了解微服务和Devops

微服务

定义

一种软件开发技术- 面向服务的体系结构（SOA）架构样式的一种变体，将应用程序构造为一组松散耦合的服务。在微服务体系结构中，服务是细粒度的，协议是轻量级的。

微服务（或微服务架构）是一种云原生架构方法，其中单个应用程序由许多松散耦合且可独立部署的较小组件或服务组成。这些服务通常

● 有自己的堆栈，包括数据库和数据模型；

● 通过REST API，事件流和消息代理的组合相互通信；

● 和它们是按业务能力组织的，分隔服务的线通常称为有界上下文。

尽管有关微服务的许多讨论都围绕体系结构定义和特征展开，但它们的价值可以通过相当简单的业务和组织收益更普遍地理解：

● 可以更轻松地更新代码。

● 团队可以为不同的组件使用不同的堆栈。

● 组件可以彼此独立地进行缩放，从而减少了因必须缩放整个应用程序而产生的浪费和成本，因为单个功能可能面临过多的负载。

特点

- 单一职责的。一个微服务应该都是单一职责的，这才是“微”的体现，一个微服务解决一个业务问题（注意是一个业务问题而不是一个接口）。
- 面向服务的。将自己的业务能力封装并对外提供服务，这是继承SOA的核心思想，一个微服务本身也可能使用到其它微服务的能力。
- 微服务的最重要的单一特征是**可独立部署**

可独立部署

微服务向组织承诺了解决方案，以解决因细微变化而引起的内在挫折，这需要花费大量时间。在计算机科学中可以看到或理解更好地促进速度和敏捷性的方法的价值。

但是，速度并不是以这种方式设计服务的唯一价值。一种常见的新兴组织模型是将跨职能的团队聚集在业务问题，服务或产品上。微服务模型非常适合这种趋势，因为它使组织能够围绕一项服务或一组服务创建跨职能的小型团队，并使它们以敏捷的方式运作。

最后，服务的小规模加上清晰的边界和沟通模式，使新团队成员更容易理解代码库并快速做出贡献，这在速度和员工士气方面均具有明显的好处。

Devops

DevOps（Development和Operations的组合词）是一组过程、方法与系统的统称，用于促进开发（应用程序/软件工程）、技术运营和质量保障（QA）部门之间的沟通、协作与整合。

它是一种重视“软件开发人员（Dev）”和“IT运维技术人员（Ops）”之间沟通合作的文化、运动或惯例。透过自动化“软件交付”和“架构变更”的流程，来使得构建、测试、发布软件能够更加地快捷、频繁和可靠。

它的出现是由于软件行业日益清晰地认识到：为了按时交付软件产品和服务，开发和运维工作必须紧密合作。

核心：把开发、测试、运维统一集成，可以使得同一组人自动化完成。

简介

DevOps 是一个完整的面向IT运维的工作流，以 IT 自动化以及持续集成（CI）、持续部署（CD）为基础，来优化程式开发、测试、系统运维等所有环节。

DevOps一词的来自于Development和Operations的组合，突出重视软件开发人员和运维人员的沟通合作，通过自动化流程来使得软件构建、测试、发布更加快捷、频繁和可靠。

DevOps是为了填补开发端和运维端之间的信息鸿沟，改善团队之间的协作关系。不过需要澄清的一点是，从开发到运维，中间还有测试环节**。DevOps其实包含了三个部分：开发、测试和运维。**

DevOps希望做到的是**软件产品交付过程中IT工具链的打通**，使得各个团队减少时间损耗，更加高效地协同工作。

工具

- 代码管理（SCM）：**GitHub**、GitLab、BitBucket、SubVersion
- 构建工具：**Ant**、Gradle、**maven**
- 自动部署：Capistrano、CodeDeploy
- 持续集成（CI）：Bamboo、Hudson、Jenkins
- 配置管理：Ansible、Chef、Puppet、SaltStack、ScriptRock GuardRail
- 容器：**Docker**、LXC、第三方厂商如AWS
- 编排：Kubernetes、Core、Apache Mesos、DC/OS
- 服务注册与发现：**Zookeeper**、etcd、Consul
- 脚本语言：python、ruby、shell
- 日志管理：ELK、Logentries
- 系统监控：Datadog、Graphite、Icinga、Nagios
- 性能监控：AppDynamics、New Relic、Splunk
- 压力测试：JMeter、Blaze Meter、loader.io
- 预警：PagerDuty、pingdom、厂商自带如AWS SNS
- HTTP加速器：Varnish
- 消息总线：ActiveMQ、SQS
- 应用服务器：Tomcat、JBoss
- Web服务器：Apache、Nginx、IIS
- 数据库：MySQL、Oracle、PostgreSQL等关系型数据库；cassandra、mongoDB、redis等NoSQL数据库
- 项目管理（PM）：Jira、Asana、Taiga、Trello、Basecamp、Pivotal Tracker

微服务和Devops配合使用

把一个大的单体应用拆分成多个微服务之后，每个服务都需要独立进行开发、测试和运维。

但是这样，业务的开发人员不仅需要负责业务代码的开发，还需要负责业务的测试以及上线发布等全生命周期，真正做到掌控服务全流程。

而DevOps能集开发、测试和运维三者角色于一体。实现DevOps，就可以在开发完成代码开发后，能自动进行测试，测试通过后，能自动发布到线上。对应的这两个过程就是CI和CD

web1.0-web4.0发展历程

Web1.0

> Web1.0–信息共享。虽然人们为着信息共享已经奋斗了很多年，但直到Web技术的出现并逐步完善之今，信息共享也还远未令人满意。但比起之前的其它技术，如ftp等，自描述性赋予了Web系统强大的生命力，使得Web成为信息共享的第一设施。
>
> Web1.0的主要协议是HTTP, HTML 和 URI
>
> Web1.0以静态、单向阅读为主，网站内信息可以直接和其他网站信息进行交互，能通过第三方信息平台同时对多家网站信息进行整合使用。
>
> Web1.0 的主要特点在于用户通过浏览器获取信息。
>
> **Web 1.0 是单向的。**大公司创造了供人们阅读的内容，几乎没有交互，用户只能读取数据，只有内容创造者才能编辑和创作内容。所以，就像在图书馆看书一样，用户没有太多权利，他们只是消费者。

Web2.0

> Web2.0–信息共建。直到Web1.0时代，信息也还都是单向的，由话语权集团发出。普通百姓只有听的份儿，而Web2.0则赋予了普通百姓一样的话语权，意识表达空前活跃，特别是在意识形态禁锢的社会里。如此必然导致网络信息的泛滥：陷阱病毒成灾，如今杀毒软件倒成了计算机第一应用了；垃圾信息遍野，如果找到适合于自己的信息，就成了网民的需要，因此搜索引擎崛起。但搜索引擎并不能杜绝陷阱病毒，也不能区分垃圾信息，更不能系统化Web信息，因此技术探索就成为必然。
>
> 更注重用户的交互作用，用户既是浏览者，也是内容的制造者，在模式上有单纯的“读”向“写”以及共同建设发展。
>
> 使用了包括html、css、JavaScript、.NET、PHP、Java等技术
>
> Web2.0 则更注重用户的交互作用，用户既是网站内容的浏览者，也是网站内容的制造者。
>
> Web2.0包含了我们经常使用到的服务，例如博客、播客、维基、P2P下载、社区、分享服务等等。
>
> 以分享为特征的实时网络，用户在互联网上拥有自己的数据，并能在不同的网站上使用。

Web3.0

> Web3.0–知识传承。计算机是人类的意识外化，其每一点进步，都必然聚合了更多人的智慧。集聚人类智慧为人类共享，是计算机科学技术的内在本质。Web 3.0里，我们不仅要消灭陷阱病毒，踢出垃圾信息，更要有序化系统化整个Web世界，以全Web资源为基础建设出一座“Web图书馆”来，实现人类自身的“知识传承”。我的知识界系统产品，就是这样一个实现人类自身知识传承的Web3.0系统。即时性是其主要特性，因此即时通信（iM）系统是知识界的技术平台。
>
> 在网络搜索方面，Web 3.0引入个人信息偏好处理系统和个性化搜索引擎，对个体用户进行特征分析，同时也对整个互联网的搜索习惯进行整理，归类，最终得出更适合网民需求的搜索平台，实现了快捷、准确、高效的搜索，用户可以可以在极短时间内找到自己需要的信息资料，节省了时间和精力。
>
> 适合多种终端平台，实现信息服务的普适性

> 良好的人性化用户体验，以及基础性的个性化配置
>
> 引入个人信息偏好处理系统和个性化搜索引擎
>
> Web 3.0 是更加以用户为中心、专注于使事情更加人性化的、透明的、安全的互联网。
>
> 当你在 Google 上搜索特定产品后，你 Facebook 网页上的广告也会发生变化，一切都是相互关联的。因此，未来的营销策略将不再采用大规模营销技术，而是以人为本。

Web4.0

> Web4.0–知识分配。在Web3.0里，人类可以随心所欲地获取各种知识，当然这些知识都是先人们即时贡献出来的。这里的即时性，指的就是学堂里老师教学生的即时性。从Web3.0开始，网络就具备了即时特性。但人们并不知道自己应该获取怎样的知识，即自己适合于学习哪些知识。比如一个10岁的孩子想在20岁的时候成为核物理学家，那么他应该怎样学习知识呢？这些问题就是Web4.0的核心–知识分配系统所要解决的问题了。通过连接情报，在任何时候、任何地方能够提供任何需要的东西。目前的设想为带共生网络(Symbiotic Web)、大规模网络(Massive Web)、同步网络(Simultaneously Web)和智慧网络(Intelligent Web)等特征的下一代互联网络。
>
> web 4.0的含义关键在于它在任何时候、任何地方能够提供给你任何需要的东西。

云平台架构

云平台架构概述：

首先应明白建立云平台的目的，与传统的服务器相比，云平台可以将物理资源虚拟化为虚拟机资源池，灵活调用软硬件资源，实现对用户的按需访问。而且在运行过程中根据用户并发量不同，实时迁移虚拟机资源，一方面保证提供高质量服务，另一方面最小化资源成本，提高CPU、内存等利用率。

该架构主要分为4层，从底层到上层分别是资源层、虚拟层、中间件层、应用层。以下从底到上分别说明各层的构造和作用。

资源层：

由**服务器集群**组成。传统服务器要想提供高质量服务，需要性能特别好的服务器（内存高，CPU快，磁盘空间大等），价格昂贵。而服务器集群可以使用以前性能不太好的服务器，利用分布式处理技术，依然可以提供可靠服务，节省费用。

物理资源指的是物理设备，如服务器等。

服务器服务指的是操作系统的环境，如linux集群等。

网络服务指的是提供的网络处理能力，如防火墙，VLAN，负载等。

存储服务为用户提供存储能力。

虚拟层：

有了物理机集群后，我们需要在物理机上建立**虚拟机。**建立虚拟机的目的是为了最小化资源成本（最大化资源利用率）。试想一下某台物理机有16G内存，当某段时间连续有小任务量的应用需要处理时，物理机的内存利用率会很低，所以为最大化资源利用率，可以在物理机上独立开辟几个虚拟机，每台虚拟机相当于一个小型服务器，依然可以处理应用请求。我们采用KVM来给每一台虚拟机分配适量的内存、CPU、网络带宽和磁盘，形成虚拟机池。(KVM就是虚拟机监控器hypervisor，可以给虚拟机分配资源，当然也可以开关虚拟机。同样还有XEN和OVM)。

中间件层：

中间件是**独立的系统软件服务程序**，分布式应用软件借助这种软件在不同的技术之间共享资源，中间件位于客户机服务器的操作系统之上，管理计算资源和网络通信。

这层应该是云平台的核心层，主要功能为：**对虚拟机池资源状态进行监测、预警、优化决策。**①资源监测：实时监测当前各台虚拟机CPU、内存等使用情况，当然也监测用户应用请求，以便根据应用规模大小进行决策。②预警：防患于未然，根据当前虚拟机资源使用情况预测下一秒用户请求量，以便做出相应资源调整，防止宕机。比如CPU使用率上限为70%，所以当预测下一秒达到该触发点时，应有相应响应。当然，触发阈值应该有更科学的设定。③优化决策：预警之后，虚拟机要进行资源调度(迁移或伸缩)，采用何种调度策略，才能保证服务和资源利用率是研究重点。由于该层需要对应用进行响应处理，所以需要在虚拟机上搭建操作系统，文件存储系统，以及服务器，当然最应该有负载均衡系统Nginx，其实现中间件层功能，相当于网络中的路由器不处理数据，只进行数据转发，数据处理交有虚拟机上的tomcat服务器执行。（也相当于hadoop中的Namenode，其他虚拟机相当于datanode）。

应用层：

给用户提供**可视化界面**，应用若为存储：比如百度云会给用户提供交互界面，建立文件夹，进行数据存储，在线播放视频等界面，供用户选择操作。应用若为租用服务器：界面应该有租用的服务器资源状态。

过程工具

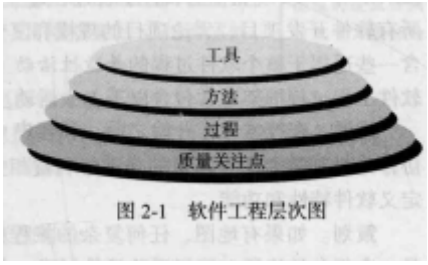


图 2-1　软件工程层次图

对图2.1的理解

图2.1即软件工程层次图，是在定义的基础上对软件工程的抽象。由于任何工程方法（包括软件工程）必须构建在质量承诺的基础之上，因此支持软件工程的根基在于质量关注点。为了提高软件质量，作为基础的过程层将各个技术层次结合在一起，使得合理、及时地开发计算机软件称为可能。方法则为构建软件提供技术上的解决方法，软件工程工具则是方法技术的工具化，为过程和方法提供自动化或半自动化的支持。在软件开发过程中用到的各种各样的工具统称为计算机辅助软件工程(CASE)。

过程工具

沟通

邮件outlook/实时通讯工具qq wechat

策划

软件策划和软件项目策划是一个意思。它的输入是软件《合同》/《立项建议书》、《任务书》、《用户需求报告》，输出是《软件开发计划》(包括质量保证计划、软件配置管理计划、测试计划、评审计划)。

用于估算软件项目规模的方法有Line Of Code（LOC）、Function Point（FP）；用于估算软件开发成本的模型有IBM、Putnam、COCOMO

建模

各种uml工具

Rational Rose EclipseUML

构建

构建工具——调查结果

Apache Maven ——主要用于构建Java项目的自动化工具。

Hudson ——用Java编写的持续集成（CI）工具。

Jenkins ——用Java编写的一个开源持续集成工具。项目是在和Oracle发生争执后的来自于Hudson 的分支。

Gradle ——一个开源的自动化构建系统，建立在Apache Ant和Maven Apache概念的基础上，并引入了基于Groovy的特定领域语言（DSL），而不是使用Apache Maven宣布的项目配置XML形式。

Apache Ant ——用于自动化软件构建过程的软件工具，源于2000年初的的Apache Tomcat项目。

SBT ——用于Scala和Java项目的开源构建工具，类似于Java的Maven和Ant。

Atlassian Bamboo ——持续集成和交付工具，它将自动化构建、测试和发布捆绑到单个流程中。

TeamCity ——来自于JetBrains的一个基于Java构建的管理和持续集成服务器。

Grape ——嵌入到Groovy的JAR依赖项管理器。

Ivy ——Apache Ant项目的一个子项目，一个可传递的依赖项管理器。

Leiningen ——一个自动化构建和依赖性管理工具，用于使用Clojure编程语言写的软件项目。

部署

deploy-tool

使用XP或Scrum开发的项目

XP极限编程是一门针对业务和软件开发的规则，它的作用在于将两者的力量集中在共同的、可以达到的目标上。它是以符合客户需要的软件为目标而产生的一种方法论，XP使开发者能够更有效的响应客户的需求变化，哪怕是在软件生命周期的后期。它强调，软件开发是人与人合作进行的过程，因此成功的软件开发过程应该充分利用人的优势，而弱化人的缺点，突出了人在软件开发过程中的作用。极端编程属于轻量级的方法，认为文档、架构不如直接编程来的直接。

从长远看，早期发现错误以及降低复杂度可以节约成本。极限编程强调我们将任务/系统细分为可以在较短周期解决的一个个子任务/模块，并且强调测试、代码质量和及早发现问题。通常，通过一个个短小的迭代周期，我们就可以获得一个个阶段性的进展，并且可以及时形成一个版本供用户参考，以便及时对用户可能的需求变更作出响应。

XP用“沟通、简单、反馈、尊重和勇气”来减轻开发压力和包袱；无论是术语命名、专著叙述内容和方式、过程要求，都可以从中感受到轻松愉快和主动奋发的态度和气氛。这是一种帮助理解和更容易激发人的潜力的手段。XP用自己的实践，在一定范围内成功地打破了软件工程“必须重量”才能成功的传统观念。

XP四大价值观：

· 沟通： 项目中发现的问题往往是由于开发人员与设计人员、 设计人员与客户之间的沟通不畅造成的，因此， 在 XP 项目中没有沟通是不可能的。
· 简单： XP 认为应该尽量保持代码的简单， 只要它能工作就可以。 Kent Beck 指出与其实现一个复杂的的系统， 不如设计一个能够满足目 前需要的、 简单的系统， 因为你所考虑的情况可能永远都不会发生。
· 反馈： 尽快获得用户的反馈， 并且越详细越好， 使得开发人员能够保证自 己的成果符合用户的需要。
· 勇气： 这是最重要的核心价值。 因为 XP 强调要"拥抱变化"， 因此对于用户的反馈，要勇于对自己的代码进行修改， 丢掉坏的代码。

五大原则：

快速反馈、简单性假设、逐步修改、提倡更改、优质工作。

XP弱化针对未来需求的设计，非常注重当前的简化，开发人员只注重眼前需求而依赖重构来适应需求的变动。

XP适合规模小、进度紧、需求变化大、质量要求严的项目。它希望以最高的效率和质量来解决用户目前的问题，以最大的灵活性和最小的代价来满足用户未来的需求，XP 在平衡短期和长期利益之间做了巧妙的选择。

某公司在一个项目中启用 XP ，该项目是为一家国际知名手机生产厂商的合作伙伴提供手机配件定购、申请、 回收等服务， 项目的情况如下表所示：

条 目	描 述
项目名称	合作伙伴管理系统
处理工作流程	9个
项目周期	43个工作日
项目金额	25万
项目小組人员	5人，其中资深顾问2名

从上表中可以看出，该项目是一个小型项目，在项目执行过程中，提交了一个"用户需求变更"，该变更对于项目 周期的影响为 6 个工作日。项目实施后，在用户接收测试中， 只提交了 2 个 BUG， 而且在提交当天就得到了解决。因此认为XP 在该项目中的实施有效地保证了项目质量和项目周期。

总结

XP 是在总结了很多项目实践的过程中发展起来的软件开发过程，**基础是面向对象方法，重视代码、 文档的最小化和设计的简化，采用动态适应变化的演进式迭代周期等等。**
执行 XP 的理想情况应该是： 在保持组织既有的开发过程和生命周期模型的情况下， 根据应用类型、项目特点和组织文化，借鉴、采取个别对项目有效的 XP 做法，在项目的实施过程中， XP 对于执行者的要求是比较高的， 因为它要求开发团队必须具备熟练的代码设计技能和严格的测试保障技术， 了解面向对象和模式， 掌握了重构和 OO 测试技术， 习惯了测试先行的开发方式等等。因此， 对于目前国内的软件开发组织来说， 应该首先加强对于软件开发过程化和系统架构设计的掌握， 然后， 才是利用 XP 等敏捷方法来完善软件开发过程。

Scrum

Scrum是**迭代式增量软件开发过程，通常用于敏捷软件开发。**Scrum包括了一系列实践和预定义角色的过程骨架。Scrum中的主要角色包括同项目经理类似的Scrum主管角色负责维护过程和任务，产品负责人代表利益所有者，开发团队包括了所有开发人员。虽然Scrum是为管理软件开发项目而开发的，它同样可以用于运行软件维护团队，或者作为计划管理方法：Scrum of Scrums.

Scrum 是一个用于开发和维护复杂产品的框架，是一个增量的、迭代的开发过程。在这个框架中，整个开发过程由若干个短的迭代周期组成，一个短的迭代周期称为一个Sprint，每个Sprint的长度是2到4周。

模式

过程模式

每个软件团队在软件过程里都会遇到很多问题。针对这些问题，如果软件团队能够得到已有的经过验证的解决方案，将有助于他们快速地分析和解决问题。

过程模式（process pattern）描述了软件工程工作中遇到的过程相关的问题，明确了问题环境并给出了针对该问题的一种或几种可证明的解决方案。通俗地讲，**过程模式提供了一个模板——一种在软件过程的背景下统一描述问题解决方案的方法。**通过模式组合，软件团队可以解决问题并定义最符合项目需求的开发过程。

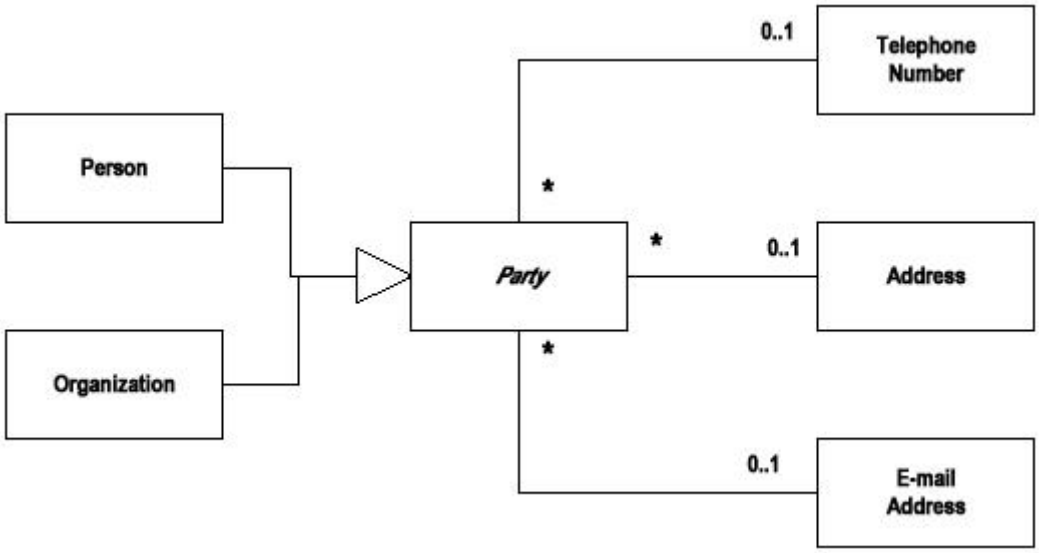
分析模式

分析模式来自概念商业模型的模式；他们提供来自贸易、测量、记帐、组织关系等多个问题域的关键抽象。这些模式之所以是概念性的因为他们关注的是人们对业务的思考和认识，而不是计算机系统的设计方法。

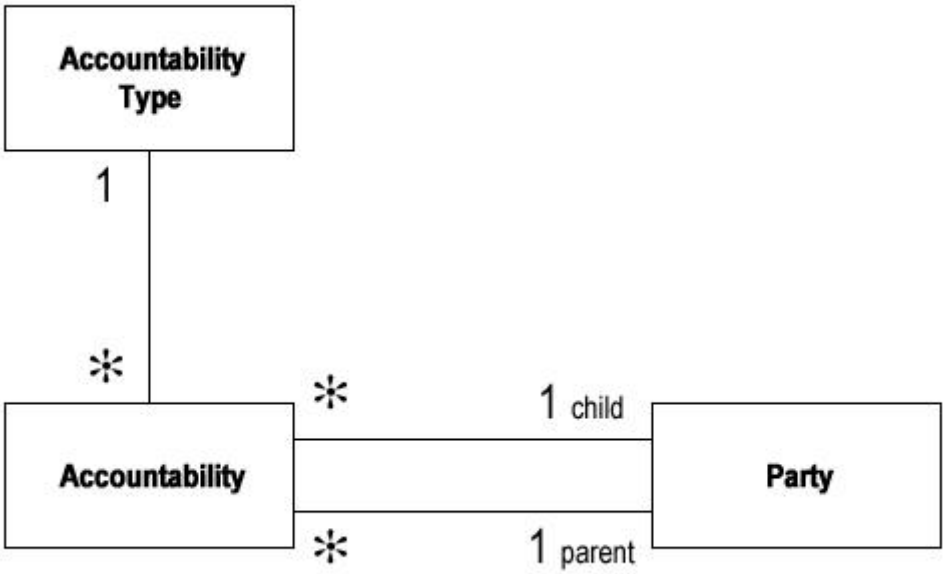
责任模式

最直观的例子是组织结构，上级组织与下层组织之间的关系，从具体层面讲是一种所属关系，Martin将它抽象为一种责任关系。人和组织之间的所属关系是责任关系；管理者与部署之间的关系是责任关系；部门与部门负责人、经理之间的关系是责任关系。对象类型与派生类型之间并不一定是继承关系、各种派生类型也不必是在同一个数据库表中，仅仅是实现某个接口而已。

Party模式，在进行系统分析和概念模型设计的时候，经常发现人和各种各样的组织有着同样的行为，例如，固定电话的计费可能是针对个人，也可能是一个单位；需要各种服务的时候，你可能求助于一个服务公司，或者服务公司一个特定的业务员。总之，因为人（Person）和组织（Organization）表现上的一致性，从中抽象出Party，作为Person 和Organization的抽象父类。



Party: 参与者，责任关系中任何一方都属于参与者
Accountability: 责任关系，从数据模型来理解就是两个或多个参与者之间的对应关系，以及相关属性
Party是Person和Organization的抽象父类，因此正式形成责任（Accountability）模式。



分析SafeHome家居安全功能的用例图

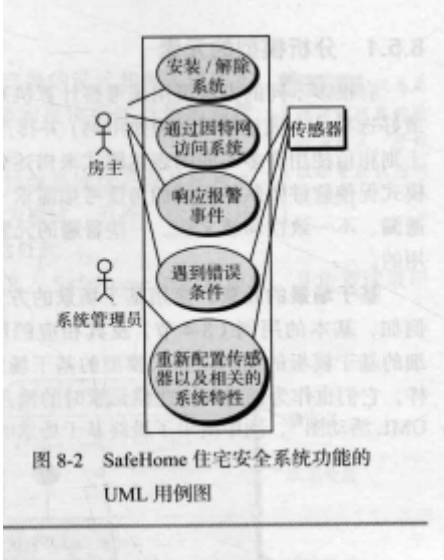


图 8-2 SafeHome 住宅安全系统功能的 UML 用例图

参与者

- Homeowner房主（即用户）
- Systemadministrator系统管理员（很可能就是房主，但扮演不同的角色）
- sensors传感器（附属于系统的设备）

用例

- 安装/解除系统
- 通过英特网访问系统
- 响应报警事件
- 遇到错误条件
- 重新配置传感器以及相关的系统特性

学生管理系统中课程考试时间安排

共有6门课程（A，B，C，D，E，F），对于一名学生，如果选择了若干门课程，则这些课程就存在关系R。根据R建立一个6 * 6的矩阵，求得其中每个元素的等价类再依次与这6门课程取交集，最后即为结果。

具体操作
根据学生选课情况，存在如下关系：
(A, B), (B, E), (A, E), (C, D), (C, E), (E, F), (C, F), (A, D), (D, F), (A, F), (B, F)

使用邻接矩阵存储信息，通过分组算法算出在同一段时间内可以考试的不同的几门课程，有了这些分组就可以很好的安排考试，按照分组，在同一组的数门课程可以在同一段时间在不同的教室参加考试。

```
A  B  C D  E  F

1  1  0  1  1  1  A
1  1  0  0  1  1  B
0  0  1  1  1  1  C
1  0  1  1  0  1  D
1  1  1  0  1  1  E
1  1  1  1  1  1  F
```

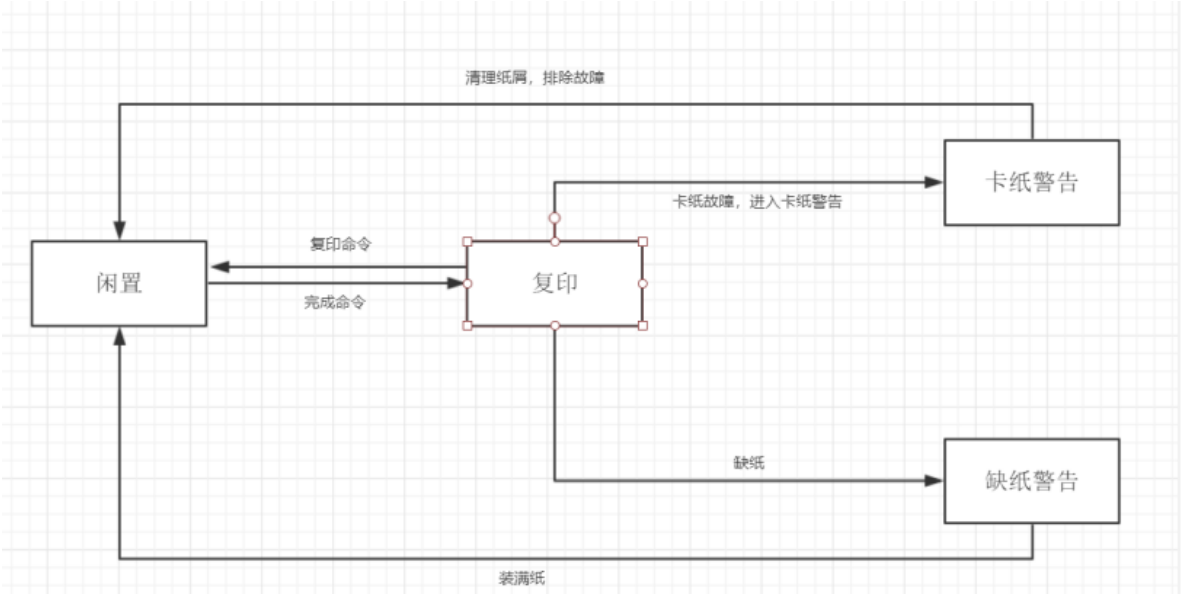
对于求等价类的算法，用伪代码描述如下：

```
根据定义求等价类，如果A和B存在X，则（A，B）为一等价类，如果A与C也存在关系X， 且B与C也存在关系X，那么(A，B，C)就是一个等价类

for (i = 1 to n ) {
    create an empty set Si
    for (j = 1 to n) {
        if (a[i][j] == 0) {
            if (Si is empty) put i into Si
            for (every element k in Si)
                if (a[j][k] == 0) put k into Si
        }
    }
    if (Si is not empty) count++
}

求得等价类为：
(A，C）
(B，D)，(B，C)
(D，E)

依次与S =（A，B，C，D，E，F）进行集合减的运算
（A  C）       --〉S =  (B，D，E，F）
（B  D）       --〉S =（E，F）
（E）--〉S =（F）
（F）          --〉S为空，结束

所以最终结果为
第一场：算法分析     计算机图形学
第二场：形式语言     模式识别
第三场：计算机网络
第四场：人工智能
```

构建打印机或复印件软件系统的状态图

复印机的工作过程大致如下：未接到复印命令时处于闲置状态，一旦接到复印命令则进入复印状态，完成一个复印命令规定的工作后又回到闲置状态，等待下一个复印命令；如果执行复印命令时发现没纸，则进入缺纸状态，发出警告，等待装纸，装满纸后进入闲置状态，准备接收复印命令；如果复印时发生卡纸故障，则进入卡纸状态，发出警告，等待维修人员来排除故障，故障排除后回到闲置状态。

状态图：

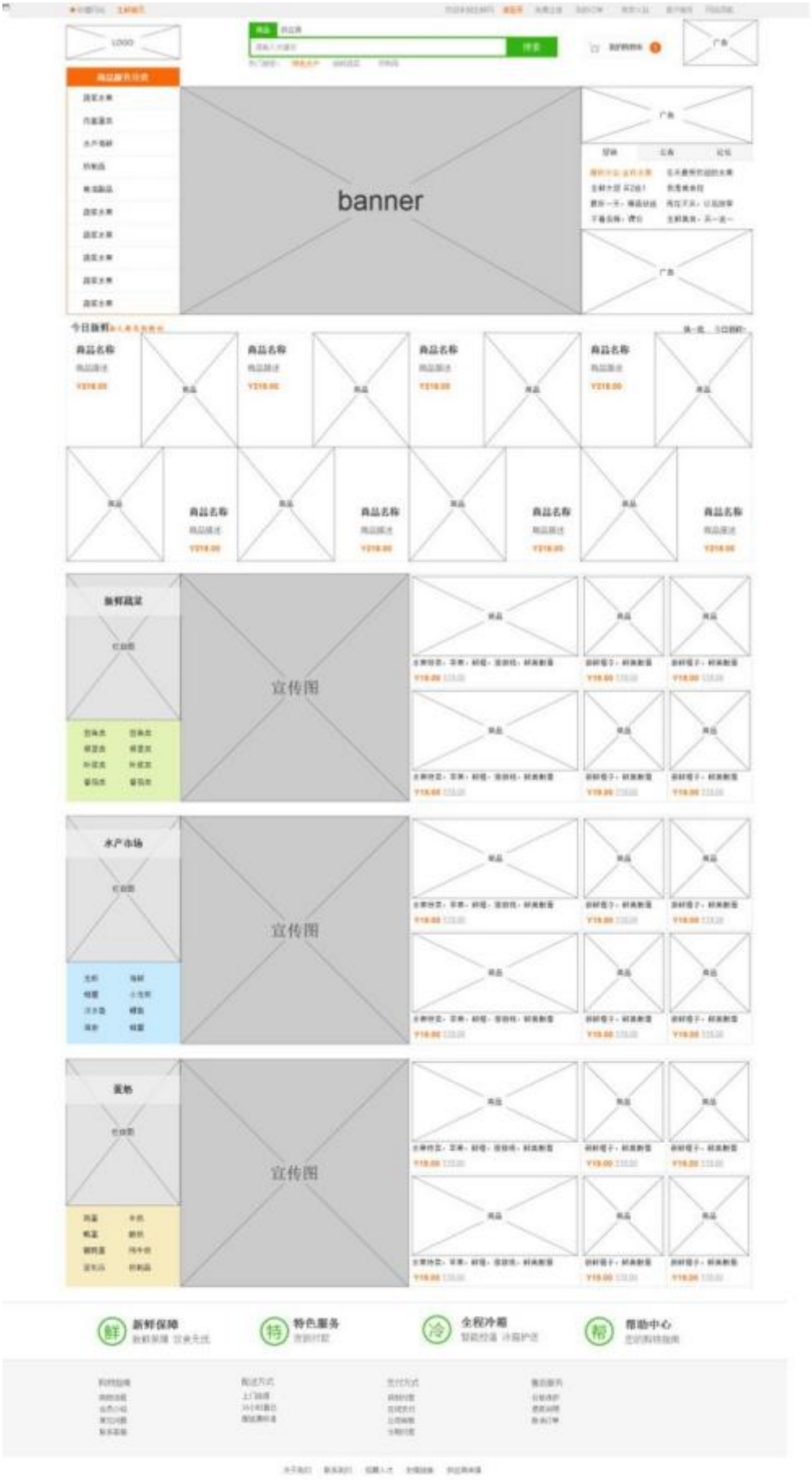


创建界面原型的工具

Axure

Axure RP是美国Axure Software Solution公司旗舰产品，是一个专业的快速原型设计工具，让负责定义需求和规格、设计功能和界面的专家能够快速创建应用软件或Web网站的线框图、流程图、原型和规格说明文档。作为专业的原型设计工具，它能快速、高效的创建原型，同时支持多人协作设计和版本控制管理。

Axure的可视化工作环境可以让你轻松快捷的以鼠标的方式创建带有注释的线框图。不用进行编程，就可以在线框图上定义简单连接和高级交互。在线框图的基础上，可以自动生成HTML原型和Word格式的规格。



1.主菜单和工具栏

执行常用操作，如文件打开、保存文件，格式化控件，自动生成原型和规格说明书等操作。

2.站点地图面板

对所设计的页面（包括线框图和流程图）进行添加、删除、重命名和组织页面层次。

3.控件面板

该面板包含线框图控件和流程图控件，另外，你还可以载入已有的部件库（*.rplib文件）创建自己的部件库。

4.模块面板

一种可以复用的特殊页面，在该面板中可进行模块的添加、删除、重命名和组织模块分类层次。

5.线框图工作区

线框图工作区也叫页面工作区，线框图工作区是你进行原型设计的主要区域，在该区域中你可以设计线框图、流程图、自定义部件、模块。

6.页面注释和交互区

添加和管理页面级的注释和交互。

7.控件交互面板

定义控件的交互，如：链接、弹出、动态显示和隐藏等。

8.控件注释面板

对控件的功能进行注释说明。