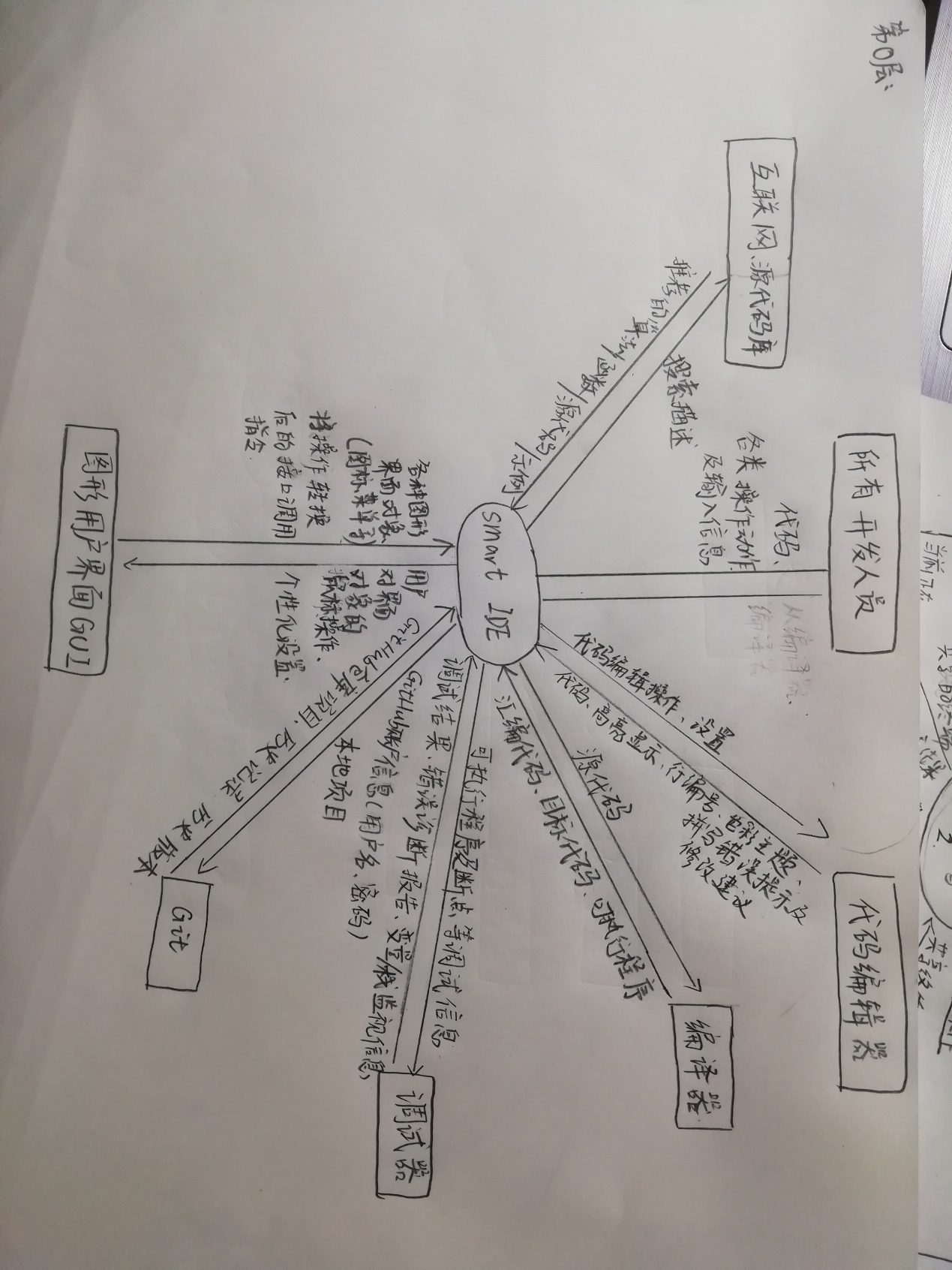
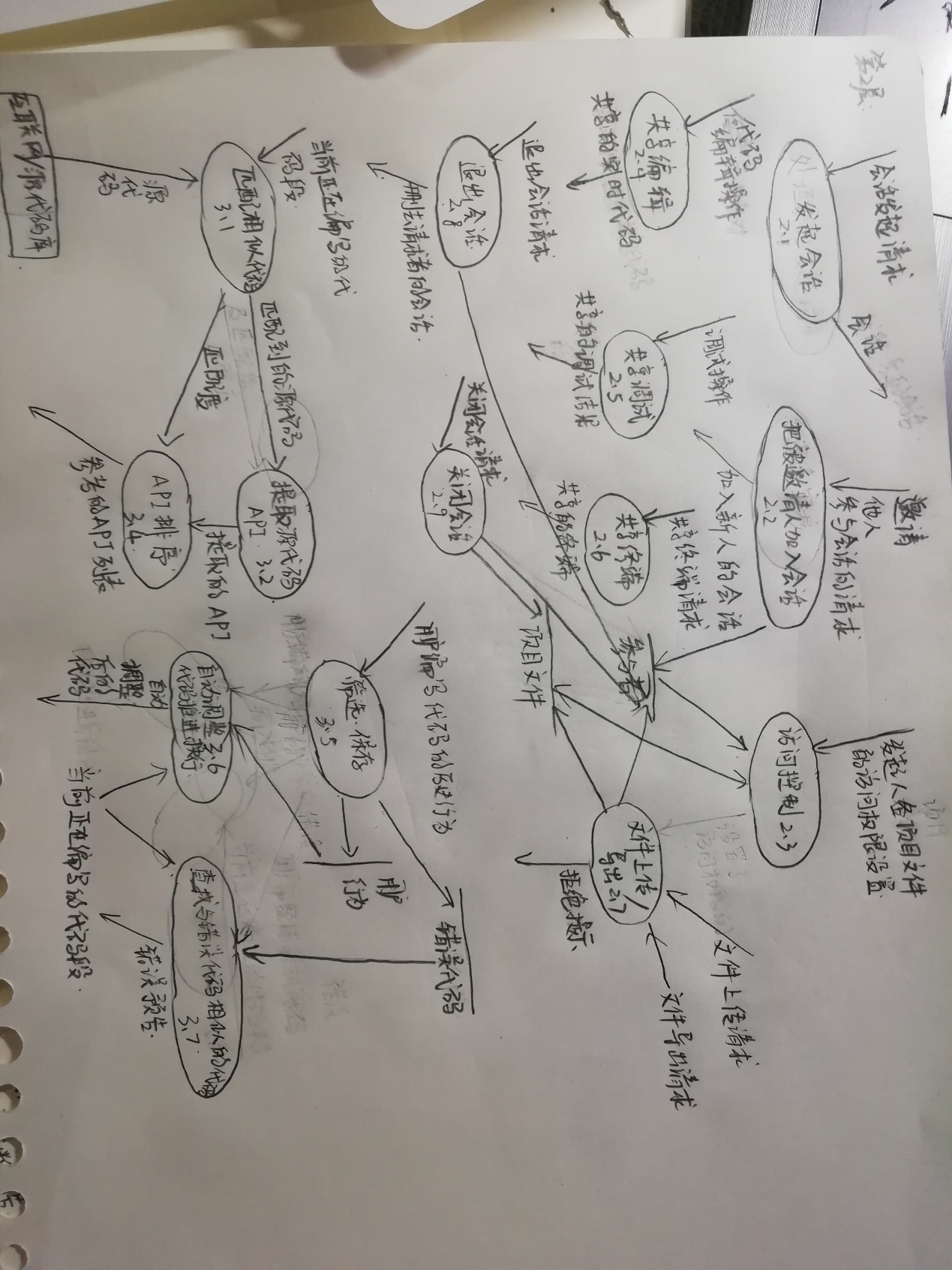
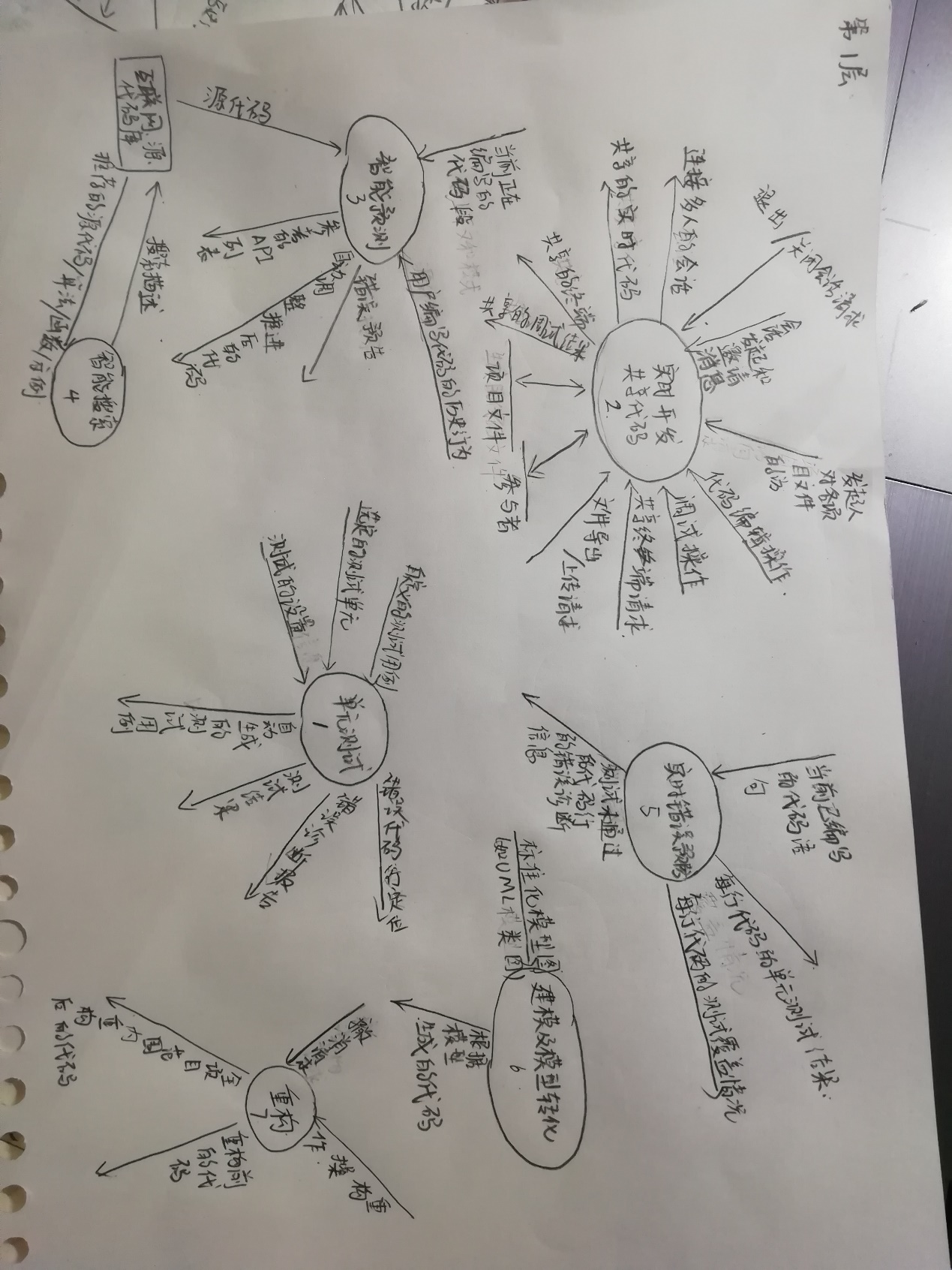
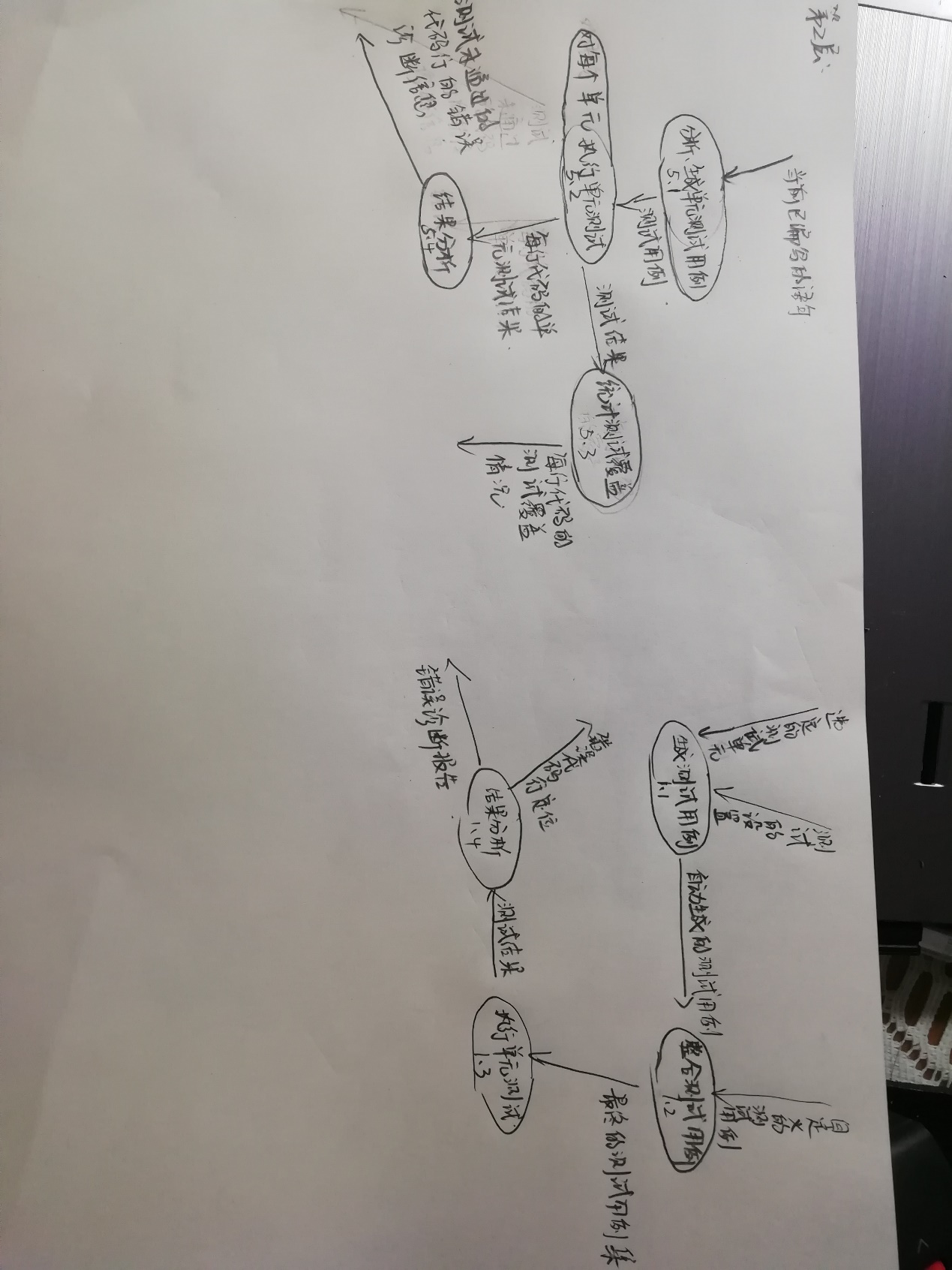
第3次作业

姓名:符冰瑶 学号：171860023

1. **DFD图**

**2.功能需求罗列**

**2.功能需求罗列**

* **代码编辑：**集成代码编辑器，支持多种编程语言（C、C++、Java、C#、XML、HTML、PHP、JS等）的代码编写，同时具有语法高亮显示、行编号、查找替换、色彩主题、检查拼写错误并提供修改建议等高级功能。
* **编译：**集成编译器，提供预处理、编译、汇编、链接四种功能，程序员可分步处理或一步到底直接生成可执行程序。
* **调试：**集成调试器，提供语法检查、断点调试功能；程序员可查看程序运行中的寄存器信息，[堆栈](https://baike.baidu.com/item/%E5%A0%86%E6%A0%88)信息、内存信息、当前EIP附近的[反汇编](https://baike.baidu.com/item/%E5%8F%8D%E6%B1%87%E7%BC%96)信息等；运行异常或故障时生成错误诊断报告。
* **图形用户界面（GUI）：**用户使用鼠标等输入设备操纵屏幕上的图标或菜单选项，以选择命令、调用文件、启动程序或执行其它一些日常任务。
* **实时共享**：实时协作开发共享代码。

1. 会话发起和邀请：每个协作项目包含一个发起人和一个或多个协作者。发起人创建一个协作会话，然后邀请其他协作者进入会话。
2. 访问控制：IDE对项目中的文件进行访问控制。发起人作为协作项目的管理员，具有所有文件的读写权限，并能够设置所有文件的访问权限（只读／可读写／不可访问），或者增加新的管理员。
3. 共享编辑：协作者加入会话后，能够实时看到别人的编辑，并且可以对自己具有写权限的文件进行编辑。
4. 共享调试：开发者完成某一文件的编辑后，可以发起共享调试会话，进入会话的每个开发者可以进行自己的调试，同时能够实时查看其他开发者的调试情况。
5. 共享终端：开发者可以以只读／可读写的方式与别人共享一个终端。
6. 提供会话框：开发者可以互相交流。
7. 文件本地导出／上传：开发者可以将会话中的文件导出到本地，也可以也可以将本地文件上传到会话中。

* **IDE中的Git管理和仓库创建、克隆**

1. 登录：IDE与GitHub连接，程序员可在IDE中登录GitHub.com和Github账户。
2. 同步功能：IDE与Github连接后，可以将本地项目发布到Github仓库。
3. 克隆仓库功能：将连接的Github的仓库克隆到IDE中。
4. 创建仓库功能：在IDE中创建Github仓库并且将其克隆到IDE中。
5. 在IDE中查看、提交pull request。
6. 版本控制功能：可查看参考、作者、测试、提交历史记录和其他重要信息。支持版本回退。

* **基于人工智能的智能预测**

1. 当程序员编写代码时，IDE能够当前的代码上下文和模式，利用人工智能技术，从网上的源代码或者程序员自己的代码学习、预测程序员此时最可能需要的 API，提供一个API参考列表。
2. 利用人工智能学习用户的行为，如：输入习惯、代码风格、偏好、常犯错误等。根据学习的结果，在用户写代码时，可能发生错误的地方自动提示，按照用户习惯调整推进、自动填充空格、换行等，同时保护用户隐私。

* **智能搜索**

提供一个搜索框，用户输入问题/关键字，IDE能在网上/自己的数据库中搜索相关信息，利用人工智能辅助给出推荐的算法/函数/源代码/示例。

* **智能测试**

智能单元测试组件

1. 自动生成测试用例：打开单元测试组件，选择要测试的单元，IDE利用Al辅助自动生成测试用例。程序员可以根据自己的需求修改、增、删测试用例。
2. 点击测试，组件就能根据测试用例自动进行测试。
3. 测试结束后，提供结果和错误分析报告，包括错误类型、错误代码定位以及Al辅助的原因推断和debug建议。

* **实时错误预防：**能够立即了解写代码时所做的每个更改的影响，并在用户键入代码时对其进行测试。在错误发生时立即提示/修改错误，并查看现有测试是否覆盖了新更改。

1. 实时对代码中的语句，生成执行该语句的测试输入；实时针对不同的方法，生成测试用例，并进行实时的单元测试。
2. 在代码编辑器中，在每行代码的最左端，显示它是否由单元测试覆盖/覆盖这些代码的测试是否通过。程序员可以点击查看每行代码的测试信息（如：未通过的测试名和用例、未通过的原因、人工智能给出的修改建议）。

* **快速、安全地重构**

在代码中的多个位置进行类似的编辑时（对某个变量或函数进行重命名、移动、删除、提取方法、嵌入变量、向上/向下推动成员、 改变签名等），只需几下点击操作，即可执行全项目范围内的改变，并且可以安全地撤销操作。

* **建模：**开发人员建立标准化模型，IDE将其转换为合适的代码。

**3.小结**

通过画DFD图对IDE进行结构化分析：

1. 画第0层DFD图，使人明确了软件的范围，即与外部实体的界限和接口、通过接口的数据流，这样就把开发工作限定在了规定的范围内。
2. DFD图是逐层分解的，通过逐层画图，我们对软件的功能由上至下地逐层进行分解、细化，从整体和局部的角度了明确分析了各个功能需求，可以说DFD图是一种初步的架构设计。
3. DFD图的数据流使我们明确了软件系统中存在的各种数据以及他们的流向。对于各个“加工”而言，我们已经描述清楚了他们的输入输出，这就类似于定义了函数的接口，使功能更加明确，也方便后续的设计工作。