西安交通大學管理学院
THE SCHOOL OF MANAGEMENT
XI'AN JIAOTONG UNIVERSITY

Topic 4:系统开发与低代码/零代码平台

System Development and Low-code/No-code Development Platform (LCDP)

刘跃文 博士 Dr. LIU, Yuewen

教授、博士生导师 Professor

liuyuewen@xjtu.edu.cn

西安交通大学管理学院

School of Management, Xi'an Jiaotong University

V2.0, 2023-Oct

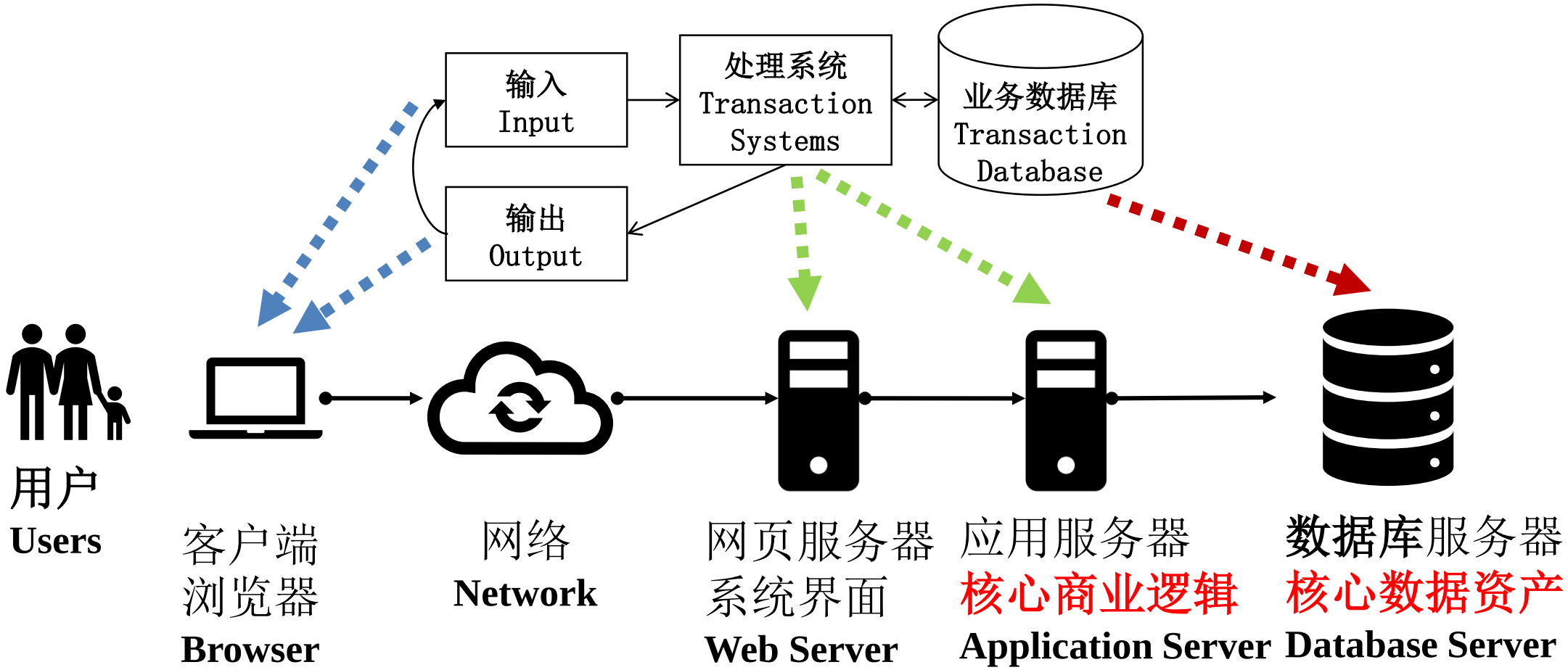
提纲 Outline

1. 典型的信息系统结构
   A General Network Structure of Information System
2. 传统的系统开发方法
   Traditional System Development Method
3. 低代码/零代码开发
   Low-code/No-code Development Platform (LCDP)
4. 低代码开发的发展方向
   Future Directions of LCDP
5. 作业 I
   Homework I

1. 信息系统网络结构

A General Network Structure of Information System

1. 信息系统网络结构
The Network Structure of an Information System



1. 输入关键字“管理信息系统教材”发出页面访问请求 https://search.jd.com/Search?keyword=管理信息系统&enc=utf-8&pvid=****

2. 预设没有内容的基础网页框架 https://search.jd.com/Search 请求填写网页中多项具体内容，如产品列表、推荐产品列表等

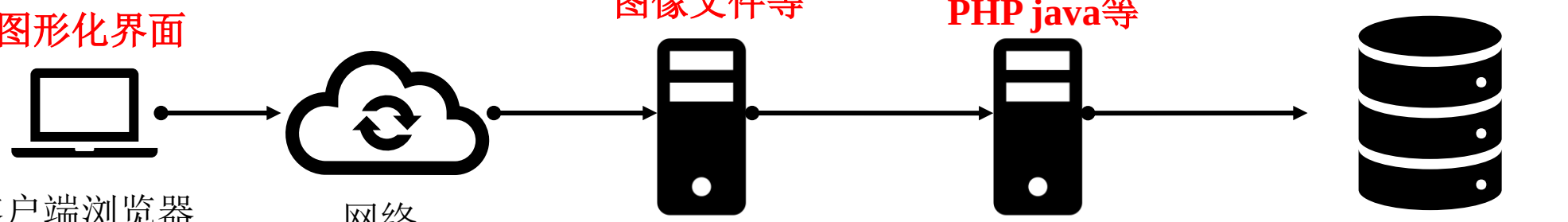
3. 根据预设的数据获取/计算方法，向数据库发出操作请求，如查询产品表中包括某关键字的商品

4. 执行查询并反馈结果，如 select * from goods where name like '%管理信息系统%'

5. 对查询结果进行计算和处理（如排序），形成符合网页要求的内容

6. 组装成网页内容

7. 查询到管理信息系统教材产品以及推荐产品等结果



售前工程师　前端工程师/UI/美工　后端工程师　数据库工程师

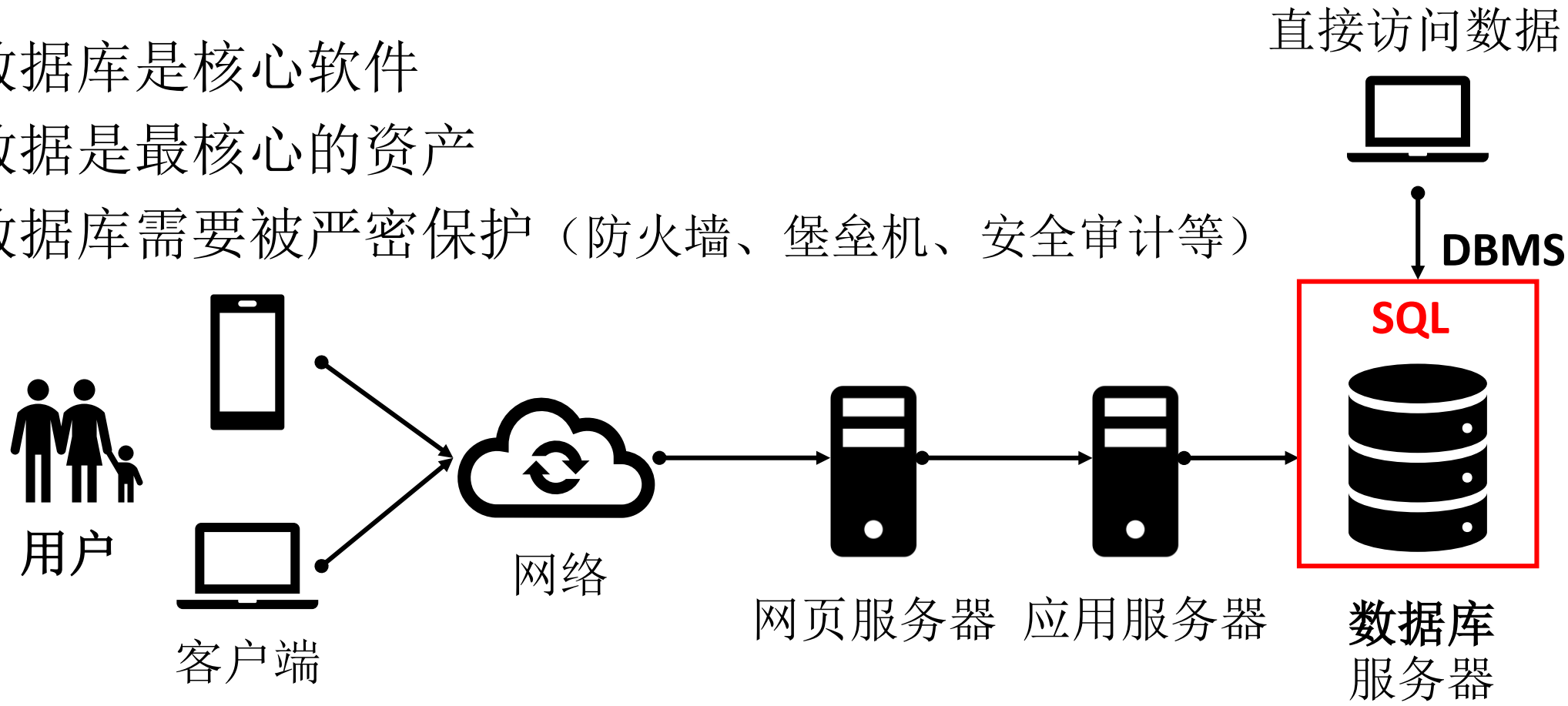
项目经理/系统架构师/网络工程师/安全工程师/全栈工程师

2. 三种服务器
Three Server Types

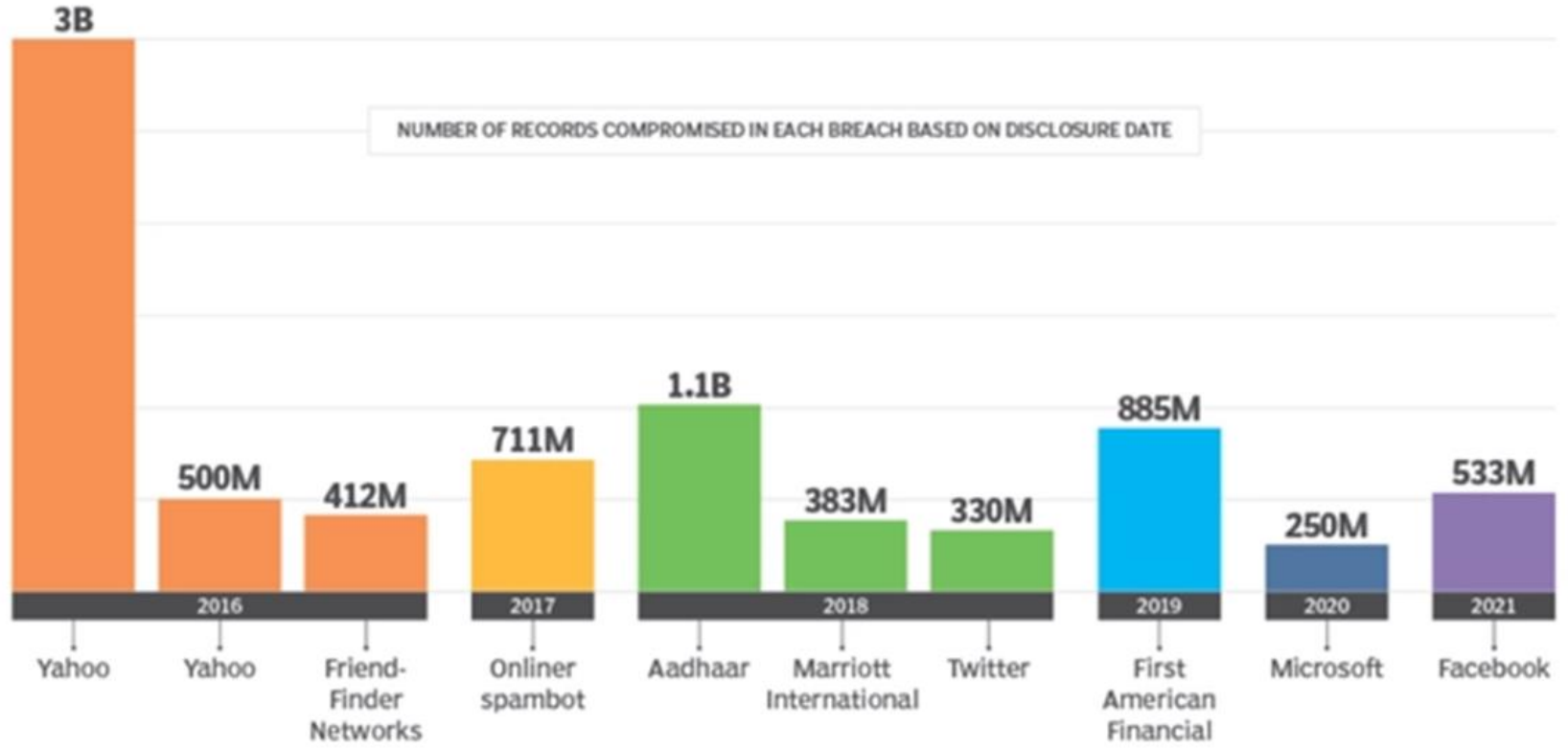
- Web服务器-用户界面
- 应用服务器-商业逻辑
  - 一些商业逻辑非常简单（例如：查询）
  - 一些数据计算方法非常复杂（例如：排程、调度、推荐、优化）
- 数据库服务器-数据资产
  - 一些数据库非常简单
  - 一些数据库非常复杂（完整性约束、依赖关系、权限控制、操作的原子性、并发控制等）

那为什么我们看不到数据库？

- 数据库是核心软件
- 数据是最核心的资产
- 数据库需要被严密保护（防火墙、堡垒机、安全审计等）



被“拖库”有非常严重的后果



来源：史上十大数据泄露事件及其教训 https://baijiahao.baidu.com/s?id=1740121852776165027

2. 传统的系统开发方法
Traditional System Development Method

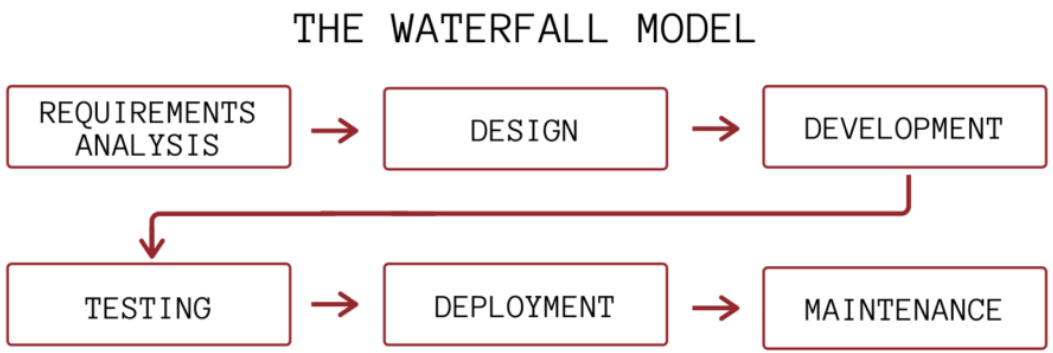
建设/开发一套系统是谁的工作？

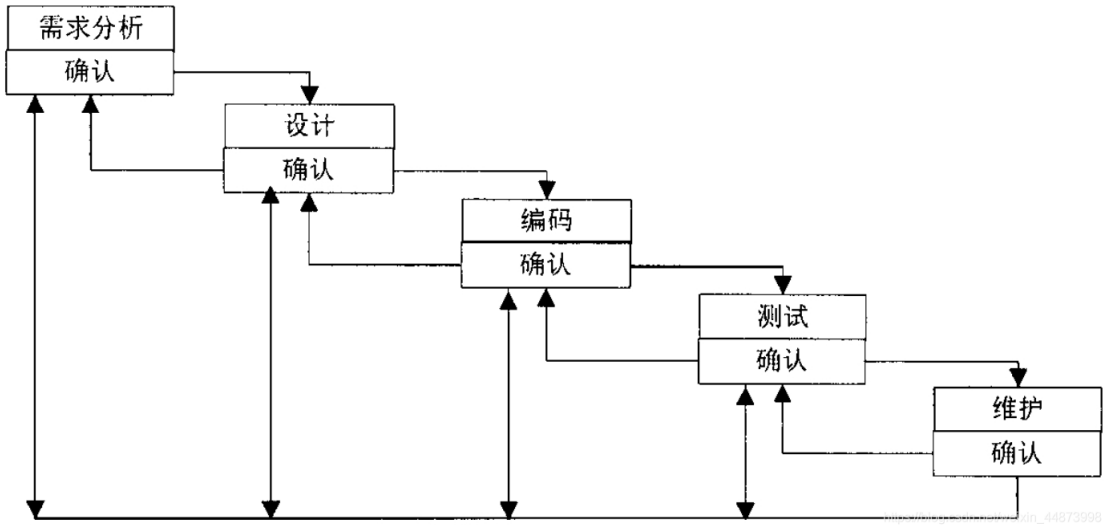
- CTO/CIO/领导：IT部门负责建设。
- IT部门：我花了钱，就是软件/系统开发公司的工作。

- 常见的流程：
  - 撰写可行性研究报告，预算
  - 专家评审可行性研究报告
  - 编写招标说明书，招标：某公司中标，支付启动经费
  - 初步设计-初步设计专家评审
  - 详细设计-详细设计专家评审
  - 代码开发、测试
  - 专家验收，支付尾款

1. 瀑布模型
Waterfall Model

- 瀑布模型也称软件生存周期模型或线性顺序过程模型，由W.Royce于1970年首先提出的，提供软件开发的系统化的和顺序的方法。
- 它是将软件生存周期各活动规定为线性顺序连接的若干阶段的模型，包括问题定义、可行性研究、需求分析、概要设计、详细设计、编码、测试和维护。
- 瀑布模型从需求分析开始，逐渐进行，直至通过测试并得到用户确认的软件产品为止。瀑布模型的上一阶段的输出结果是下一阶段的输入，如同瀑布流水，逐级下落。
- 瀑布模型提供了一个模板，模板使得分析、设计、编码、测试和维护的方法可以在该模板下有一个共同的指导。



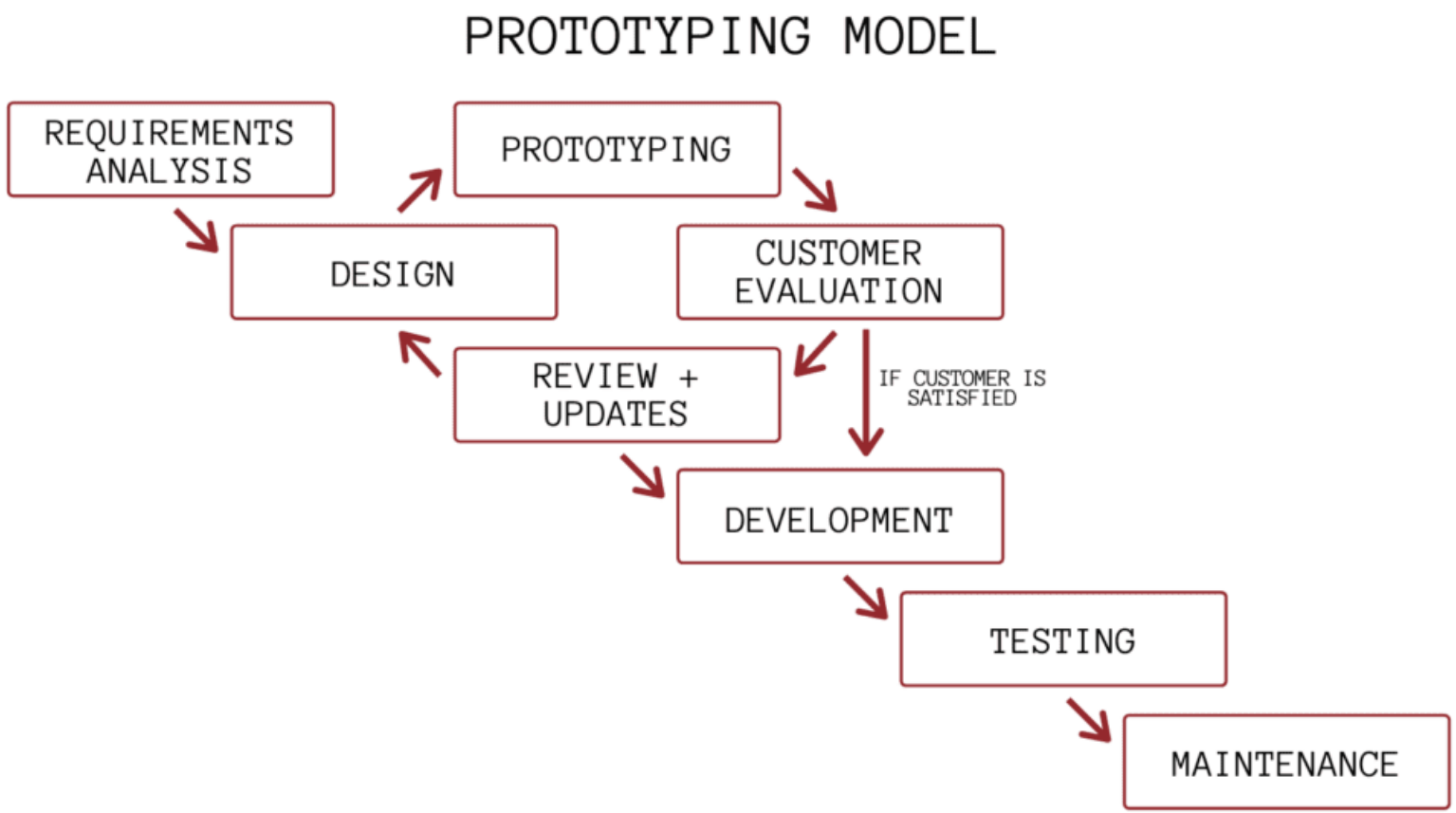


模糊式的开始 + 灾难式的结束

- 实际的项目大部分情况难以按照该模型给出的顺序进行，而且这种模型的迭代是间接的，这很容易由微小的变化而造成大的混乱。
- 在通常情况下，用户难以表达真正的需求，而这种模式却要求如此，这种模型是不欢迎具有二义性问题存在的。
- 用户要等到开发周期的晚期才能看到程序运行的测试版本，而在这时若发现大的错误，可能引起用户的惊慌，而后果也可能是灾难性的。
- 采用这种线性模型，经常在过程的开始和结束时，要等待其他成员完成后，才能进行下去，有可能花在等待的时间比开发的时间要长。我们称之为“堵塞状态”。

2. 快速原型模型
Prototyping Model

- 原型模型原型就是可以逐步改进成运行系统的模型。开发者在初步了解用户需求的基础上，凭借自己对用户需求的理解，通过强有力的软件环境支持，利用软件快速开发工具，构成、设计和开发一个实在的软件初始模型（原型，一个可以实现的软件应用模型）。
- 开发人员和用户在“原型”上达成一致。这样一来，可以减少设计中的错误和开发中的风险，也减少了对用户培训的时间，从而提高了系统的实用性、正确性及用户满意度。
- 原型模型采用逐步求精的方法完善原型，使得原型能够快速开发，避免了像瀑布模型一样冗长的开发过程中难以对用户的反馈作出快速响应。
- 原型模型通过“样品”不断改进，降低了成本。

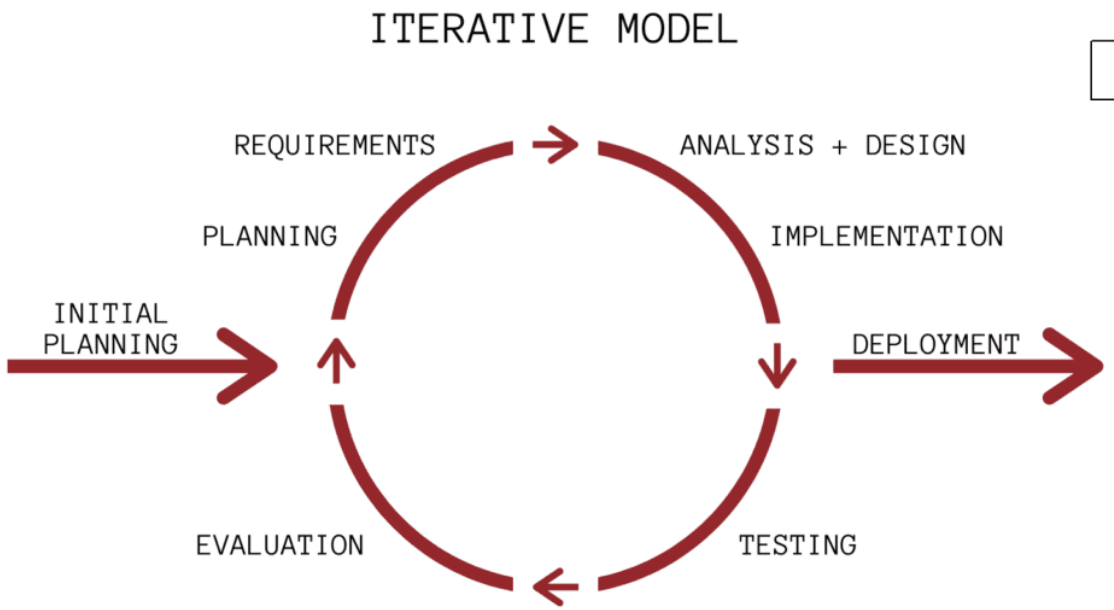


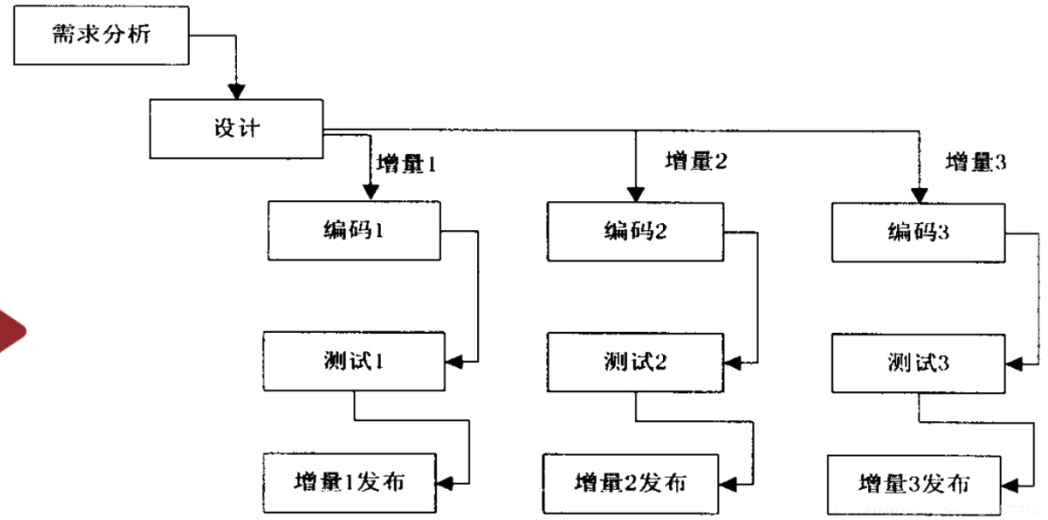
原型系统必然要被抛弃

- 用户看到的是一个可运行的软件版本，但不知道这个原型是临时搭建起来的，也不知道软件开发者为了使原型尽快运行，并没有考虑软件的整体质量或以后的可维护性问题。当被告知该产品必须重建才能使其达到高质量时，用户往往叫苦连天。
- 开发人员常常需要在实现上采取折中的办法，以使原型能够尽快工作。开发人员很可能采用一个不合适的操作系统或程序设计语言，仅仅因为它通用或有名，也可能使用一个效率低的算法，仅仅为了实现演示功能。经过一段时间之后，开发人员可能对这些选择已经习以为常了，忘记了它们不合适的原因。于是，这些不理想的选择就成了软件的组成部分。
- 使用原型模型开发系统时，用户和开发者必须达成一致：原型被建造仅仅是用户用于定义需求，不宜利用它来作为最终产品，之后被部分或全部抛弃，最终的软件是要充分考虑了质量和可维护性等方面之后才被开发。

3. 迭代模型/增量模型
Iterative (and Incremental) Model

- 增量模型也称为渐增模型。该模型融合了线性顺序模型的基本成分和原型实现模型的迭代特征。增量模型采用随着日程时间的进展而交错的线性序列。每一个线性序列产生软件的一个可发布的“增量”。
- 当使用增量模型时，第一个增量往往是核心的产品，也就是说，第一个增量实现了基本的需求，但很多的补充的特征还没有发布。用户对每一个增量的使用和评估，都作为下一个增量发布的新特征和功能。这个过程在每一个增量发布后不断重复，直到产生了最终的完善产品。增量模型强调每一个增量均发布一个可操作的产品。

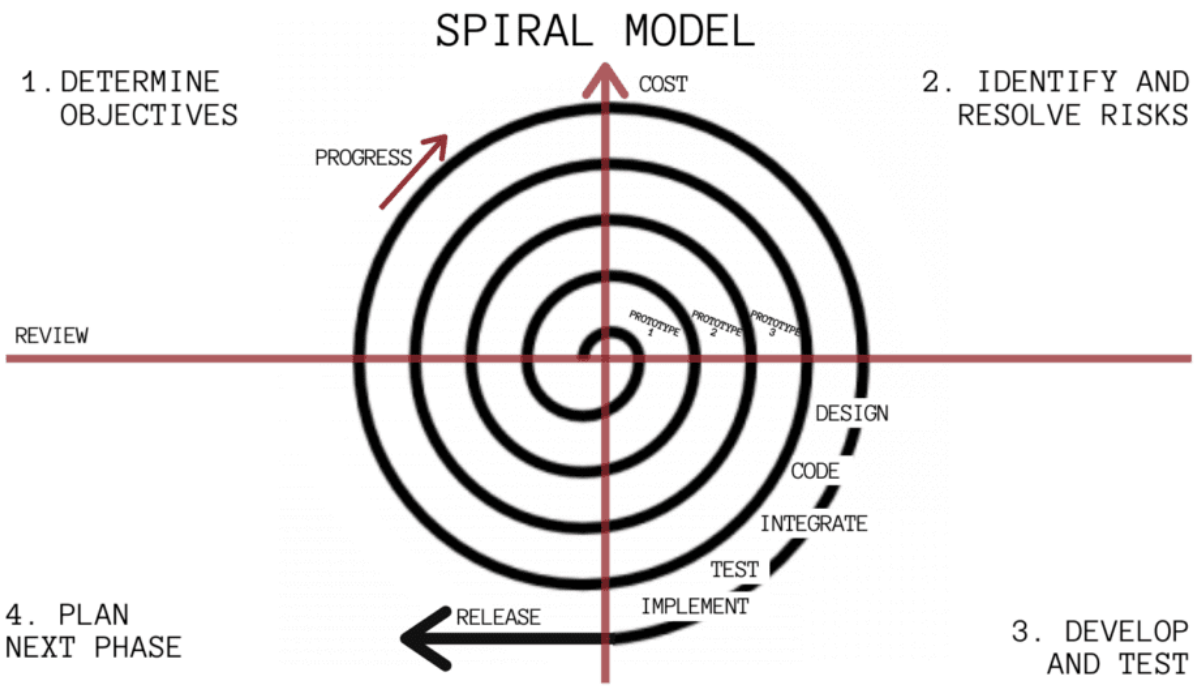


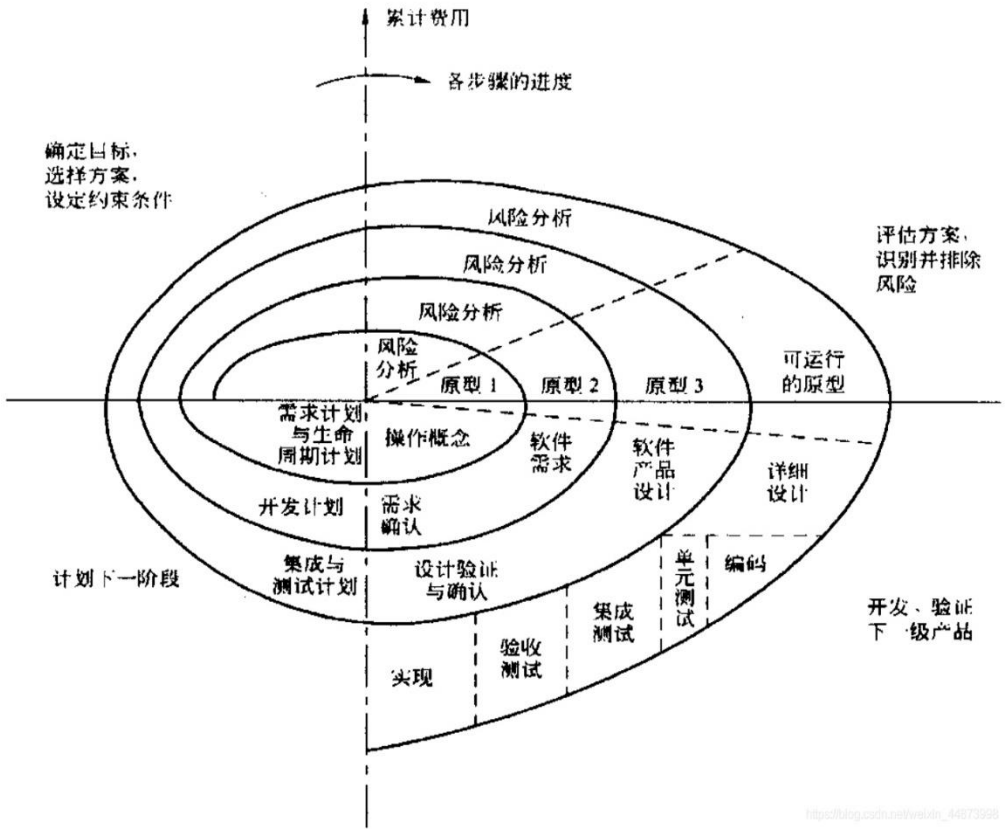


- 反复迭代的过程中，一部分开发资源被浪费。
- 无法确定的资源消耗，无法确定的最终期限。
- 自始至终，开发者和用户纠缠在一起，直到完全版本出来。

4. 螺旋模型
Spiral Model

- 螺旋模型（Spiral Model）是一个演化软件过程模型，它将原型实现的迭代特征和线性顺序模型中控制的和系统化的方面结合起来，使得软件的增量版本的快速开发成为可能。
- 在螺旋模型中，软件开发是一系列的增量发布。在每一个迭代中，被开发系统的更加完善的版本逐渐产生。螺旋模型被划分为若干框架活动，也称为任务区域。对于大型软件，只开发一个原型往往达不到要求。螺旋模型将瀑布模型和增量模型结合起来，并加入了风险分析。
- 螺旋模型将开发过程分为几个螺旋周期，每个螺旋周期可分为4个工作步骤：1. 确定目标、方案和限制条件。2. 评估方案、标识风险和解决风险。3. 开发确认产品。4. 计划下一周期工作。

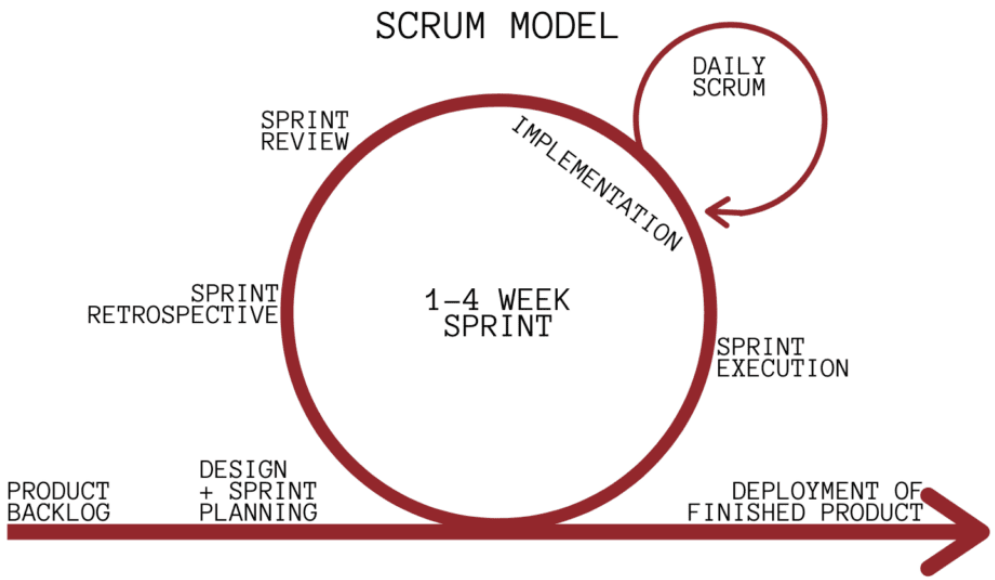


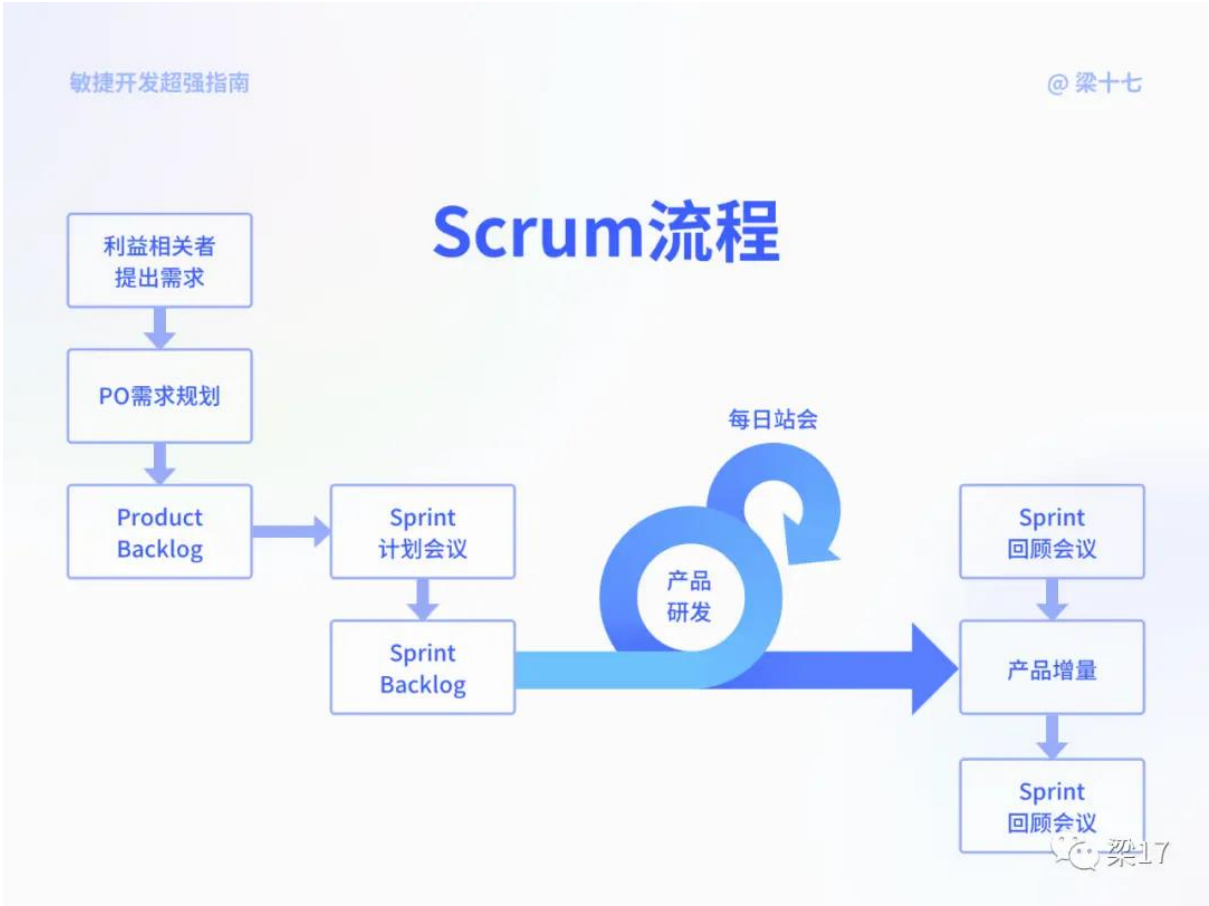


- 需要相当的风险分析评估的技术，且成功就依赖于这种技术。
- 显然，若存在一个没有被发现的大风险，将会出现问题，甚至可能导致演化过程失去控制。这种模型相对比较新，应用不广泛，其功效需要进一步的验证。

5. 敏捷开发

- 敏捷软件开发是基于敏捷宣言定义的价值观《敏捷软件开发宣言》和原则《敏捷软件的十二条原则》的一系列方法和实践的总称。自组织、跨职能团队运用适合他们自身环境的实践进行演进得出解决方案。换句话说敏捷开发是一种应对快速变化的需求的一种软件开发能力，只要在符合价值观和原则的基础上能让开发团队拥有应对快速变化需求的能力，这就叫做敏捷开发。
- 例如：Scrum敏捷开发模型
  - Scrum整个流程分为6步：1. 待办事项列表（Product Backlog）2. 迭代待办事项列表（Sprint Backlog）3. Sprint计划会议 4. Sprint（冲刺）5. Sprint评审会议 6. Sprint回顾会议





6. 露华浓ERP项目分析

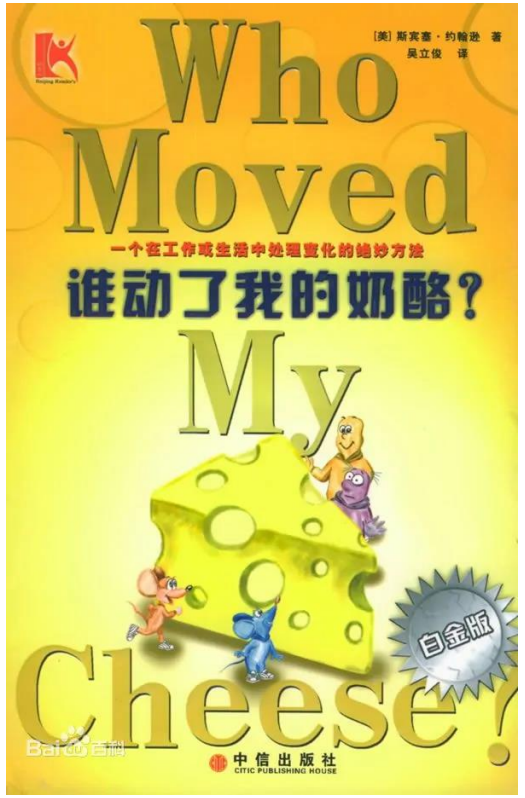
- 2019年发生过露华浓集体诉讼事件，就是由于ERP二次开发成本过高间接造成的。当时露华浓（Revlon）作为知名的美容产品，选择了某世界最顶尖的ERP厂商，然而随着ERP项目的推进露华浓需求的不断变更，ERP厂商的报价也越来越高昂，最后不堪重负的露华浓决定暂缓项目，直接导致了项目进度延误，进而引起6400万美元的产品未能及时交付，引起了集体诉讼。
- 露华浓ERP开发可能用了什么系统开发模型？
- 采用预算-开发式的项目管理方式可能会有什么问题？
- 有没有“最好的”系统开发模型？

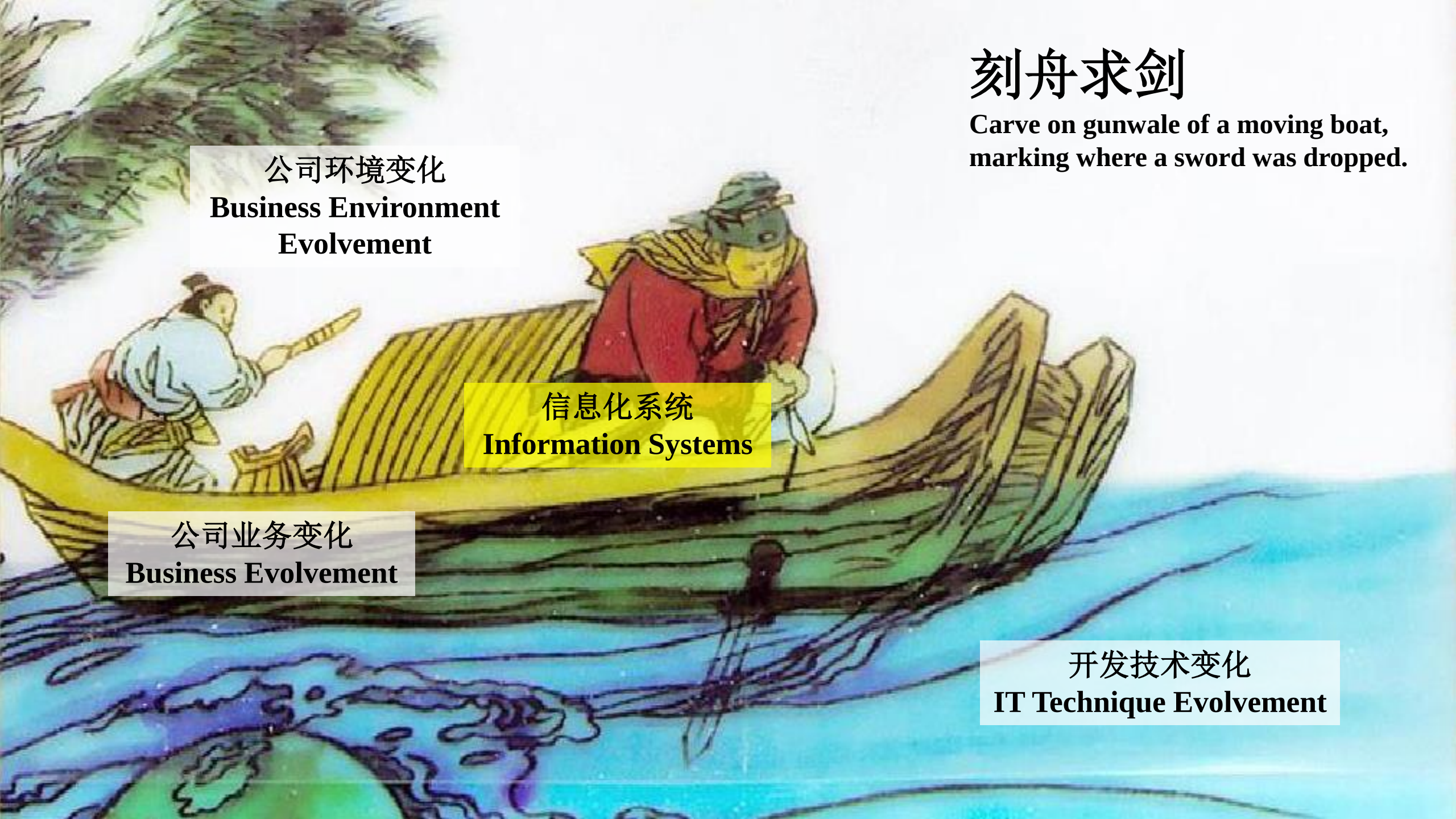
3. 低代码/零代码开发

Low-code/No-code Development Platform (LCDP)

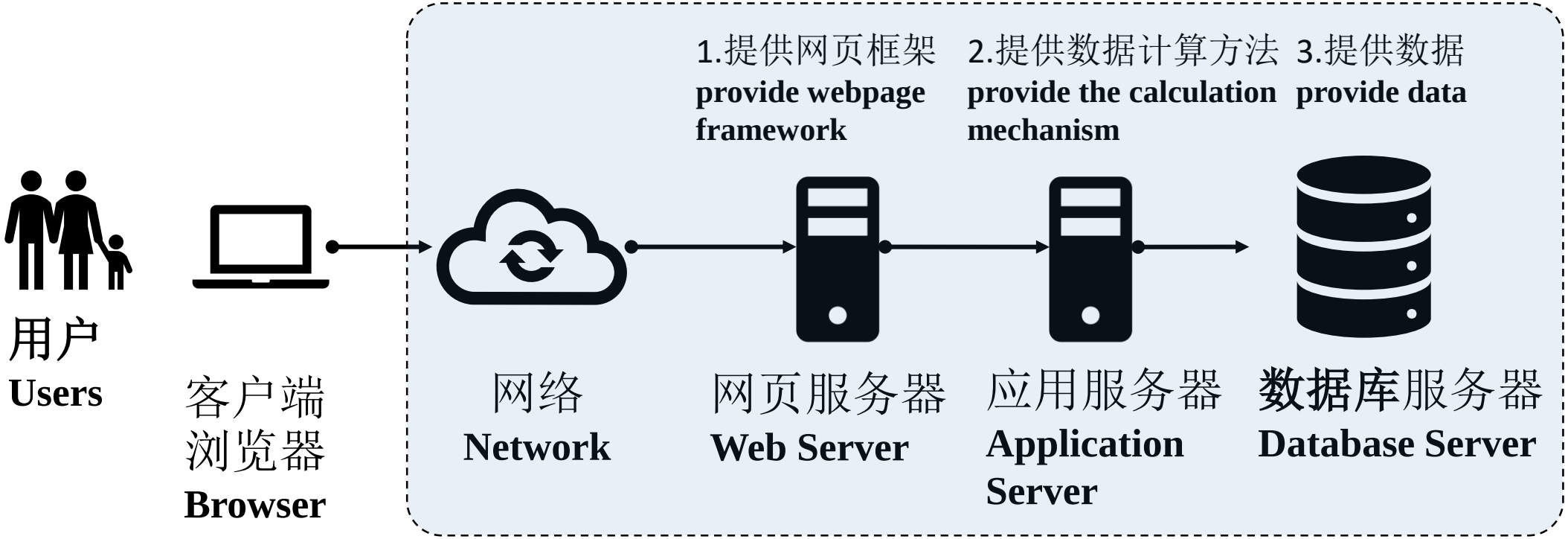
1. 唯一不变的是变化
Change is the only constant thing.

- Who Moved My Cheese
- The old adage asserts that, ‘change is the only constant thing’; consequently, adapting and enjoying change repeatedly is the best thing to do. Spencer wrote Who Moved My Cheese to illustrate the effects of denying change or accepting it.
- Smell The Cheese Often So You Know When It Is Getting Old.
- 经常闻一闻你的奶酪，你就会知道，它什么时候开始变质。




刻舟求剑
Carve on gunwale of a moving boat, marking where a sword was dropped.
公司环境变化
Business Environment Evolvement
信息化系统
Information Systems
公司业务变化
Business Evolvement
开发技术变化
IT Technique Evolvement

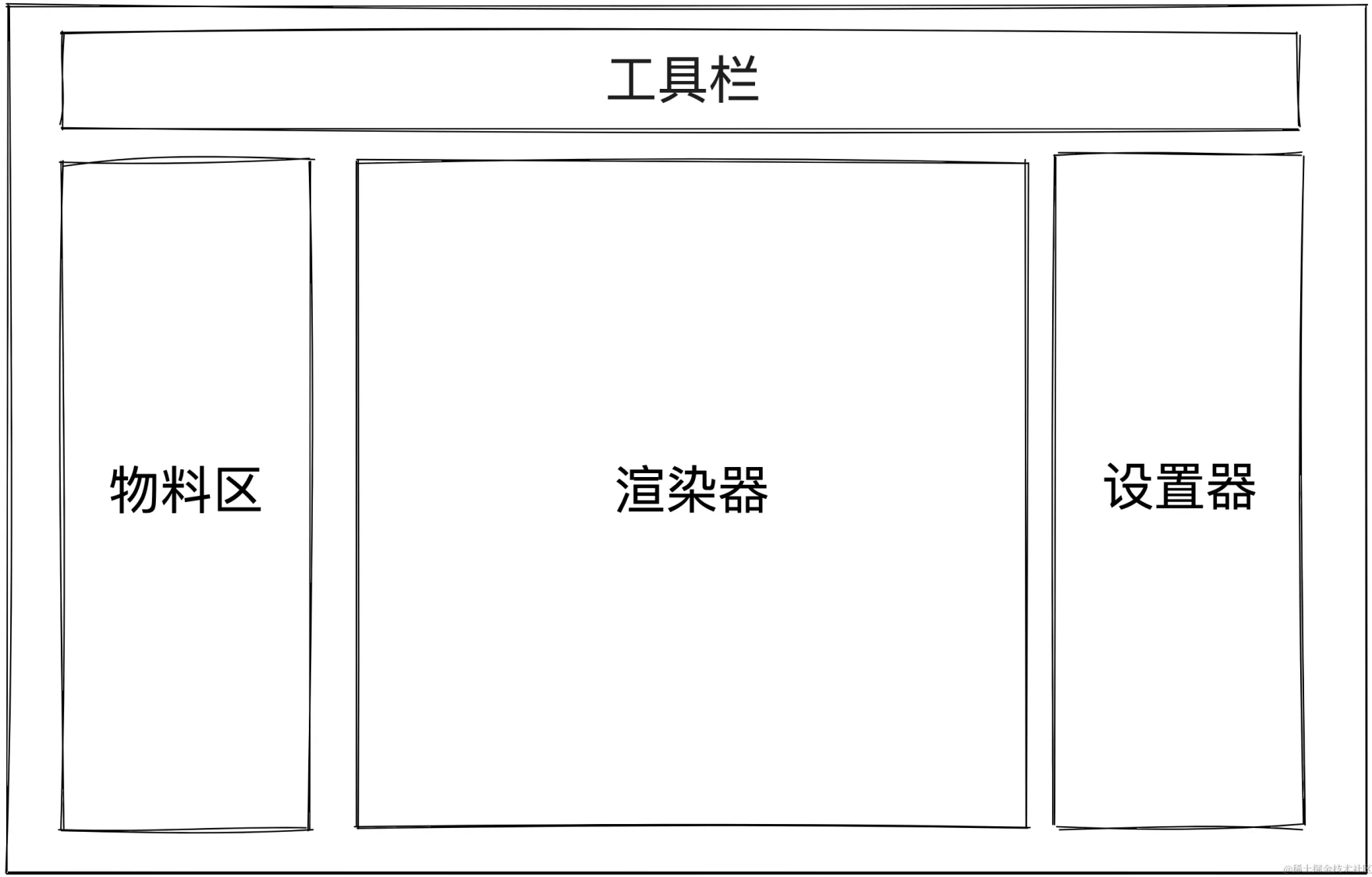
云服务 Could Service



1. 将技术要素进行封装，以业务视角进行应用快速构建。
2. 很少或几乎不用写代码就可以快速开发应用。

2. 低代码/零代码平台
Low-code/No-code Development Platform

- 低代码是一种只需用很少甚至不需要代码即可快速开发系统，并将其快速配置和部署的技术和工具。（来自Gartner于2014年提出的概念）。
- 低代码是一组数字技术工具平台，基于图形化拖拽、参数化配置等更为高效的方式，实现快速构建、数据编排、连接生态、中台服务。通过少量代码或不用代码实现数字化转型中的场景应用创新。
- 低代码平台也常被称为aPaaS平台。aPaaS，Application Platform as a Service，应用程序平台即服务。国际知名咨询机构Gartner对aPaaS所下的定义是：“这是基于PaaS（平台即服务）的一种解决方案，支持应用程序在云端的开发、部署和运行，提供软件开发中的基础工具给用户，包括数据对象、权限管理、用户界面等。”用户可以直接在aPaaS平台上以低代码/零代码的方式快速完成应用程序的搭建、部署、运行和管理。



网易视频

B站视频 零代码开发平台


B站视频 6种低代码平台
先进生产力工具系列


钉钉宜搭简单视频教程


钉钉宜搭审批流程

3. 低代码开发的优点

- 减少开发的成本和操作门槛
  - 低代码开发工具把功能都抽象、统一、开发、封装起来，形成一个个模块，让用户可以直接使用，省掉大量重复的开发劳动。
  - 让不会编程的用户也可以利用平台现成的功能，构建出自己需要的应用。
  - 业务人员自己也可以用零代码开发平台实现，彻底打消数字应用的操作门槛。
- 提高系统适应业务变更的灵活性
  - 构建高度灵活的业务管理应用，随时根据业务变动来修改配置。
  - 对于业务和技术部门来说，低代码都能有效提高双方解决问题的效率，减少沟通成本。
- 为业务模式革新提供灵感和工具
  - 低代码开发平台赋予非开发者敢于想象、敢于试错的工具和精神，而这正是推动行业创新、企业进步所需要的动力。


一汽大众低代码应用
四川 成都
一汽-大众成都分公司
一汽-大众汽车有限公司
FAW-VOLKSWAGEN AUTOMOTIVE CO.,LTD.

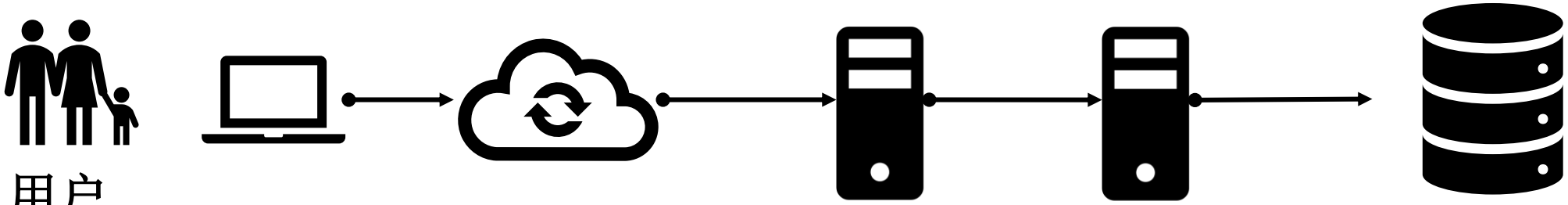

默安科技使用低代码平台

4. 低代码平台仍然存在的问题
The Weaknesses of LCDP

- 仍然有学习和上手成本。
- 受限于低代码平台的组件和功能。
- 底层代码未知，卡bug需要平台公司配合解决或者规避。
- 并发数和连接数受限于平台能力，性能优化比较困难。
- 升级和发展的问题。
  - 前端框架日新月异，过一两年来个改革，低码平台就得适配一遍，维护成本相当大；
  - 平台自身升级了，你也不知道会对自己的项目产生什么影响，但是不跟着升级后面就没法迭代；
  - 从一个平台的低代码，要想切换到其他平台那就得从 0 到 1。

低代码平台适用的场景

- 二八法则：实现80%的功能，适用80%的场景



用户
Users

客户端
浏览器
Browser

网络
Network

网页服务器
系统界面
Web Server

应用服务器
核心商业逻辑
Application Server

数据库服务器
核心数据资产
Database Server

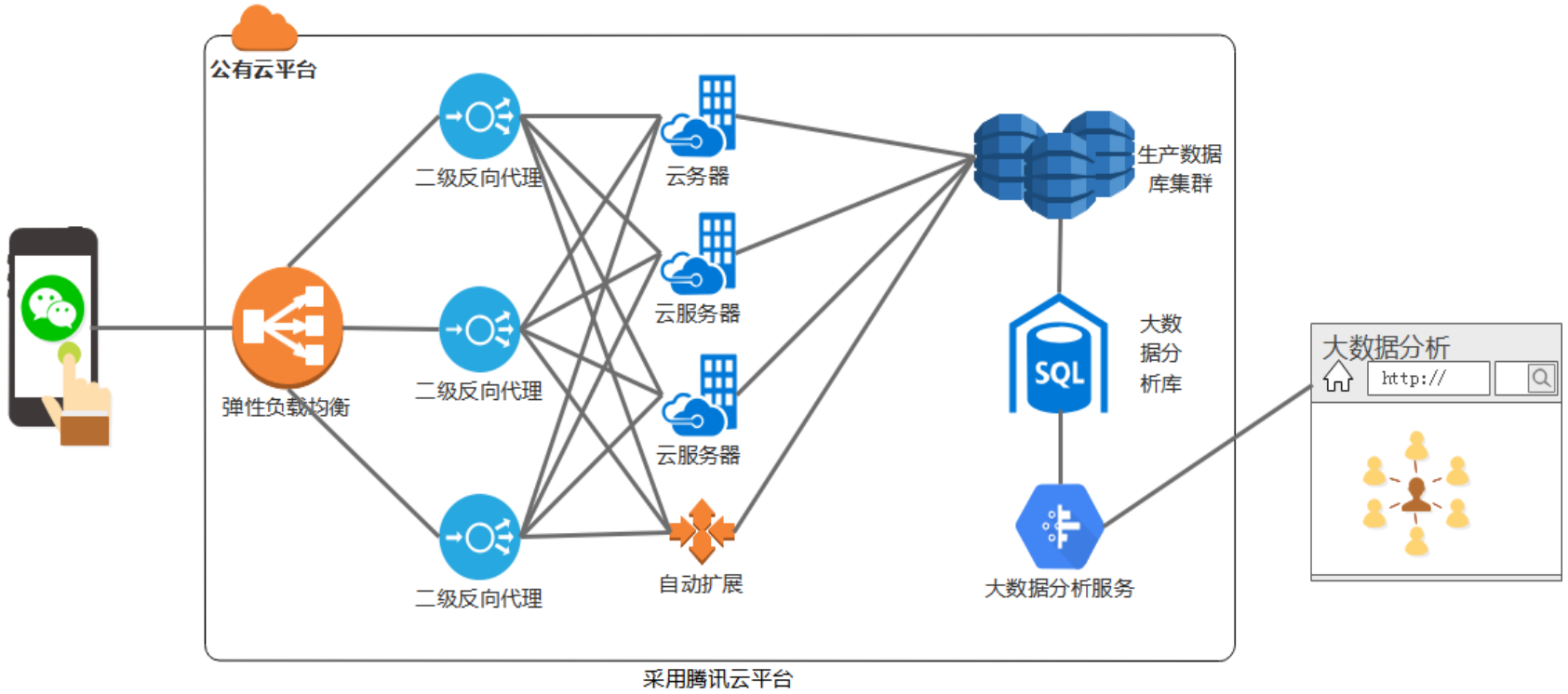
核心页面为通用型组件

计算逻辑/流程相对比较简单

数据结构相对比较简单

案例：高并发、高响应的系统

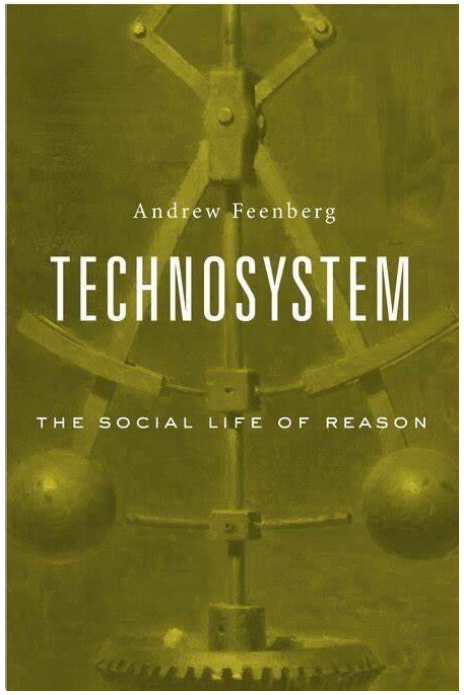
- 百度如何做到1秒之内查询结果返回？
- 负载均衡服务



4. 低代码平台的发展趋势
Future Directions of LCDP

1. 技术民主化
Democratization of Technology

- 著名技术哲学家安德鲁•费恩伯格 Andrew Feenberg 教授，曾提过一个很有创意的概念，“技术民主化 Democratization of Technology”。
- 技术民主就是扩大社会个体的自由边界，让大家积极有效地参与到技术设计和技术决策之中，包括不同身份、不同阶层的“外行行动者”也都能参与其中，以确保他们的利益诉求得以实现。简而言之，就是让广泛大众参与技术设计，最终实现更大程度的技术协同。



Andrew Feenberg (2017). Technosystem: The Social Life of Reason: Cambridge, MA: Harvard University Press. 235 pp. ISBN 9780674971783 (Hardcover)

- 技术民主化（Democratization of Technology）指的是人人都能容易地获取技术，不论他们从事何种职业、不论他们身处何方。
- 高德纳咨询公司Gartner认为，在 2022 年以后的后现代，技术民主化的四个关键方面将成为最新的技术趋势：应用程序开发、设计、知识、数据和分析。
- 新技术和改进的用户体验将使技术行业之外的人员能够访问和使用技术产品和服务。技术的民主化意味着无需广泛或昂贵的培训，即可使人们轻松获得技术或业务专业知识。随着大众开发者的崛起，这一点已经得到广泛认可.

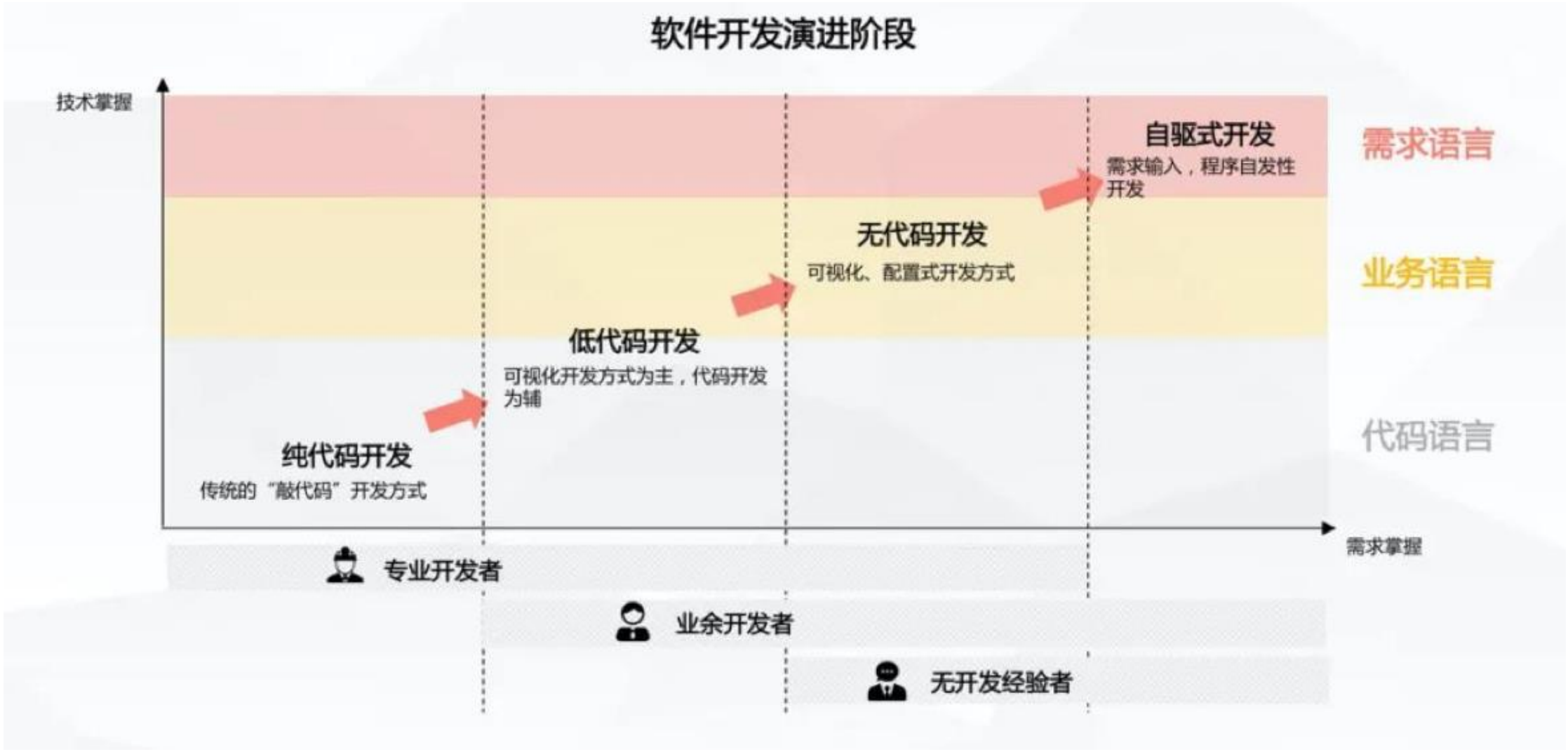
技术民主化的四个方面

- 根据高德纳预测，从2020到2023年技术民主化趋势将在以下4个方面迎来飞速发展：
  - **数据与分析技能**民主化（原本针对数据科学家的工具正在快速扩展，并逐步服务于普通开发人员社区）。
  - **开发技能**民主化（利用AI工具实现应用程序的定制化开发）。
  - **设计技能**民主化（通过低代码、无代码模式实现开发，并通过在应用程序中内置自动化开发功能增强开发者的能力）。
  - **知识**民主化（非IT专业人员也可以使用工具与专业级系统，使其得以利用超越自身专业知识与培训范畴的特定技能）。

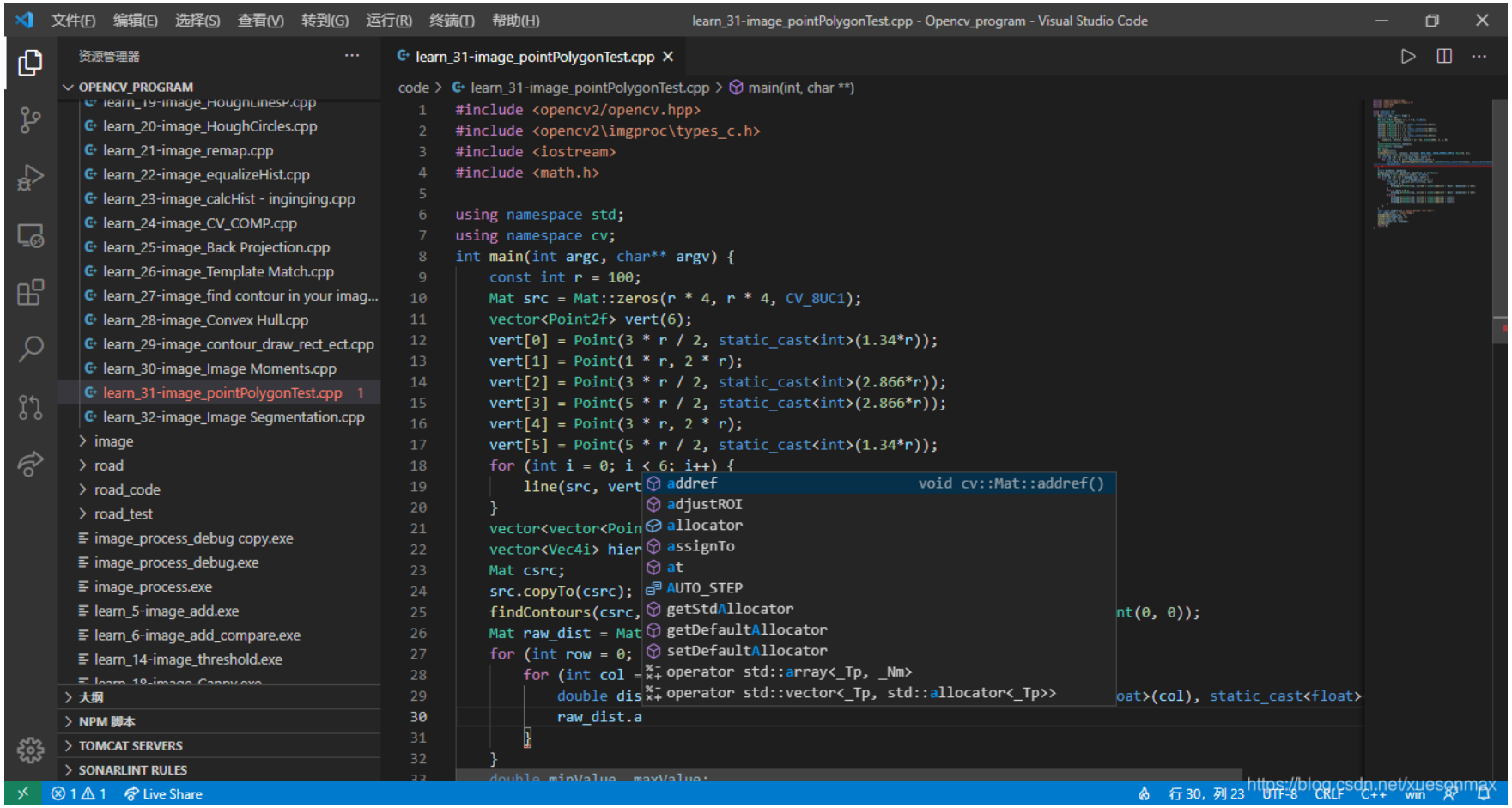
技术民主化的优点

- **提高员工生产力。**员工接触数字工具的机会越多，使用这些工具的技能越熟练，他们的工作效率就越高。
- **提升创新力。**新工具，如无代码平台，使员工能够以在引入这些工具之前不可能的方式进行创新。某些平台，如数字采用平台，使员工能够自动化工作流程，甚至创建与员工互动的“自动化机器人”。
- **增强敏捷性。**无论在哪里应用，技术都会提高效率和性能。组织内的技术越民主化，组织采用新工具、移动和转变的速度就越快。
- **提高灵活性。**数字颠覆是常态。无法采用新技术并迅速使其大众化的组织可能会被能够采用新技术的公司抢走市场份额。另一方面，已经构建了数字采用计划的组织可以更快地集成新技术，并在面对数字中断时保持弹性。
- **提高组织绩效。**能够将技术转化为资产——并提高其生产率、效率、创新和敏捷性——的组织将在市场中表现更好。
- **提高客户满意度。**技术民主化的主要目标之一是提高客户满意度。技术民主化为员工提供了有效完成工作的工具和资源。这将提高生产率，让客户更加满意。

2. 低代码/零代码是必然趋势

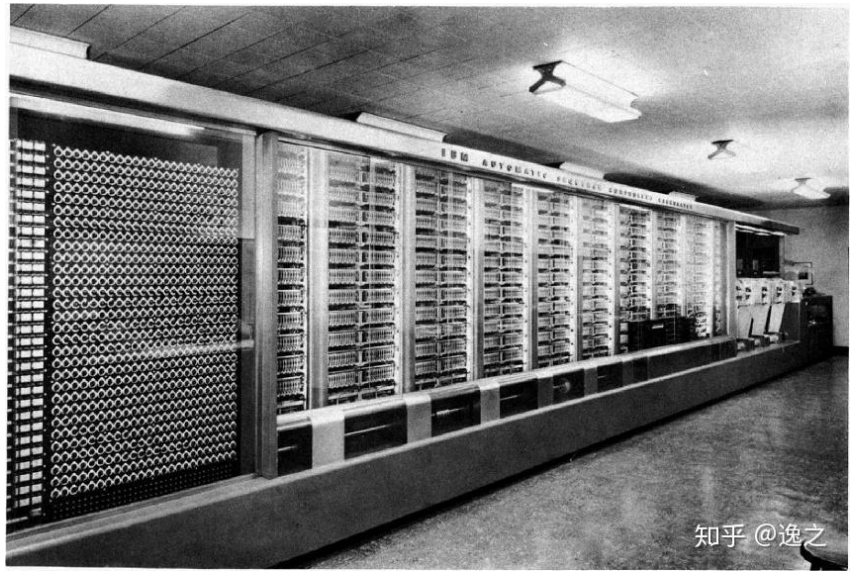


什么是全代码编程？
Full-code Programming



哈佛机 1944：穿孔纸带上的编程
Machine Language-Havard Mark I

- Mark I是当年的大型计算机，它由约765000个机电元件组成，内部电线总长达800公里。机器长约15.5米，高约2.4米，重达5吨，撑满了整个机房的墙面。
- 编程语言的设计者：霍华德·艾肯（Howard Hathaway Aiken）



操作	编程代码	穿孔
S10+C24（结果存入C24）	752 54 7	○●○●○○●○ ○○○●●○○○ ○●○○○○○○
用1号打字机打印S10	752 7432	○●○●○○●○ ○●○○●●●○ ○○○○○○○○
清空C24	54 54 7	○○○●●○○○ ○○○●●○○○ ○●○○○○○○
关闭2号打字机	8731	●●○○○●○● ○○○○○○○○ ○○○○○○○○
求C64的正弦值	7 7631	○●○○○○○○ ○●●○○●○● ○○○○○○○○

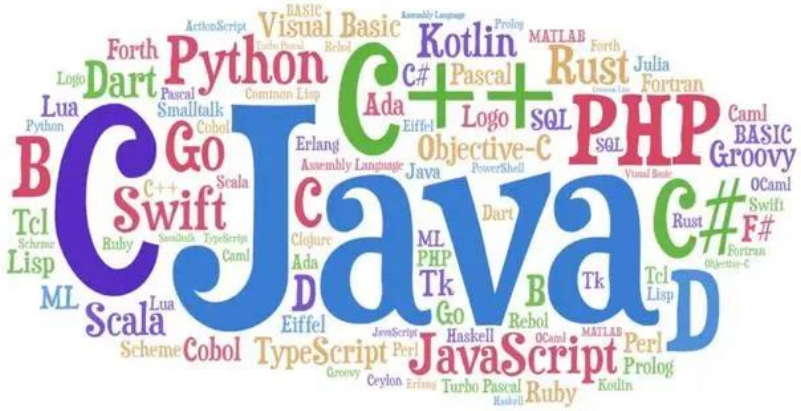
汇编语言
Assembly Language

- 汇编语言， 即第二代计算机语言，用一些容易理解和记忆的缩写单词来代替一些特定的指令，例如：用"ADD"代表加法操作指令，"SUB"代表减法操作指令，以及"INC"代表增加1，"DEC"代表减去1，"MOV"代表变量传递等等，通过这种方法，人们很容易去阅读已经完成的程序或者理解程序正在执行的功能，对现有程序的bug修复以及运营维护都变得更加简单方便。

```
C_S   SEGMENT
      ASSUME   CS: C_S, DS: C_S
S_T:
      MOV   AX, C_S
      MOV   DS, AX
      LEA   DX, P_S
      MOV   AH, 9
      INT   21H
      MOV   AH, 4CH
      INT   21H
P_S   DB    'Hello World!', 36
C_S   ENDS
      END    S_T
```

高级语言
High-Level Language

- 相对于汇编语言和机器语言而言更加抽象化的编程语言，它使用类似于英语的语法和结构，使得程序员可以更容易地理解和编写程序。
- 高级语言通常比汇编语言和机器语言更加易于阅读、编写和维护，适合用来编写大型复杂的应用程序。
- 高级语言的编译器或解释器可以将高级语言代码转化为机器语言或汇编语言，然后由计算机执行。 常见的高级语言包括C语言、Java、Python、Ruby、PHP等。



AI大语言模型的发展 + 低代码平台

- AI大模型学习各行业软件模板，成为业务系统设计专家；通过跟客户聊天交互，生成需求文档；基于文档，生成代码，完成软件开发。
- 只要通过拍照或者写一句话，不需要任何一行代码，就能自动生成一个业务应用。
- AI大模型+低代码，软件开发新范式；解决中小企业，90%的软件需求。

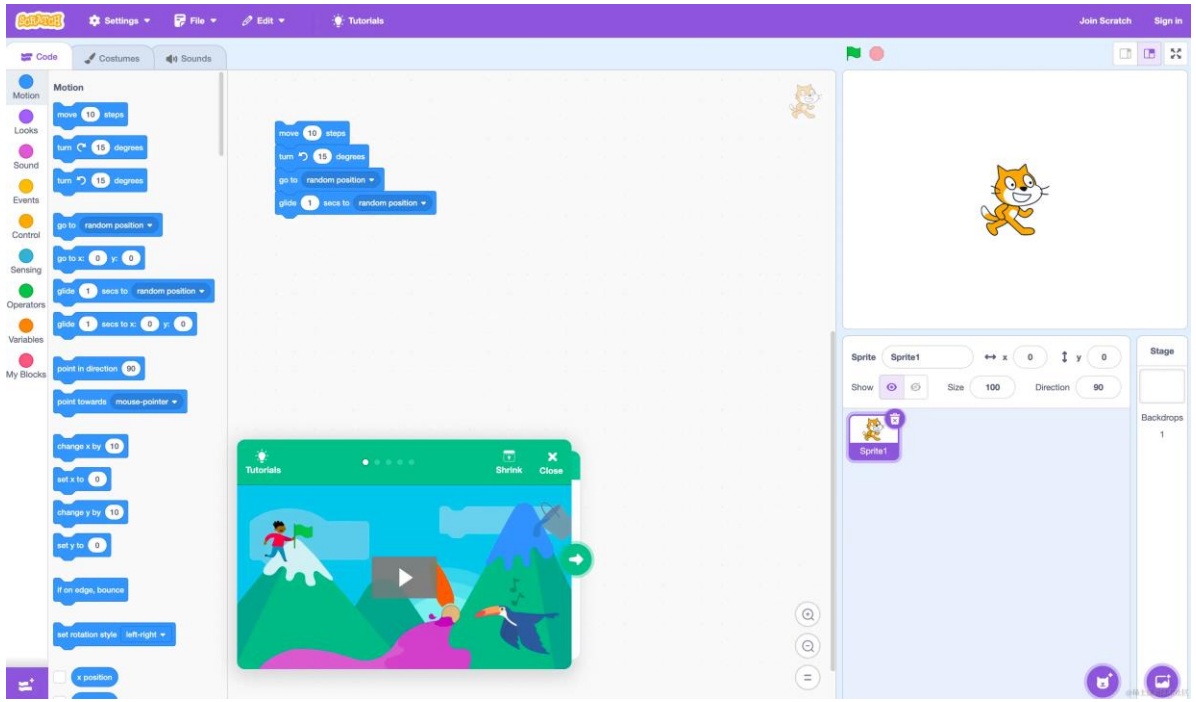
No Code AI

- No-code AI, also called codeless AI, is a category in the AI landscape that aims to democratize AI.
- No-code AI means using a no-code development platform with a visual, code-free, and often drag-and-drop interface to deploy AI and machine learning models.
- A wide range of tools provide no code AI capabilities, including dedicated no code AI tools, as well as some automation tools (e.g. some RPA software providers) that include integrated AI capabilities in a no-code user interface.
- No code AI enables non-technical users to quickly classify, analyze data and easily build accurate models to make predictions.

3. 其它一些低代码的例子
Some other examples of LCDP

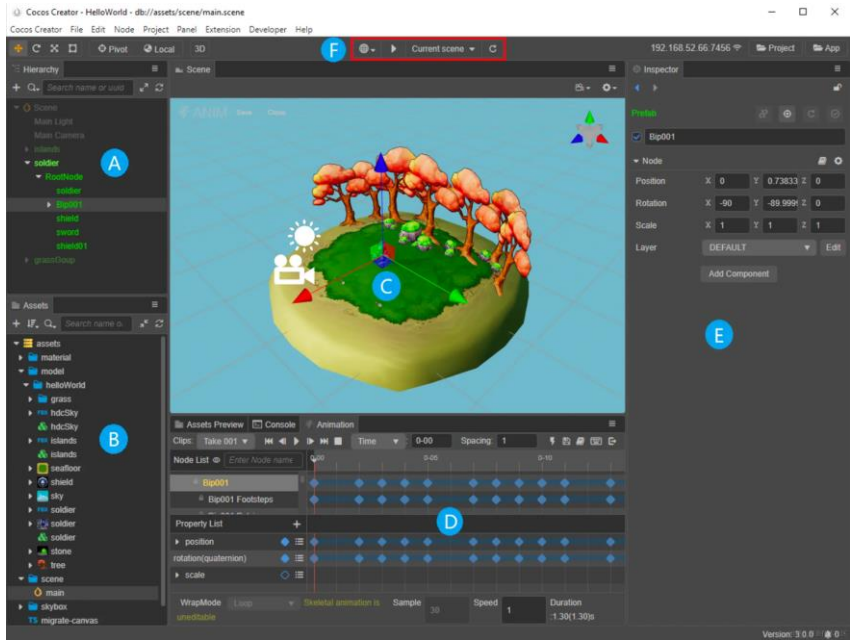
儿童编程软件 如**Scratch**

Kids' Programming Software

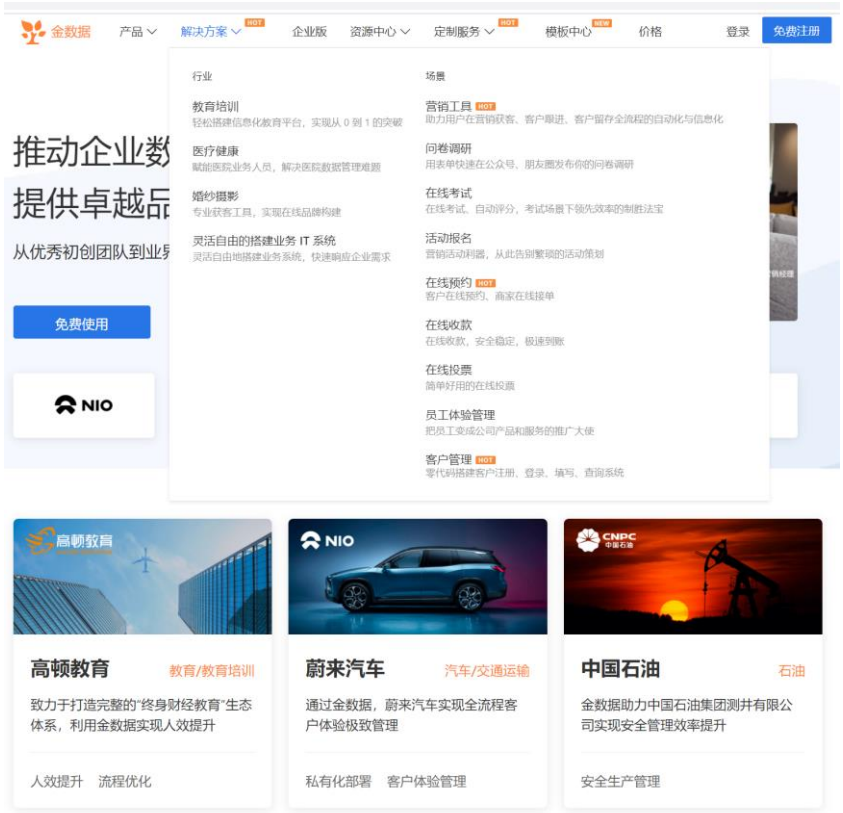


游戏开发软件 如**Cocos Creator**

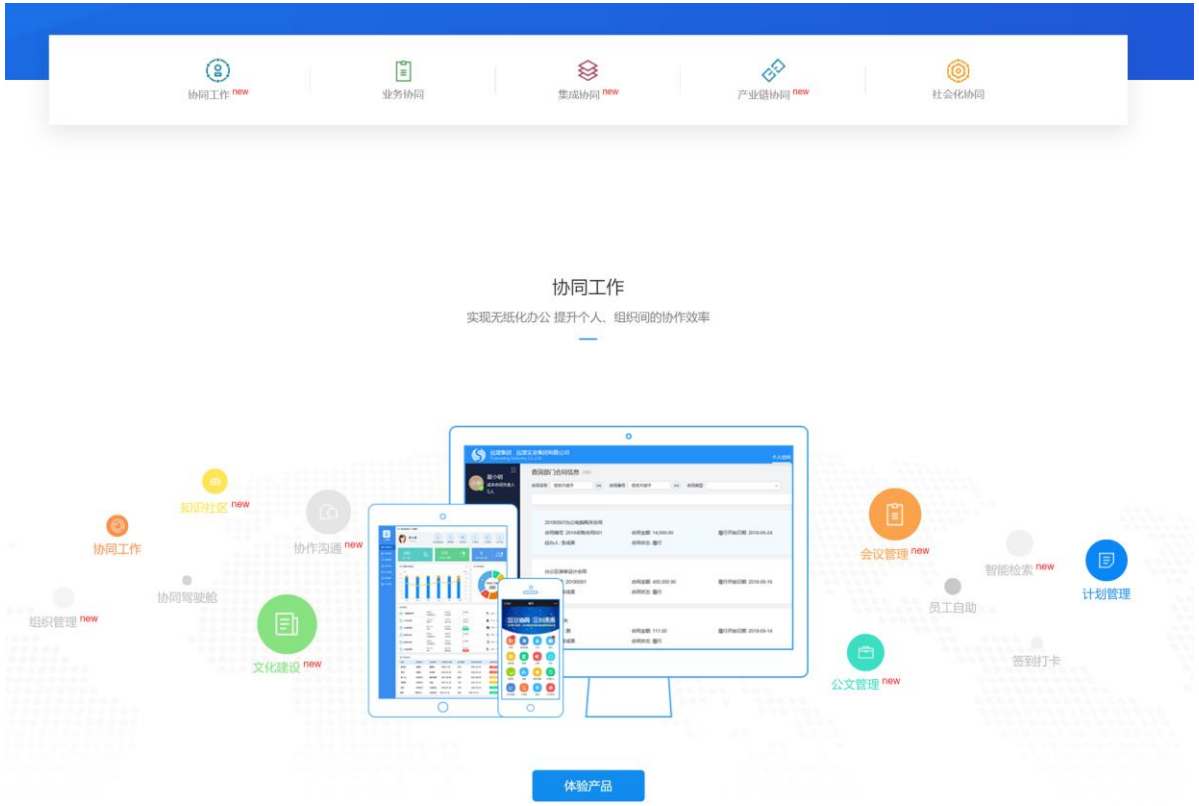
Game Development Tool



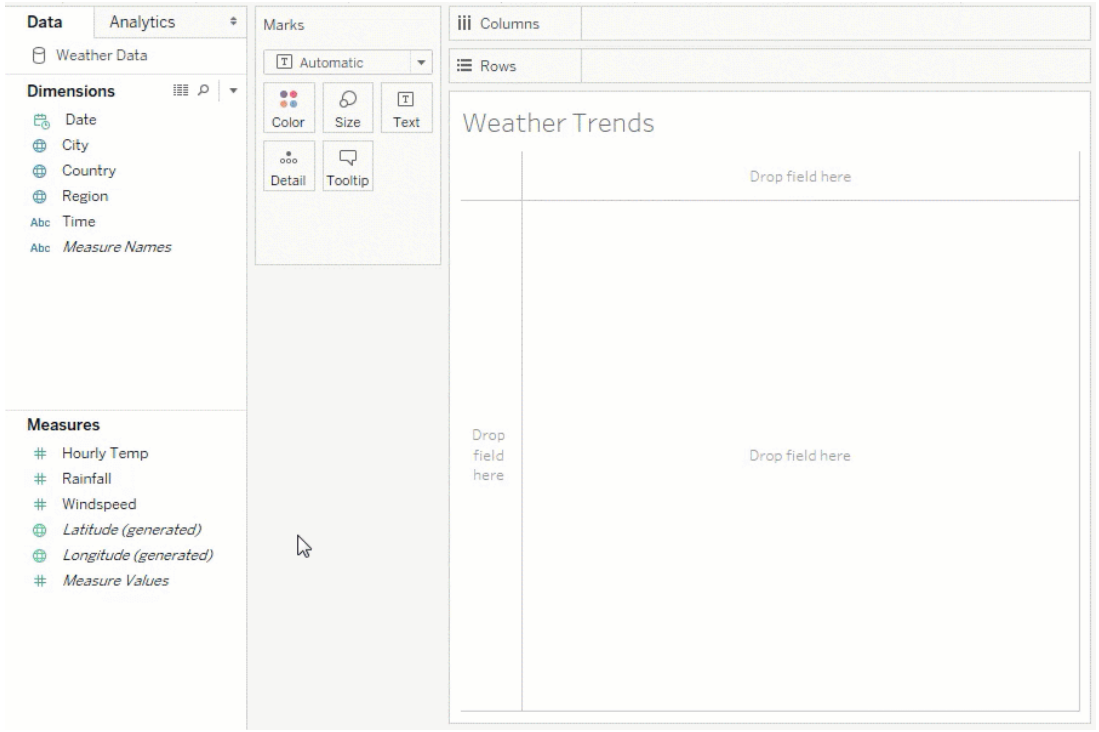
调查问卷和数据采集 如金数据
Survey and Data Collection



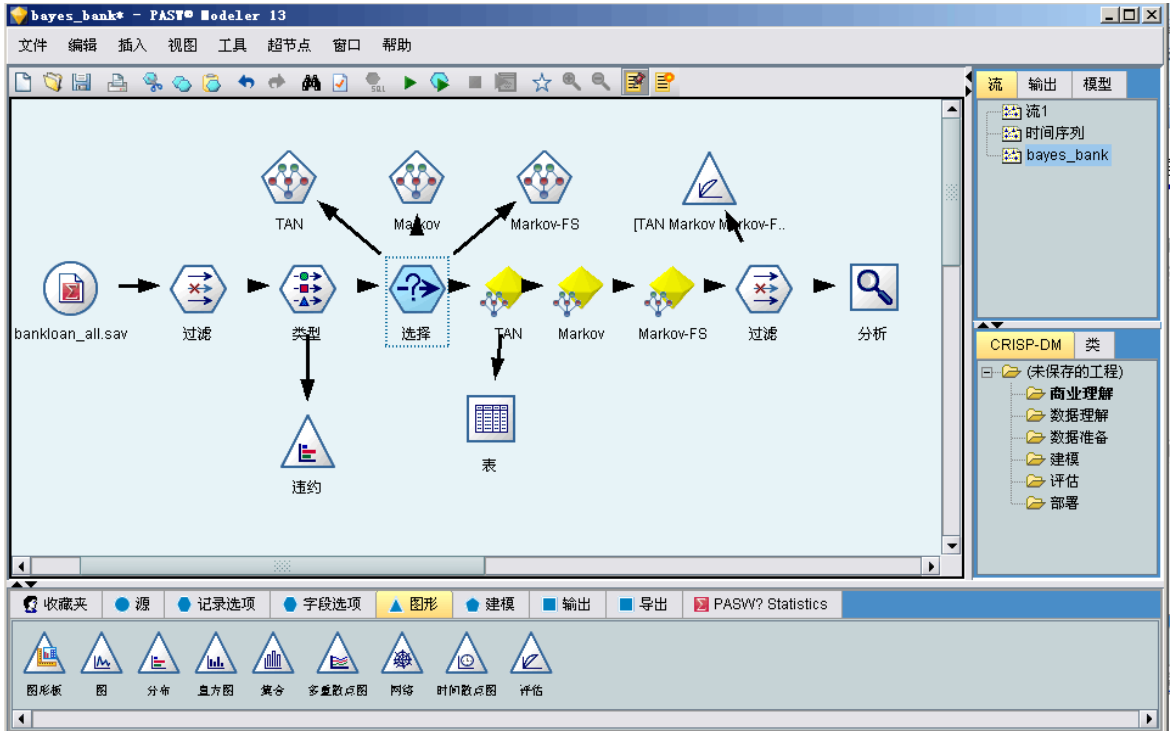
业务中台 如致远互联
Business Middle Platform



数据分析 如Tableau
Exploratory Data Analysis



机器学习 如Modeler
Machine Learning Modeling



5. 课后作业-1
Homework-1

1. 课后作业 Homework

- 学习使用一个低代码平台
- 每个组员分配一个系统中的角色
- 模拟业务流转的情况，开发一个系统
- 画简单的功能结构图、业务流程图
- 做PPT汇报+演示系统（10分钟左右）
- 各小组互相打分

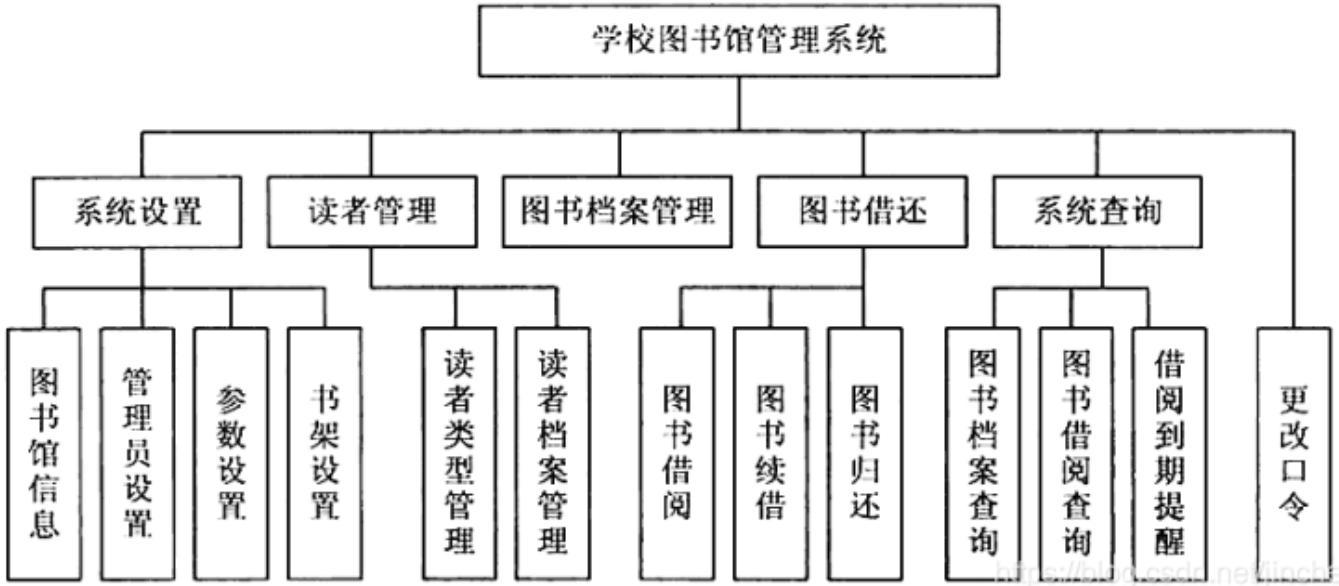
2. 低代码开发平台教程

- 低代码-宜搭-小课堂 （30分钟）
  https://www.bilibili.com/video/BV1rp4y1x7cz/

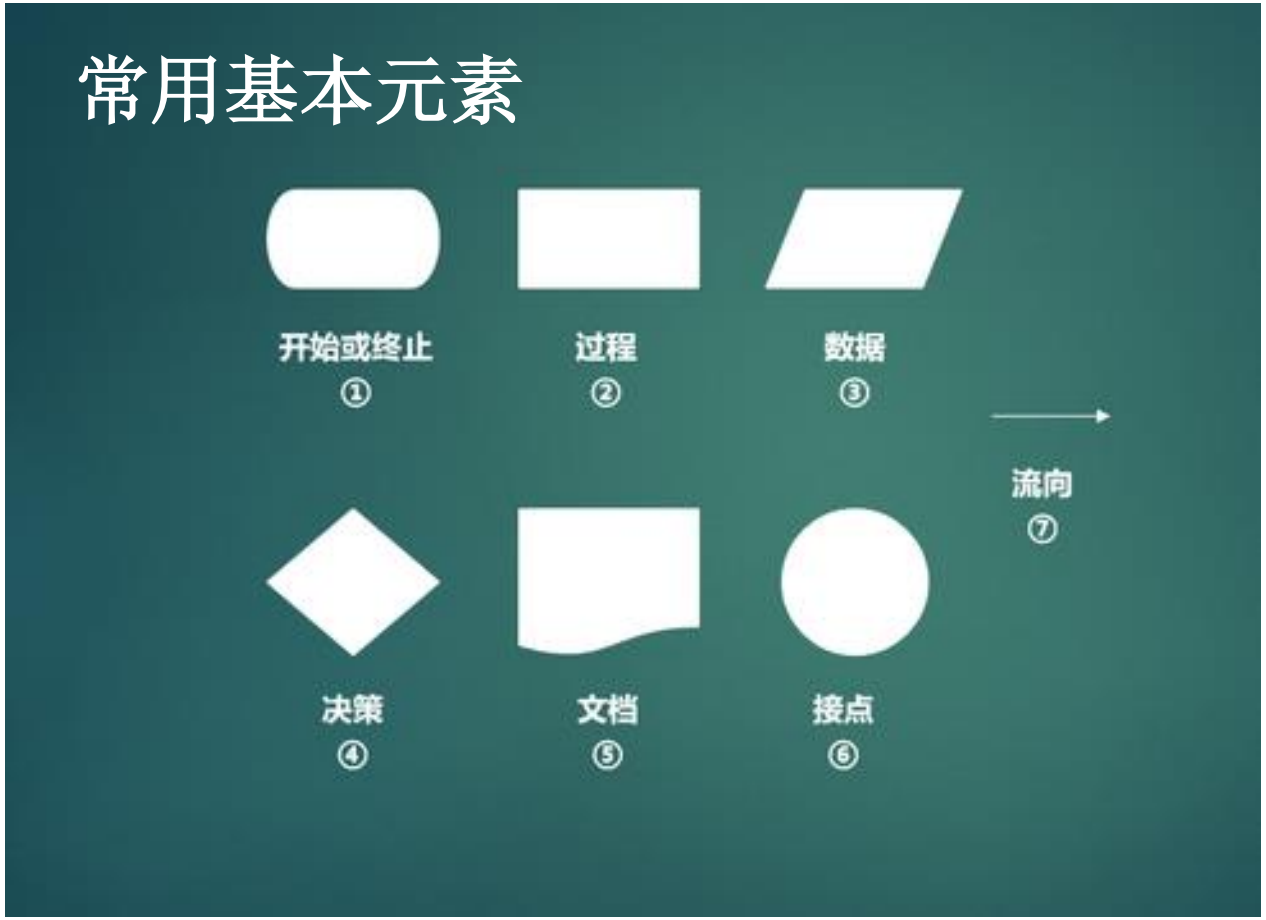
钉钉宜搭简单视频教程

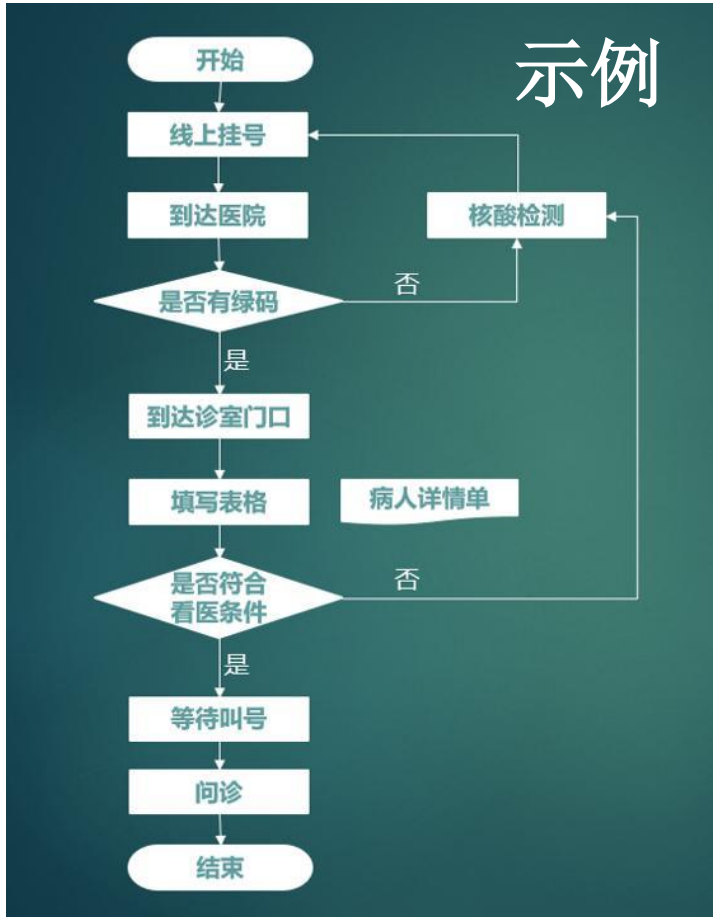
3. 功能结构图

- 功能结构图就是将系统的功能进行分解，按功能从属关系表示的图表。管理信息系统的各子系统可以看作是系统目标下层的功能，对其中每项功能还可以继续分解为第三层、第四层……甚至更多的功能。
- 功能结构的设计目的是分层次、解耦合。
- 类似于思维导图。



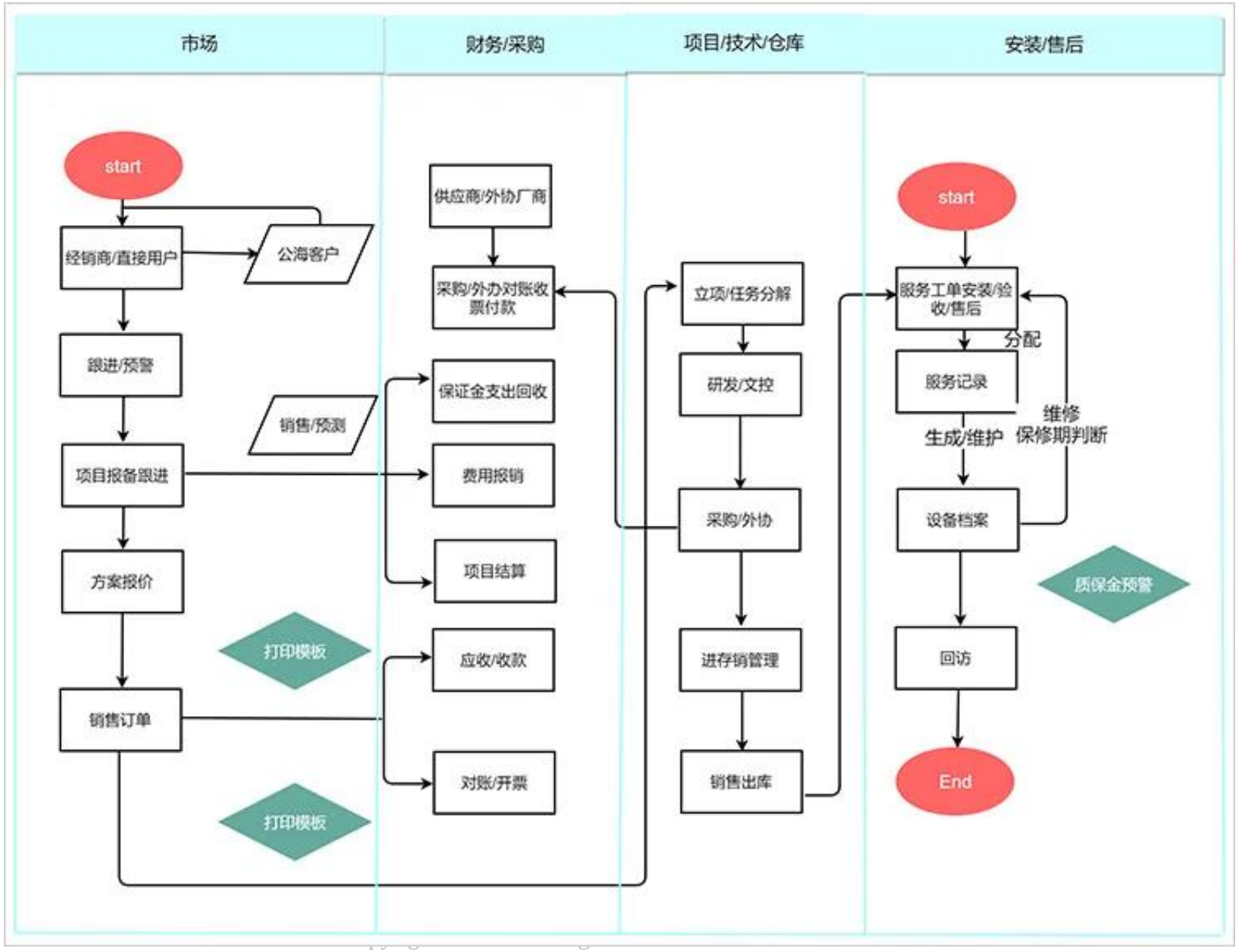
3. 业务流程图



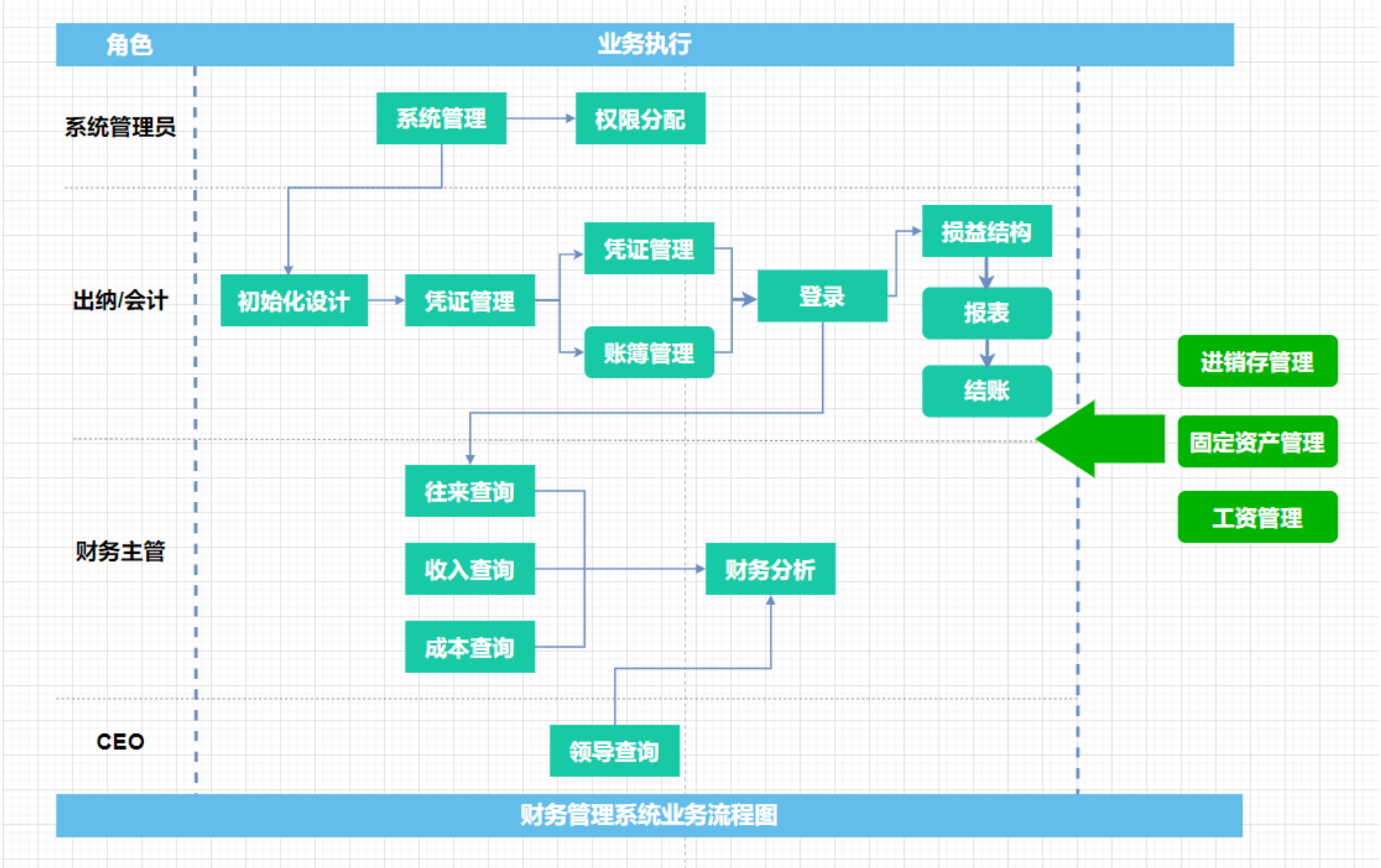


单一角色业务流程图

多种角色业务流程图


市场
财务/采购
项目/技术/仓库
安装/售后
start
经销商/直接用户
公海客户
跟进/预警
销售/预测
项目报备跟进
方案报价
打印模板
销售订单
打印模板
供应商/外协厂商
采购/外办对账收票付款
保证金支出回收
费用报销
项目结算
应收/收款
对账/开票
立项/任务分解
研发/文控
采购/外协
进存销管理
销售出库
start
服务工单安装/验收/售后
分配
服务记录
生成/维护
维修
保修期判断
设备档案
质保金预警
回访
End

多种角色业务流程图



业务流程图的一些说明

- 业务流程图没有固定的格式，但是要求能说明问题。
- 不要被流程图吓倒，流程图是核心关键！
- 复杂的、说不明白的业务流程图本身就能说明问题。
- 如果流程是靠人处理的，很可能会造成效率低下、扯皮。
- 超出业务流程图的（所谓的不规范）业务怎么办？

值得一提的是，西安一码通开发运维人员是西安本地的企业，现在其工作人员也因为无法扫码而且没有 48 小时核酸证明所以进不了公司，所以目前无法处理这一问题，总而言之似乎出现了一点比较尴尬的现象。[4]

谢谢！

Thank you for your attention.

liuyuewen@xjtu.edu.cn