

实验报告

（\_\_2023\_\_\_\_/\_\_2024\_\_学年 第一学期）

|  |  |
| --- | --- |
| 课程名称： | 软件工程实验 |
| 学 院： | 计算机科学与技术学院 |
| 课 程 组： | 软件工程课程组 |
| 专业班级： | 计算机21-1 |
| 学 号： | 20211401206 |
| 姓 名： | 梁浩铂 |
| 指导教师： | 郑炅 |

# 实验 1 软件工程文档及工具

|  |  |  |  |
| --- | --- | --- | --- |
| 班级 | 学号 | 姓名 | 成绩 |
| 计算机21-1 | 20211401206 | 梁浩铂 |  |

## 一、实验目的

1. 理解软件工程的基本概念，熟悉软件，软件生命周期，软件生存周期过程和软件生命周期各阶段的定义和内容。

2. 了解软件工程文档的类别、内容及撰写软件工程文档的方法。

3. 通过Internet搜索与浏览，了解网络环境中主流的软件工程技术网站，掌握通过专业网站不断丰富软件工程最新知识的学习方法，尝试通过专业网站的辅助与支持开展软件工程应用实践。

4. 熟悉软件工具、软件开发环境和CASE工具。

5. 了解《GB/T 11457-2006 信息技术 软件工程术语》相关内容。

6.了解软件工程相关的国家标准。

## 二、实验内容与步骤

1. 查阅有关资料，给出“软件”的权威定义，请标明出处。

软件（中国大陆及香港用语，台湾称作软体，英文：software）是一系列按照特定顺序组织的计算机数据和指令的集合。一般来讲软件被划分为系统软件、应用软件和介于这两者之间的中间件。软件并不只是包括可以在计算机（这里的计算机是指广义的计算机）上运行的电脑程序，与这些电脑程序相关的文档一般也被认为是软件的一部分。简单的说软件就是程序加文档的集合体。

国标中对软件的定义为：与计算机系统操作有关的[计算机程序](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A8%8B%E5%BA%8F/3220205?fromModule=lemma_inlink" \t "_blank)、[规程](https://baike.baidu.com/item/%E8%A7%84%E7%A8%8B?fromModule=lemma_inlink" \t "_blank)、[规则](https://baike.baidu.com/item/%E8%A7%84%E5%88%99/4405?fromModule=lemma_inlink" \t "_blank)，以及可能有的文件、[文档](https://baike.baidu.com/item/%E6%96%87%E6%A1%A3?fromModule=lemma_inlink" \t "_blank)及数据。

其它定义：

1．运行时，能够提供所要求功能和性能的[指令](https://baike.baidu.com/item/%E6%8C%87%E4%BB%A4?fromModule=lemma_inlink" \t "_blank)或[计算机程序](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A8%8B%E5%BA%8F?fromModule=lemma_inlink" \t "_blank)集合。

2．程序能够满意地处理信息的数据结构。

3．描述程序功能需求以及程序如何操作和使用所要求的文档。

以开发语言作为描述语言，可以认为：软件=[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F/13831935?fromModule=lemma_inlink" \t "_blank)+数据+文档

摘选自百度百科：[https: //baike.baidu.com/ item/ %E8%BD%AF%E4%BB%B6/ 12053? fr= ge\_ala](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6/12053?fr=ge_ala)

2. 查阅有关资料，给出“软件生命周期”的权威定义，请标明出处。

软件生存周期(SDLC，[软件生命周期](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E7%94%9F%E5%91%BD%E5%91%A8%E6%9C%9F/861455?fromModule=lemma_inlink" \t "_blank))是软件的产生直到报废的生命周期，周期内有问题定义、可行性分析、总体描述、[系统设计](https://baike.baidu.com/item/%E7%B3%BB%E7%BB%9F%E8%AE%BE%E8%AE%A1/3073283?fromModule=lemma_inlink" \t "_blank)、[编码](https://baike.baidu.com/item/%E7%BC%96%E7%A0%81/80092?fromModule=lemma_inlink" \t "_blank)、调试和测试、验收与运行、维护升级到废弃等阶段，这种按时间分程的思想方法是[软件工程](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B/25279?fromModule=lemma_inlink" \t "_blank)中的一种思想原则，即按部就班、逐步推进，每个阶段都要有定义、工作、审查、形成文档以供交流或备查，以提高软件的质量。

软件生存周期（software life cycle）又称为软件生命期，生存期。是指从形成开发软件概念起，所开发的软件使用以后，直到失去使用价值消亡为止的整个过程。

一般来说，整个生存周期包括计划（定义）、开发、运行（维护）三个时期，每一个时期又划分为若干阶段。每个阶段有明确的任务，这样使规模大、结构复杂和管理复杂的软件开发变得容易控制和管理。

摘选自百度百科：https:// baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E7%94%9F% E5% AD%98%E5%91%A8%E6%9C%9F/3230322?fr=aladdin

3. 查阅有关资料，给出“软件过程”的权威定义，请标明出处。

软件过程（Software Process）是指一套关于项目的阶段、状态、方法、技术和开发、维护软件的人员以及相关Artifacts（计划、文档、模型、编码、测试、手册等）组成。目前有[三种方法](https://baike.baidu.com/item/%E4%B8%89%E7%A7%8D%E6%96%B9%E6%B3%95/22625417?fromModule=lemma_inlink" \t "_blank)：UP（the unified process），The OPEN Process，OOSP(TheObject-Oriented Software Process)。软件过程(Software Process)是指[软件生存周期](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E7%94%9F%E5%AD%98%E5%91%A8%E6%9C%9F/3230322?fromModule=lemma_inlink" \t "_blank)所涉及的一系列相关过程。过程是活动的集合；活动是任务的集合；任务要起着把输入进行加工然后输出的作用。活动的执行可以是顺序的、重复的、并行的、嵌套的或者是有条件地引发的。

软件过程可概括为三类：基本过程类、[支持过程](https://baike.baidu.com/item/%E6%94%AF%E6%8C%81%E8%BF%87%E7%A8%8B/16967346?fromModule=lemma_inlink" \t "_blank)类和[组织过程](https://baike.baidu.com/item/%E7%BB%84%E7%BB%87%E8%BF%87%E7%A8%8B/22325089?fromModule=lemma_inlink" \t "_blank)类。基本过程类包括获取过程、[供应过程](https://baike.baidu.com/item/%E4%BE%9B%E5%BA%94%E8%BF%87%E7%A8%8B/6745041?fromModule=lemma_inlink" \t "_blank)、开发过程、运作过程，[维护过程](https://baike.baidu.com/item/%E7%BB%B4%E6%8A%A4%E8%BF%87%E7%A8%8B/23162240?fromModule=lemma_inlink" \t "_blank)和管理过程。支持过程类包括文档过程、[配置管理](https://baike.baidu.com/item/%E9%85%8D%E7%BD%AE%E7%AE%A1%E7%90%86?fromModule=lemma_inlink" \t "_blank)过程、质量保证过程、验证过程、确认过程、联合评审过程、[审计过程](https://baike.baidu.com/item/%E5%AE%A1%E8%AE%A1%E8%BF%87%E7%A8%8B/5409821?fromModule=lemma_inlink" \t "_blank)以及[问题解决](https://baike.baidu.com/item/%E9%97%AE%E9%A2%98%E8%A7%A3%E5%86%B3/10903575?fromModule=lemma_inlink" \t "_blank)过程。组织过程类包括基础设施过程、改进过程以及培训过程。

软件过程主要针对[软件生产](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E7%94%9F%E4%BA%A7/50886296?fromModule=lemma_inlink" \t "_blank)和管理进行研究。为了获得满足工程目标的软件，不仅涉及工程开发，而且还涉及工程支持和工程管理。对于一个特定的项目，可以通过[剪裁过程](https://baike.baidu.com/item/%E5%89%AA%E8%A3%81%E8%BF%87%E7%A8%8B/23162239?fromModule=lemma_inlink" \t "_blank)定义所需的活动和任务，并可使活动并发执行。与软件有关的单位，根据需要和目标，可采用不同的过程、活动和任务。

软件过程是指软件整个[生命周期](https://baike.baidu.com/item/%E7%94%9F%E5%91%BD%E5%91%A8%E6%9C%9F/9878142?fromModule=lemma_inlink" \t "_blank)，从[需求获取](https://baike.baidu.com/item/%E9%9C%80%E6%B1%82%E8%8E%B7%E5%8F%96/10661146?fromModule=lemma_inlink" \t "_blank)，[需求分析](https://baike.baidu.com/item/%E9%9C%80%E6%B1%82%E5%88%86%E6%9E%90/2012709?fromModule=lemma_inlink" \t "_blank)，设计，实现，测试，发布和维护一个[过程模型](https://baike.baidu.com/item/%E8%BF%87%E7%A8%8B%E6%A8%A1%E5%9E%8B/2868894?fromModule=lemma_inlink" \t "_blank)。一个软件过程定义了软件开发中采用的方法，但软件过程还包含该过程中应用的技术——[技术方法](https://baike.baidu.com/item/%E6%8A%80%E6%9C%AF%E6%96%B9%E6%B3%95/3322730?fromModule=lemma_inlink" \t "_blank)和自动化工具。过程定义一个框架，为有效交付软件工程技术，这个框架必须创建。软件过程构成了软件项目管理控制的基础，并且创建了一个环境以便于技术方法的采用、工作产品（模型、文档、报告、表格等）的产生、里程碑的创建、质量的保证、正常变更的正确管理。

摘自百度百科：[https://baike.baidu.com/item/%E8%BD%AF%E4% BB%B6%E8%BF%87% E7%A8%8B/122453?fr=ge\_ala](https://baike.baidu.com/item/%E8%BD%AF%E4%25%20BB%B6%E8%BF%87%25%20E7%A8%8B/122453?fr=ge_ala)

4. 查阅有关资料，给出软件工程文档的类别、内容及撰写文档的方式。

软件工程文档可分为：可行性研究报告、项目开发计划、软件需求说明书、概要设计说明书、详细设计说明书、测试计划、测试分析报告、质量保证计划书、操作手册、用户使用手册、开发进度月报、项目开发总结报告，接下来就各文档说明的问题及提到的作用做简单的介绍。

可行性研究报告：说明该软件开发项目的实现在技术上、经济上和社会因素上的可行性，评述了合理地达到开发目标可供选择的各种可能实施的方案，说明并论证所选定实施方案的理由。

项目开发计划：为软件项目实施方案制定出具体计划，应该包括各部分工作的负责人员、开发的进度、开发经费的预算、所需要的硬件及软件资源等。项目开发计划应提供给管理部门，并作为开发阶段评审的参考。

软件需求说明书（软件规格说明书）：对所开发软件的功能、性能、用户界面及运行环境等做出详细的说明。它是用户与开发人员双方对软件需求取得共同理解基础上达成的协议，也是是开发工作的基础。

数据要求说明书：给出数据逻辑描述和数据采集的各项要求，为生成和维护系统数据文卷做好准备。

概要设计说明书：是概要设计阶段的工作成果，它应说明功能分配、模块划分、程序的总体结构、输入输出以及接口设计、运行设计、数据结构设计和出错处理设计等，为详细设计奠定基础。

详细设计说明书：着重描述每一模块是怎样实现的，包括实现算法、逻辑流程等。

测试计划：为做好组装测试和确认测试，需要为如何组织测试制定实施计划。计划应包括测试的内容、进度、条件、人员、测试用例的选取原则、测试结果允许的偏差范围等。

测试分析报告：测试工作完成后，应提交测试计划执行情况的说明，对测试结果加以分析，并提出测试的结论意见。

用户手册：详细描述软件的功能、性能和用户界面，是用户了解如何使用该软件。

操作手册：本手册为操作人员提供该软件各种运行情况的有关知识，特别是操作方法的具体细节。

开发进度月报：该月报是软件人员按月向管理部门提交的项目进展情况报告。报告应包括进度计划与实际执行情况的比较、阶段成果、遇到的问题和解决的办法以及下个月的计划等。

项目开发总结报告：软件项目开发完成后，应与项目实施计划对照，总结实际执行的情况，如进度、成果、资源利用、成本和投入的人力。此外还需要对开发工作做出评价，总结出经验和教训。

摘自CSDN博主「zql\_LV1314」的原创文章，原文链接：https:/ /blog.csdn.net/ zql\_LV1314/ article/details/90440558

5. 查阅有关资料，给出“软件工具”、“软件开发环境”、“CASE工具”的定义，并标明出处。

软件工具是指为支持计算机软件的开发、维护、模拟、移植或管理而研制的程序系统。它是为专门目的而开发的，在软件工程范围内也就是为实现软件生存期中的各种处理活动（包括管理、开发和维护）的自动化和半自动化而开发的程序系统。软件工具分为六类：模拟工具、[开发工具](https://baike.baidu.com/item/%E5%BC%80%E5%8F%91%E5%B7%A5%E5%85%B7?fromModule=lemma_inlink" \t "_blank)、测试和评估工具、运行和维护工具、性能质量工具和程序设计支持工具。

摘自百度百科：[https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5% E5%85%B7/3296796?fr=ge\_ala](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%25%20E5%85%B7/3296796?fr=ge_ala)

软件开发环境(Software Development Environment，SDE)是指在基本硬件和数字软件的基础上，为支持系统软件和应用软件的工程化开发和维护而使用的一组软件，简称SDE。它由软件工具和环境集成机制构成，前者用以支持软件开发的相关过程、活动和任务，后者为工具集成和软件的开发、维护及管理提供统一的支持。

摘自百度百科：[https://baike.baidu.com/item/%E5%BC%80%E5%8F%91%E7%8E%AF%E5% A2%83/10119007?fr=ge\_ala](https://baike.baidu.com/item/%E5%BC%80%E5%8F%91%E7%8E%AF%E5%25%20A2%83/10119007?fr=ge_ala)

case工具原来指用来支持管理信息系统开发的、由各种计算机辅助软件和工具组成的大型综合性软件开发环境，随着各种工具和软件技术的产生、发展、完善和不断集成，逐步由单纯的辅助开发工具环境转化为一种相对独立的方法论。

CASE：[计算机辅助软件工程](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E8%BE%85%E5%8A%A9%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B/3351642?fromModule=lemma_inlink" \t "_blank)。

[CASE技术](https://baike.baidu.com/item/CASE%E6%8A%80%E6%9C%AF/50904848?fromModule=lemma_inlink" \t "_blank)：一种软件技术。为软件的开发、维护和项目管理提供一种自动化工程原理，包括自动化[结构化方法](https://baike.baidu.com/item/%E7%BB%93%E6%9E%84%E5%8C%96%E6%96%B9%E6%B3%95/9827043?fromModule=lemma_inlink" \t "_blank)和自动化工具。

CASE工具：一种软件工具。对某个具体的[软件生命周期](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E7%94%9F%E5%91%BD%E5%91%A8%E6%9C%9F/861455?fromModule=lemma_inlink" \t "_blank)的任务实现自动化（至少是某一部分的自动化）。

CASE系统：一种集成的 CASE工具。使用一个公共的用户接口，并在一个公共的计算机环境下运行。

CASE工具箱：一组集成的 CASE工具。用来协同工作以实现某个软件生命周期的阶段或某类具体的软件作业的自动化（或部分地实现自动化）。

CASE工作台：一组集成的 CASE工具，被设计用来协同工作以实现整个软件生存期的自动化（或提供自动化的辅助手段），包括分析、设计、编码和测试。

[CASE方法](https://baike.baidu.com/item/CASE%E6%96%B9%E6%B3%95/22298602?fromModule=lemma_inlink" \t "_blank)：一种“可自动化”的结构化方法。为软件的开发和维护的整个过程或某个方面定义了一个类似工程的方法。

摘自百度百科：[https://baike.baidu.com/item/case%E5%B7%A5%E5%85%B7/1330464? fr=ge\_ala](https://baike.baidu.com/item/case%E5%B7%A5%E5%85%B7/1330464?%20fr=ge_ala)

6. 填写下列学术术语的英文简写，并思考、理解其含义

（1）软件开发环境（Software Development Environment）：SDE

（2）软件工程环境（Software Engineering Environment）：SEE

（3）软件支持环境（Software Support Environment）：SSE

（4）集成开发环境（Integrated Development Environment）：IDE

7. 了解软件工程有关的国家标准，请罗列它们并简要介绍（给出出处），并思考遵循标准的意义。

1、基础

GB/T 11457-2006 信息技术 软件工程术语  
GB/T 22033-2017 信息技术 嵌入式系统术语  
GB/Z 31102-2014 软件工程 软件工程知识体系指南

2、软件质量与测试

GB/T 15532-2008 计算机软件测试规范  
GB/T 18492-2001 信息技术 系统和软件完整性级别  
GB/T 18905.6-2002 信息技术 软件产品评价 第 6 部分：评价模块的文档编制  
GB/T 25000.1-2010 软件工程 软件产品质量要求与评价（SQuaRE）SQuaRE 指南  
GB/T 25000.2-2018 系统与软件工程 系统与软件质量要求和评价（SQuaRE）第 2 部分:计划与管理  
GB/T 25000.10-2016 系统与软件工程 系统与软件质量要求和评价（SQuaRE）第 10 部分：系统与软件质量模型  
GB/T 25000.12-2017 系统与软件工程 系统与软件质量要求和评价（SQuaRE）第 12 部分：数据质量模型  
GB/T 25000.21-2019 系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第21 部分：质量测度元素  
GB/T 25000.22-2019 系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第22 部分：使用质量测量  
GB/T 25000.23-2019 系统与软件工程 系统与软件质量要求与评价(SQuaRE) 第23 部分：系统与软件产品质量测量  
GB/T 25000.24-2017 系统与软件工程 系统与软件质量要求和评价（SQuaRE）第 24 部分：数据质量测量  
GB/T 25000.40-2018 系统与软件工程 系统与软件质量要求和评价（SQuaRE）第 40 部分：评价过程  
GB/T 25000.41-2018 系统与软件工程 系统与软件质量要求和评价（SQuaRE）第 41 部分：开发方、需方和独立评价方评价指南  
GB/T 25000.45-2018 系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第45 部分：易恢复性的评价模块  
GB/T 25000.51-2016 系统与软件工程 系统与软件质量要求和评价（SQuaRE）第 51 部分：就绪可用软件产品（RUSP）的质量要求和测试细则  
GB/T 25000.62-2014 软件工程 软件产品质量要求与评价（SQuaRE） 易用性测试报告行业通用格式（CIF）  
GB/T 28171-2011 嵌入式软件可靠性测试方法  
GB/T 28172-2011 嵌入式软件质量保证要求  
GB/T 29831.1-2013 系统与软件功能性 第 1 部分：指标体系  
GB/T 29831.2-2013 系统与软件功能性 第 2 部分：度量方法  
GB/T 29831.3-2013 系统与软件功能性 第 3 部分：测试方法  
GB/T 29832.1-2013 系统与软件可靠性 第 1 部分：指标体系  
GB/T 29832.2-2013 系统与软件可靠性 第 2 部分：度量方法  
GB/T 29832.3-2013 系统与软件可靠性 第 3 部分：测试方法  
GB/T 29833.1-2013 系统与软件可移植性 第 1 部分：指标体系  
GB/T 29833.2-2013 系统与软件可移植性 第 2 部分：度量方法  
GB/T 29833.3-2013 系统与软件可移植性 第 3 部分：测试方法  
GB/T 29834.1-2013 系统与软件维护性 第 1 部分：指标体系  
GB/T 29834.2-2013 系统与软件维护性 第 2 部分：度量方法  
GB/T 29834.3-2013 系统与软件维护性 第 3 部分：测试方法  
GB/T 29835.1-2013 系统与软件效率 第 1 部分：指标体系  
GB/T 29835.2-2013 系统与软件效率 第 2 部分：度量方法  
GB/T 29835.3-2013 系统与软件效率 第 3 部分：测试方法  
GB/T 29836.1-2013 系统与软件易用性 第 1 部分：指标体系  
GB/T 29836.2-2013 系统与软件易用性 第 2 部分：度量方法  
GB/T 29836.3-2013 系统与软件易用性 第 3 部分：测评方法  
GB/T 30264.1-2013软件工程自动化测试能力 第 1 部分：测试机构能力等级模型  
GB/T 30264.2-2013软件工程自动化测试能力 第 2 部分：从业人员能力等级模型  
GB/T 30847.1-2014系统与软件工程 可信计算平台可信性度量 第 1 部分：概述与词汇  
GB/T 30847.2-2014系统与软件工程 可信计算平台可信性度量 第 2 部分：信任链  
GB/T 30961-2014 嵌入式软件质量度量  
GB/T 30975-2014 信息技术 基于计算机的软件系统的性能测量与评级  
GB/T 32421-2015 软件工程 软件评审与审核  
GB/T 32422-2015 软件工程 软件异常分类指南  
GB/T 32423-2015 系统与软件工程 验证与确认  
GB/T 32904-2016 软件质量量化评价规范  
GB/T 33138-2016 存储备份系统等级和测试方法  
GB/T 33783-2017 可编程逻辑器件软件测试指南  
GB/T 34943-2017 C/C++语言源代码漏洞测试规范  
GB/T 34944-2017 Java 语言源代码漏洞测试规范  
GB/T 34946-2017 C#语言源代码漏洞测试规范  
GB/T 36099-2018 基于行为声明的应用软件可信性验证  
GB/T 37970-2019 软件过程及制品可信度评估  
GB/T 38634.1-2020 系统与软件工程 软件测试 第 1 部分：概念和定义  
GB/T 38634.2-2020 系统与软件工程 软件测试 第 2 部分：测试过程  
GB/T 38634.3-2020 系统与软件工程 软件测试 第 3 部分：测试文档  
GB/T 38634.4-2020 系统与软件工程 软件测试 第 4 部分：测试技术  
GB/T 38639-2020 系统与软件工程 软件组合测试方法

3、生存周期管理

GB/T 8566-2007 信息技术 软件生存周期过程  
GB/T 14394-2008 计算机软件可靠性和维护性管理  
GB/Z 18493-2001 信息技术 软件生存周期过程指南  
GB/T 19003-2008 软件工程 GB/T 19001-2000 应用于计算机软件的指南  
GB/Z 20156-2006 软件工程 软件生存周期过程用于项目管理的指南  
GB/T 20157-2006 信息技术 软件维护  
GB/T 20158-2006 信息技术 软件生存周期过程 配置管理  
GB/T 20261-2006 信息技术 系统安全工程 能力成熟度模型  
GB/T 20917-2007 软件工程 软件测量过程  
GB/T 20918-2007 信息技术 软件生存周期过程 风险管理  
GB/T 22032-2008 系统工程 系统生存周期过程  
GB/T 25644-2010 信息技术 软件工程可复用资产规范  
GB/T 26224-2010 信息技术 软件生存周期过程重用过程  
GB/T 26240-2010 系统工程 系统工程过程的应用和管理  
GB/T 28035-2011 软件系统验收规范  
GB/T 28173-2011 嵌入式系统 系统工程过程应用和管理  
GB/T 30971-2014软件工程 用于互联网的推荐实践网站工程、网站管理和网站生存周期  
GB/T 30998-2014 信息技术 软件安全保障规范  
GB/T 30999-2014 系统与软件工程 生存周期管理 过程描述指南  
GB/Z 31103-2014 系统工程 GB/T 22032（系统生存周期过程）应用指南  
GB/T 33781-2017 可编程逻辑器件软件开发通用要求  
GB/T 36462-2018 面向组件的虚拟样机软件开发通用要求  
GB/T 37691-2019 可编程逻辑器件软件安全性设计指南  
GB/T 37979-2019 可编程逻辑器件软件 VHDL 编程安全要求

4、工具和方法

GB/T 13502-1992 信息处理 程序结构及其表示的约定  
GB/T 18234-2000 信息技术 CASE 工具评价和选择指南  
GB/T 18714.1-2002 信息技术 开放分布式处理参考模型 第 1 部分：概述  
GB/T 18714.2-2002 信息技术 开放分布式处理参考模型 第 2 部分：基本概念  
GB/T 18714.3-2003 信息技术 开放分布式处理参考模型 第 3 部分：体系结构  
GB/Z 18914-2014 信息技术 软件工程 CASE 工具的采用指南  
GB/T 25654-2010 手持电子产品嵌入式软件 API  
GB/T 26239-2010 软件工程 开发方法元模型  
GB/T 28174.1-2011 统一建模语言（UML）第 1 部分：基础结构  
GB/T 28174.2-2011 统一建模语言（UML）第 2 部分：上层结构  
GB/T 28174.3-2011 统一建模语言（UML）第 3 部分：对象约束语言  
GB/T 28174.4-2011 统一建模语言（UML）第 4 部分：图交换  
GB/T 30972-2014 系统与软件工程 软件工程环境服务  
GB/T 30973-2014 信息技术 软件状态转换的人际通信表示  
GB/Z 26248.1-2010信息技术 文档描述和处理语言 用于 XML 的规则语言描述（RELAX） 第 1 部分：RELAX 核心  
GB/T 36457-2018 复杂产品虚拟样机建模方法

5、软件文档化

GB/T 1526-1989信息处理 数据、程序和系统流程图、程序网络图以及系统资源图表用的文档符号和约定  
 GB/T 8567-2006 计算机软件文档编制规范  
 GB/T 9385-2008 计算机软件需求规格说明规范  
 GB/T 9386-2008 计算机软件测试文档编制规范  
 GB/T 14085-1993 信息处理系统 计算机系统配置图符号和约定  
 GB/T 15189-1994 DOS 中文信息处理系统接口规范  
 GB/T 15535-1995 信息处理 单命中制定表规范  
 GB/T 16680-2015 系统与软件工程 用户文档的管理者要求  
 GB/T 32424-2015 系统与软件工程 用户文档的设计者和开发者要求  
 GB/T 33784-2017 可编程逻辑器件软件文档编制规范

6、软件资产管理

GB/T 26236.1-2010 信息技术 软件资产管理 第 1 部分：过程  
GB/T 36328-2018 信息技术 软件资产管理 标识规范  
GB/T 36329-2018 信息技术 软件资产管理 授权管理

7、软件绩效和成本

GB/T 18491.1-2001 信息技术 软件测量 功能规模测量 第 1 部分：概念定义  
GB/T 18491.2-2010信息技术 软件测量 功能规模测量 第 2 部分：软件规模测量方法与 GB/T 18491.1-2001 的符合性评价  
GB/T 18491.3-2010信息技术 软件测量 功能规模测量 第 3 部分：功能规模测量方法的验证  
GB/T 18491.4-2010 信息技术 软件测量 功能规模测量 第 4 部分：基准模型  
GB/T 18491.5-2010信息技术 软件测量 功能规模测量 第 5 部分：功能规模测量的功能域确定  
GB/T 18491.6-2010信息技术 软件测量 功能规模测量 第 6 部分：GB/T 18491 系列标准和相关标准的使用指南  
GB/T 32911-2016 软件测试成本度量规范  
GB/T 36964-2018 软件工程 软件开发成本度量规范

8、在研国家标准列表

20171718-T-469 系统与软件工程 软件测试 性能测试方法  
20171808-T-424 系统与软件工程 系统与软件质量要求和评价（SQuaRE） 第1部分：SQuaRE指南  
20171809-T-424 系统与软件工程 系统与软件质量要求和评价（SQuaRE） 第20部分：测量参考模型和指南  
20162511-T-469 系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第30部分：质量要求  
20190850-T-469 系统与软件工程 系统生存周期过程  
20190849-T-469 系统与软件工程 信息技术项目绩效基准化框架 第1部分：概念和定义征求意见  
20190845-T-469 系统与软件工程 信息技术项目绩效基准化框架 第2部分：基准化要求  
20190837-T-469 软件工程 软件测试工具能力  
20190838-T-469 系统与软件工程 信息技术项目绩效基准化框架 第4部分：数据收集和维护指南  
20190839-T-469 系统与软件工程 信息技术项目绩效基准化框架 第3部分：报告指南  
20190846-T-469 软件与系统工程 产品线工程与管理参考模型  
20194201-T-469 系统与软件工程 功能规模测量 COSMIC 方法  
20194189-T-469 系统与软件工程 功能规模测量 IFPUG 方法  
20194199-T-469 系统与软件工程 功能规模测量 Mk II 功能点分析方法  
20194198-T-469 系统与软件工程 功能规模测量 NESMA 方法  
20194202-T-469 系统与软件工程 功能规模测量 FiSMA1.1 方法  
20194197-T-469 系统与软件工程 软件生存周期过程  
20194204-T-469 系统与软件工程 开发运维一体化 能力成熟度模型

摘自：https://www.jianshu.com/p/f8b6fbbc999c

遵循标准的意义：

统一的规范和准则：标准为软件工程提供了统一的规范和准则。它们定义了软件开发、测试、管理和交付过程中的最佳实践和方法，确保团队在开发过程中具有一致的方向和方法论。

提高质量和可靠性：遵循标准有助于提高软件的质量和可靠性。标准中包含了测试、验证、质量管理和缺陷修复等方面的指导，帮助团队发现和解决潜在的问题，减少缺陷和错误的出现，提高软件的稳定性和可靠性

降低风险和成本：标准化的软件工程实践有助于降低风险和成本。通过遵循标准，团队可以规避常见的错误和问题，减少项目延期、预算超支和客户投诉等风险，从而降低项目的风险和成本。

支持合规性要求：在某些行业和领域中，存在特定的法规和合规性要求。遵循相关的软件工程标准可以帮助组织满足这些要求，并确保软件的合规性。这包括数据隐私、安全性和法律合规等方面的要求。

促进协作和知识共享：标准化的软件工程实践促进了团队内部和团队之间的协作和知识共享。通过共同遵循标准，团队成员可以更好地理解和交流，分享经验和最佳实践，提高团队整体的工作效率和协作能力。

8. 通过查阅资料给出《GB/T 11457-2006 信息技术 软件工程术语》规定的已在软件工程、数据库、数据结构、C语言、面向对象的编程语言等课程中已学过的10条术语的详细定义（注明术语编号和页码信息）。

寻址异常 addressing exception（2.50，第8页）：当程序计算地址时，超出了可用的地址边界所发生的异常。（在编写代码的过程中出现数组访问越界、栈溢出等异常）

算法 algorithm（2.54 第9页）：用有限步数求解某问题的一套明确定义的规则的集合，为执行特定任务的任何运算序列。

算法分析 algorithm analysis（2.55 第9页）：对一算法的检查，其目的在于确定与其预期用途有关的正确性，确定其运行特性，或为了更充分地理解某一算法以便对其进行修改、简化或改进。

别名 alisa（2.59 第10页）：某一项目的另一个名字；一个替换标号，可以使用一个标号和一个或多个别名来指示计算机程序中同一元素或点。（在数据库中可以给一个表起别名来简化程序书写）

数组 array（2.75 第11页）：由单个名和一个或多个索引标识的数据项所组成的n维有序子集，该集中的每个元素可以分别寻址。例如，矩阵、表或向量。

断言 assertion（2.87 第13页）：一种逻辑表达式，它规定必须存在的一种程序状态，或规定在程序执行过程中某一特定点上程序变量必须满足得条件集合。类型包括输入断言、循环断言、输出断言。

赋值语句 assignment statement（2.99 第14页）：一种计算机程序语句，它用于表达一系列操作，或用于把操作数赋给指定变量、或符号、或变量和符号两者。例如，Y=X+5

二进制数字[位] binary digit[bit]（2.138 第18页）：能用0或1表示的信息单元；能保持a中信息单元的计算机存储的元素；用于表示二进制数字系统，0或1两个数字之一的数字。

布尔量 Boolean（2.148 第19页）：一种枚举类型，其值只有“真”或“假”两种

分支 branch（2.157 第21页）：一种计算机程序结构，在此结构中，可选择两个或多个程序语句的可替换的集之一执行；计算机程序中选择两个或多个程序语句的可替换的集一直的执行点。

1. 了解Microsoft Visio的基本情况，试列出优势。

Microsoft Visio是一款专业的流程图和图表制作软件，由微软公司开发和发布。它提供了丰富的功能和工具，使用户能够创建、编辑和共享各种类型的图表，包括流程图、组织结构图、平面布局、网络拓扑图等。以下是关于Microsoft Visio的基本情况以及优势：

1、功能丰富：Visio提供了广泛的功能和工具，包括形状绘制、连接线、文本编辑、图层管理、数据连接、图表样式、图像导入等，使用户能够创建具有专业外观和准确性的图表。

2、模板和形状库：Visio内置了大量的预定义模板和形状库，涵盖了各种行业和应用场景。用户可以根据需要选择合适的模板，并从形状库中拖放所需的图形元素，加快图表创建的速度。

3、数据连接与可视化：Visio具有强大的数据连接功能，可以将外部数据源（如Excel、SQL数据库等）与图表进行关联。用户可以通过数据连接将图表与实时数据关联，并使用可视化选项，如数据图标、颜色编码等，将数据呈现出来。

4、协作和共享：Visio支持多人协作，多个用户可以同时编辑同一个图表，并实时查看彼此的更改。此外，Visio还提供了共享和协作功能，用户可以轻松共享图表，并与他人进行反馈和讨论。

5、Microsoft Office集成：作为Microsoft Office套件的一部分，Visio与其他Office应用程序（如Word、Excel、PowerPoint）无缝集成。用户可以将图表嵌入到其他文档中，实现更综合的信息展示。

6、可扩展性和定制性：Visio支持自定义形状、模板和宏的创建，用户可以根据自己的需求进行扩展和定制。这使得Visio适应各种特定业务或行业的要求。

7、多平台支持：除了传统的Windows桌面版本，Visio还提供了Web版和移动版，支持在不同平台上使用和编辑图表，提高了灵活性和可访问性。

1. 列出常用的UML建模工具，简要描述它们的特点。

Visual Paradigm：Visual Paradigm是一款功能强大的UML建模工具，提供了广泛的UML图表类型，包括类图、时序图、活动图等。它具有直观的界面和丰富的功能，支持团队协作、版本控制和代码工程化等特点。

IBM Rational Rose：IBM Rational Rose是一款经典的UML建模工具，具有强大的建模和分析能力。它支持多种UML图表类型，并提供了自动生成代码和文档的功能。此外，它还与其他IBM工具集成，方便进行全面的软件开发和管理。

Sparx Systems Enterprise Architect：Enterprise Architect是一款全面的建模工具，支持多种建模标准，包括UML、BPMN、SysML等。它具有强大的建模和分析功能，并提供了代码生成、模型转换等特性。同时，它也支持团队协作和版本控制。

Visual Studio：Microsoft Visual Studio是一款集成开发环境，也提供了UML建模的能力。它内置了建模工具，如类图和序列图的设计，同时与开发环境无缝集成，方便进行代码开发和建模的结合。

Lucidchart：Lucidchart是一款基于Web的在线建模工具，支持UML和其他图表类型。它具有易于使用的界面和协作功能，用户可以在任何地方进行建模工作，并与团队成员实时协作和共享图表。

Astah：Astah是一款流行的UML建模工具，提供了多种UML图表类型和建模功能。它具有直观的界面和易于使用的工具，支持代码生成、模型转换和团队协作等特性。

MagicDraw：MagicDraw是一款功能强大的建模工具，支持多种UML和其他建模标准。它提供了广泛的建模和分析功能，并具有代码生成、模型转换和模型驱动开发等特性。此外，它还支持与其他工具的集成。

Papyrus：Papyrus是一款开源的UML建模工具，具有灵活的扩展性和自定义性。它支持多种UML图表类型和建模功能，并与Eclipse集成，可与其他Eclipse插件无缝协作。

ArgoUML：ArgoUML是一款开源的UML建模工具，具有简单易用的界面和基本的建模功能。它适合小型项目或初学者使用，支持常见的UML图表类型和基本的建模操作。

11. 列出主流的数据库设计工具，简要描述它们的特点。

ER Studio：ER Studio是一款功能强大的数据库设计和建模工具。它支持多种数据库平台，并提供了直观的界面和丰富的建模功能，包括实体关系图（ER图）、逻辑模型设计、物理模型生成等。ER Studio还具有数据字典、版本控制和团队协作等特性，适用于大规模和复杂的数据库设计项目。

Oracle SQL Developer Data Modeler：Oracle SQL Developer Data Modeler是Oracle官方提供的数据库建模工具。它支持多种数据库平台，并提供了全面的建模和设计功能，包括实体关系图、逻辑模型、物理模型等。该工具与Oracle数据库紧密集成，可以直接生成DDL脚本以及进行数据库反向工程等操作。

MySQL Workbench：MySQL Workbench是一款针对MySQL数据库的建模和设计工具。它提供了丰富的建模功能，包括实体关系图、逻辑模型设计、物理模型生成等。MySQL Workbench支持直观的界面和实时协作特性，适用于MySQL数据库的设计和管理。

Microsoft SQL Server Management Studio (SSMS)：Microsoft SQL Server Management Studio是微软官方提供的SQL Server数据库管理工具，同时也包含了数据库建模的功能。它提供了实体关系图、数据库图、表设计等功能，支持SQL Server数据库的建模和管理操作。

DBDesigner：DBDesigner是一款开源的数据库设计工具，具有直观的界面和基本的建模功能。它支持实体关系图、逻辑模型设计等，可以生成DDL脚本，并支持多种数据库平台。DBDesigner适用于小型项目或初学者使用。

Toad Data Modeler：Toad Data Modeler是一款功能强大的数据库建模工具，支持多种数据库平台。它提供了全面的建模功能，包括实体关系图、逻辑模型、物理模型设计等。Toad Data Modeler还具有数据字典、版本控制、团队协作等特性，适用于大型和复杂的数据库设计项目。

PowerDesigner：PowerDesigner是一款综合性的数据建模和设计工具，支持多种数据库平台和建模标准。它提供了广泛的建模功能，包括实体关系图、数据流图、逻辑模型、物理模型等。PowerDesigner还具有强大的版本管理和团队协作特性，适用于大规模的数据库设计和管理项目。

Navicat Data Modeler：Navicat Data Modeler是一款直观易用的数据库建模工具，支持多种数据库平台。它提供了实体关系图、逻辑模型设计、物理模型生成等功能，并支持数据同步和版本控制。Navicat Data Modeler还具有直观的界面和数据导入导出功能，方便进行数据模型的管理和交互。

DbSchema：DbSchema是一款跨平台的数据库设计工具，支持多种数据库平台和建模标准。它提供了丰富的建模功能，包括实体关系图、逻辑模型设计、物理模型生成等，并支持数据库反向工程。DbSchema还具有直观的界面和团队协作特性，适用于中小型数据库设计项目。

12. 列出当前流行的原型设计工具，说明应用场景。

Adobe XD：Adobe XD是一款全面的原型设计工具，用于设计和交互式原型制作。它具有直观的界面和丰富的功能，支持快速创建交互式界面、设计动画效果以及进行用户测试。Adobe XD适用于移动应用、Web应用和软件界面的原型设计。

Sketch：Sketch是一款专注于界面设计的原型工具，特别受到Mac用户的欢迎。它提供了丰富的设计工具和插件生态系统，方便进行界面设计和原型制作。Sketch适用于移动应用、Web应用和产品界面的原型设计。

Figma：Figma是一款基于云的协作原型设计工具，它允许多个设计师实时协作并共同编辑设计文件。Figma具有强大的共享和评论功能，适用于团队协作和远程工作环境。它适用于移动应用、Web应用和产品界面的原型设计。

InVision：InVision是一款专注于用户体验设计的原型工具，它提供了设计文件的共享、用户测试和原型交互功能。InVision具有易于使用的界面和丰富的协作特性，适用于移动应用、Web应用和软件界面的原型设计。

Axure RP：Axure RP是一款功能强大的原型设计工具，特别适用于复杂交互和高保真度的原型制作。它提供了丰富的交互组件和动画效果，支持原型的注释和文档生成。Axure RP适用于移动应用、Web应用和软件界面的原型设计，尤其适合需要详细交互设计和文档生成的项目。

Marvel：Marvel是一款易于使用的原型设计工具，适用于移动应用和Web应用的原型设计。它提供了快速创建交互式原型的功能，具有简洁的界面和直观的操作方式。Marvel还支持原型共享、用户测试和版本控制等特性。

Justinmind：Justinmind是一款专业的原型设计工具，适用于复杂的交互和高保真度的原型制作。它提供了丰富的交互组件、动画效果和状态转换功能，支持多平台的原型设计。Justinmind还具有用户测试、协作和文档导出等功能。

Proto.io：Proto.io是一款基于云的原型设计工具，适用于移动应用和Web应用的原型设计。它提供了易于使用的界面和丰富的交互效果，支持实时预览和用户测试。Proto.io还具有协作和共享功能，方便团队协作和反馈收集。

Balsamiq Wireframes：Balsamiq Wireframes是一款快速且简洁的原型设计工具，用于快速绘制草图风格的界面原型。它提供了大量的预设组件和快速绘制工具，适用于快速验证概念和进行初步的用户测试。Balsamiq Wireframes注重快速迭代和快速原型制作。

13. 列出主流的源代码管理工具，简要分析它们的特点。

Git：Git是目前最流行的分布式版本控制系统。它具有分支管理、合并、提交和回滚等强大的功能，能够有效地管理和跟踪代码的变更。Git具有高性能、灵活性和强大的分布式特性，适用于个人开发和团队协作，可以轻松处理大型代码库。

Subversion（SVN）：Subversion是一种集中式版本控制系统，它采用集中式的存储结构，所有的代码都集中存放在服务器上。SVN提供了版本控制、分支管理和合并等功能，适用于小型团队和较简单的项目。SVN相对于Git来说更加简单易用，但在处理大型项目和并行开发方面性能可能略逊一筹。

Mercurial：Mercurial是另一种分布式版本控制系统，类似于Git。它具有分支管理、合并和提交等功能，同时也注重用户友好性和易用性。Mercurial适用于中小型项目和个人开发者，提供了简洁的命令行和图形界面工具。

Perforce：Perforce是一种集中式版本控制系统，它特别适用于大型项目和对安全性要求较高的场景。Perforce提供了高度可定制的权限控制和文件锁定机制，支持大规模文件和二进制文件的管理。它在处理大型项目和多人协作方面具有出色的性能和稳定性。

Microsoft Team Foundation Version Control (TFVC)：TFVC是一种集中式版本控制系统，由Microsoft开发。它与Microsoft的开发工具集成紧密，如Visual Studio和Azure DevOps。TFVC支持强大的分支管理、权限控制和团队协作功能，适用于Microsoft技术栈的开发项目。

Bitbucket：Bitbucket是一种基于Git的源代码管理平台，由Atlassian开发。它提供了Git仓库托管、分支管理、协作和持续集成等功能。Bitbucket与其他Atlassian工具（如Jira和Confluence）集成紧密，适用于团队协作和敏捷开发。

IBM Rational ClearCase：ClearCase是一种集中式版本控制系统，由IBM开发。它提供了高度可定制的版本控制和配置管理功能，适用于大型软件开发项目和对安全性要求较高的环境。ClearCase具有强大的分支和合并功能，支持多平台和多语言开发。

Apache Subversion Edge：Subversion Edge是一种集中式版本控制系统，由Apache开发。它是Subversion的增强版，提供了易于安装和管理的集成环境，包括版本控制、权限管理和报告功能。Subversion Edge适用于中小型项目和简化Subversion部署的场景。

14. 列出主流的单元测试、集成测试、性能测试工具，简要说明各自特点。

单元测试工具：

JUnit：JUnit是Java语言最常用的单元测试框架之一。它提供了一组注解和断言方法，用于编写和执行单元测试。JUnit易于使用，支持测试套件和参数化测试，并且与各种开发工具和持续集成平台集成紧密。

NUnit：NUnit是.NET平台上的单元测试框架，类似于JUnit。它提供了丰富的断言方法和测试装置，支持并行测试和数据驱动测试。NUnit还提供了可扩展性和自定义性，适用于.NET开发环境。

pytest：pytest是一个功能强大的Python单元测试框架，它提供了简洁的语法和丰富的断言方法。pytest支持参数化测试、夹具（fixture）和插件化扩展，具有灵活性和易用性。

mocha：mocha是一个流行的JavaScript测试框架，用于编写单元测试和集成测试。它支持异步测试、钩子函数和丰富的断言库，适用于前端和后端JavaScript应用程序的测试。

集成测试工具：

Selenium：Selenium是一个用于Web应用程序的自动化测试框架。它支持多种编程语言，如Java、C#和Python，并提供了API用于模拟用户操作和验证Web页面的行为。Selenium可以用于跨浏览器和跨平台的自动化测试。

Apache JMeter：JMeter是一种功能强大的负载测试工具，用于测试性能和压力。它支持多种协议和服务器类型，可以模拟大量用户并生成负载。JMeter具有可扩展性和灵活性，适用于Web应用程序和服务的性能测试。

Postman：Postman是一种广泛使用的API开发和测试工具，它提供了强大的集成测试功能。Postman可以发送HTTP请求、验证响应和编写测试脚本，支持自动化测试和生成测试报告。

SoapUI：SoapUI是一种专门用于测试Web服务的工具，支持SOAP和RESTful服务的集成测试。它提供了可视化的接口和测试构建器，支持数据驱动测试和安全性测试。

性能测试工具：

Apache JMeter：除了用于集成测试，JMeter也是一种常用的性能测试工具。它可以模拟多个并发用户和请求，测量响应时间和吞吐量，并生成测试报告。JMeter具有广泛的插件和扩展支持，适用于各种性能测试场景。

Apache ab：ab（Apache Bench）是Apache服务器自带的基准测试工具，用于测试Web服务器的性能。它可以发送并发请求并计算服务器的响应时间和吞吐量。ab简单易用，适合快速测试和基本性能评估。

Gatling：Gatling是一种基于Scala的高性能负载测试工具，用于测量系统的性能和承载能力。它支持分布式测试、实时监控和报告生成，适用于各种Web应用程序和API的性能测试。

Apache Bench：Apache Bench（ab）除了用于基准测试，也可以用作简单的性能测试工具。它可以模拟并发请求并计算服务器的响应时间和吞吐量。

15. 列出主流的软件项目管理工具；

Jira：Jira是一个功能强大的敏捷项目管理工具，由Atlassian开发。它支持任务管理、缺陷跟踪、团队协作和项目报告等功能。Jira可以自定义工作流程、设置权限和集成其他工具，适用于敏捷开发和项目管理。

Trello：Trello是一个简单易用的项目管理工具，提供了看板式的任务管理界面。用户可以创建任务卡片、拖动任务状态、添加注释和附件等。Trello适合个人任务管理和小型团队协作。

Microsoft Azure DevOps：Azure DevOps是一个综合的软件开发和项目管理平台，提供了版本控制、持续集成、任务管理和报告等功能。它与Microsoft的开发工具集成紧密，如Visual Studio和Azure云服务。

Basecamp：Basecamp是一个在线项目管理和团队协作工具，具有简洁直观的界面。它提供了任务列表、日程安排、文件共享和讨论等功能。Basecamp适用于小型团队和远程团队协作。

Asana：Asana是一个强大的团队协作和项目管理工具，支持任务管理、项目计划、协作和报告等功能。它提供了灵活的任务视图和工作流程，适用于各种团队和项目管理需求。

GitHub Projects：GitHub Projects是GitHub平台上的项目管理工具，与Git代码托管紧密集成。它提供了任务面板、项目看板和里程碑等功能，支持敏捷开发和团队协作。GitHub Projects与GitHub的版本控制和问题跟踪功能无缝集成。

GitLab：GitLab是一个综合的DevOps平台，提供了代码托管、CI/CD、项目管理和协作等功能。它支持自托管和云托管两种部署方式，并提供了丰富的项目管理功能，如任务追踪、里程碑和团队协作。

Redmine：Redmine是一个开源的项目管理和问题跟踪工具。它提供了任务管理、缺陷管理、文档管理和报告等功能。Redmine支持自定义工作流程和插件扩展，适用于中小型项目和团队协作。

Monday.com：Monday.com是一个直观易用的团队协作和项目管理平台。它提供了任务看板、日程安排、文件共享和团队通信等功能。Monday.com适用于各种团队和项目管理需求，包括软件开发、市场营销和人力资源等。

Teamwork：Teamwork是一个综合的项目管理和协作工具，提供了任务管理、文档协作、时间追踪和报告等功能。它支持团队协作、项目计划和资源管理，适用于中小型团队和项目。

16.常用的其他支撑工具

协作和沟通工具：

Slack：Slack是一个团队协作平台，提供实时聊天、频道管理、文件共享和集成等功能。它可以帮助团队成员之间进行即时沟通和协作，提高团队的效率和协同工作能力。

Microsoft Teams：Microsoft Teams是微软提供的协作和通信平台，集成了聊天、会议、文件共享和应用集成等功能。它适用于团队内部沟通和远程协作。

文档协作工具：

Google Docs：Google Docs是一种在线文档协作工具，多人可以同时编辑和评论文档。它提供了实时协作、版本控制和评论功能，方便团队成员共同撰写和编辑文档。

Microsoft Office 365：Office 365是微软提供的云端办公套件，包括Word、Excel、PowerPoint等工具，支持在线文档协作和共享。

项目进度和计划工具：

Gantt图工具：例如Microsoft Project、GanttProject等，用于制定项目计划、管理任务和跟踪进度。这些工具可以帮助团队可视化项目进度和任务依赖关系。

平台集成的项目管理模块：有些综合性平台，如Jira、Azure DevOps等，在其项目管理工具中提供了进度追踪、任务分配和报告等功能。

文档管理工具：

Confluence：Confluence是Atlassian开发的企业级团队协作和文档管理工具。它提供了创建、共享和组织文档的功能，支持团队协作、评论和版本控制。Confluence与Jira等工具紧密集成，适用于知识管理和团队文档协作。

SharePoint：SharePoint是微软提供的企业级文档管理和协作平台。它支持文档存储、共享、权限管理和工作流程等功能，适用于团队协作、知识管理和企业文档存档。

项目计划和资源管理工具：

Microsoft Project：Microsoft Project是一种专业的项目管理软件，用于制定项目计划、资源分配和进度跟踪等。它提供了强大的项目管理功能，可视化项目进度、任务依赖和资源利用情况。

Smartsheet：Smartsheet是一种在线项目管理和协作平台，具有类似于Excel的表格界面。它支持任务跟踪、进度管理、资源分配和报告生成等功能，适用于各种项目和团队。

团队远程协作工具：

Zoom：Zoom是一种流行的视频会议和远程协作工具，支持多人视频通话、屏幕共享和在线聊天等功能。它适用于远程团队协作、远程会议和远程培训等场景。

Microsoft Teams：除了作为协作工具，Microsoft Teams也提供了视频会议和远程协作功能。团队成员可以通过Teams进行视频通话、屏幕共享和在线协作。

测试管理工具：

TestRail：TestRail是一种专业的测试管理工具，用于跟踪测试计划、测试用例和缺陷。它提供了测试执行和结果记录、报告生成和缺陷跟踪等功能，帮助团队进行有效的软件测试管理。

Zephyr：Zephyr是一个广泛使用的测试管理插件，可与Jira等工具集成。它提供了测试计划、用例管理、执行和报告等功能，方便团队进行测试管理和跟踪。

运维和部署工具：

Ansible：Ansible是一种自动化运维工具，用于自动化部署、配置和管理服务器。它采用基于SSH的远程执行方式，支持编排任务、配置管理和应用部署等操作。

Docker：Docker是一种容器化平台，用于构建、发布和运行应用容器。它提供了轻量级、可移植的容器环境，简化了应用部署和环境管理的过程。

代码审查工具：

SonarQube：SonarQube是一个开源的代码质量管理平台，用于进行静态代码分析和质量度量。它可以检测代码缺陷、漏洞、代码重复和技术债务等问题，并提供相应的报告和指导意见。

Crucible：Crucible是一个代码审查工具，用于团队成员之间进行代码审查和讨论。它支持代码差异比较、注释和评论，帮助团队提高代码质量和知识共享。

## 三、实验总结

在本次实验过程中，通过在网络上搜索寻找与软件工程相关的概念，让我对软件工程有了进一步了解，熟悉了软件、软件生命周期、软件生存周期过程和软件生命周期各阶段的定义和内容，明白了软件工程不只需要代码编写，还需要前期的需求分析、设计，以及后期的测试、部署、运维。理清了软件工程文档的类别，以及该怎样书写软件工程文档。在查找过程中，也了解到一些在软件工程中需要用到的工具，UML建模工具、数据库设计工具、原型设计工具、源代码管理工具、单元测试工具、集成测试工具、性能测试工具、软件项目管理工具以及其他多种支撑工具。通过本次实验，我也了解到国标这一概念，在查找过程中，逐渐明白标准在软件工程中的重要性，正是有了一套统一的规范和准则，才使得在软件开发过程中，提高了质量和可靠性，降低了风险和成本，在开发过程中促进了协作和知识共享。