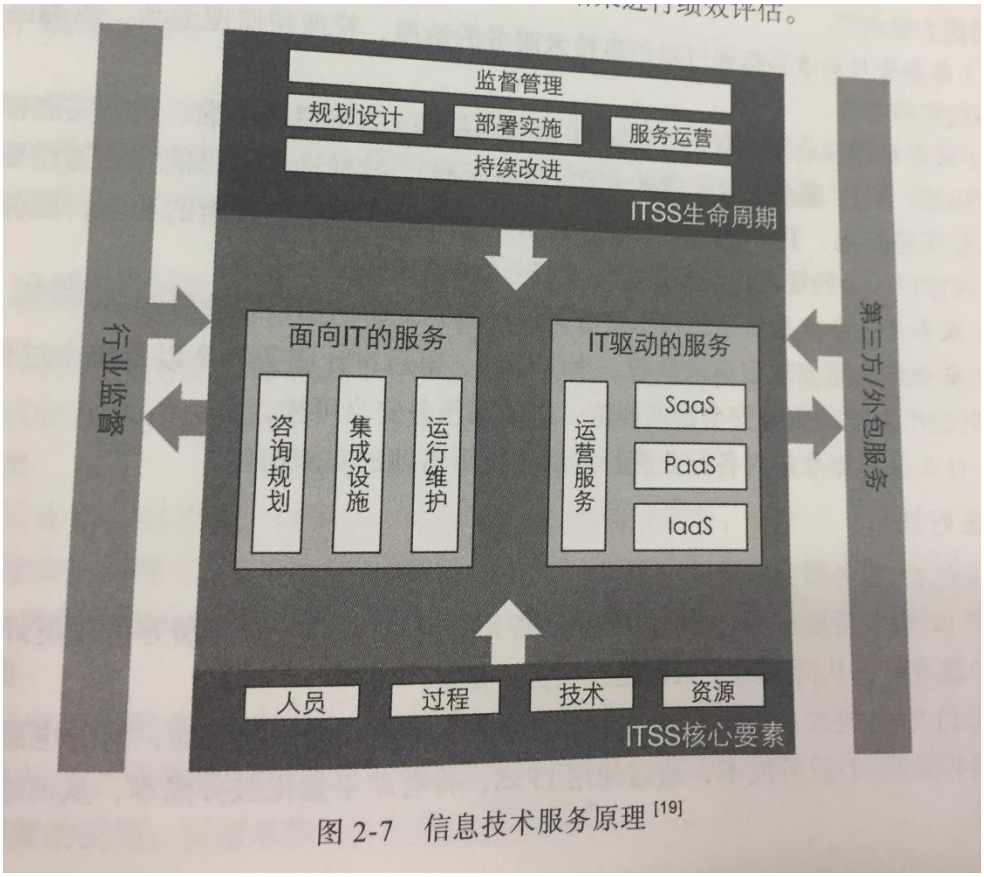


图 2-7　信息技术服务原理 [19]

ITSS 核心要素——PPTR

人员(People)、 过程（Process)、 技术（Technology） 、 资源（Resource）

ITSS 的内容（6 大标准）

基础标准、 服务管控标准、 业务标准、 服务外包标准、 安全标准、 行业应用标准

ITSS 的好处（对服务需方/服务供方， 三个方面）

提升 IT 服务质量

优化 IT 服务成本

强化 IT 服务效能

降低 IT 服务风险

5. 请描述 ITIL 模型的要点

ITIL：Information Technology Infrastructure Library(信息技术基础架构库)，是一套公开的、基于业界最佳实践制定的、 用于规范 IT 服务管理的流程和方法论。 以流程为导向， 以客户为中心， 目的是确保 IT 能更好地服务于业务部门， 从而让企业的 IT 投资最大化

围绕的五个部分（服务策略是核心）

服务策略、 服务设计、 服务转换、 服务运营、 服务改进

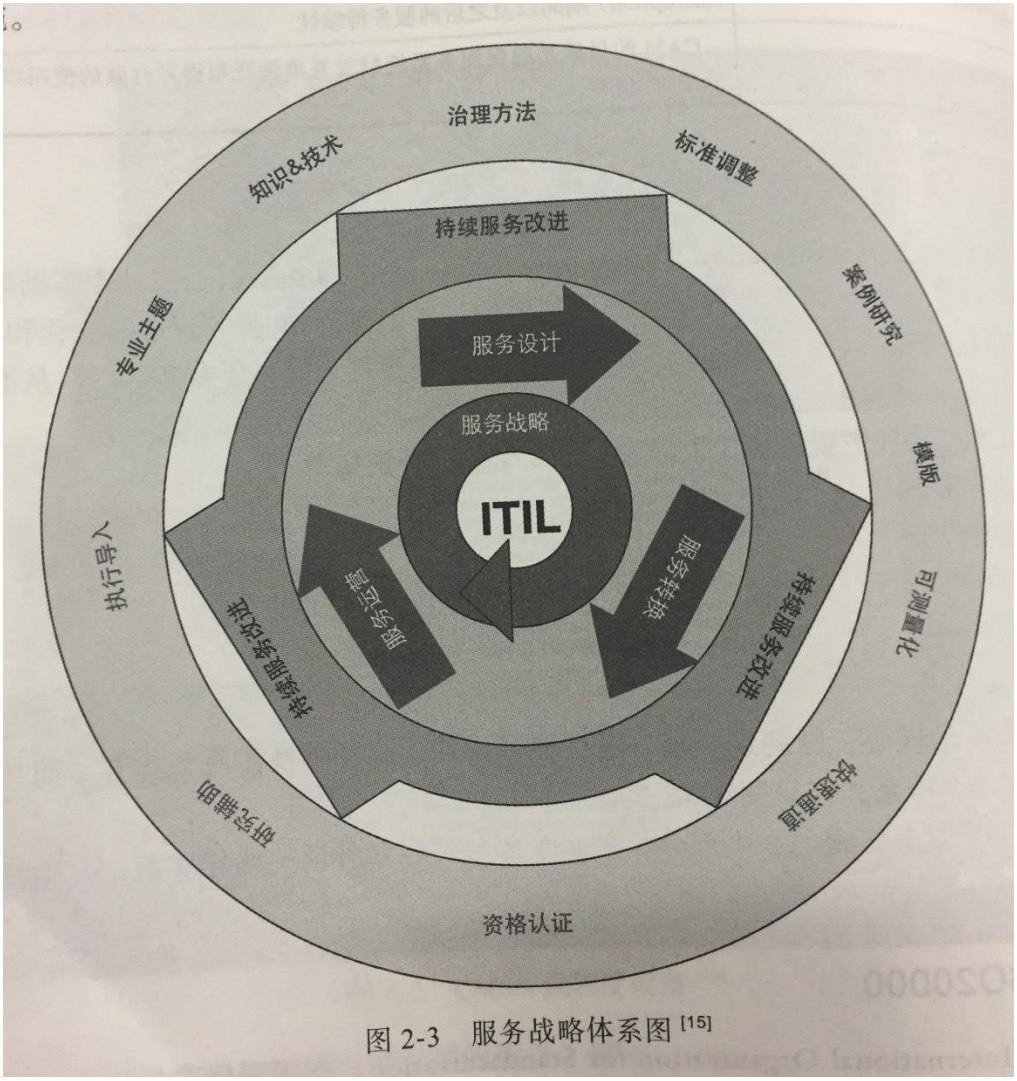


图 2-3　服务战略体系图 [15]

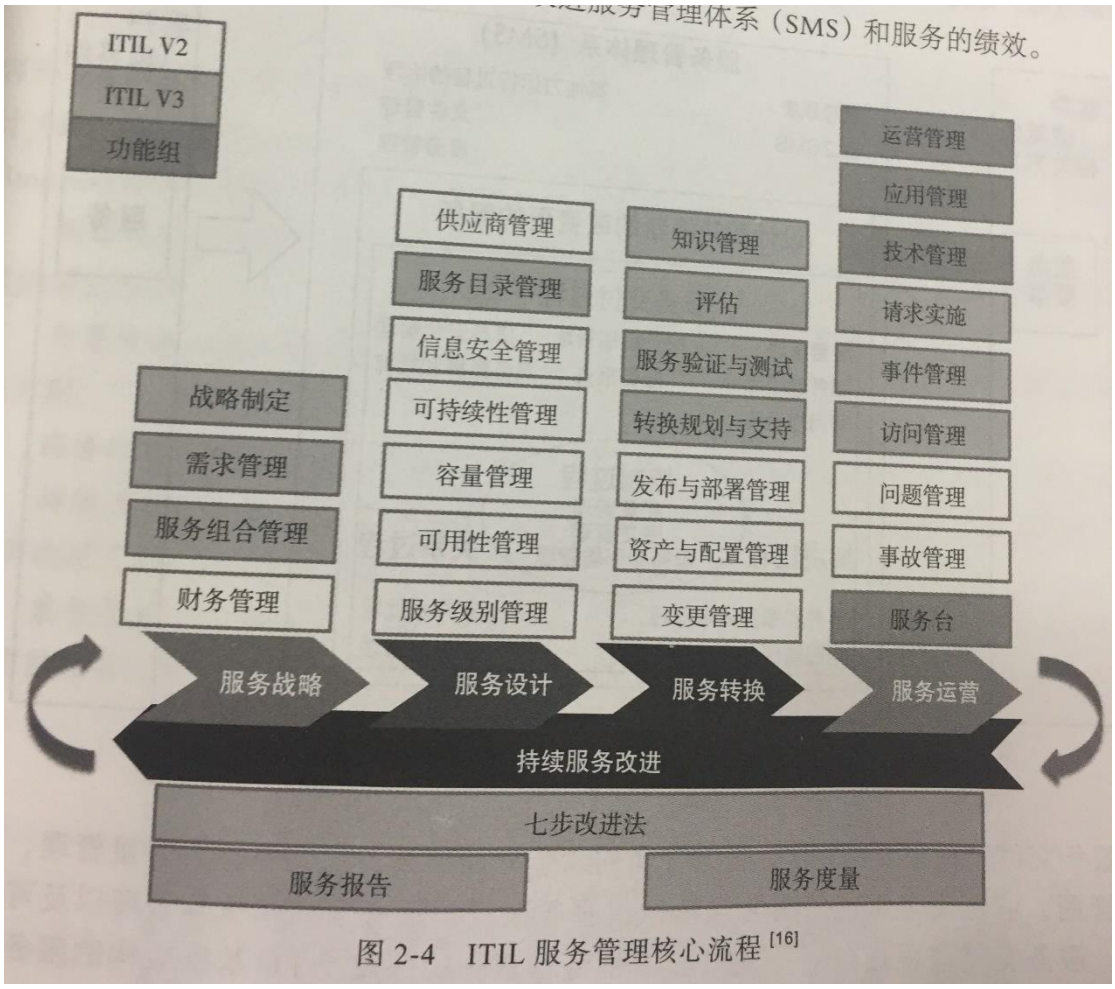


图 2-4　ITIL 服务管理核心流程 [16]

6. 请描述 ISO20000 模型的要点

ISO: International Organization for Standardization 国际化标准组织
ISO20000 帮助识别和管理 IT 服务的关键过程， 提出了服务文化， 定义了一系列相互关联的服务管理过程。
PDCA 方法论
P（策划） ： 建立书面和协定的服务管理体系（SMS） ,服务管理体系包括满足服务需求的方针、 目标、 计划和过程
D（实施） ： 实施和运行服务管理体系（SMS） ， 以设计、 转换、 交付和改进服务
C（检查） ： 根据方针、 目标、 计划和服务需求， 对服务管理体系（SMS） 进行监视、测量和回顾， 并报告结果
A（处置） ： 采取措施， 以持续改进服务管理体系和服务绩效

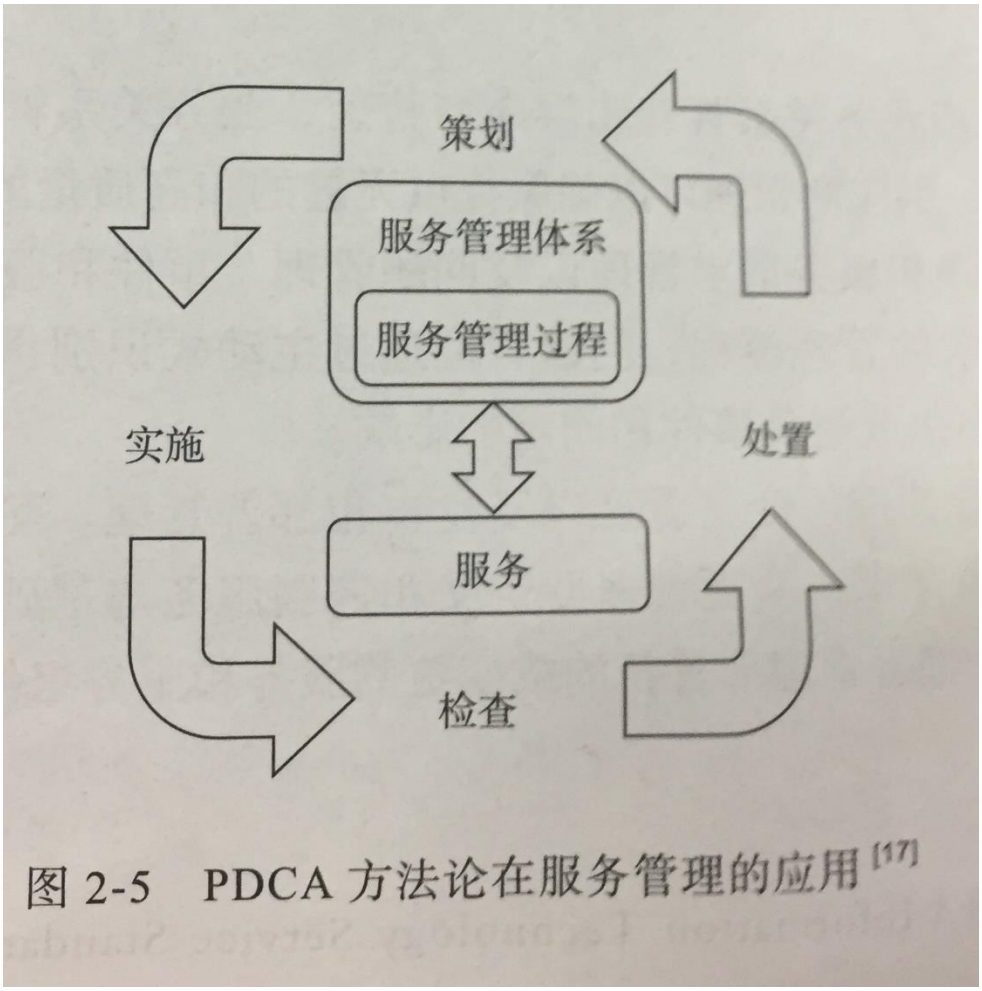


图 2-5　PDCA 方法论在服务管理的应用[17]

4 个关键的服务管理过程

服务交付过程： 围绕 IT 服务管理的六个方面展开： 容量管理、 服务级别管理、 信息安全管理、 服务连续性、 服务报告、 服务预算和核算管理以及可用性管理

关系过程： 包括业务关系管理和供应商管理

解决过程： 包括事件和服务请求管理以及问题管理

控制过程： 包括配置管理、 变更管理以及发布和部署管理

图 2-6　服务管理体系 [17]

第三章

注：（）内容为相关说明，供理解，不用背

1.什么是一个好的软件架构

（在现实生活中，没有一个架构是完全可以通用的，更没有一个最好的架构可以适用于任何使用场景。）

软件架构是对用户的需求，公司对业务的需求，公司未来发展的需求，以及成本条件的折中，在这些限制条件下，我们才能探求一个好的架构设计，同时，一个好的架构设计也不是一成不变的，随着时间的发展，公司的业务方向会发生变化，需求会发生变化，相应的成本预算也会发生变化，所以架构也自然随着这些先决条件的变化而不断演化。

一个好的架构的首要目标是能够支持这种演化，让演化能够按照预期的方式进行扩展，而不是整个架构级别的重构。

3.常见的软件架构的视图有哪几种？描述每种视图的作用和涉众

4+1 模型：逻辑视图，开发视图，进程视图，物理视图+场景

1）逻辑视图：逻辑视图主要是为了支持功能性需求，即系统需要给用户提供哪些服务。涉众：最终用户

（在这个视图中，会利用面向对象的方式（抽象，封装，继承）把系统分解成一系列领域对象和类）

（一般用类图描述/数据驱动则是 E-R 图）

2）进程视图：进程视图主要从性能，可用性等一些非功能需求的角度，从多个抽象层次去描述系统。涉众：系统设计和继承人员

（着重强调系统的并发和分布，系统完整性，容错性，以及把逻辑视图中的抽象元素放到进程视图中时，哪个控制进程去运行一个 操作）

（组件：任务，进程）

3）开发视图：开发视图主要关注软件开发实现过程中的真正的软件模块以及如何组织，同时也用来做任务分配，成本评估，制定开发计划，跟踪开发进度，考虑软件的重用性和可移植性等，是制定产品线的一个基础。涉众：开发者和技术经理

（组件：模块，子系统）

4）物理视图（部署视图）：物理视图主要考虑把一些非功能性的需求，比如可用性，可靠性（容错性），性能（吞吐量）以及横向扩展性。涉众：系统设计者

（组件：节点）（C、F、K：CPU、内存、磁盘）

5）场景视图：把以上视图的设计元素结合在一起，利用对象场景图和对象交互图来描述系统中一些重要用例的场景，对最重要的一些需求进行抽象。涉众：最终用户和开发者，主要用来

a) 作为驱动的工具，用在架构过程中发现架构元素（场景驱动的架构设计）
b) 作为验证的角色，用在架构设计完成后，在纸上进行架构原型的测试和验证。

4.相比传统软件架构，互联网软件系统有什么特点

1) 用户量大，并发高
2) 业务变化快，强调敏捷
3) 系统多（可能有几个，甚至成百上千个不同功能系统同时运行）
4) 对系统可用性要求更高，追求 24*7 零宕机时间

5.阐述单体架构的优缺点

优点：（后面章节找的）

易于测试；
IDE 友好；
便于共享
为人所熟知

容易部署

缺点：

多个团队协作开发效率低，需要额外的协调工作

系统的可维护性非常差，尤其是对大型的应用

任何小的改动部署都会影响整个系统重新部署

系统复杂了以后，很难做扩展

遇到性能问题，只能整体加机器，难以拆分

6.阐述事件驱动架构中 Mediator 模式和 Broker 模式的区别

Mediator 模式适合需要很多步骤来处理一个事件，同时又希望这些步骤的组合在未来容易更改的场景，当一个事件通过消息队列发给 Event Mediator 的时候，Event Mediator 会根据预配置的步骤分成多个子事件，通过 Event Channel 分发给具体的 Event Processor。步骤控制放在独立的一个 Event Mediator，Event Processor 只处理单一的业务逻辑。

Broker 模式和 Mediator 模式最大的区别就是没有 Event Mediator，它的消息流直接通过消息通道分发到下游的消息处理器。Event Processor 除了做好自己的业务逻辑，还要明白事件的来源和去向。

当业务流程比较固定的时候，Broker 模式更易于维护，当业务流程变化比较大的时候，Mediator 模式更易于修改。

7.从运维角度，阐述微服务架构的优点

1) 每个服务可以被独立的开发、部署，其他服务几乎不受影响，也不需要跟其他服务的开发人员进行额外的协作和等待
2) 局部修改，局部更新。当运维对单体应用进行修改时，可能要先把应用停止运行后才去修改，而微服务只需逐步修改和更新即可；
3) 故障隔离，非全局。单体应用中只要一个模块有故障，应用就无法正常运行。但微服务的故障隔离性、业务可持续性都非常高；
4) 资源利用率高。单体应用的资源利用率低，而使用微服务，可以按需分配资源，资源利用率会非常高。

8，9，10-以 4+1 架构模型描述和演进教学支持系统

略-附上书上 4+1 模型示例：

1) 逻辑视图

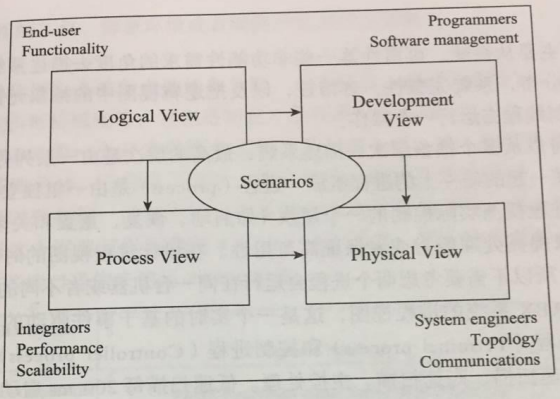


图 3-4　4+1 架构模型 [3]

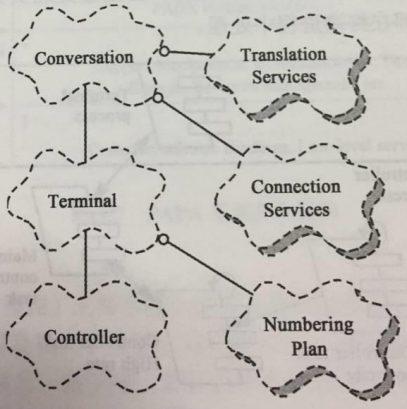


图 3-5　PABX 系统逻辑视图 [3]

2)　进程视图

图 3-6 是 PABX 系统的进程视图，这是一个实时的基于事件驱动的系统。总共有
个进程，终端进程（Terminal process）和控制进程（Controller process）。控制进程有
个任务组成：低速扫描、高速扫描、主控处理。低速扫描每 200 ms 启动一次，如果有
件就放到高速扫描任务列表里，等高速扫描（10 ms 启动一次）启动的时候，如果发现
任何重要的状态变化，就把事件发给主控处理，由主控处理负责解析并和终端进程进行
交互。所有的终端操作都在终端进程中处理。

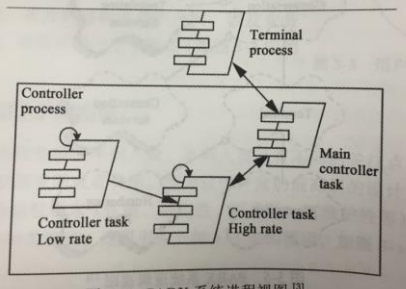


图 3-6　PABX 系统进程视图 [3]

3. 开发视图

3)　开发视图

同时，开发视图也会被用来做任务分配，成本评估，制订开发计划，跟踪开发进度，考虑软件重用性和可移植性等，它也是制定产品线的一个基础。

图 3-7 是 PABX 的一个开发视图的示例，总共有 5 层。第 1 层和第 2 层是领域无关的一个分布式基础平台，它屏蔽了硬件、操作系统、数据库等的差异。基于这个基础平台，整个产品可以拥有统一的编程接口而不需要考虑底层的平台。第 3 层是领域相关的基础框架，利用这个框架可以构建第 4 层的 PABX 功能。第 5 层是对外的接口，系统外的终端通过跟这些接口通信来使用 PABX 功能。

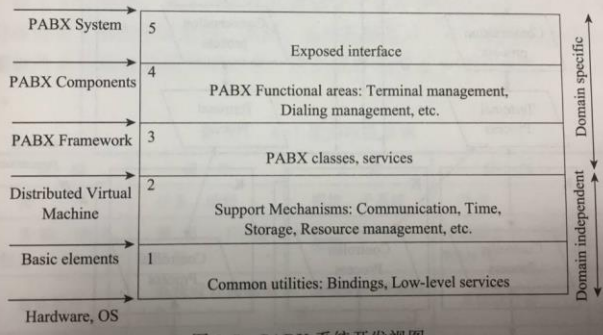


图 3-7 PABX 系统开发视图

4. 物理视图

4) 物理视图

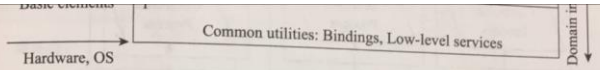


图 3-7 PABX 系统开发视图

4. 物理视图

物理视图（也叫部署视图）主要考虑一些非功能性的需求，比如可用性、可靠性（容错性）、性能（吞吐量）以及横向扩展性。这个视图中，我们会把进程视图的进程映射到一组基于网络的计算节点上。由于同一套系统会被部署到不同的环境中（开发环境、测试环境、生产环境），所以这个映射关系必须足够灵活并且尽可能少地修改源代码。

以下展示 PABX 的两个不同集群规模的物理视图：图 3-8 为小集群部署的物理视图，图 3-9 为大集群部署的物理视图。

其中，C、F、K 代表了 3 种不同类型的计算节点，每一种类型有不同的计算资源（CPU、内存、磁盘等）。

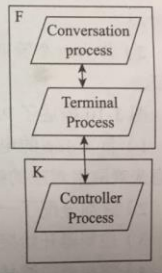


图 3-8 PABX 系统小集群物理视图 [3]

5. 场景视图

场景视图是把以上一些视图的设计元素结合在一起，利用对象场景图和对象交互图来描述系统中一些重要用例的场景，对最重要的一些需求进行抽象。

5) 场景视图

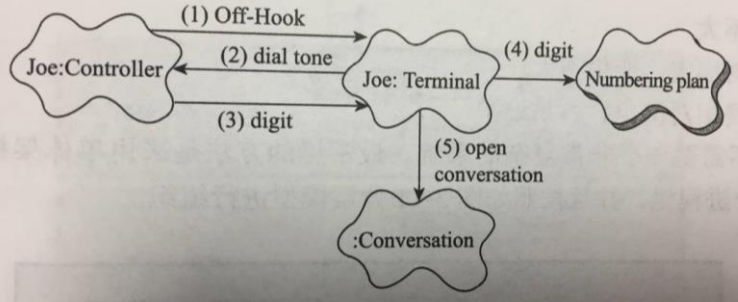


图 3-10 PABX 系统场景视图 [3]

第四章

1、 在个体和小组两个层次上，有哪些典型的软件过程和方法？试论述每种方法的特点。

个体：

（1） PSP 过程：PSP 是包括了数据记录表格、过程操作指南和规格在内的结构化框架。肯定“过程质量决定最终质量”，突出个体软件工程师在管理和改进自身过程中的能动性。

（2） PSP 过程度量：过程度量在过程管理和改进中起极为重要的作用，帮助过程的实践者了解过程状态，理解过程偏差。没有度量就没有办法管理软件

（3） PROBE 估算方法：如果新建立的组件与以前建立的组件类似，那么新组件所需的工作量与旧组件一样。在 PROBE 估算中，需要建立自己的代码库，以跟踪所有程序的规模和工作量。

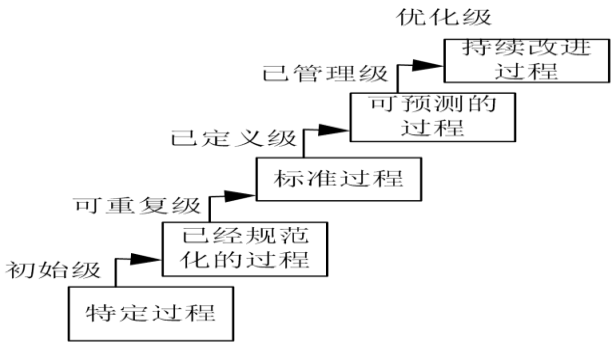
小组：

（1） XP 实践：eXtreme Programming，是敏捷过程中最富盛名的一个，极限的含义是指把好的开发实践运用到极致。极限编程已经成为一个典型的开发方法，广泛应用于需求模糊且经常变更的场合。

（2） Scrum 方法：迭代式增量的敏捷软件开发过程，包括了一系列实践和预定义角色的过程骨架

（3） TSP 过程：TSP 是一个已经被良好定义并证明的支持构建和管理团队的最佳实践，指导团队中的成员如何有效地规划和管理所面临项目的开发任务，告诉管理人员如何指导软件开发队伍。

2、 软件过程改进的参考模型有哪些？

CMM 模型	1） CMM 是专门针对软件产品研究开发的评估模型。CMM 描述了一个有效的软件过程中的关键要素，描述了成为有规律的、成熟的软件机构的改进阶段过程，CMM 可以科学地评价软件开发企业的软件能力成熟等级。  2）

<table>
<tr><td></td><td>3）
<table>
<tr><th></th><th>等级</th><th>特征</th><th>主要需解决的问题</th></tr>
<tr><td>五</td><td>优化级</td><td>经反馈得以改进的过程</td><td>保持优化的机构，但仍为人员密集的过程</td></tr>
<tr><td>四</td><td>已管理级</td><td>(量化的)已度量的过程</td><td>技术变更、问题分析、问题预防</td></tr>
<tr><td>三</td><td>已定义级</td><td>(量化的)已定义且制度化的过程</td><td>过程度量、过程分析、量化质量计划</td></tr>
<tr><td>二</td><td>可重复级</td><td>(直觉的)过程依赖于个人</td><td>培训、测试、技术常规和评审、过程关注、标准和过程</td></tr>
<tr><td>一</td><td>初始级</td><td>个别的、混乱的过程</td><td>项目管理、项目策划、配置管理、软件质量保证</td></tr>
</table></td></tr>
<tr><td>CMMI</td><td>解决使用多种能力成熟度模型的问题，对 CMM 模型做了较多改进和补充，可以支持多个群集，群集可与其成员分享最佳实践方法</td></tr>
<tr><td>SPICE</td><td>1） 软件过程改进和能力鉴定标准，是新兴的软件过程能力评估国际标准
2） 过程类别共有五种，分别是
➢ 客户-供应商(CUS)过程
➢ 工程（ENG）过程
➢ 支持（SUP）过程
➢ 管理（MAN）过程
➢ 组织（ORG）过程</td></tr>
<tr><td>ISO/IEC15504</td><td>前身是 SPICE，和 CMM 内容相关，都是为软件组织的过程能力进行评估，此外也为组织提供了一种兼容的、可重复的软件能力评估方式，可以根据组织的具体情况选择评估范围，可以在组织的局部范围内进行评估定级</td></tr>
<tr><td>ISO/IEC 12207</td><td>软件生命周期过程的国际标准</td></tr>
<tr><td>ISO 9000</td><td>1） 重点关注“过程质量”，强调“持续改进”
2） 获得 ISO 9000 标准认证的企业应该具有 CMM 第 2~3 级的水平</td></tr>
</table>

3、 过程改进的元模型有哪些？

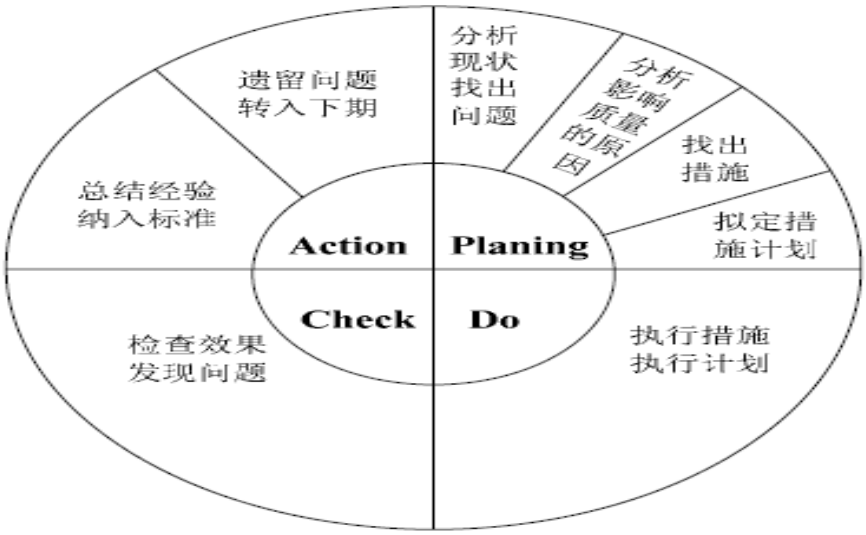
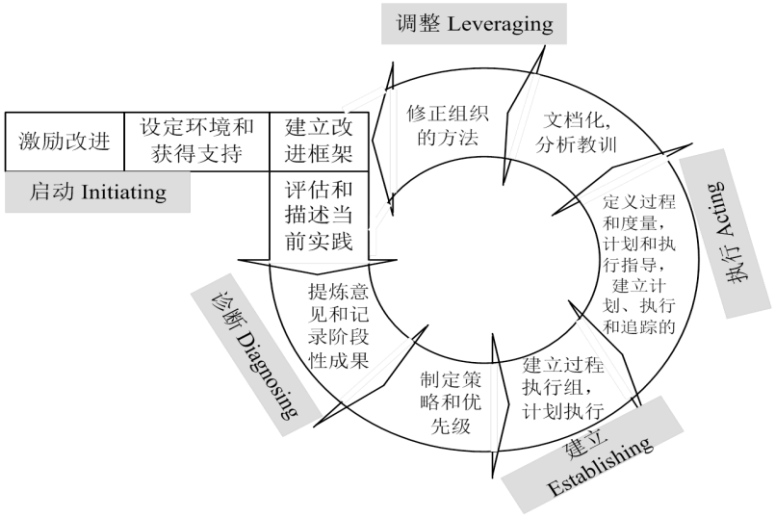
<table>
<tr>
<td>PDCA 模型</td>
<td> 

图 1.20 戴明的 PDCA 循环示意图</td>
</tr>
<tr>
<td>IDEAL 模型</td>
<td>1） 包括软件过程改进周期的五个阶段
➢ I: Initiating 开始
➢ D: Diagnosing 诊断、评价
➢ E: Establishing 建立
➢ A: Acting 执行
➢ L: Leveraging 调整
 

2）</td>
</tr>
</table>

4、 试着比较 Scrum 方法和 TSP 方法，两者有什么相似和相异之处？

相同：

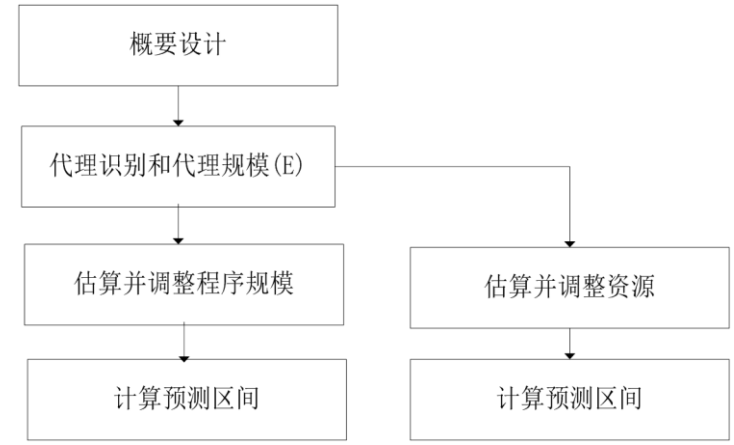
1、 都是小组过程和实践

不同：

1、 Scrum：迭代式增量的敏捷软件开发过程，包括了一系列实践和预定义角色的过程骨架
2、 TSP :TSP 是一个已经被良好定义并证明的支持构建和管理团队的最佳实践，指导团队中的成员如何有效地规划和管理所面临项目的开发任务，告诉管理人员如何指导软件开发队伍。TSP 提供了一个已经定义的团队构建过程、一个团队作业框架、一个有效的管理环境

5、 PSP 估算方法是什么？这种方法有何特点？

PROBE 估算方法（Proxy Based Estimation）

思想	如果新建立的组件与以前建立的组件类似，那么新组件所需的工作量与旧组件一样。在 PROBE 估算中，需要建立自己的代码库，以跟踪所有程序的规模和工作量，而代码库中的每个组件都有设定的类型(计算、逻辑或数据等)和规模(非常小、小、中、大、非常大)。当开始一个新项目时，我们可以将任务划分成与代码库中组件相似的类型和规模，然后利用线性回归方法来估算项目的工作量。
估算流程	

6、 PSP 有哪些原则？

1、 软件系统的整体质量由该系统中质量最差的某些组件所决定
2、 软件组件的质量取决于开发这些组件的软件工程师，更确切地说，是由这些工程师所使用的开发过程决定。
3、 作为合格的软件工程师，应当自己度量、跟踪自己的工作，应当自己管理软件组件的质量
4、 作为合格的软件工程师，应当从自己开发过程中的偏差学习、总结，并将这些经验教训整合到自己的开发实践中，也就是说，应当建立持续的自我改进机制。

7、 PSP 的质量策略是什么？对实践有哪些启发？

为了使一个软件产品可以工作，该产品没有缺陷是最基本的要求，这样整个软件产品的质量目标可以归结为首先要确保基本没有缺陷，然后在考察其他的质量目标。PSP 中就使用这种方式，用**缺陷管理来替代质量管理**，这大大简化了质量管理的方法，使得质量管理更加易于

操作。

1. 软件质量目标：软件项目的日程、成本和质量
2. PSP 中定义质量为满足用户需求的程度（所以就需要明确用户需求的范围、优先级、度量方式）
3. PSP 中用**缺陷管理**来代替质量管理，高质量的产品也就意味着组成软件产品的各个组件基本无缺陷；而各个组件的高质量是通过**高质量评审**来实现的

8、 CMM/CMMI 的五个成熟度级别的特征是什么？

1. CMM 初始级（CMMI 初始级） 个人行为，软件开发过程混乱无序无准则指导。
2. CMM 可重复级（CMMI 已管理级） 已建立基本的项目管理过程跟踪成本、进度和功能性；建立了必要的过程纪律，能重复利用以前的成功。
3. CMM 已定义级（CMMI 已定义级） 软件过程文档化、标准化，并集成到组织的标准软件过程。项目活动是标准和一致的，不同项目采用相同的标准。
4. CMM 已管理级（CMMI 定量管理级）已经采用详细的有关软件过程和产品质量的度量，可有效预测。
5. CMM 优化级（CMMI 优化级） 定量反馈，持续过程改进。

9、 试比较 CMMI 和 SPICE 两个模型的异同。

	CMMI	SPICE
过程	过程管理； 项目管理； 工程； 支持；	客户 – 供应商过程； 工程过程； 管理过程； 支持过程； 组织过程
级别	初始级； 已管理级； 已定义级； 定量管理级； 优化级	0. 不完整过程； 1. 已实施过程； 2. 已管理过程； 3. 已建立过程； 4. 可预测过程； 5. 优化过程；

10、 DevOps 模型对现有的软件开发过程和方法可能带来哪些影响？

目前并不存在专门为 DevOps 所定义的开发过程和方法，很多 DevOps 的实践者认为软件开发中的精益方法是 DevOps 的基础方法学。

第五章

一、精益思想：

a) 起源：丰田对制造业的重新认识和成功实践

b) 本质： 准时化 自働（not 动）化

i. 准时化：即时生产，只在需要的时间和地点生产需要数量的东西，降低库存，加速流动，即时暴露问题（重要工具： 看板）

看板：后道工序需要时通过 看板 向前道工序发出信号，前道工序得到 看板 后 再 按需生产（下游的客户价值 拉动 上游的生产活动， 使产品向下游流动）

ii. 自働化（auto-no-mation）：出现问题时机器和生产线自动停止，以迫使现时现地解决问题，并发现问题根源

iii. 与 自动化 的区别：

1. 自働化：质量内建于每个制造环节，出现异常时杜绝继续产出不合格产品（过程中停止并解决）
2. 自动化：质量依赖于最后的检测环节（一套流程结束后检测问题-发现问题-定位问题-解决问题，造成时间和资源的浪费）

c) 定义：有效组织人类活动的一个新的思维方法，目标是消除浪费，以更多地交付有用的价值

d) 三个层面：p128

i. 价值观：精益价值观

1. 两个主题（支柱）：尊重人、持续改进（《丰田之道》，精益思想的两个支柱）
2. 5 个价值观：

①挑战现状

②改善

③现地现物：去源头发现事实，做出正确决策，建立共识，最快速地达成目标

④尊重

⑤团队合作

ii. 方法学：5 个原则（5 个步骤）：

1. 定义价值：用户视角定义，并描述为具体产品或服务
2. 识别价值流：识别创造价值的步骤，消除不增加用户价值的步骤
3. 让价值持续流动：让用户价值在流程步骤中流动起来，使它们持续、顺畅地流向最终用户
4. 用户价值拉动：由用户价值拉动流动，避免不带来用户价值的流程浪费
5. 精益求精：不断重复 1-4，消除过程中的所有浪费

iii. 实践：行业区分，如制造业的 自働化、准时生产、看板拉动

iv. 方法学最重要：聚焦于用户价值，关注价值的流动并持续改进，方法学是打开精益开发实践之门的钥匙

二、精益产品开发实践体系

a) 目标：顺畅和高质量地 交付 有用的价值

i. 顺畅：指价值的交付过程要顺畅，用最短时间完成用户价值的交付

ii. 高质量：符合要求，避免不必要的错误
iii. 有用的价值：交付的价值应该符合市场和用户的要求，并能产生业务影响，促进组织绩效

b) 方法学（原则、支柱）
i. 探索和发现有用价值
与制造业的区别：制造业是重复性的确定活动，可以预先定义价值，而软件开发是一个不确定性很强的活动，需要把探索和发现价值融入产品开发和交付过程
ii. 聚焦和提升价值流动效率
流动效率：从用户的视角审视用户价值历经各个流程步骤直至交付的过程，整个过程时间越短等待越少则流动效率越高
从外部绩效（用户价值）出发，协调内部资源最快地交付用户价值
与传统管理方法区别：传统关注内部资源效率，精益产品开发聚焦和提成价值流动效率

c) 实践
i. 管理实践：协调和运作开发过程，帮助组织探索和发现用户价值，聚焦和提升价值流动效率
1. 精益创业和创新实践：探索、发现和验证价值（包括商业模式设计、精益产品设计、定性验证、精益数据分析等）
2. 精益需求分析和管理实践：如何有效拆分、规划和沟通需求（包括场景分析、用例设计、领域建模、古诗地图、发布规划、实例化需求等）
3. 精益看板方法实践：如何让价值顺畅高质量地流动（包括 可视化价值流动、显式化流程规则等）
ii. 技术实践
1. 和敏捷相似，强调小批量持续交付
2. 挑战：如何维护架构和设计的一致性、如何降低验证成本

三、看板方法的起源（制造业）

a) 看板是精益制造系统的核心工具
b) 看板向上传递形成的信息流 拉动了向下的物流，直至交付用户价值，最终拉动生产的源头是用户的需求=>拉动式生产系统（看板系统
c) 拉式生产方式优点：
i. 控制库存
ii. 加速流动：进入生产环节的物料很快被拉入下一环节，直至变成成品，实现了保证安全库存的前提下物料最快的流动，提高了工厂的运转效能
iii. 灵活响应：用户需求的变化通过看板信息流快速传递至上游各环节。低库存也降低了负载，让响应更加迅捷和低成本
iv. 促进改善：库存降低+加速流动=>生产环节的问题可以再第一时间暴露

四、产品开发中的看板方法：2 组 5 个核心实践

a) 2 步：
i. 建立看板系统：
1. 可视化价值流动、显式化流程规则、控制在制品数量
ii. 运作看板系统：让用户价值在看板系统中顺畅和高质量地流动
1. 管理工作流动、建立反馈和持续改进

b) 5 个核心实践

i. 可视化价值流动（使产品开发的价值流动可视化）：三个重点要素
   1. 用户价值（原因：产品开发目标是交付用户价值）
   2. 用户价值端到端的流动过程：价值提出到价值交付的整个过程
   3. 问题和瓶颈
      a) 问题：阻碍用户价值流动的因素
      b) 瓶颈：价值流动不畅的环节，工作在瓶颈处积压形成队列
ii. 显式化流程规则（价值流转规则）
   1. 流程规则：
      a) 流转规则：工作项从看板墙上的一列进入下一列所必须达到的标准
      b) 其他协作规则, 如 :各种活动的节奏和组织方式、优先级的确定方式、问题处理的机制等
   2. 显式化流程规则：明确以上两类规则，并在团队内达到共识
   3. 显式化的流程规则是团队协作的依据，更是团队改进的基线
iii. 控制在制品数目
   1. 在制品：特定环节内所有的工作项（包括进行中和等待的）
   2. 当环节内在制品小于某个既定数字时，可以从上一环节拉入新的工作，否则不允许拉入新的工作
   3. 意义：
      a) 环节内并行工作减少，单个工作项的完成加等待时间缩短，交付时间随之缩短
      b) 帮助团队暴露瓶颈和问题，下游顺畅时才能从上游拉入新的工作，否则将会连环阻塞
      c) 加速用户价值的流动
iv. 管理工作流动：为了让用户价值顺畅和高质量地流动
   1. 就绪队列填充会议（管理价值输入）：就绪队列：看板系统的输入环节和价值流动的源头
   2. 每日站立会议（管理中间过程）：重点关注价值流动过程中的问题和阻碍
   3. 发布计划会议（管理输出）
v. 建立反馈和持续改进：
   1. 目标：反应和度量价值流动的状态，从中发现问题和模式，从而激发和指导团队系统性的改进，并衡量改进行动的效果，形成持续改善的闭环。
   2. 两类反馈和度量体系：
      a) 流动是否顺畅：阻碍问题分类，影响和原因分析
      b) 质量问题反馈：如开发环节或测试环节遗漏缺陷的正交分类和分析

思考题：

1. 丰田生产方式（精益生产）的两大支柱是什么？并简述其作用意义

即时生产（准时化）：只在需要的时间和地点生产需要数量的东西，降低工厂库存，从而加速流动和即时暴露问题（按需生产）

自働化：出现问题时及其和生产线自动停止，质量內建于每个制造环节，出现问题时杜绝继续产出不合格的产品，并现时现地发现问题根源，解决问题

提高生产率，出现问题时能及时发现问题并解决问题，反应更加灵敏，同时也更加保证了产出质量

2. 《精益思想》一书中定义了精益的五大原则，它们分别是什么？

1）定义价值：用户角度定义价值，并把它描述为具体的产品和服务
2）识别价值流：识别和映射创造价值的流程步骤，消除不增加用户价值的步骤
3）让价值持续流动：让用户价值在流程步骤中流动起来，使它们持续、顺畅地流向最终用户
4）用户价值拉动：由用户价值拉动流动，避免没有用户价值的步骤所造成的浪费
5）精益求精：1-4 步循环，消除过程中的所有浪费

3. 精益产品开发的目标是什么：

顺畅、高质量地 交付 有用的价值

4. 支持精益产品开发目标的两大原则和支柱是什么

1）探索和发现有用的价值
与制造业的区别：制造业是重复性的确定活动，可以预先定义价值，而软件开发是一个不确定性很强的活动，需要把探索和发现价值融入产品开发和交付过程
2）聚焦和提升价值流动效率
流动效率：从用户的视角审视用户价值历经各个流程步骤直至交付的过程，整个过程时间越短等待越少则流动效率越高
从外部绩效（用户价值）出发，协调内部资源最快地交付用户价值
与传统管理方法区别：传统关注内部资源效率，精益产品开发聚焦和提成价值流动效率

5. 精益产品开发相关的管理实践分为三大类，它们分别是哪三大类？

1）精益创业和创新实践，解决问题：探索、发现和验证价值
2）精益需求分析和管理实践，解决问题：如何有效地拆分、规划、分析和沟通需求
3）精益看板方法实践，解决问题：如何让价值顺畅、高质量地流动

6. 在丰田生产方式中，“看板”的原始含义是什么？

信号卡……吧

看板工具实质：后道工序在需要时，通过看板向前道工序发出信号，前道工序只有得到看板后，才按需生产 。看板信号由下游向上游传递，拉动上游的生产活动，使产品向下游流动

7. 产品开发中的看板方法由哪五个核心实践构成？

1）可视化价值流动（使产品开发的价值流动可视化）：三个重点要素

① 用户价值（原因：产品开发目标是交付用户价值）

② 用户价值端到端的流动过程：价值提出到价值交付的整个过程

③ 问题和瓶颈

问题：阻碍用户价值流动的因素

瓶颈：价值流动不畅的环节，工作在瓶颈处积压形成队列

2）显式化流程规则（价值流转规则）

① 流程规则：

流转规则：工作项从看板墙上的一列进入下一列所必须达到的标准

其他协作规则，如：各种活动的节奏和组织方式、优先级的确定方式、问题处理的机制等

② 显式化流程规则：明确以上两类规则，并在团队内达到共识

③ 显式化的流程规则是团队协作的依据，更是团队改进的基线

3）控制在制品数目

① 在制品：特定环节内所有的工作项（包括进行中和等待的）

② 当环节内在制品小于某个既定数字时，可以从上一环节拉入新的工作，否则不允许拉入新的工作

③ 意义：

环节内并行工作减少，单个工作项的完成加等待时间缩短，交付时间随之缩短

帮助团队暴露瓶颈和问题，下游顺畅时才能从上游拉入新的工作，否则将会连环阻塞

加速用户价值的流动

4）管理工作流动：为了让用户价值顺畅和高质量地流动

① 就绪队列填充会议（管理价值输入）：就绪队列：看板系统的输入环节和价值流动的源头

② 每日站立会议（管理中间过程）：重点关注价值流动过程中的问题和阻碍

③ 发布计划会议（管理输出）

5）建立反馈和持续改进：

① 目标：反应和度量价值流动的状态，从中发现问题和模式，从而激发和指导团队系统性的改进，并衡量改进行动的效果，形成持续改善的闭环。

② 两类反馈和度量体系：

流动是否顺畅：阻碍问题分类，影响和原因分析

质量问题反馈：如开发环节或测试环节遗漏缺陷的正交分类和分析

8. 看板方法中的可视化价值流动是最基础的实践，请简要描述要可视化哪三个方面

1）用户价值（原因：产品开发的目标是用户价值）
2）用户价值流端到端的流动的过程：价值提出到价值交付的整个过程
3）问题和瓶颈
问题：阻碍用户价值流动的因素
瓶颈：价值流动不畅的环节，工作在瓶颈处积压形成队列

9. 显式化流程规则是看板方法的第二个核心实践，它是团队协作的基础，团队还应该把它看做什么？

明确流转规则和协作规则，并达成共识
团队还应该把它看做团队改进的基线，必要时，团队应调整改进他们

10. 限制在制品数目是看板方法最为核心的实践，它除了加速价值的流动，还起到什么关键作用？

帮助团队暴露瓶颈和问题，下游顺畅时才能从上游拉入新的工作，否则将会连环阻塞

11. 管理工作的流动这一实践，包括管理价值的输入、中间过程和输出，在看板方法中它分别对应什么活动？

1）就绪队列填充会议——管理价值输入
2）每日站立会议——管理价值的流动过程
3）发布计划会议——管理价值输出

12. 反馈的目的是为了持续改善，看板方法的反馈可以分为两类，一类是关于流动是否顺畅的，另一类是什么？

另一类是关于质量问题的反馈，如开发环节或测试环节遗漏的缺陷分析

第六章

1. 什么是微服务架构？

微服务架构风格是一种将一个单一应用程序开发为一组小型服务的方法， 每个服务运行在自己的进程中， 服务间通信采用轻量级通信机制（通常用 HTTP 资源 API） 。 这些服务围绕业务能力构建， 并且可通过全自动部署机制独立部署。 这些服务共用一个最小型的集中式的管理， 服务可用不同语言， 使用不同的数据存储技术

2. 比较微服务与单体应用架构、 SOA 架构的异同

单体架构： 应用程序的全部功能被一起打包作为单个单元或应用程序， 部署在同一个 JVM 或同一台机器上

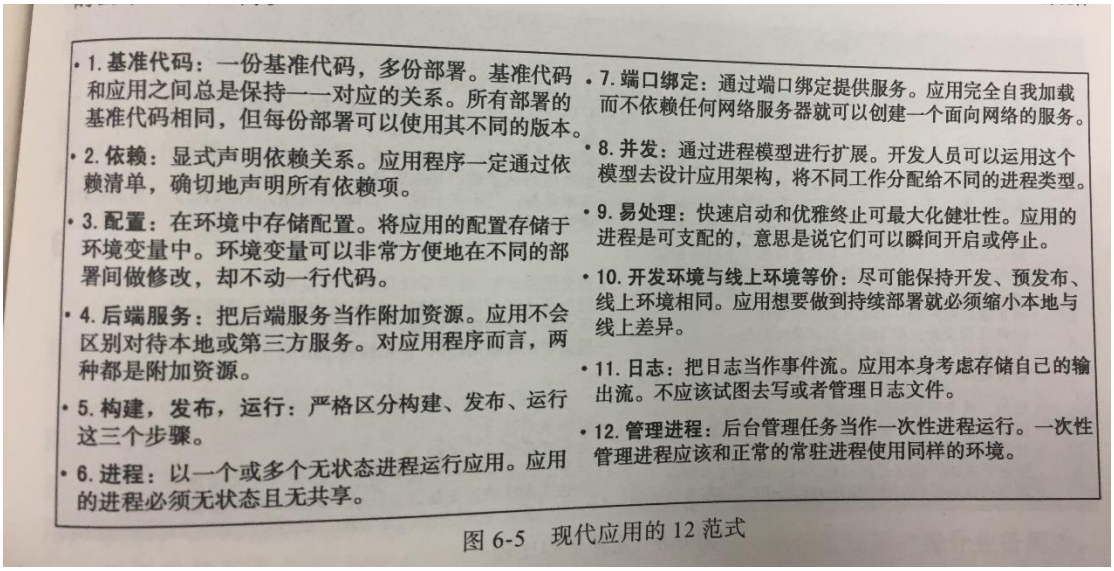
SOA 架构： 面向服务的体系结构， 构造分布式计算的应用程序的方法， 将应用程序功能作为服务发送给最终用户或其他服务。 强调的是系统服务化和系统之间用良好定义的借口进行交互

课本上的优劣比较

	优势	劣势
单体	• 人所众知：传统工具、应用和脚本都是这种结构 • IDE友好：Eclipse、IntelliJ等开发工具多 • 便于共享：单个归档文件包含所有功能，便于共享 • 易于测试：单体应用部署后，服务或特性即可展现，没有额外的依赖，测试可以立刻开始 • 容易部署：只需将单个归档文件复制到单个目录下	• 不够灵活：任何细修改需要将整个应用重新构建/部署，这降低了团队的灵活性和功能交付频率 • 妨碍持续交付：单体应用比较大时，构建时间比较长、不利于频繁部署，阻碍持续交付 • 受技术栈限制：必须使用同一语言/工具、存储及消息，无法根据具体的场景做出其他选择 • 技术债务：“不坏不修”（Not broken，don't fix），系统设计/代码难以修改，耦合性高
SOA	• 服务重用性：通过编排基本服务以用于不同的场景 • 易维护性：单个服务的规模变小，维护相对容易 • 高可靠性：使用消息机制及异步机制，提高了可靠性 • 高扩展和可用性：分布式系统的特性 • 软件质量提升：单个服务的复杂度降低 • 平台无关：可以集成不同的系统 • 提升效率：服务重用、降低复杂性，提升了开发效率	• 过分使用ESB：使得系统集成过于复杂 • 使用基于SOAP协议的WS：使得通信的额外开销很大 • 使用形式化的方式管理：增加了服务管理的复杂度 • 需要使用可靠的ESB：初始投资比较高
微服务	• 简单：单个服务简单，只关注于一个业务功能 • 团队独立性：每个微服务可以由不同的团队独立开发 • 松耦合：微服务是松散耦合的 • 平台无关性：微服务可以用不同的语言/工具开发 • 通信协议轻量级：使用REST或者RPC进行服务间通信	• 运维成本过高 • 分布式系统的复杂性 • 异步、消息与并行方式使得系统的开发门槛增加 • 分布式系统的复杂性也会让系统的测试变得复杂

图 6-6　几种架构比较

3. 什么是 12 范式？



- 1. 基准代码：一份基准代码，多份部署。基准代码和应用之间总是保持一一对应的关系。所有部署的基准代码相同，但每份部署可以使用其不同的版本。
- 2. 依赖：显式声明依赖关系。应用程序一定通过依赖清单，确切地声明所有依赖项。
- 3. 配置：在环境中存储配置。将应用的配置存储于环境变量中。环境变量可以非常方便地在不同的部署间做修改，却不动一行代码。
- 4. 后端服务：把后端服务当作附加资源。应用不会区别对待本地或第三方服务。对应用程序而言，两种都是附加资源。
- 5. 构建，发布，运行：严格区分构建、发布、运行这三个步骤。
- 6. 进程：以一个或多个无状态进程运行应用。应用的进程必须无状态且无共享。
- 7. 端口绑定：通过端口绑定提供服务。应用完全自我加载而不依赖任何网络服务器就可以创建一个面向网络的服务。
- 8. 并发：通过进程模型进行扩展。开发人员可以运用这个模型去设计应用架构，将不同工作分配给不同的进程类型。
- 9. 易处理：快速启动和优雅终止可最大化健壮性。应用的进程是可支配的，意思是说它们可以瞬间开启或停止。
- 10. 开发环境与线上环境等价：尽可能保持开发、预发布、线上环境相同。应用想要做到持续部署就必须缩小本地与线上差异。
- 11. 日志：把日志当作事件流。应用本身考虑存储自己的输出流。不应该试图去写或者管理日志文件。
- 12. 管理进程：后台管理任务当作一次性进程运行。一次性管理进程应该和正常的常驻进程使用同样的环境。

图 6-5　现代应用的 12 范式

4. 微服务架构的特征有哪些？

通过服务组件化
围绕业务能力组织
是产品不是项目
智能端口和哑通道
去中心化治理
去中心化数据管理
基础设施自动化
为失效设计
进化式设计

5. 微服务的核心模式有哪些？

服务注册与发现： 服务提供者、 服务消费者和服务发现组件
配置中心
API 网关
优势：
客户端无法察觉应用程序是如何被拆分成多项微服务的
客户端不受服务实例的位置的影响
为每套客户端提供最优 API
降低请求/往返次数
将从客户端调用多项服务的逻辑转换为从 API 网关处调用， 从而简化整个客户端
熔断器
状态： 闭合状态、 断开状态、 半断开状态

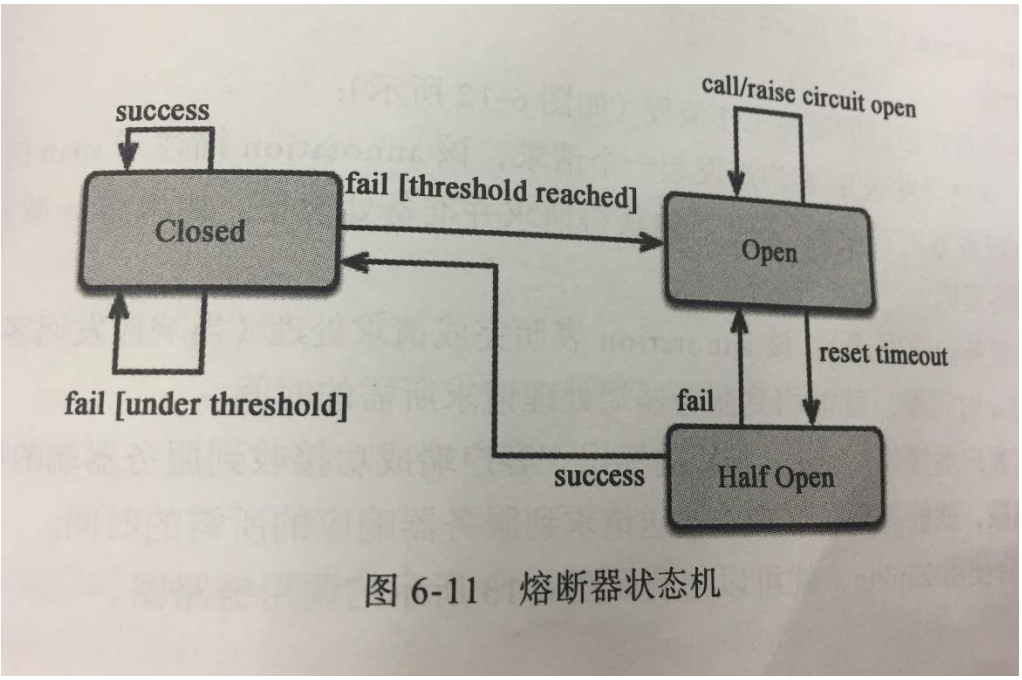


图 6-11　熔断器状态机

分布式追踪

第七章

1.Linux namespace 有几种，分别是什么？

6 种

1) UTS namespace：提供主机名与域名的隔离
2) IPC namespace：提供信号量、消息队列和共享内存的隔离
3) PID namespace：对进程 PID 进行重新编号，提供对进程编号的隔离
4) network namespace：提供对网络资源的隔离，包括网络设备，IPv4 和 IPv6 协议栈，路由表，防火墙等
5) mount namespace：通过隔离文件系统挂载点对隔离文件系统提供支持
6) user namespace：隔离了安全相关的标识符和属性，包括用户 ID，用户组 ID，root 目录，key 和特殊权限

2.Dockerfile 的 CMD 与 Entrypoint 有什么异同？

同：

1) 都是指定容器启动的命令和相关参数，都可以指定 shell 或 exec 函数调用的方式执行命令；
2) 每个 Dockerfile 只能有一条 CMD/ENTRYPOINT 命令。如果指定了多条命令，只有最后一条会被执行。

异：

1) CMD 指令指定的容器启动时的命令可以被 docker run 指定的命令覆盖，ENTRYPOINT 在运行时也可以替代，但更繁琐，需要通过 docker run 的参数 --entrypoint 来指定
2) CMD 指令可以作为 ENTRYPOINT 指令的默认参数，而且可以被 docker run 指定的参数覆盖；

3.编写一个运行 Java 应用的 Dockfile

略

4.Docker 网络模式有几种，分别是什么

4 种

1) host 模式：和宿主机共用一个 network namespace
2) container 模式：和已存在的容器共享一个 network namespace
3) none 模式：容器拥有自己的 network namespace，但并进行任何的网络配置
4) bridge 模式：默认的网络设置，为每一个容器分配 network namespace ，进行网络设置，并将一个主机上的 Docker 容器连接到一个虚拟网桥上。

5.Doker Storage Driver 有几种？分别是什么？使用场景有哪些

6 种

1) AUFS：被使用在宿主机 fs 格式为：ext4 或 xfs 上
2) Overlay/Overlay2：被使用在宿主机 fs 格式为：ext4 或 xfs 上
3) Btrfs：被使用在宿主机 fs 格式为：btrfs 上
4) Device mapper：被使用在宿主机 fs 格式为：direct-lvm 上
5) VFS：被使用在宿主机 fs 格式为：debugging 上
6) ZFS：被使用在宿主机 fs 格式为：zfs 上

6.Doker 编排引擎有几种，分别是什么？

4 种

1) Docker Swarm
2) Kubernetes
3) Marathon
4) Nomad

补：

1.Dockerfile、Docker 镜像和 Docker 容器的关系

Dockerfile 是软件的原材料，Docker 镜像是 软件的交付品，Docker 容器是软件的运行态，从应用软件的角度看，分别代表软件的三个阶段，Dockerfile 面向开发，Docker 镜像成为交付标准，Docker 容器则涉及到部署和运维，三者缺一不可。

简单来说，Dockerfile 构建出 Docker 镜像，通过 Docker 镜像运行 Docker 容器。

2.Dockerfile 常用指令

FROM ：指定基础镜像
RUN：执行命令行（shell/exec 格式）
COPY：复制文件/目录
ADD：增加文件/目录+自动解压
CMD：指定默认的容器主进程的启动命令的
ENTRYPOINT: 同 CMD 一样，指定容器启动程序和参数
EXPOSE：声明运行时容器提供服务端口

3.CNM - Container Network Model - 容器网络模型 ： 一套容器网络接口的标准

第八章

8.1 DevOps 流程涵盖哪些节点？

需求 -> 计划 -> 编码 -> 构建 -> 测试 -> 发布 -> 运营

8.2 容器技术对于 DevOps 实践有哪些帮助？

容器技术最重要的一点是标准化，它的理念是"build, ship, run"。即构建出来的镜像包含了依赖的第三方软件、操作系统以及代码构建后的制品，可以在任意安装了相同版本容器服务的操作系统上运行，并具备相同的行为。

现代化的分布式应用架构带来了很高的复杂性，从而增加了交付和运维的难度，因此容器技术，包括编排、调度等能力，恰好能够很好的满足分布式应用交付的需求。

8.3 什么是持续交付流水线？

持续交付流水线是一组能够帮助开发团队极大提高软件交付速度金额质量的模式和最佳时间组成。

持续交付在以下方面提供帮助：

- 尽可能快的交付软件，尽可能早的将有价值的新功能运用于生产；
- 提高软件质量、系统正常运行时间和稳定性；
- 降低发布风险，避免同时在测试和生产环境部署失败；
- 减少浪费，提高开发和交付过程的效率；
- 使软甲 始终处于生成就绪状态，以便可以随时部署。

8.4 什么是持续集成？

持续集成 (CI) 是 DevOps 流程中的一个重要组成部分，目的是对开发团队的代码进行集成，包括代码的构建、单元测试、与集成测试的执行，以及生产执行结果的报表等。

8.5 什么是持续部署？

持续部署（CD）与持续集成通常是紧密联系在一起的。CD 过程通过在流水线中定义的步骤将 CI 过程所产生的结果部署集成、预发布乃至生产环境中。

CD 过程是“持续”的，一旦有代码签入源控制系统，后续过程就会自动进行测试、构建并部署。

优点：开发者可以迅速收到故障与 bug 的通知，形成快速的反馈循环；客户也将更快的使用到新特性。

8.6 一个典型的 Java 应用的持续集成/持续部署包含哪些步骤？

持续集成步骤：

- 代码提交触发持续集成流程
- 代码构建（例如 Maven 构建）
- 单元测试（例如 MockServer 这样的 Mock 服务器）
- 测试结果及代码检查
- 结果通过邮件反馈到相应同事

持续部署步骤：

- 分支合并触发持续部署流程
- Maven 编译及打包
- 构建容器镜像
- 发布容器镜像
- 部署容器
- 集成、接口测试
- 反馈到代码仓库

第九章

9.1 常见的 DevOps 工具集

协同开发工具：

- JIRA
- Kanboard

- Rally

持续集成工具：
- Jenkins
- Bamboo
- Travis CI

版本管理工具：
- Git
- GitHub
- GitLab
- Subversion
- Mercurial

编译工具：
- Ant
- Maven
- Gradle
- MSBuild

配置管理工具：
- Chef
- Puppet
- Ansible

测试工具：
- JUnit
- Selenium
- Cucumber
- FitNesse

监控工具：
- Nagios
- Zabbix