**软件工程练习**

1. **查阅资料了解微服务和Devops**

1.1 **微服务**

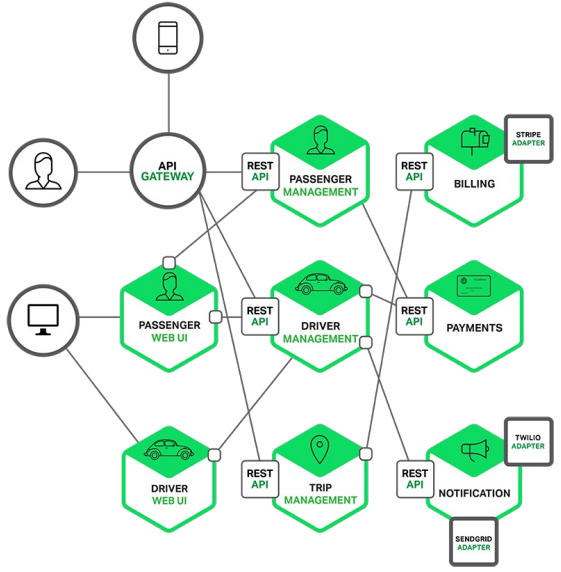
1.1.1 **微服务由来：**

微服务是当今最流行、功能强大的软件架构，为了理解微服务，我们需要先理解软件架构的演变历程。 早期的软件，所有功能都写在一起，这称为单体架构（monolithic software）。其缺点是显然的：

1. 所有功能耦合在一起，互相影响，最终难以管理。

2. 哪怕只修改一行代码，整个软件就要重新构建和部署，成本非常高。

3. 因为软件做成了一个整体，不可能每个功能单独开发和测试，只能整体开发和测 试，导致必须采用瀑布式开发模型。 单体架构的软件不仅开发速度慢，而且会形成难以维护和升级的复杂代码，成为程序员的沉重负担。 大概在20多年前，随着互联网的出现，功能单元可以用远程"服务"的形式提供，于是就诞生出了"面向服务架构"（service-oriented architecture，简称 SOA）。



"面向服务架构"就是把一个大型的单体程序，拆分成一个个独立服务，也就是较小的程序。每个服务都是一个独立的功能单元，承担不同的功能，服务之间通过通信协议连在一起。这种架构有点有许多优点，如：每种服务功能单一，相当于一个小型软件，便于开发和测试；不同服务可以单独开发和部署，便于升级；扩展性好等等需要注意的一点是，“面向服务架构”的不同服务可以使用不同的语言和工具开发，可能需要部署在不同的系统和环境。这意味着，面向服务架构默认运行在不同服务器上，每台服务器提供一种服务，多台服务器共同组成一个完整的网络应用。2014年，Martin Fowler与James Lewis共同提出了微服务的概念，定义了微服务是由以单一应用程序构成的小服务，自己拥有自己的行程与轻量化处理，服务依业务功能设计，以全自动的方式部署，与其他服务使用 HTTP API 通信。同时服务会使用最小的规模的集中管理 (例如Docker)能力，服务可以用不同的编程语言与数据库等组件实现。

1.1.2 **微服务与单体框架的区别：**

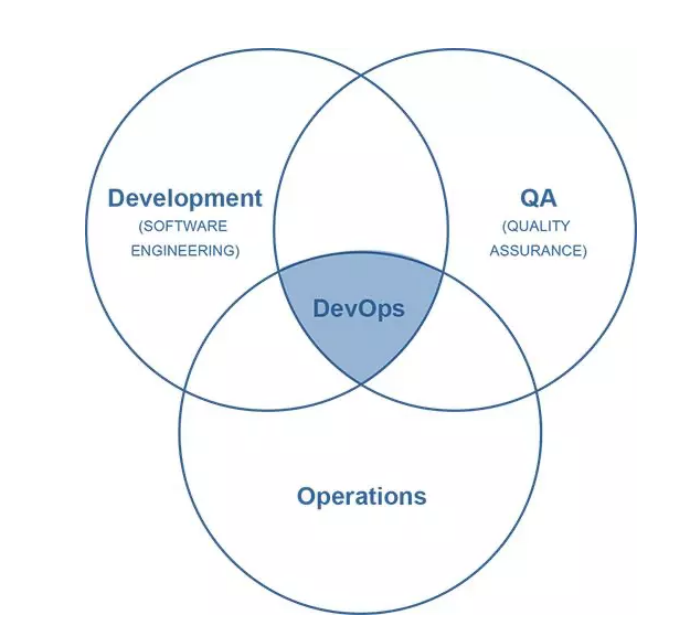
* 单体架构所有的模块全都耦合在一块，代码量大，维护困难。微服务每个模块就相当于一个单独的项目，代码量明显减少，遇到问题也相对来说比较好解决。
* 单体架构所有的模块都共用一个数据库，存储方式比较单一。微服务每个模块都可以使用不同的存储方式（比如有的用redis，有的用mysql等），数据库也是单个模块对应自己的数据库。（单体架构也可以实现，但是比较麻烦）
* 单体架构所有的模块开发所使用的技术一样。微服务每个模块都可以使用不同的开发技术，开发模式更灵活。

1.1.3 **微服务本质：**

* 微服务，关键其实不仅仅是微服务本身，而是系统要提供一套基础的架构，这种架构使得微服务可以独立的部署、运行、升级，不仅如此，这个系统架构还让微服务与微服务之间在结构上“松耦合”，而在功能上则表现为一个统一的整体。这种所谓的“统一的整体”表现出来的是统一风格的界面，统一的权限管理，统一的安全策略，统一的上线过程，统一的日志和审计方法，统一的调度方式，统一的访问入口等等。
* 微服务的目的是有效的拆分应用，实现敏捷开发和部署 。
* 微服务提倡的理念团队间应该是 inter-operate, not integrate。inter-operate是定义好系统的边界和接口，在一个团队内全栈，让团队自治，原因就是因为如果团队按照这样的方式组建，将沟通的成本维持在系统内部，每个子系统就会更加内聚，彼此的依赖耦合能变弱，跨系统的沟通成本也就能降低。

1.2 **DevOps**

DevOps 强调的是高效组织团队之间如何通过自动化的工具协作和沟通来完成软件的生命周期管理，从而更快、更频繁地交付更稳定的软件。



如果从字面上来理解，DevOps 只是Dev（开发人员）+Ops（运维人员），实际上，它是一组过程、方法与系统的统称，其概念从2009年首次提出发展到现在，内容非常丰富，有理论也有实践，包括组织文化、自动化、精益、反馈和分享等不同方面。

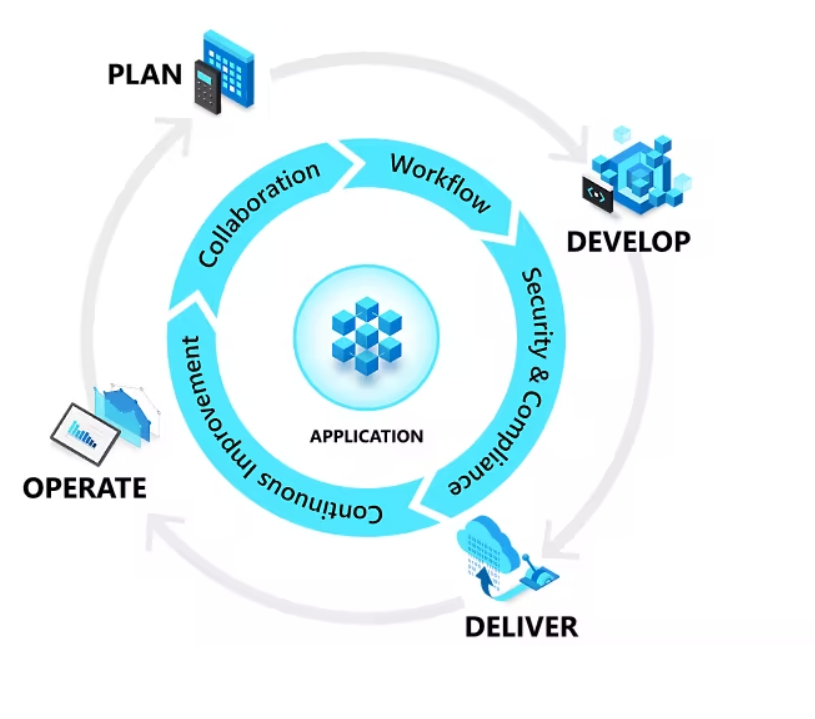
1. 组织架构、企业文化与理念等，需要自上而下设计，用于促进开发部门、运维部门和质量保障部门之间的沟通、协作与整合，简单而言组织形式类似于系统分层设计。

2. 自动化是指所有的操作都不需要人工参与，全部依赖系统自动完成，比如上述的持续交付过程必须自动化才有可能完成快速迭代。

3. DevOps 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，开发部门和运维部门必须紧密合作

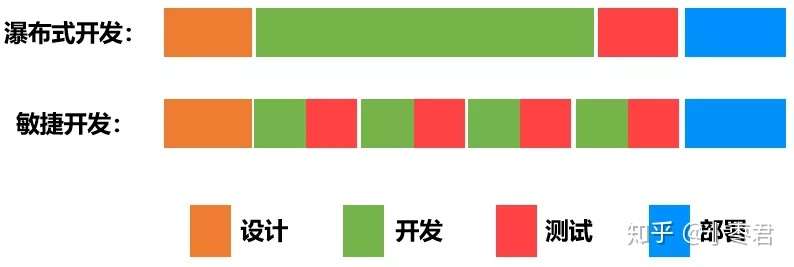
1.2.1 **Devops开发模式**

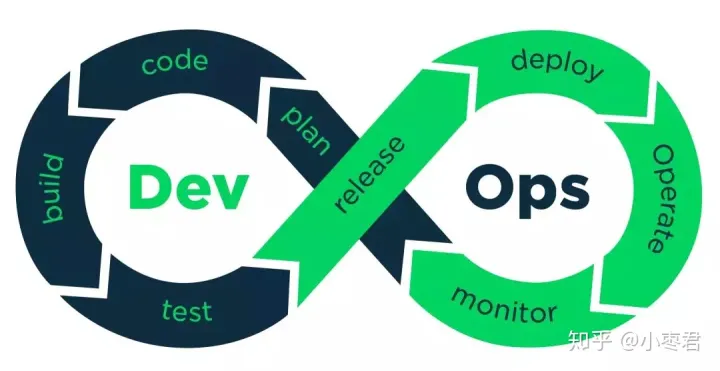
将项目拆成一个个小的服务单独部署，拆分部署催生出 DevOps，此时远程部署一些机器，专门用来管理代码，进行上线工作，由运维事先把上线的规则都给定义好了，开发只要按照他的规则都访问这台服务器进行各自的代码合成和发布，自己上线呢，能用代码自动完成的事情就绝不要手动解决，这是每个开发人员都在想的东西。运维需要做的事情，慢慢的都沉淀到了各个平台上面，例如监控，有专门的监控组件和可视化，基础服务例如服务器，CDN，负载均衡等基础服务可以外包到云服务厂商，日志也有专门的日志工具，链路追踪也有专门的组件和可视化，还有网关等，渐渐的，只要这些都配置好了，开发也可以做运维的部分工作，毕竟开发才是最了解代码的人，哪里出了问题看看监控日志，可以最快速度定位到问题，于是 **DevOps开发模式**诞生了



1.2.2 **Devops本质：**

从目标来看，**Devops**就是让开发人员和运维人员更好地沟通合作，通过自动化流程来使得软件整体过程更加快捷和可靠。对比瀑布式开发和敏捷开发，**Devops**贯穿了软件全生命周期，而不仅限于开发阶段。





在 DevOps 的流程下，运维人员会在项目开发期间就介入到开发过程中，了解开发人员使用的系统架构和技术路线，从而制定适当的运维方案。而开发人员也会在运维的初期参与到系统部署中，并提供系统部署的优化建议DevOps 的实施，促进开发和运维人员的沟通，增进彼此的理解。

1.2.3 **DevOps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

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

2. **By looking up information, to learn the history of web development. (web1.0~web4.0).**

2.1 **Web 1.0**

Web 1.0是互联网发展的第一阶段。这一时期的特点是，内容创作者可以一只手算数，所有其他互联网用户都是消费者。在互联网发展的这个阶段，个人网站（名片网站）很受欢迎，由静态页面组成，托管在由ISP管理的Web服务器上或免费托管上。

在互联网发展的这个阶段，禁止在网站上做广告;它几乎无处可寻。Web 1.0 可以被描述为允许在网站上显示信息的内容交付网络。Web 1.0允许人们搜索信息并消费它。没有与用户或用户之间的交互，也没有生成用户生成的内容。

*Web 1.0时期的主要特点：*

1. 静态页面。

2. 网站内容从服务器文件系统提供。

3. 页面是使用服务器端包含或通用网关接口 （CGI） 构建的。

4. 框架和表格用于定位和对齐页面上的元素。

Web 1.0 是满足当时大多数网站所有者需求的互联网。他们需要一种工具，让人们能够在一天中的任何时间访问他们的信息。他们有这种补救措施。Web 1.0还取消了对信息共享的地理限制。

2.2 **Web 2.0**

Web 2.0是互联网发展的第二阶段，是一种社交网络。Web 2.0 是一个重视用户生成内容、网站可用性和互操作性的互联网。在这个阶段，改变的不是网站的技术规格，而是网页的设计和使用方式。

互联网正在成为人们分享观点、意见、想法和经验的在线工具和平台的聚集地。与 Web 3.0 应用程序不同，Web 2.0 应用程序主要与最终用户交互。因此，感谢最终用户，以下内容正在积极开发：

● 播客

● 博客

● 使用 RSS 策划

● 社交网络

在互联网发展的这个阶段，Web浏览器技术被用于网站的开发，例如AJAX和JavaScript框架。这些框架在开发人员中最受欢迎，对Ruby on Rails编程语言的兴趣也在增长。Web 2.0可以表征如下 - AJAX，标签优化，前端创新。

*Web 2.0时期的主要特点：*

1.自由获取大量信息，允许用户对信息进行集体获取和分类。

2. 响应用户输入的动态网站内容的出现。

3. 改变网站所有者和网站用户之间信息传输的方式。

4. 允许使用软件使用网站信息的 API 的出现。

5. 对社会的负面影响和新威胁的出现。

Web 2.0 是交付内容并与其他互联网用户互动的能力。Web 2.0在短时间内彻底改变了人们对互联网的看法。Web 2.0发展的一个典型例子是YouTube，它一直依赖并且仍然依赖用户内容。YouTube今天仍然很受欢迎，尽管Web 4.0的阶段很快就会到来。

2.3 **Web 3.0**

Web 3.0是互联网使用和交互演变的结果。互联网已经成为一个数据库。这个阶段的特点是网站内部部分的优化，对此给予了很多关注。正是在互联网发展的这个阶段，网站和整个互联网的工作中引入了最多的创新。Web 3.0的特点是数据不属于特定的组织或人员，而是由用户共享的。互联网发展的这个阶段也被称为语义网。

语义网改变了分析信息和对网站进行排名的过程。这一时期的互联网开始使用机器人来控制站点。Web 3.0 迫使开发人员使用声明式本体编程语言 （OWL） 来创建特定的本体，机器人可以使用这些本体从网站内容中得出推论，而不是简单地匹配  
 关键字。

*Web 3.0时期的主要特点：*

1.网站语义的重要性出现。语义极大地影响了创建和共享网站内容所需的 Web 技术。“智能搜索和分析”的出现，让机器人理解单词的语义，而不是将其视为键和数字，对互联网空间的发展产生了强烈的影响。

2.积极运用人工智能。得益于自然语言处理能力，机器人可以区分信息并提供快速和相关的结果。

3.3D图形。三维设计不仅在网站中而且在电脑游戏、电子商务、广告等中都越来越受欢迎。

4.内容之间的关系。由于语义元数据，网站上的内容相互链接。结果，用户体验被提升到另一个层次。

5. 可用性。在这个阶段，互联网可用于所有电子设备：手机，手表，微波炉，平板电脑等。

自然语言处理能力弥合了人类与计算机化应用程序之间的沟通鸿沟。Web 2.0的问题在于机器人缺乏理解互联网上发布  
 的数据上下文的能力，因此，它们无法理解哪些内容是相关的。利用文本语言处理（语义）的可能性，可以使机器人和应用程序能够理解信息。

2.4 **Web 4.0**

Web 4.0是互联网技术快速发展的结果。今天，Web 4.0是关于用户和机器人之间的移动性和语音交互。以前，要与互联网交互，用户必须在计算机前的桌面上，但移动设备允许用户使用和分发信息，而不管位置如何。

Web 4.0改变了人类与机器人之间的关系，它们变得共生。人类可以不断接触机器人，日常生活越来越依赖机器。多亏了人工智能，自学系统可以研究人们并更准确地了解他们需要什么。例如，问爱丽丝一个问题就足够了，她会给出答案。

虚拟现实和增强现实正变得越来越流行。互联网上有越来越多的网站使用虚拟现实和增强现实提供各种服务。虚拟现实专用设备的功能越来越强大，销量也在逐渐增加。

*Web 4.0时期的主要特点：*

1. Web 4.0是人与机器的共生互动。人与机器之间的界限是模糊的。

2. 互联网是一个网络操作系统，信息可以从世界各地传输。

3. 互联网在其工作中积极使用人工智能，从而提高了所提供机会的质量。

4. 互联网与用户的通信方式与人们之间的通信方式相同。

5、互联网透明;所有操作都是可追溯的。

6. 在这个发展阶段，互联网提供了比以往任何时候都更高的速度和可靠性。

Web 4.0 改变了我们周围的一切：基础设施、商业、物流、医药等。在这个发展阶段，互联网由消费者控制，并让他完全控制自己的行为和数据。Web 4.0正在从根本上改变人们的日常生活。通过提供人与机器之间的联系，Web 4.0丰富了与互联网相关的任何活动领域的可能性。

3. **By looking up information, to learn cloud platform architecture.**

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。



该架构主要分为三层，从底层到上层分别是资源层、中间件层、应用层。接下来将分别说明各层的构造和作用。

3.1 **资源层或基础设施层（IaaS, Infrastructure as a Service）**

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。 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128G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。如，采用KVM技术（Kernel-based Virtual Machine的简称，是一 个开源的系统虚拟化模块，KVM的虚拟化需要硬件支持，是基于硬件的完全虚拟化）， 同样还有XEN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。广为流传的商业系统虚拟化软件VMware ESX系列是基于软件模拟的Full?Virtualization。

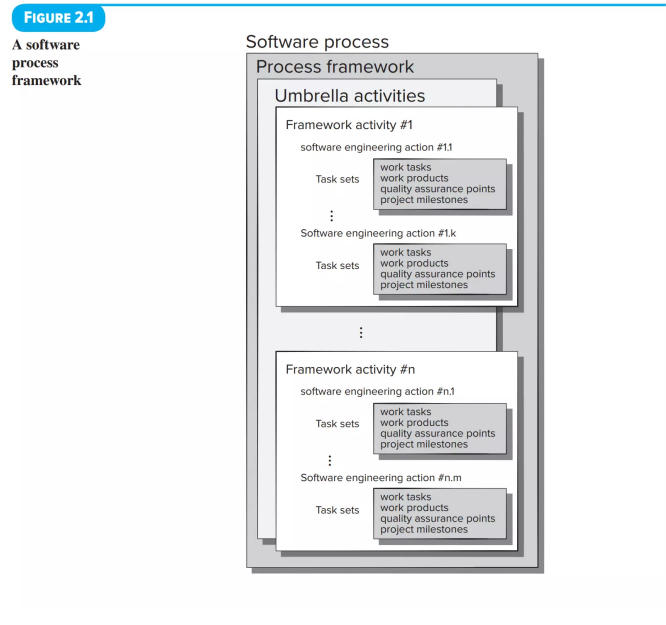
3.2 **中间件层或平台层（PaaS, Platform as a Service）**

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，当然还有负载均衡系统，如，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，数据处理交由虚拟机上的tomcat服务 器执行。k8s（Kubernetes），k8s是一种开放源码的容器集群管理系统，能够实现自 动化部署、扩展容器集群、维护等功能，所以它是一个完整的分布式系统支撑平台，提 供完善的管理工具，涵盖了开发、部署、测试、运行监控等各个环节，可以基于K8s来 部署Nginx。

3.3  **应用层或软件服务层（SaaS，Software-as-a-Service）**

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。

4. **Understand Figure 2.1 and find some process tools including open source tools and try to install and discuss them.**



每一个framework activity是由一系列software engineering action组成的，进一步，每一个software engineering action由一个任务列表所定义，这个任务列表中定义了：

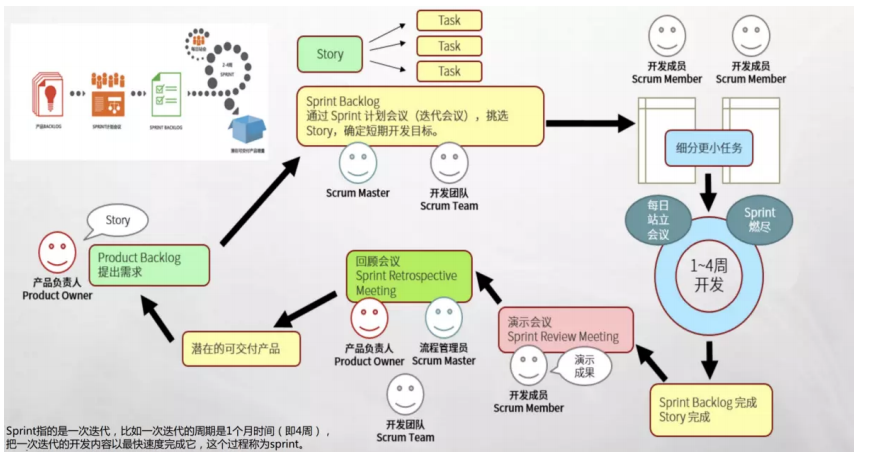
* 需要完成的任务
* 将要被生产出的产品
* 要求的质量保证
* 一些用来反映项目进度的里程碑。

一个通用的软件工程过程框架中包含5个framework activity，分别是：

* 交流
* 计划
* 建模
* 建设
* 部署

此外，在整个过程中，一系列的umbrella activity将会被使用，包括但不限于以下几点：

* 项目跟踪及控制
* 风险管理
* 项目质量保证
* 项目配置管理
* 技术评审



我们找到了一个过程管理Web端工具——**PingCode**。下面将简要的介绍一下 PingCode如何做敏捷开发项目管理。

4.1 **项目目标管理**

开发团队要清楚打造什么样的产品。在PingCode有一款专门为目标管理服务的子产 品（Goals），它能够帮助项目团队进行目标管理，建立可视化的目标管理过程并与研 发工作数据连接：Goals 不仅可实时查看目标进度，目标还支持添加关联多个项目的工 作项，并查看项目研发进度，从而定位给目标进度带来风险的具体项目；



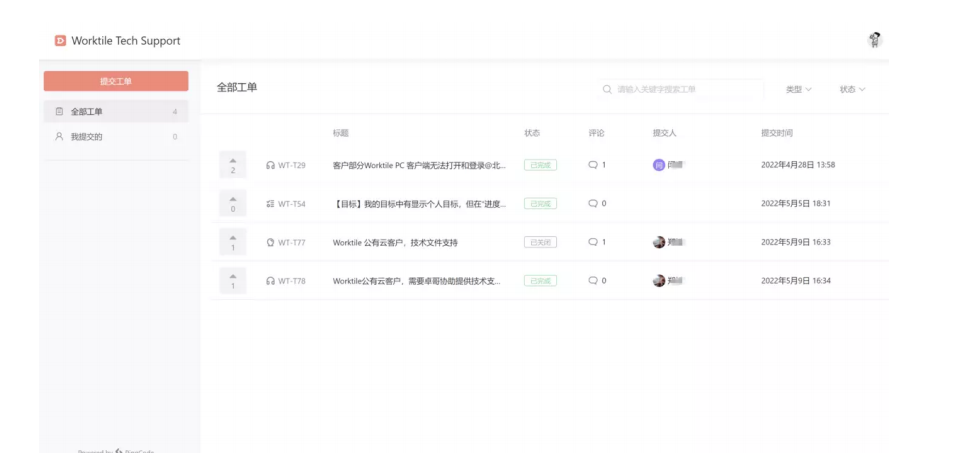
4.2 **需求代办列表管理**

在需求收集和需求管理的过程中，我们会遇到如下两类问题：

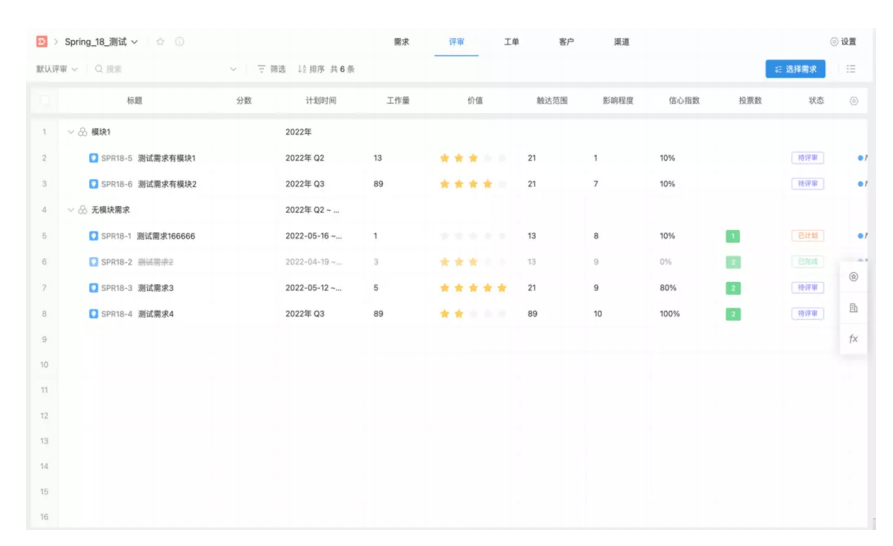
* 需求描述的问题：需求信息不清晰、不完整；
* 需求的优先级及排期问题：什么样的功能能对用户产生最大的价值

PingCode提供了较好的解决方案：

* 获取清晰完整的需求信息：PingCode 为用户提供了统一的需求、缺陷和建议反馈通 道，其中就包括Web Portal、小程序、邮件、Webhook等渠道。开发团队可以根据 需求自定义工单页面，以及与需求提交人直接沟通，尽可能的完善需求信息。在需 求收集后，团队可以按照史诗/特性/用户故事分级管理需求。



* 基于数据洞察实现科学的需求优先级评审排期：PingCode 能够帮助团队整合工作 量、价值、投票数、信心指数、影响程度，以及其他客户自定义的维度等信息。在 评审过程中，团队将综合各维度信息确定每个需求的权重分数，需求经过评审之后 通过计算的权重分数确定需求排期，以实现科学的优先级管理。

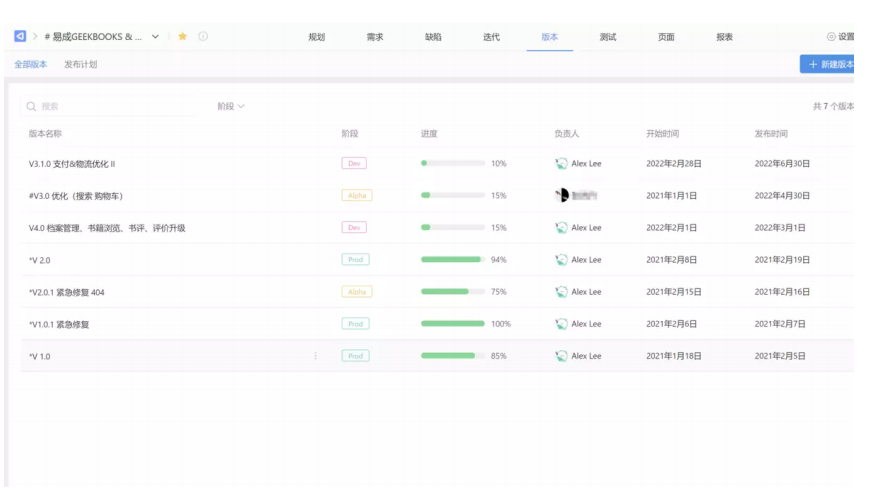


4.3 **迭代计划**

由于在开发过程中，迭代计划会频繁变动，经常有临时任务插队，团队成员的工作 状态被频繁打破等问题。解决这些问题需要团队在迭代前有着清晰的规划，并确定迭代 时间和目标，将需求拆解的足够细，在迭代后根据准确的度量来发现问题持续改进；

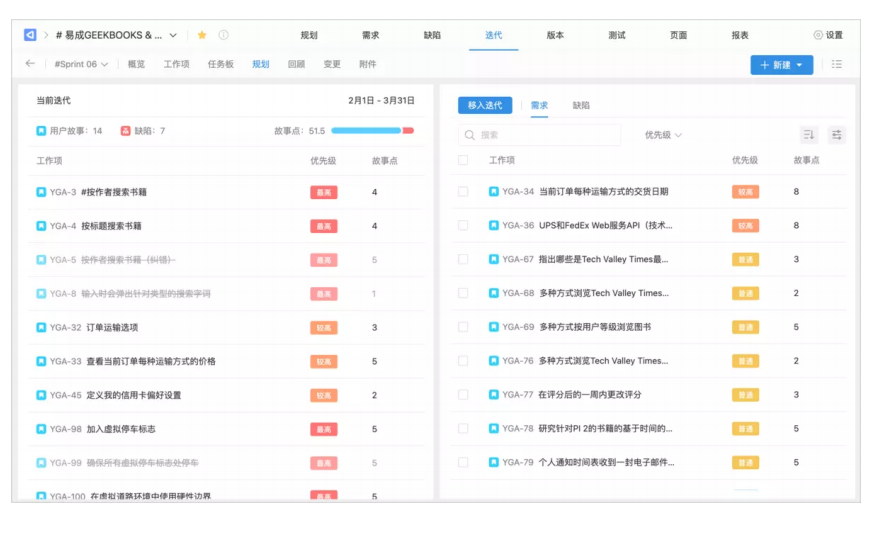
4.4 **迭代开始前**

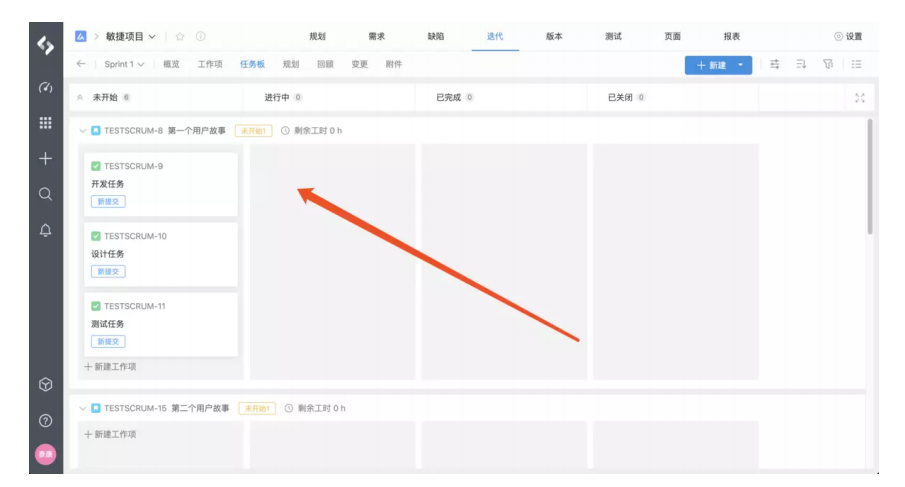
团队可以将已梳理完成且优先级高的用户故事规划到迭代看板内



4.5 **迭代会议**

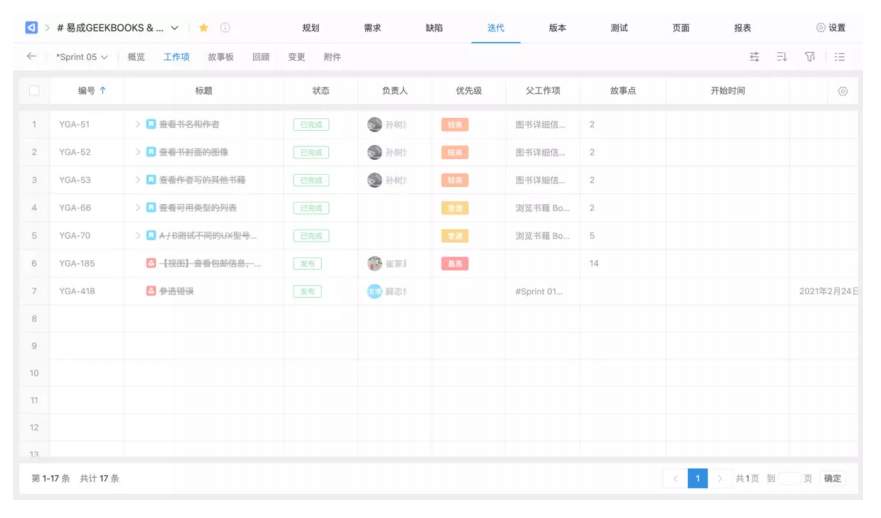
细化用户故事为更小的开发任务，提供敏捷估算器辅助估算故事点，规划形成 Sprint Backlog，填写预估工时。





4.6 **迭代评审**

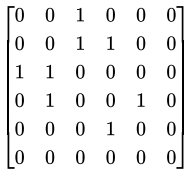
迭代评审会议在 Sprint 快结束时举行 ，这个事件是让开发团队展示他们在Sprint中 取得的成就，当任务负责人演示工作成果时，可能会提出一些缺陷，而这个时候团队可 以直接在用户故事上直接创建缺陷/Bug，并确定Bug紧急度。



5. **考试安排**

用等价类解决该题。

首先定义关系R：对于两门不同课程 i 与 j ，若两者可以在同一时段考试，则我们称 i 与 j 满足关系R。显然，关系R满足自反性与交换性，但是不满足传递性。对于一名学生选择的所有课程，任意两个不同的课程都不满足关系R，根据学生的选课信息我们可以得到如下的关系矩阵：



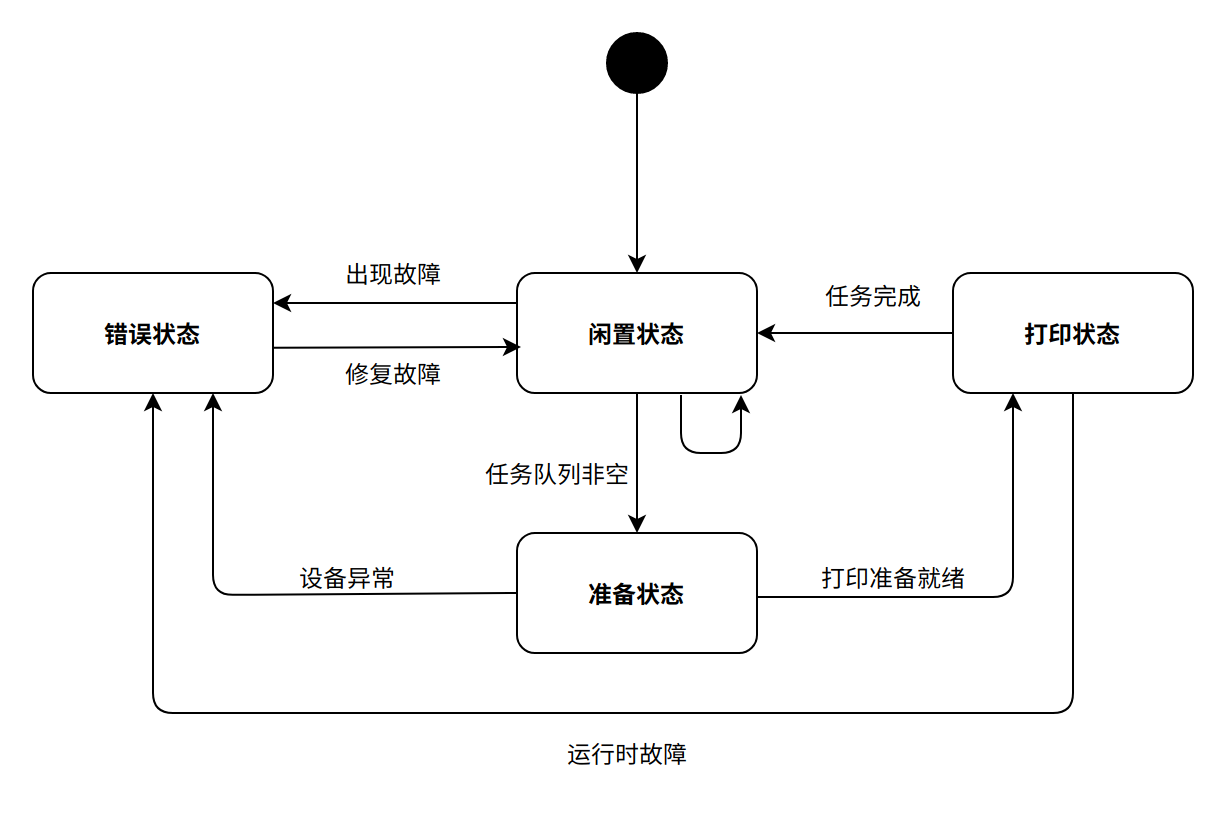
基于此矩阵，我们可以求出每个课程的等价类，在同一个等价类中的所有课程可以在安排在相同的时间考试，算法伪代码如下：

|  |
| --- |
| C++ for (i = 1 to n ) {   create an empty set Si   // represent equivalent class   for (j = 1 to n) {   if (a[i][j] == 1) {   bool flag = 1;   for (every element k in Si) { // scan at the column   if (a[j][k] == 0) {   flag = 0;   break;   }   }  if(flag) add j to Si   }   }  } |

按照以上方法求得的等价类为：(A,C) , (B,C) , (B,D) , (D,E),按照以上列出等价类的顺序划分课程集合，得到如下结果：

|  |  |  |
| --- | --- | --- |
| 场次 | 课一 | 课二 |
| 第一场 | 算法分析A | 计算机图形学C |
| 第二场 | 形式语言B | 模式识别D |
| 第三场 | 计算机网络E |  |
| 第四场 | 人工智能F |  |

6. **构建打印机或复印件软件系统的状态图**



打印机以闲置状态启动，在闲置状态下，打印机会监听任务列表，一旦任务列表中有打印任务，则从闲置状态转移到准备状态。在准备状态，打印机会加载当前任务，检查设备情况，若任务加载成功且设备正常，则会转移到打印状态，否则会转移到错误状态。在打印状态，若打印过程中遇到运行错误，则转移到错误状态，若未发生错误，则在完成当前打印任务后回到闲置状态。在错误状态，若相应问题得到修复，则状态会转移到闲置状态。

7. **Find a tool that creates the user interface prototype**

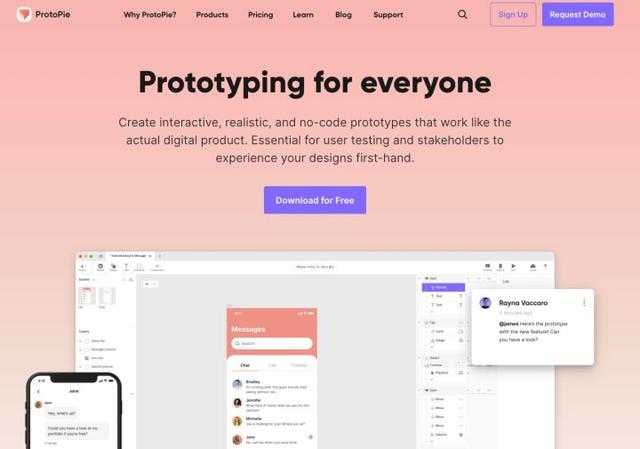
**ProtoPie**

我们找到的用户界面原型设计工具是ProtoPie。这是一款在互联网产品开发中非常流行的工具，因此我们查阅了相关资料，对这款工具做了以下的探讨。



ProtoPie 是一个灵活而强大的原型工具，因其易于使用的界面和平衡的学习曲线而受到设计界的称赞。

* 功能：原型。集成了 Sketch、Figma 和 Adobe XD
* 比较适合初学者，因为它非常容易学习和使用
* 适用于 macOS，Windows，iOS，iPadOS 和 Android 系统



与列表中介绍的其他原型工具略有不同，ProtoPie 只关注原型，因此首先需要从其他工具导入静态设计。ProtoPie 提供了 Figma、Adobe XD 和 Sketch 的插件，可以比较方便地从这 3 款软件中导入文件。

一旦导入了线框图，就可以通过添加触发器、响应和对象将它们转换为交互式原型。

触发器是触发交互的用户操作。对象是用户为了设置运动中的触发器而与之交互的元素。响应是互动的结果ーー例如，打开一个链接或跳到另一个页面。你可以通过简单地从工具栏的下拉菜单中选择预制触发器和响应来构建这些交互。