《软件工程》

Ch2 软件工程概述

教育部“101计划”教材

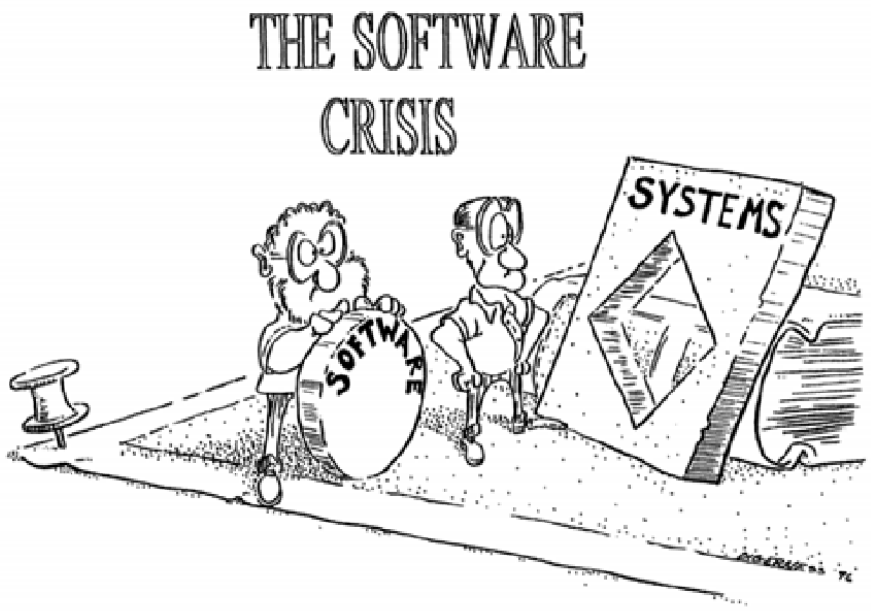
清华大学出版社

学习目标

- 理解**软件危机**的表现和根源；
- 理解**软件工程**概念、思想、目标和原则，了解软件工程发展历史；
- 理解计算机辅助软件工程概念，了解计算机**辅助软件工程工具**；
- 理解软件工程**从业人员**需遵守的法律、法规和职业道德，能够在软件开发中遵守相应的法律、法规和职业道德规范。

目录

1. 软件危机



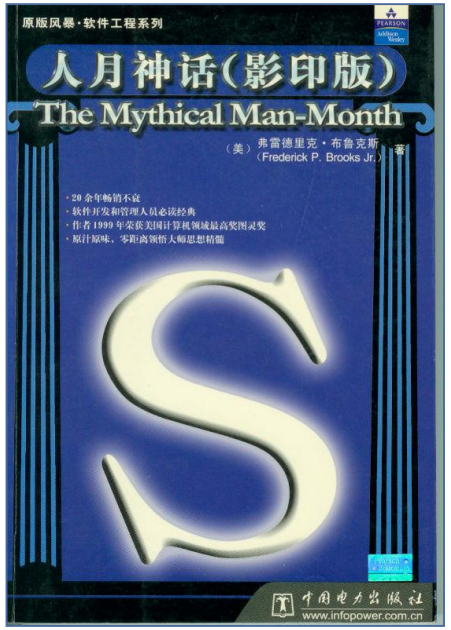
面临的问题

软件危机的表现

软件危机的根源

面临的问题

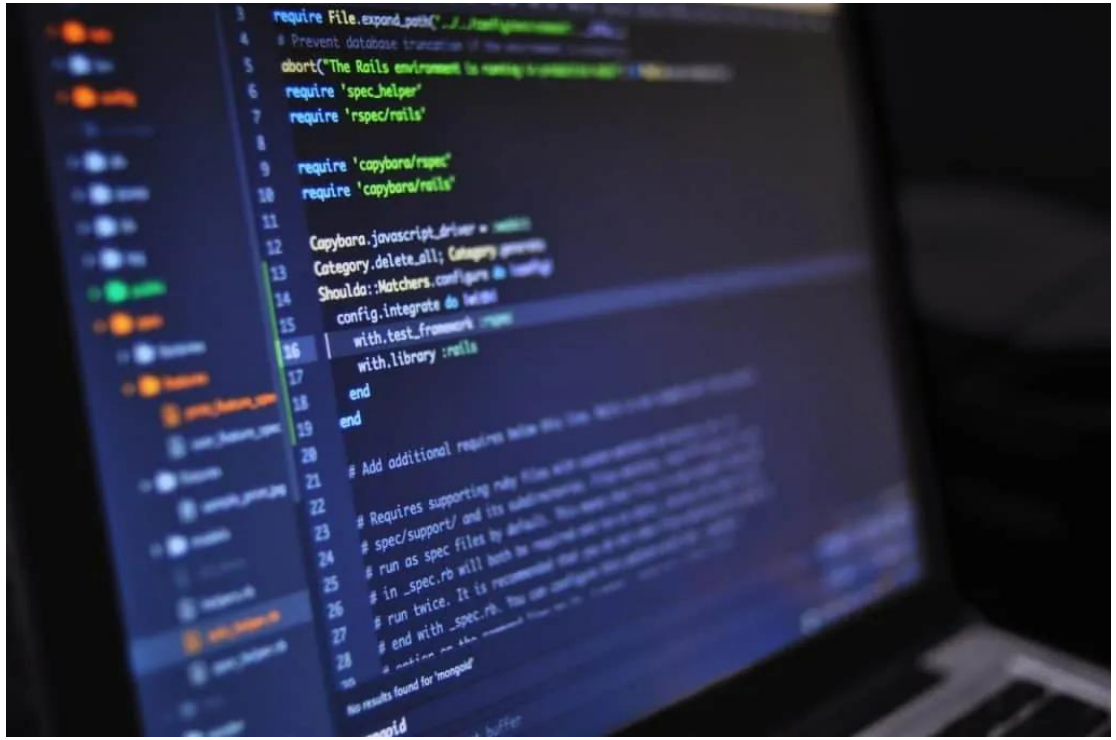
“正像一只逃亡的野兽落到泥潭中做垂死挣扎，越挣扎，陷的越深，最后无法逃脱灭顶灾难，…程序设计工作也正像这样一个泥潭，… 一批批程序员被迫在泥潭中拼命挣扎，… 谁也没有料到问题竟会陷入这样的困境…”



费雷德里克.布鲁克斯

面临的问题

- 软件开发技术更新速度快
- 软件开发需求变更频繁
- 软件开发复杂度高
- 软件开发需要团队协作完成
- 软件开发成果需要长期维护
- 软件开发保证软件质量
- 软件开发需要评估和控制风险

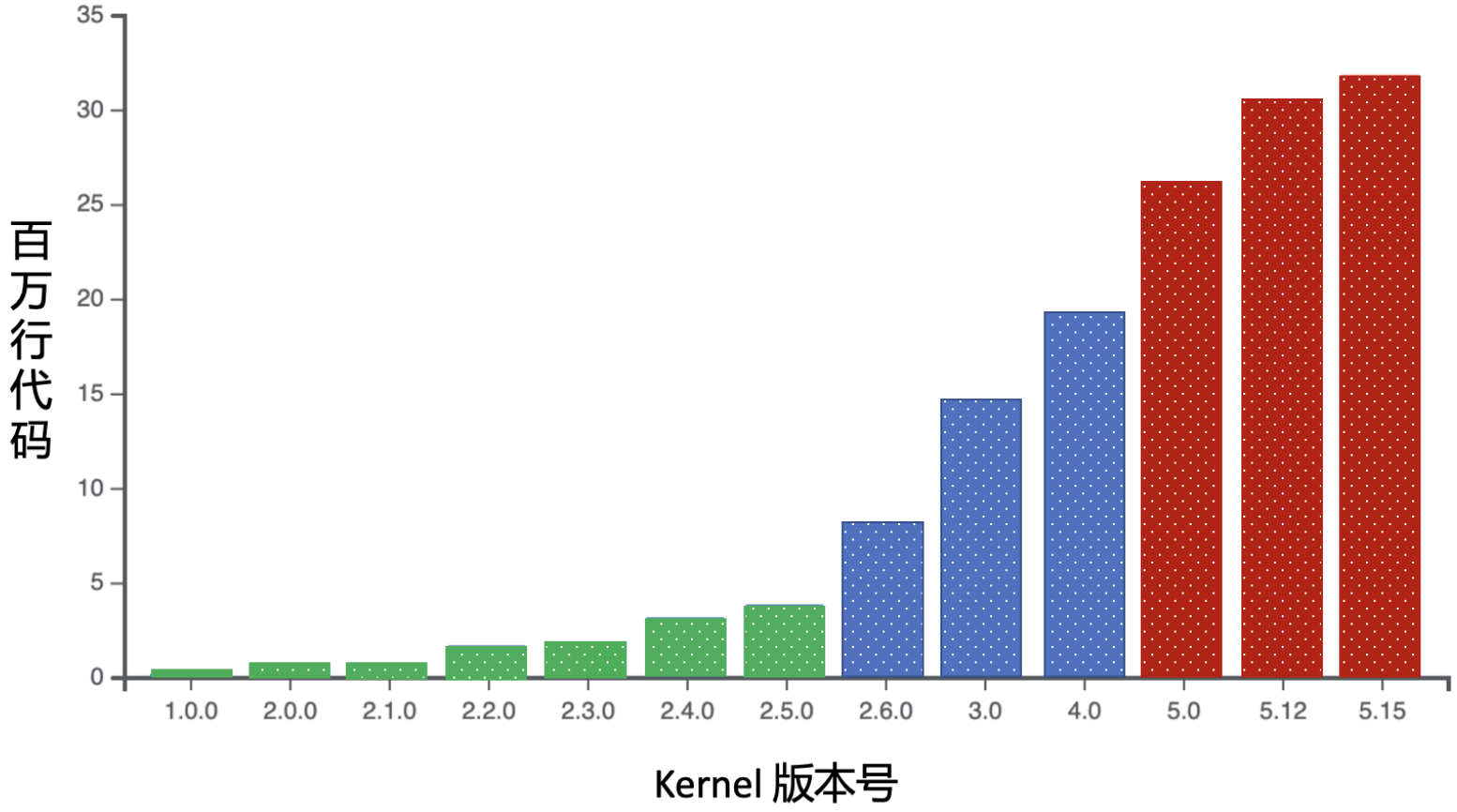


软件危机表现

- 用户对交付软件不满意
- 软件质量不可靠
- 项目超预算或延误

软件危机的根源

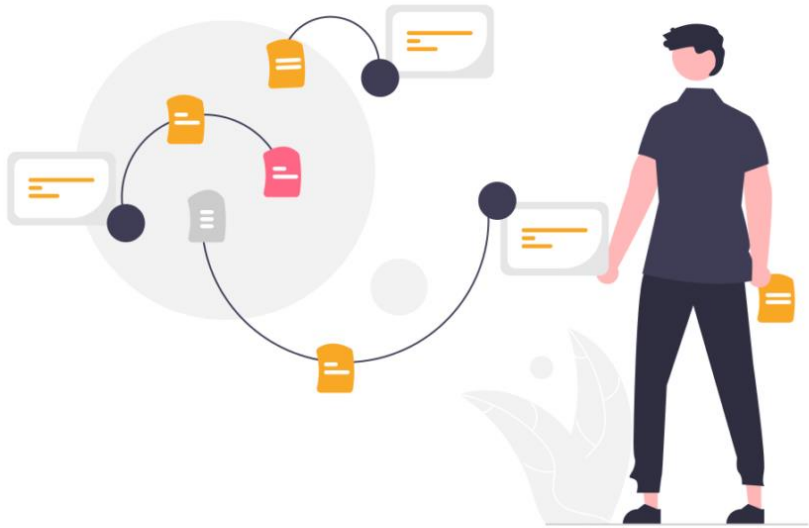
❑软件复杂度高



软件危机的根源

❑软件开发技术问题

- ✓过时的软件开发方法仍被沿用
- ✓设计、架构、测试等难题



软件危机的根源

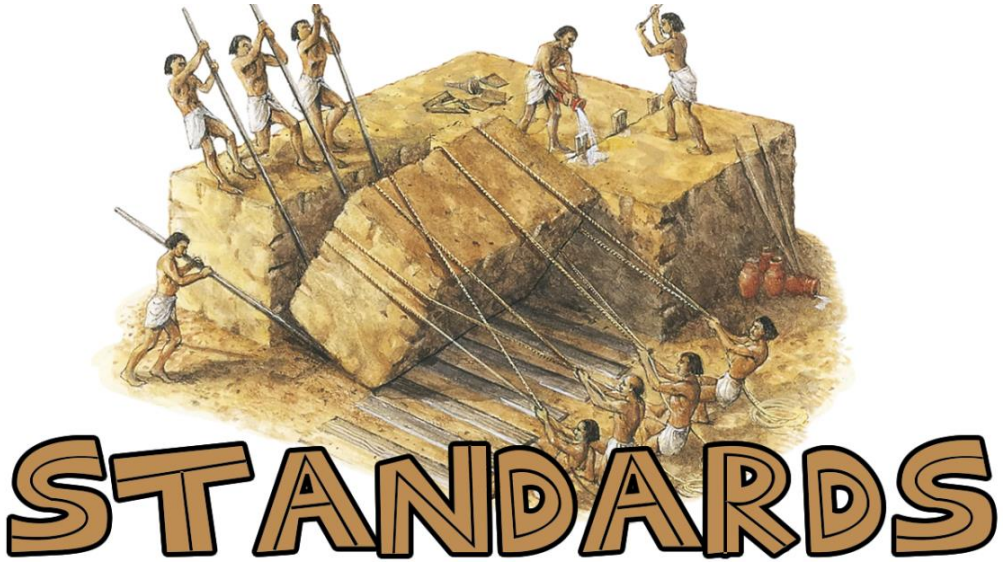
- **组织管理问题**
  - ✓需求不清晰
  - ✓进度不可控
  - ✓沟通不畅

软件危机的根源

- **市场需求问题**
  - ✓市场需求不断变化
  - ✓软件需要快速适应
  - ✓开发周期变短
  - ✓开发成本变高

软件危机的根源

- 缺乏标准和规范
  - ✓缺乏标准化文档
  - ✓缺乏标准化编码规范
  - ✓缺乏标准化开发流程



2. 软件工程概念

软件工程的诞生

软件工程的定义

软件工程发展史

软件工程发展的特点

软件工程的诞生

1968年

德国加米施

NATO会议科技委

如何解决软件危机

第一次提出“软件工程”

软件工程的诞生

- 1968年，一群计算机科学家和相关领域的专家们齐聚德国加米施，参加著名的北约软件工程会议。
- 与会者包括我们耳熟能详的Edsger Dijkstra、CAR Hoare、Alan Perlis、Peter Naur和Niklaus Wirth等。
- 这次会议的主题是“软件危机”——世界越来越依赖软件，但软件系统越来越复杂，项目超期、超预算且软件系统难以修改。专家们齐聚一堂讨论了导致这些问题的原因、可能的解决方案及相关技术和方法，并提出将软件的创建归于工程学范畴。
- NATO会议基于会议讨论和论文摘录出版了一份100多页的报告，其标题即为《软件工程》。
- 报告指出，选择“软件工程”这个术语意味着软件制造和传统的工程领域一样，既需要理论基础也需要实践经验。

软件工程的定义

- **IEEE STD 61012-1990 软件工程术语标准：**
  - ✓软件工程是 “将系统化的、规范的、可量化的工程方法应用于软件的开发、操作和维护。”
- **Barry Boehm**
  - ✓软件工程是“运用现代科学技术知识来设计并构造计算机程序及为开发、运行和维护这些程序所必需的相关文件资料。”
- **《计算机科学技术百科全书》中定义**
  - ✓软件工程是“应用计算机科学、数学及管理科学等原理，开发软件的工程。软件工程借鉴传统工程的原则、方法，以提高质量、降低成本。其中，计算机科学、数学用于构建模型与算法，工程科学用于制定规范、设计范型、评估成本及确定权衡，管理科学用于计划、资源、质量、成本等管理”。

软件工程的定义

- 软件工程是一门研究如何用系统化、规范化、数量化等工程原则和方法进行软件开发和维护的学科
- 其目标是创造“**足够好**”的软件
  - ✓低成本、高质量、按时交付
- 其内容包括市场调研、正式立项、需求分析、项目策划、概要设计、详细设计、编程、测试、试运行、产品发布、用户培训、产品复制、销售、实施、系统维护和版本升级等。
- 软件工程不仅仅关注软件开发的技术过程，也关注软件项目管理以及支持软件开发的工具、方法和理论。

软件工程发展历史

❑20世纪40年代到60年代：史前时期

- ✓40年代：科学和军事项目，开发人员是科学家和工程师，使用低级机器语言和汇编语言，没有太多协作
- ✓50~60年代：SAGE防空系统和NASA大型项目，出现了第一种高级程序语言FORTRAN，第一个操作系统，基于IBM704的操作系统GM-NAA I/O。
- ✓这一时期软件开发的主要问题是开发成本高、难度大，缺乏系统化的方法。但这一时期出现的高级编程语言和操作系统等基础技术，为后来出现的更复杂的软件开发实践奠定了基础。

软件工程发展历史

- **20世纪70年代到80年代：软件工程的诞生和兴起**
  - ✓软件危机
  - ✓NATO软件工程会议
  - ✓Edsger W. Dijkstra提倡使用结构化编程技术
  - ✓一系列软件工程方法论的发展：如瀑布模型、软件成本估算模型
  - ✓软件工程在这个时期开始成为一门正式学科，并有了显著的发展。其标志是结构化编程技术、软件工程方法论和软件成本估算模型的出现，这些模型为更复杂的软件开发实践奠定了基础。

软件工程发展历史

❑20世纪90年代：软件工程的成熟

- ✓这一时期见证了面向对象编程OOP的发展，开发人员能够创建可重用的代码并更轻松地构建复杂的软件系统。统一建模语言UML被开发出来，提供了一种标准化的方式来记录和交流软件设计。一些软件开发过程模型（如RUP）和过程改进模型（如CMM）是软件工程专业化的重要一步，有助于为行业建立通用语言和最佳实践集合。

软件工程发展历史

❑2000年至今：软件工程面临新挑战

- ✓**敏捷开发方法**：软件开发出现了一系列新特点，包括追求创新、快速响应用户变化、需求不确定性高、版本发布成本低且更新速度快。敏捷开发方法应运而生，如快速应用程序开发(RAD)、动态系统开发法(DSDM)、SCRUM、CrystalClear、极限编程法(XP)、功能驱动开发(FeatureDrivenDevelopment)等。
- ✓**DevOps**：DevOps 是一种强调协作、自动化和持续改进的软件交付整体方法，其目标是更快、更可靠地交付高质量的软件。DevOps 的实践包括，持续集成和部署（CI/CD）、基础设施即代码、任务自动化、监控和日志等。

软件工程发展历史

❑2000年至今：软件工程面临新挑战

- ✓**云计算**：在云计算的架构下，开发者无需购买和维护昂贵的硬件设备，只需通过租赁云服务提供商的服务器、存储、数据库等资源，即可快速构建起自己的软件开发环境。这种灵活性和可扩展性使得软件开发更加便捷，开发者可以根据项目的实际需求快速调整资源分配，从而提高开发效率。
- ✓**大数据**：随着生成的大量数据及其使用率的不断提高，开发人员不得不创建新的技术和工具来处理和分析这些数据。这一现状不仅催生了一批新的软件架构，例如Hadoop和Spark，也使得一些擅长处理数据和进行统计分析的的编程语言，如R和Python，变得流行。总的来说，大数据为软件开发带来了新的挑战，要求开发人员考虑管理和处理大数据集的新技术和工具。

软件工程发展历史

❑2000年至今：软件工程面临新挑战

- ✓**人工智能**技术，特别是以ChatGPT为代表的大语言模型技术，以其强大的自然语言生成能力，为软件开发与演化提供了新的途径。在软件开发的各个阶段，大模型可以通过自然语言交互的方式，帮助开发人员更好地理解需求、生成代码、分析代码等。
  - ❑需求分析与建模
  - ❑代码生成与辅助编程
  - ❑代码分析与测试
- ✓由于大模型本质是基于概率与统计原理和训练数据所形成的数学模型，具有不可解释性和不确定性，其生成的预测性内容缺失可信性判断。软件开发人员要正确分析、理解和确认大模型生成的预测性内容，并在此基础上完成决策性任务，进而开发出高质量的软件系统。

软件工程发展的特点

- **软件抽象的层次越来越高，软件重用粒度越来越大**
  - ✓二进制编码、结构化编程、结构化开发方法学、面向对象编程、面向对象开发方法学,……
  - ✓面向功能、面向对象、面向构件、面向服务、……
- **软件开发理念的不断变化**
  - ✓以文档为中心与以代码为中心
  - ✓从个体、团队到群体的开发组织
  - ✓从还原论到演化论

3. 计算机辅助软件工程及工具

工欲善其事必先利其器

计算机辅助软件工程的概念

计算机辅助软件工程的工具

计算机辅助软件工程的概念

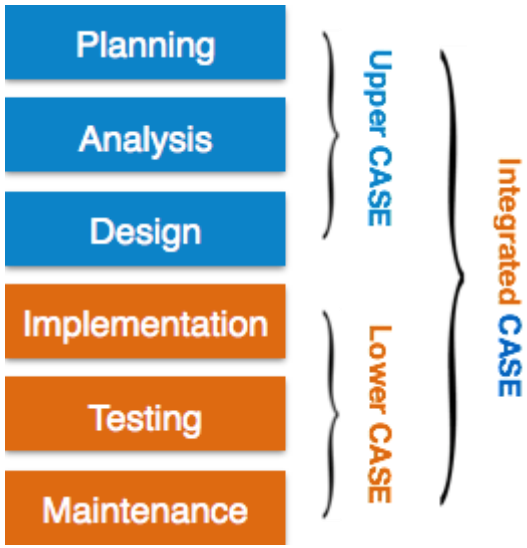
❑计算机辅助软件工程

(Computer Aided Software Engineering，CASE)

- ✓起源于20世纪80年代，最初是指在信息管理系统开发过程中由各种计算机辅助软件和工具组成的软件开发环境。
- ✓随着软件工程技术、工具和开发理念的不断发展，CASE逐步演进成为**辅助软件工程全生命周期的开发工具和方法集合**
- ✓旨在帮助软件工程从业者们进行软件开发和维护，**提高软件开发和运维效率**，**提升软件质量**，为实现软件开发全生命周期的**工程化**、**自动化**和**智能化**提供基础支撑。

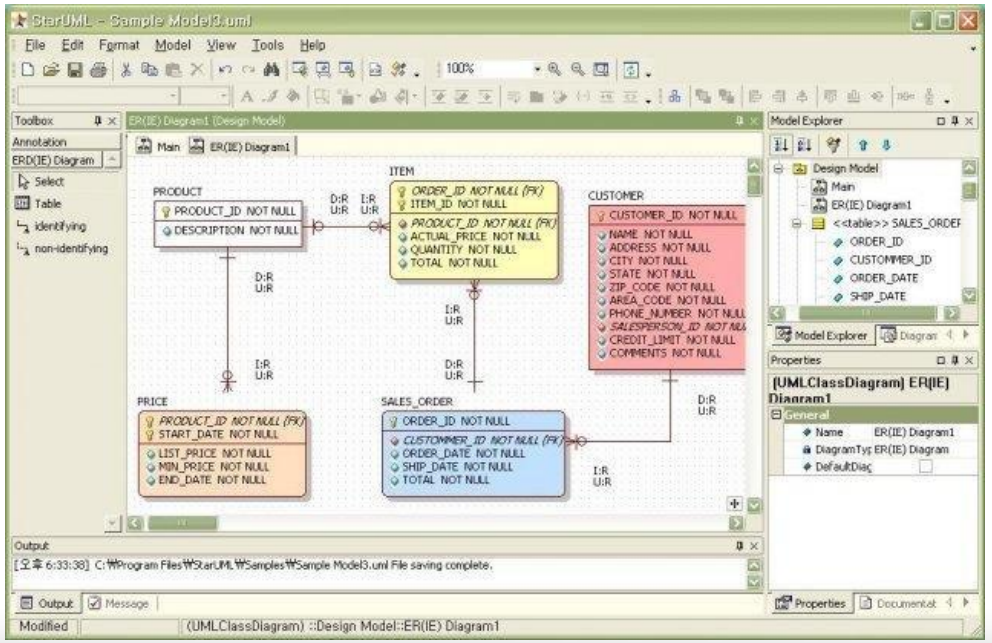
CASE工具

- 系统分析与设计工具
- 程序设计工具
- 软件测试与质量保证工具
- 项目管理工具
- 软件运维工具
- 一体化集成开发环境



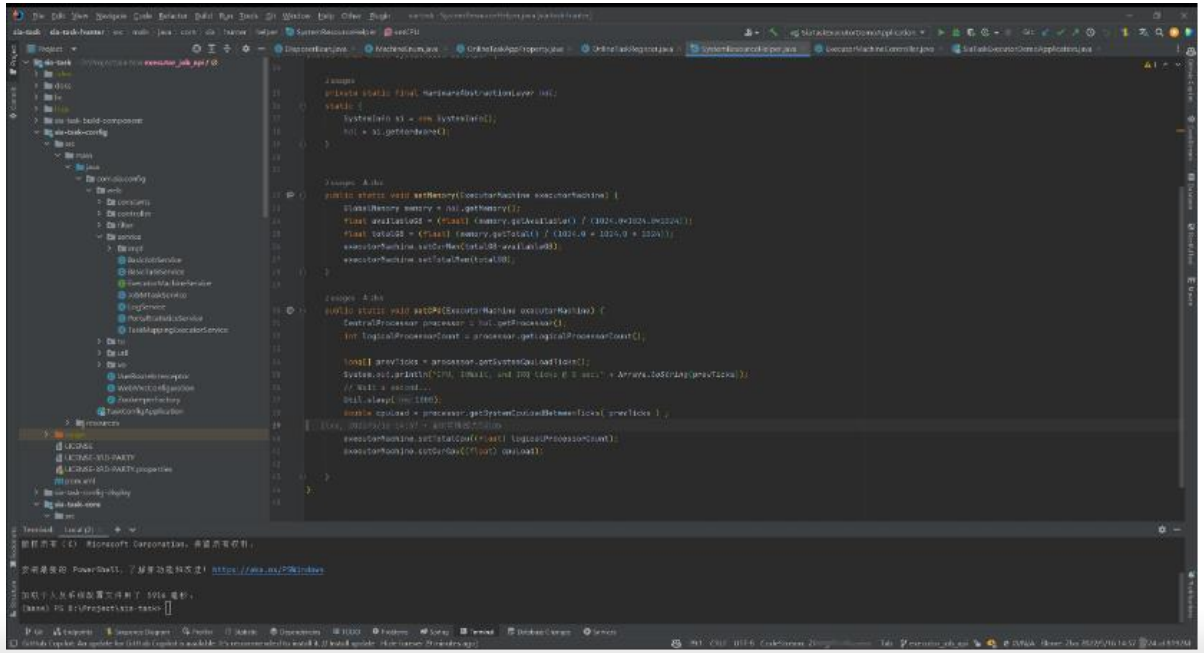
系统分析与设计工具

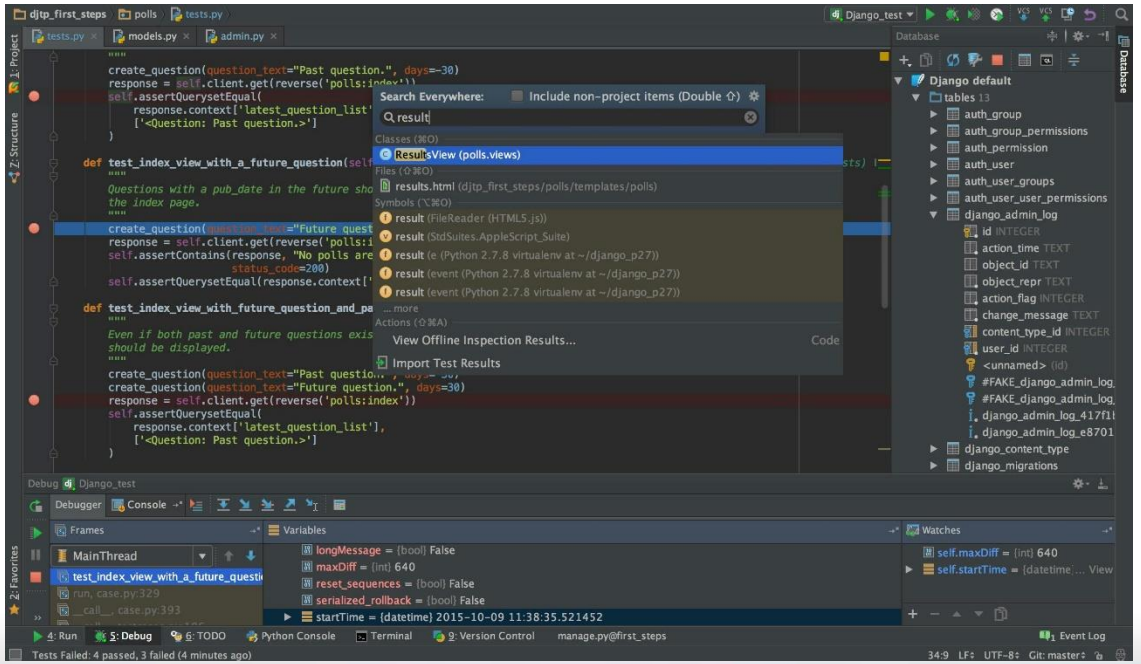
- 功能：可视化建模，需求分析
- 利益相关方：需求分析人员，软件设计人员
- 代表性工具
  - ✓ Microsoft Visio、Rational Rose、StarUML、Visual Paradigm



程序设计工具

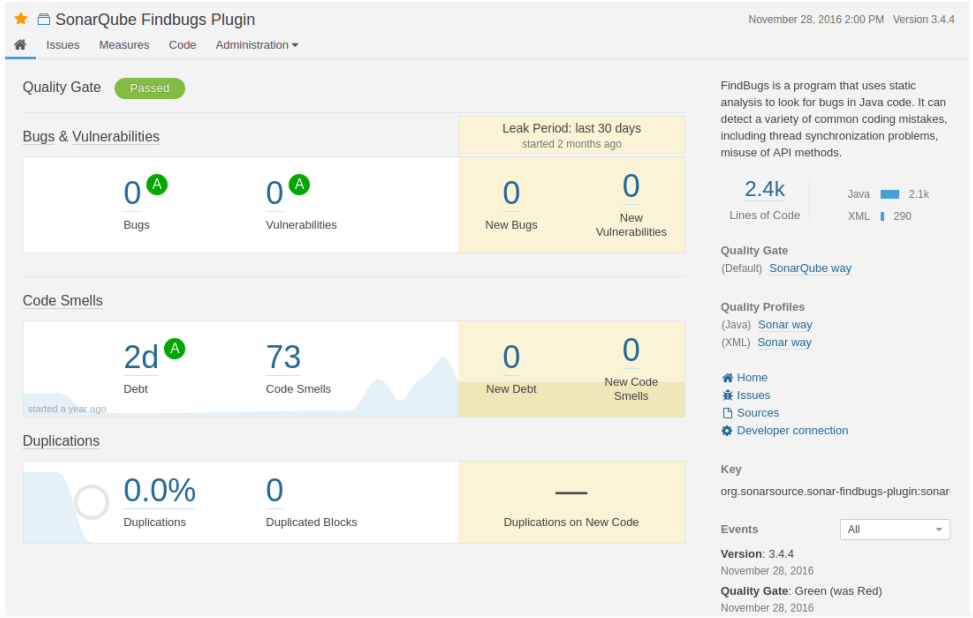
- 功能：代码编写，代码补全，程序调试，程序调试
- 利益相关方：编程人员，维护人员
- 代表性工具
  - ✓IntelliJ IDEA，Visual Studio，Eclipse，PyCharm

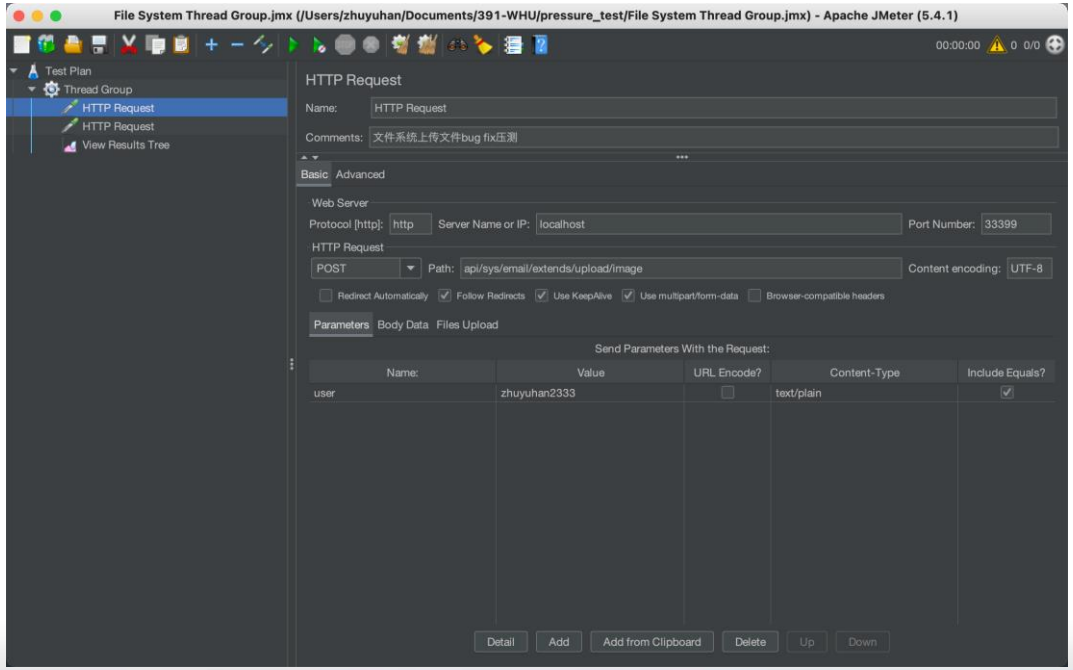




软件测试与质量保证工具

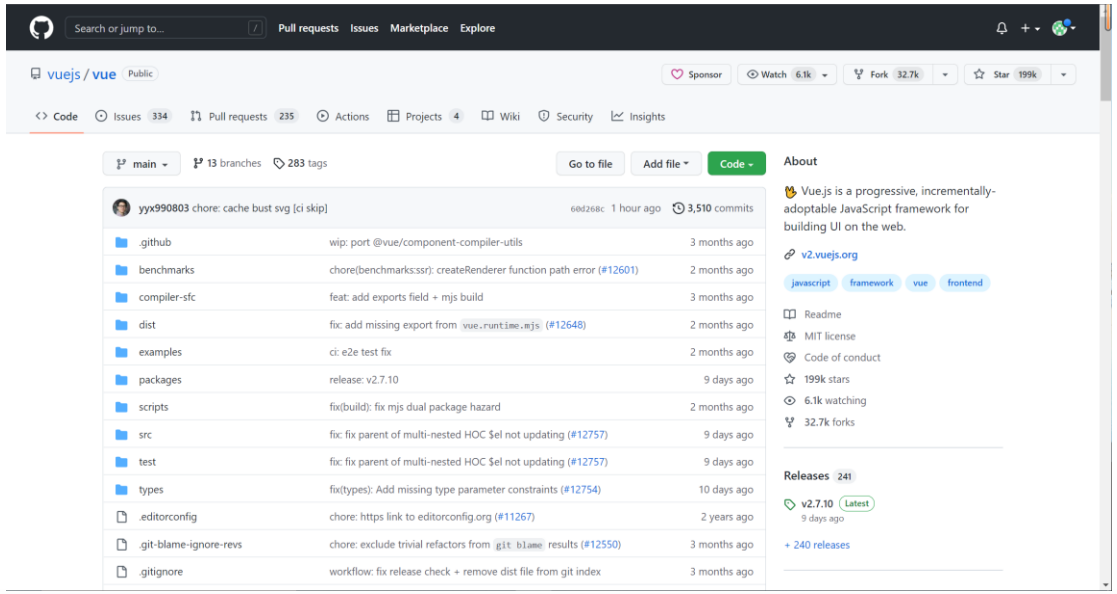
- 功能：单元测试，集成测试，性能测试
- 利益相关方：编程人员，测试人员
- 代表性工具
  - ✓SonarQube，CodeClimate，Jmeter，JaCoCo，Junit





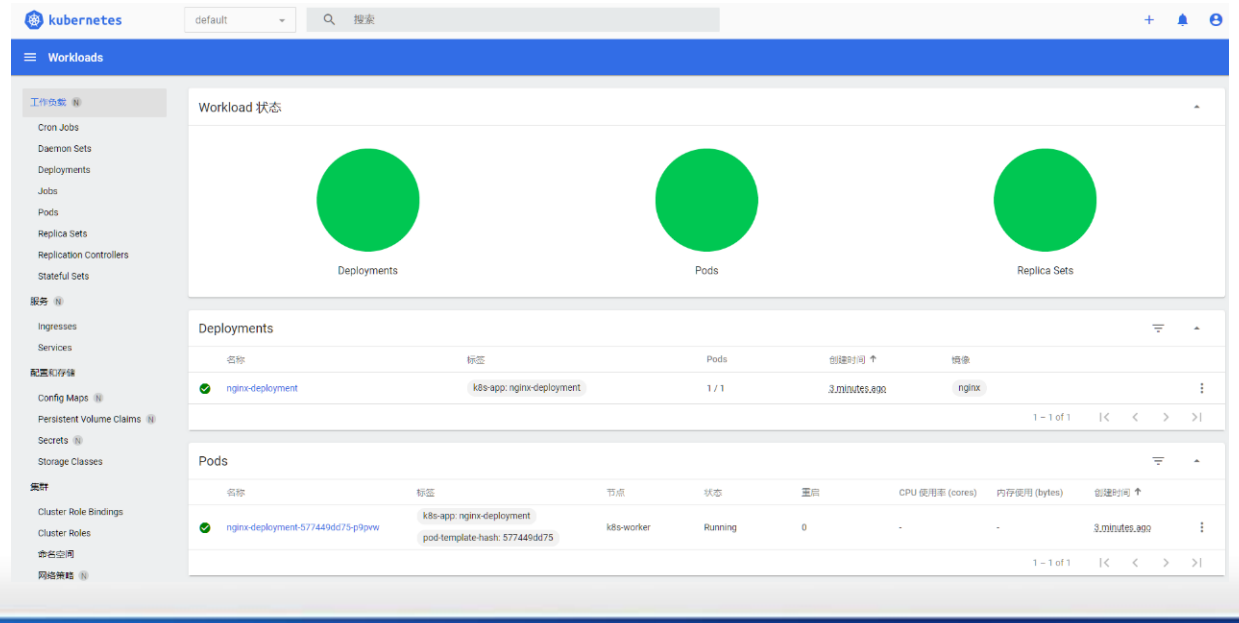
项目管理工具

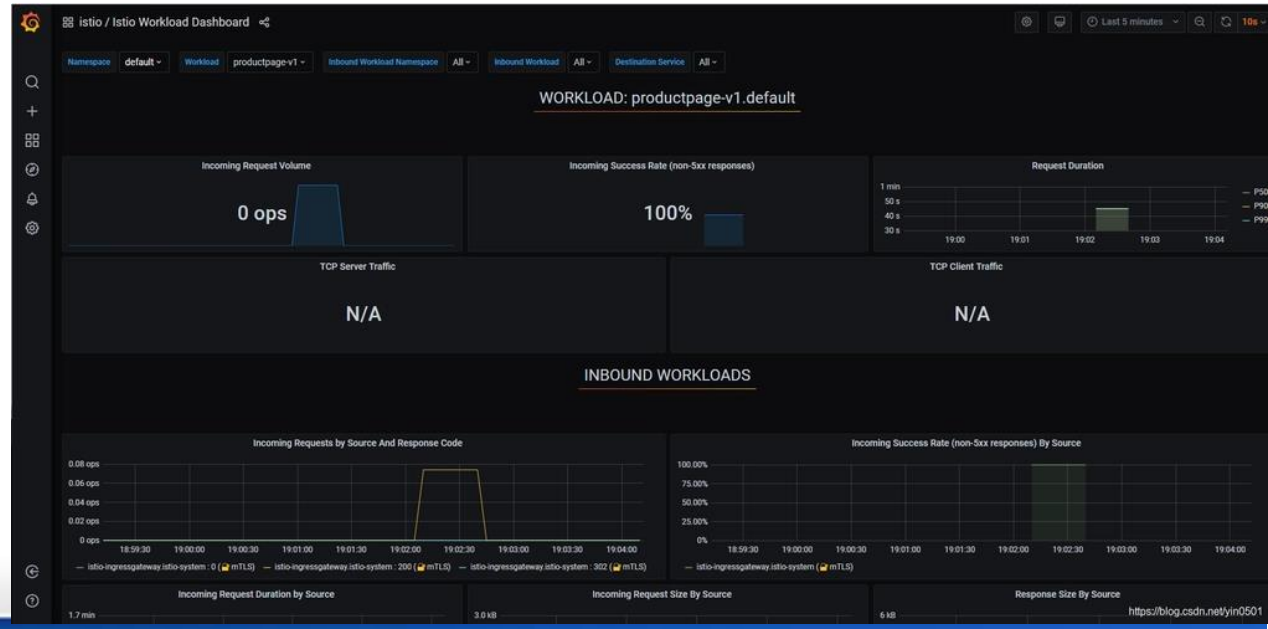
- 功能：代码和文件托管，团队协作
- 利益相关方：开发人员，管理人员
- 代表性工具
  - ✓GitHub，GitLab，Gitee



软件运维工具

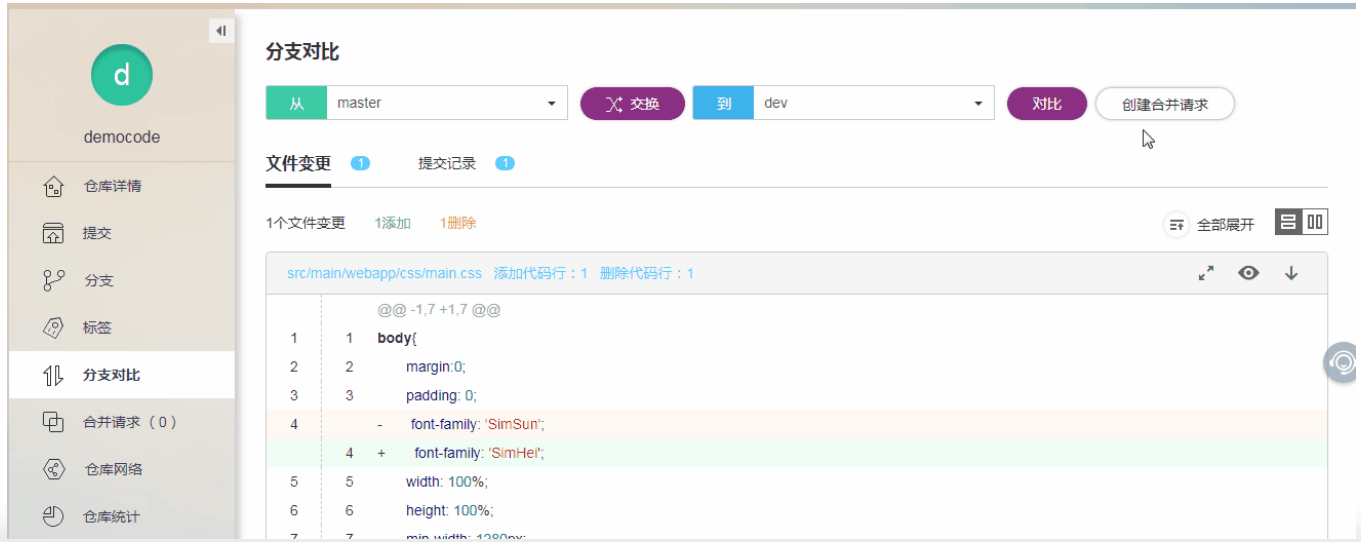
- 功能：应用程序部署，管理，监控
- 利益相关方：开发人员，维护人员
- 代表性工具
  - ✓Docker，Kubernetes，Istio

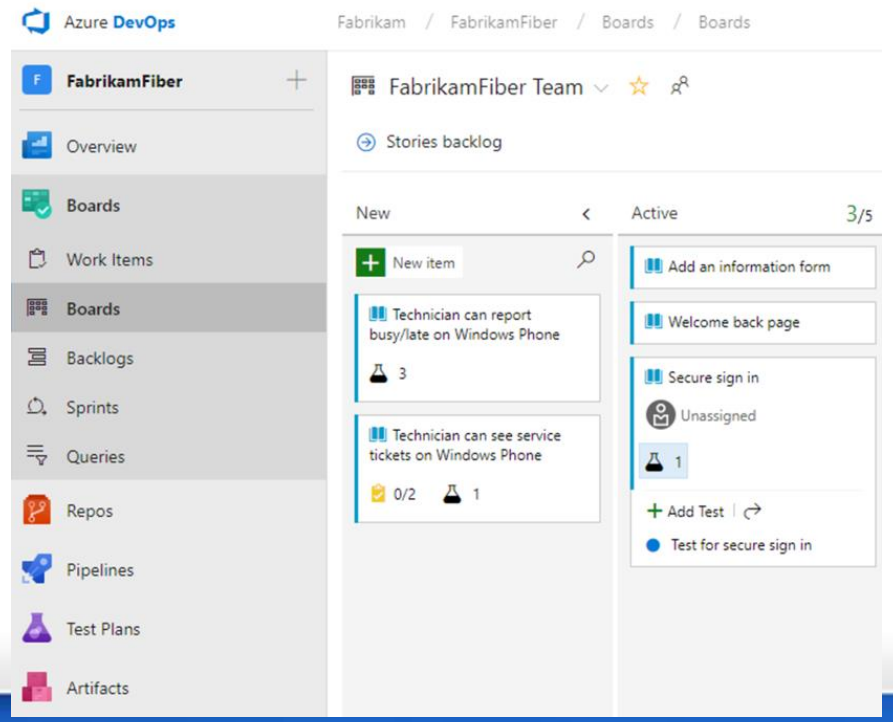




一体化集成开发环境

- 功能：全生命周期管理
- 利益相关方：软件开发团队全体成员
- 代表性工具
  - ✓华为软件开发云DevCloud、IBM Rational Team Concert（RTC）、Microsoft Azure DevOps





CASE工具小结

工具类别	基本功能	利益相关方	代表性工具
系统分析与设计工具	将用户需求进行可视化建模、分析和管理，生成需求规格促进从需求到代码的转化	需求分析人员、软件设计人员	Microsoft Visio、Rational Rose、StarUML、Visual Paradigm
程序设计工具	代码编写、代码智能补全、程序编译、程序调试、程序运行等	编程人员、维护人员	IntelliJ IDEA、Visual Studio、Eclipse、PyCharm
软件测试与质量保证工具	单元测试、集成测试、系统测试、性能测试等	编程人员、测试人员	SonarQube，CodeClimate，JMeter、JaCoCo、JUnit
项目管理工具	软件项目代码和文件的托管、团队协作开发、软件工程管理	开发人员、管理人员	GitHub、Gitee（码云）、GitLab
软件运维工具	应用程序打包、部署、管理、监控	开发人员、维护人员	Docker、Kubernetes、Istio
一体化集成开发环境	项目管理、代码托管、协同开发、软件测试、软件运维等软件开发全生命周期管理	需求人员、设计人员、开发人员、测试人员、管理人员、维护人员	华为软件开发云、DevCloudIBM Rational Team Concert（RTC）、Microsoft Azure DevOps

4. 软件工程教育

软件从业人员需要具备的技术和能力

软件从业人员需遵守的法律和法规

软件从业人员需遵守的职业道德

软件从业人员需要具备的技术和能力

- 洞悉软件系统的特点
- 具有特定领域的知识
- 掌握软件工程的方法、技术和工具
- 具有表达、交流、沟通、协作、独立解决问题等能力
- 开展软件项目的组织和管理
- 具备创新能力、系统能力、解决复杂工程问题能力

软件从业人员需要具备的技术和能力

- **IEEE SWEBOK V3.0 (SoftWare Engineering Body of Knowledge)**
  - ✓知识域(Knowledge Area: KA)

✓软件需求
✓软件设计
✓软件构造
✓软件测试
✓软件维护

✓软件配置管理
✓软件工程管理
✓软件工程过程
✓软件工程建模与方法
✓软件质量

✓软件工程专业实践
✓软件工程经济学
✓计算基础
✓数学基础
✓工程基础

软件从业人员需遵守的法律和法规

- 著作权法
- 数据保护法
- 反垄断法
- 网络安全法
- 计算机软件保护条例
- 信息产业部软件产品保护条例

- 警示案例
  - ✓ 2018年Facebook“数据门”
  - ✓ 2022年某出行平台因违规被罚80亿

软件从业人员需遵守的职业道德

计算机协会道德与职业行为准则

1. 一般道德原则。

1.1为社会和人类的幸福做出贡献,承认所有人都是计算的利益相关者。
1.2 避免伤害。
1.3 诚实可靠。
1.4 做事公平，采取行动无歧视。
1.5 尊重需要产生新想法、新发明、创造性作品和计算工件的工作。
1.6 尊重隐私。
1.7 尊重保密协议。

2. 职业责任。

2.1 努力在专业工作的过程和产品中实现高质量。
2.2 保持高标准的专业能力、行为和道德实践。
2.3 了解并尊重与专业工作相关的现有规则。
2.4 接受并提供适当的专业审查。
2.5 对计算机系统及其影响进行全面彻底的评估，包括分析可能的风险。
2.6 仅在能力范围内开展工作。
2.7 培养公众对计算、相关技术及其后果的认识和理解。
2.8 仅当获得授权或仅为公众利益之目的才能访问计算和通信资源。
2.9 设计和实施具有稳固又可用的安全的系统。

3. 专业领导原则。

3.1 确保公众利益是所有专业计算工作的核心问题。
3.2 明确、鼓励接受并评估组织或团体成员履行社会责任的情况。
3.3 管理人员和资源，提高工作生活质量。
3.4 阐明、应用和支持反映本准则原则的政策和流程。
3.5 为组织或团队成员创造机会，让其成长为专业人员。
3.6 谨慎修改或停用系统。
3.7 识别并特别关注那些融入社会基础设施里的系统。

4.遵守《准则》

4.1 坚持、促进和尊重《准则》的原则。
4.2 将违反本《准则》的行为视为不符合计算机协会会员资格。

软件从业人员需遵守的职业道德

软件工程师职业道德规范和实践要求

计算机在商业、工业、政府、医药、教育、娱乐和整个社会中的核心作用日渐突出。软件工程师是直接参与或讲授系统的分析、规格说明、设计、开发、认证、维护和测试的人员。基于在软件系统开发中的地位，软件工程师可能将事情做好也可能做坏，还可能让他人或影响他人将事情做好或做坏。为了最大限度地保证自己的工作是有益的，软件工程师必须保证使软件工程业成为对社会有益的、受人尊敬的行业。基于以上保证，软件工程师应当遵守下面的道德和职业行为准则。

1. 公众：软件工程师应当以公众利益为目标。
2. 客户和雇主：在保持与公众利益一致的原则下，软件工程师应注意满足客户和雇主的最高利益。
3. 产品：软件工程师应当确保他们的产品和相关的改进符合最高的专业标准。
4. 判断：软件工程师应当维护他们职业判断的完整性和独立性。
5. 管理：软件工程的经理和领导人员应赞成和促进对软件开发和维护合乎道德规范的管理。
6. 专业：在与公众利益一致的原则下，软件工程师应当推进其专业的完整性和声誉。
7. 同行：软件工程师对其同行应持平等和互助和支持的态度。
8. 自我：软件工程师应当参与终生职业实践的学习，并促进合乎道德的职业实践方法。

本章小结

- **软件危机产生的背景、表现和根源**
  - ✓软件危机，持续存在，关注点不同
- **软件工程的本质**
  - ✓是将“系统化的、规范的、可量化”的工程方法应用于软件的开发、操作和维护。软件工程的发展在不同阶段和时期，有不同的特点和内容
- **CASE逐步演进成为辅助软件工程全生命周期的开发工具和方法集合**
- **软件工程教育**
  - ✓软件从业人员需要具备相应的技术和能力
  - ✓软件从业人员应遵循的国家法规和行业规范