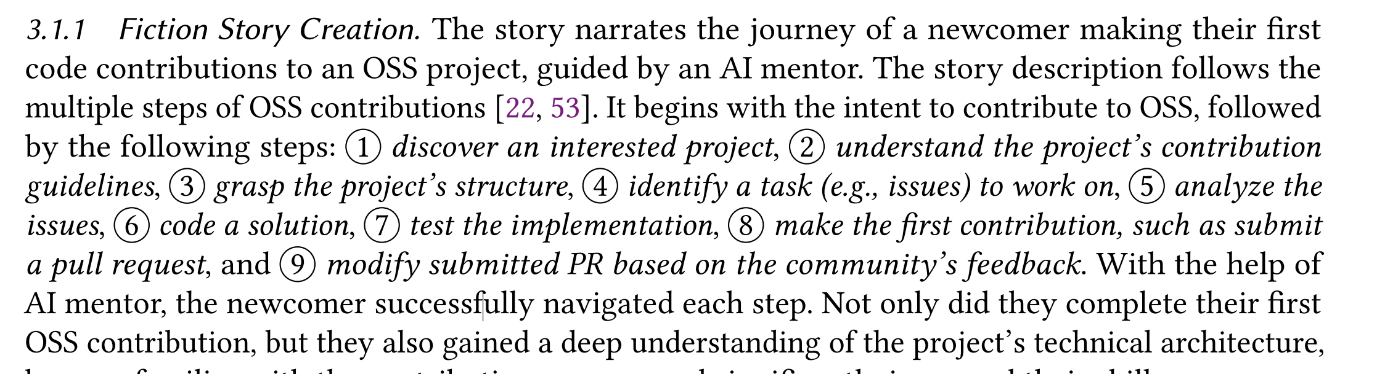
**课题一 Process doc**

**Week01**

**0912讨论：**

1. **不同类型程序员的process有哪些？**
   1. newcommer \*类似OSS
   2. experienced dev



对于每个用户程序员学习不同的行为模型（强化学习、定期更新），状态机？时序预测？

1. **程序员意图：**

**2.1 程序员当前执行的任务是什么？Tasks集合**

build，add 3rd lib, 。。？

**2.2 For each task的具体意图**

**2.2.1 file-level**

references

**2.2.2 element-level**

add new feature, element= API？，artifact具体的工件（路线图画出来的那个输出）

**输入：子问题**

**输出：file或artifact**

question：这个文件xxx.py是实现了什么功能？.eg（课题3，分解子问题）

reference：

1. repo级别给出相关file（课题2，从子问题进行代码检索）
2. person级别给出相关file（课题1，从IDE进行意图建模，其他agent来问@virtualMe）
3. 外部知识库，srs, ts, description, 规则库
4. 合成的，readme, summary

**思考记录**

**程序员意图的定义是什么？**

1. 程序员意图是指编程者在编写代码时的目标、目的或计划。
2. 它涵盖了程序员希望通过代码实现的功能、解决的问题以及期望达到的效果。
3. 程序员意图可以是具体的，比如实现一个算法或功能；也可以是抽象的，比如保持代码的可读性和可维护性。
4. 意图与程序员%在代码层面想要实现的结果直接相关，且常常有明确的短期目标。例如：
   1. 修复一个已知的bug
   2. 添加一个新功能模块优化现有代码的性能
   3. 通过重构提升代码的可维护性
5. 意图也可能具有长期性，例如开发人员如何规划系统架构、模块间的依赖关系、设计模式等，这些意图会对代码的可维护性、可扩展性等产生比较大的影响。
6. 需要注意到，实际开发中，程序员的意图并不是固定的，而是随着编程环境、调试结果等不断变化。例如程序员可能因为某次编译失败而改变当前的意图或者说编程路径。

**IDE如何更好的获取程序员的意图？**

1. 捕捉程序员操作事件，构建事件流机器学习模型，以了解程序员的编程风格和意图。
2. 相关机器学习模型提前分析项目代码的文档（例如需求文档），将需求文档中的功能和项目代码里的函数相匹配。
3. 提供输入意图的功能，即需要程序员显式地将意图告诉模型，为程序员的下一步操作做出预测。
4. 可能一个比较好的方法就是程序员将直接表明自己的开发意图。表明的方式可以是：提供文档、编写需求注释、直接输入要求。

**获取开发者在IDE内的动作信息有什么用？**

1. 分析开发者的操作模式，提供更智能的代码补全、快速修复等功能
2. 根据开发者最近的活动，更容易找到开发者需要的内容（文件、代码片段、变量等），例如自动补全时优先提示这些内容（合适的情况下）
3. ... ...

**需要收集开发者在IDE内的哪些动作信息？**

* 通用开发行为
  + 命令调用：记录开发者通过UI界面、快捷键等方式触发的IDE命令，包括编译、运行、调试
* 文件和项目管理
  + 文件编辑：以SST的形式记录对代码的操作
  + 项目结构管理：追踪项目结构编辑，包括文件的增删、重命名，理解各个文件的用途
* 调试
  + 断点设置和调试步骤：分析开发者调试思路
* 代码补全
  + 代码补全选择：记录开发者选择或者拒绝了哪些建议
* 版本控制
  + 版本控制操作：捕捉代码提交、推送、拉取、合并、回退等操作，以及这些操作涉及的文件和更改内容。
  + 记录commit的更改以及附带的message: message内容可以看作意图，对应commit的代码更改是实现意图的方式，或许有助于训练意图->编码
  + 分支管理：了解各个分支的作用，客制化给出建议
* 测试：
  + 记录测试失败的点以及为此的修复操作，未来在执行测试遇到错误时给出修复建议。
* 导航：
  + Ctrl + 点击：使用该方法通常是为了理解某个复杂的功能或模块，此时可以给出对应的介绍
  + 全局搜索：使用该方法可能是为了下一步调用相关代码，那么可以在代码补全时提升优先级
* 确定了需要收集的信息，还需要细节地讨论这些数据的保存形式，比如代码片段是直接保存源代码还是保存源代码的SST（简化语法树）。

**如何收集开发者在IDE内的动作数据？**

* 将VS Code作为研究主体。首先调研是否已有能够收集开发者动作信息的插件，能收集哪些信息。大概率是找不到能完美满足需求的插件的，考虑自行编写插件来收集需要的信息。
  + VS Code提供了API能够获取开发者在编辑器中的多种操作信息，例如文本编辑，命令执行情况，文件系统事件等
* VS Code 插件开发官方文档 <https://code.visualstudio.com/api>

**本周进度**

**TODO**

* 阅读一篇论文: Enriched Event Streams: A General Dataset For Empirical Studies On In-IDE Activities Of Software Developers。
* 阅读PPT IDE意图预测与代码仓库理解开题报告.pptx
* 自己模拟程序员进行开发工作，尝试 2-3种不同类型的任务(开发、测试、重构、Debug等)。思考在编码过程中，自己哪些在IDE中的动作能体现自己的意图，如何记录这些动作数据。
* 功能开发（用户登陆，以Django为例）

1. 在 User 包(App) 下的 urls.py文件中添加相应路径路径
   1. path('login', views.user\_login, name='user\_login'),
2. 在 User 包下的 views.py文件中实现user\_login视图函数
3. 在User包下的 test.py文件中写测试，测试功能是否正常
4. 过程中记录：代码编辑事件，具体来说：新添内容，所编辑的文件及其所属目录结构

* Debug（测试时发现错误，开始Debug）

1. 定位到未通过的断言语句，然后定位错误原因（例如返回值）
2. 定位到视图函数
3. 根据测试样例，查看视图函数实现逻辑
4. 修正视图函数，重新执行测试
5. 过程中记录：测试失败后Debug事件，具体来说：记录未通过的断言语句、记录对应的视图函数、记录修正过程中的代码更改、记录重新测试的结果

**Week02**

**0913讨论**

模块的输入属什么？只有操作历史还是还有子问题？

**0919讨论**

1. 考虑聚类还是规则
   1. 对于一个动作序列，如果使用聚类获取它的类别，那么需要将所有的动作序列和该动作序列作为输入，输入是一个逐渐增加的过程，耗时也会越来越长，因此聚类或许不适合该任务
   2. 而利用规则判断一个动作序列属于什么类别就没有上述问题
2. ICSE24 IDE Workshop 扫论文

**本周任务**

每人形成一份详细技术路线，同时考虑华为IDE个性化数据PPT框架。调研相关论文或思路。

具体：

1. 根据上面写的意图层次，找对应的场景，把意图清晰建模出来。
2. 考虑我们的模块有什么输入和输出（以及为什么）
3. 使用什么模型？状态机(有哪些状态，如何转移)？时序预测(能不能把问题需求也纳入输入)？微调大模型？以及为什么

**Week03**

**本周进度**

做PPT的同时思考和细化意图预测部分的模型

**Week04**

**问题**

1. 建模开发者行为模型的规则来源比较highlevel，那张PPT需要改
2. 开发者行为/意图模型：重命名，实际是两块，行为和操作件
3. 行为模型分层：比较少、PPT美观问题、为什么这样建？上下对应问题
4. 行为模式细分，哪些是计算出来的，哪些是收集来的，每项的功能，每项是用于意图预测还是能力水平
5. 能力分析和意图预测两张PPT，里面的逻辑联系比较乱
6. 应用场景需要找更好的

**TODO**

下周前刷新一版成熟的PPT

1. 把这个规则来源删掉（如果找不到具体的标准），那张PPT改成总述
2. 重命名为（PPT里用两个半圆表示）：行为；操作件
3. 行为模型：进一步细化，task多找，file和底下的element和上层对应上去
4. 行为模式细分
5. 总分结构，先来一张图表示总体的输入输出，里面包含了这两个模块；再分两张解释两个模块里面的东西。
6. 意图预测那张PPT的美观问题
7. 应用场景：同时体现意图预测和个体理解（启发：当前所处状态不同带来的影响）

**Week05**

完成PPT制作，其中包含的较为完整的方案设想，后续需要开始考虑方案实现。

**Week06**

**1006讨论**

**当前规划**

* 读论文（日常任务）
  + 一方面找idea
  + 另一方面寻找可行技术
* 数据集的构建
  + 先选择一种语言的项目，先做好一种语言的模型
  + 收集哪些数据，这个之前已经讨论过了，形成了一些结论
* 模型的构建
  + 需要构建哪些模型
    - 计算状态机所属当前状态的模型
    - 预测下一步行为的模型
  + 模型的输入输出是什么，完成什么功能？
* 项目原型构建计划
  + 数据（怎么得到，数据要进行哪些预处理）
  + 模型（不太可能自行设计，调研已有的模型，进行修改或微调）
  + 测试（验证原型的可行性，数据在输入到模型后能否得到想要的结果）

**后续规划**

* 是否要做插件，做哪个IDE的插件
  + 要不要做？先问问老师的意见
  + 什么时候做？等项目原型验证通过后
  + 做哪个IDE的插件？VSCode还是华为的DevEco
* 下周要和华为IDE部门交流，需要准备PPT，展示当前进度，并准备好我们的问题（比如：要不要插件，在哪个IDE里做）

**本周任务**

* 调研数据集
  + 可以是寻找已有的数据集
  + 可以是尝试自行构建简单的数据集
  + 任务分配：PZP, LSW
* 调研模型
  + 要做的模型：
    - 计算状态机所属当前状态和下一步状态的模型
    - 预测下一步行为的模型（操作对象，操作行为）
  + 可以考虑的技术：
    - 基于Transformer的预测模型
  + 任务分配：LYH, SYH

**本周进度**

* MSR18论文数据集
  + 测试数据集提供者给的测试程序，程序前半部分可以读取数据集的数据条目，分析数据集里用户动作的数量和类型。但是测试程序后半部分运行有问题，不知道具体功能是什么，程序缺少注释，暂时不知道怎么解决；
  + 了解数据集中数据的构成，每种类似动作的具体结构
* 调研如何把事件流转化成任务、工件和命令。一种可能的技术方案是端到端的无监督分类框架（TnT-LLM）。

**本周讨论**

TODO

* 了解MSR18论文提供的插件，是否可以复用，初步规划插件开发事宜
* 浏览 ICSE24 Workshop，寻找是否有可用的内容
* 将调研得到的结果放到 Obsidian 中 @PZP
* 这周没讲的内容留到下周讲，将内容合并到 Obsidian 中 @SYH
* 整理好问题，制作下周交流需要的PPT

**Week07**

**1011讨论**

任务安排

1. 了解MSR18论文提供的插件，是否可以复用
   1. 插件未开源，且已经下架了，不具有价值
2. 浏览 ICSE24 Workshop，寻找是否有可用的内容（各自安排，如果有特别有价值的可以分享到群里）
3. 将调研得到的结果放到 Obsidian 中 @PZP
4. 这周没讲的内容留到下周讲，将内容合并到 Obsidian 中 @SYH
5. 整理好问题，制作下周交流需要的PPT
   1. 当前成果(方案)：直接复用之前做的PPT
   2. 需要问的问题 @ALL
      1. 是否要开发插件，在哪个IDE上开发
      2. 如果我们确定了数据集长什么样子，华为能不能提供原始数据
      3. 现在的思路有没有问题：通过开发动作得出->操作件预测+个人画像
6. 数据集 PZP LSW
   1. MSR18数据收集，深入了解，先了解示例程序对数据集的处理逻辑，考虑如果要自行编写程序，如何读取数据集和使用其中数据
   2. .... ....
7. 模型 LYH SYH
   1. 要做的模型：
      1. 计算状态机所属当前状态的模型
      2. 预测下一步行为的模型
   2. 可以考虑的技术：
      1. 基于Transformer的预测模型

**本周进度**

1. ICSE'24 IDE Workshop <https://dl.acm.org/doi/proceedings/10.1145/3643796>
   1. Understanding and Evaluating Developer Behaviour in Programming Tasks<https://dl.acm.org/doi/abs/10.1145/3643796.3648450>
   2. Envisioning the Next-Generation AI Coding Assistants: Insights & Proposals <https://dl.acm.org/doi/10.1145/3643796.3648467> 短论文，对AI编程助手的漫谈和愿景，其中提到了他们的CodeVista的一些技术细节
2. 将论文阅读结果放到了 Obsidian 中
3. 华为交流汇报PPT制作
4. 开发初步计划分享
5. 数据集 PZP LSW
   1. 调研了新的开发者动作收集插件MIMESIS，找到了该插件仓库，但是尝试使用该插件没能成功加载 <https://gitlab.informatik.uni-bremen.de/dfgpc/mimesis>

**进度分享**

1. 论文分享
   1. **ICSE'24-IDE-Workshop** Understanding and Evaluating Developer Behaviour in Programming Tasks<https://dl.acm.org/doi/abs/10.1145/3643796.3648450>
   2. **IEEE'21** 介绍MIMESIS插件 Recording, Visualising and Understanding Developer Programming Behaviour <https://ieeexplore.ieee.org/abstract/document/9425961>
2. 自行决定其他内容吧

**下周计划**

* VS Code插件开发?
  + 需要和其他组合作
  + 最首要的就是数据集的收集
  + 基于Continue插件

**Week08**

**分享记录**

1. 考虑调查-编辑-验证【Understanding and Evaluating Developer Behaviour in Programming Tasks】 与 PDCA环 的对应关系；
2. 通过程序员行为的聚焦度等去评估当前任务对该程序员的难度，“行为降级”。对于困难任务，可以考虑帮助程序员分解任务；
3. 与华为汇报当前对开发者行为收集插件的调研结果，并询问能否先用MIMESIS 做demo；
4. 关于动作序列，
   1. 调研时序数据挖掘算法；思考得到的特征如何理解和使用；
   2. 考虑用马尔可夫链预测下一步的动作和**要操作的工件**；
5. 模型训练：pre-train + finetune or 针对每个开发者train自己的模型;BIOLSTM;相邻动作之间的时间间隔

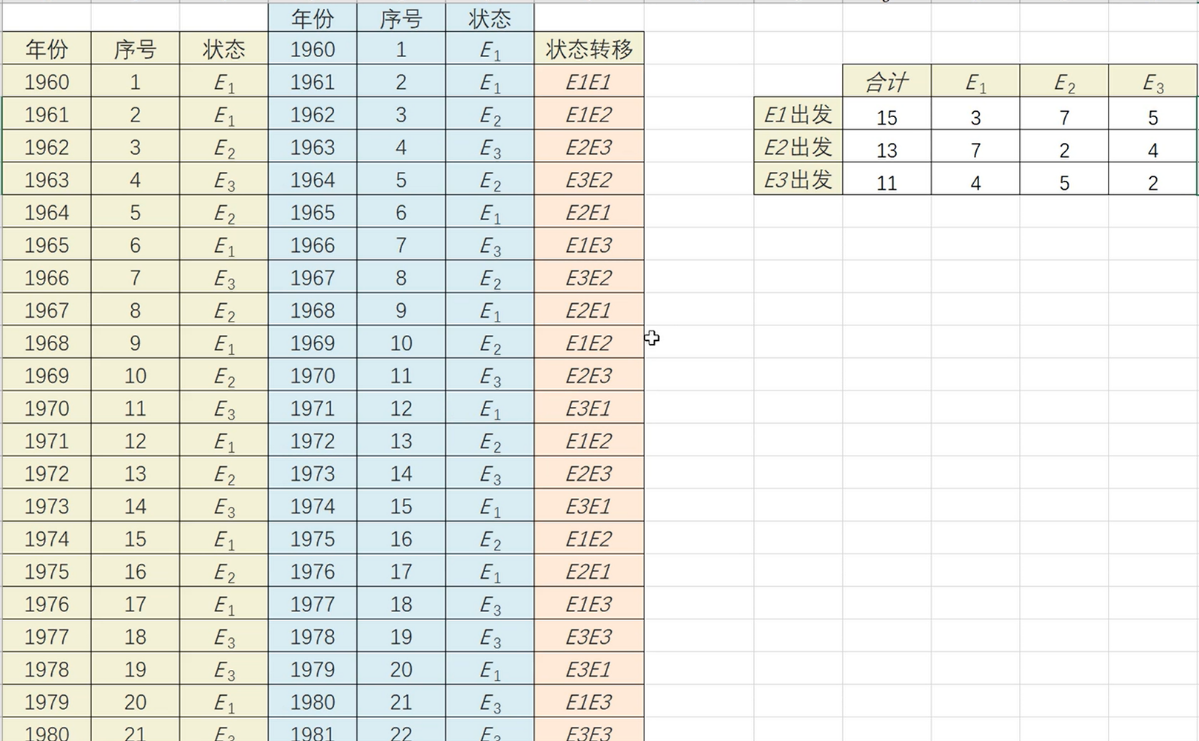
**1018讨论**

1. 重点在模型
2. DevEco CodeGenine 调研，看看当前水平
3. 最近主要的任务是明确模型应用场景

**本周计划**

1. 针对帮**助开发者重写prompt增强问答**的使用场景 SYH
   1. 找一些典型例子和场景，找跟华为找预期效果最好最有用的点，下周交流；
   2. 使用场景做成PPT形式，不必太精致
2. 模型初步尝试：状态机模型 PZP
   1. 解决方法：**寻找一些现成的模型，先跑起来**。
      1. 聚类模型？
      2. 时序数据挖掘算法？
      3. 隐马尔可夫？
   2. 输入：用户动作序列
   3. 输出：用户当前所在状态、用户接下来可能转移到的状态
3. 数据利用 LYH
   1. MSR18已有数据？
   2. MIMESIS插件收集？
4. 熟悉鸿蒙开发场景 PZP
   1. 熟悉DevEco IDE
   2. 试用DevEco CodeGenie 插件，评估该插件效果

**本周进度**

1. DevEco和CodeGenie插件调研
   1. DevEco运行程序需要连接华为设备或者模拟器（模拟器需要先获得授权）
   2. ArkTS应用的目录结构
   3. [CodeGenie介绍](https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V5/ide-codegenie-V5)
2. MIMESIS插件可以在2022-09版本的Eclipse上运行
3. 状态机模型构建
   1. 隐马尔科夫模型
      1. 动作序列 -> 分析当前状态 -> 预测下一个状态
      2. 
      3. python hmmlearn 库
         1. 隐马尔科夫有连续和离散两种，连续用于预测插值，例如拟合曲线；离散用于预测动作。
         2. 隐马尔科夫有一个稳状态，在这个稳状态下各个事件发生的概率趋于稳定。由于我们的模型需要不断接收新的动作序列，所以我们的隐马尔科夫矩阵是不断更新的，也就不存在什么稳状态了。
   2. K-Means（聚类）
4. VirtualMe问答增强场景

（1）当前的DevEco CodeGenie

本周讨论

1. 根据数据集分析开发者的开发行为和开发习惯
2. 需要着手开始数据收集工具开发（或者使用已有数据集或使用已有插件收集数据）

**Week09**

**1024讨论**

1. 状态机应该有哪些状态（现在先简单做，按戴明环分类） （LYH）
   1. Plan 计划
      1. 开发人员处在开发前的状态，比如刚创建好仓库
   2. Do 执行
      1. 开发人员进行开发任务
   3. Check 检查
      1. 运行程序
      2. Debug
   4. Action 处理
      1. 构建
      2. 打包
2. VS Code插件开发 （SYH、PZP）
   1. 原型开发
   2. 确定语言
      1. C / C++
      2. Typescript（备选）
   3. 要收集哪些数据
      1. 先把收集插件原型做出来，再讨论
      2. 时间、事件类型、操作工件

**1028线下讨论**

1. github团队/仓库创建 <https://github.com/iseg-ide-sub1>
   1. 项目文档仓库 <https://github.com/iseg-ide-sub1/docs>
   2. 数据收集插件demo仓库 <https://github.com/iseg-ide-sub1/VirtualMe-demo>
2. 插件开发讨论
   1. 收集哪些数据：见项目文档仓库
   2. 数据如何保存：保存为json文件
3. 状态机模型讨论
4. 操作件预测
5. 收集到的数据在投喂给模型之前先筛选出具有典型pattern的数据(对应构建数据集时需要有partten的有用数据和无规律的噪声数据)
6. 不需要记录添加/删除的代码内容，因为是预测模型不是内容生成模型
7. 数据收集-预处理（时间粒度、字符级操作太频繁就记得太多）-pattern识别（训练分类器or基于统计的规则）-选择三个路子中的一个（只做最近操作的统计or调用代码检索找未知项or调用时序预测找已知项）-答案整合（加权or LLM）

**课上讨论**

1. 目前插件设计针对C++，考虑将其他文件也包含进去，比如测试文件、配置文件、依赖包
2. Auto diff（参考记录的格式），如何判断代码变更的连续性（按时间合并可能会导致两个不相关的操作合并到一起），“操作粘性、编辑粘性”（非标准术语），能否通过大模型总结用户在某个操作序列中干了什么，一个完整合并后的操作可能是跨文件的操作，识别用户操作被打断的情况（大模型生成摘要？）
3. 在空间里建模用户操作的轨迹，用户的轨迹在时间维度上是否存在周期性
4. 调研清华闻立杰：工作流挖掘相关研究；图论：建模开发者的操作，向量里塞什么，兼容命令和操作件，相当于高维空间里的轨迹。

**Week10**

**1101讨论**

本周TODO

**插件部分** SYH PZP

1. 插件的改进，考虑将其他文件也包含进去，比如测试文件、配置文件、依赖包
   1. 对于有编写规则的文件（代码、配置文件）可以按VS Code提供的API对具体结构进行分析
   2. 对于没有规则或者无法得知规则的文件（文本文件、测试数据）客观的记录用户对文件的操作即可
2. 插件对于代码操作的合并逻辑优化
   1. 原来是按时间进行合并，合并时间间隔小的操作
   2. 新思路：按操作的逻辑进行合并

**模型部分** LYH PZP

状态机模型暂时先不考虑

1. “操作粘性”模型：对于两个时间跨度较大的操作，调用该模型，用于判断两个相邻的操作是否是一致的 / 可以合并的
   1. 简单实现：不用机器学习模型，单纯基于某些规则来进行判断
   2. 机器学习方法实现：前者效果不好再考虑
2. 液态神经网络

**调研部分**

1. 如何获取本地文件的git记录（commit message, diff信息），这些信息能否用于和怎么用于我们的模型（后续有初步模型了再考虑）
2. 在空间里建模用户操作的轨迹，用户的轨迹在时间维度上是否存在周期性，图论：建模开发者的操作，向量里塞什么，兼容命令和操作件，相当于高维空间里的轨迹
3. 调研清华闻立杰：工作流挖掘相关研究 LSW

**课上讨论**

TODO

1. 基于规则判断操作粘性
   1. 本文件内，例如两个事件编辑的代码行号之间的distance（如果用机器学习来做这个，也需要用到distance）
      1. 基于规则类似于SVM，决策树
      2. 没有数据的问题，自己做的数据才合适。有了自己的数据之后才能继续做
   2. 跨文件
2. 可以继续调研工作流论文，日志内容挖掘
   1. pinjia he
   2. 根据关键词顺藤摸瓜，找到领域论文
   3. 本周调研的结果——闻立杰-工作流挖掘先记下来，后面如果液态神经网络效果不理想再考虑
3. 我们收集的数据集可以和MSR18论文对标，也发篇论文
4. <跳转方向>可以有很多拓展
   1. 做一个迭代
5. 怎么收集数据？
   1. 再等一等，先数据建模
6. 设计插件分发、数据收集流程
   1. 让科研课堂上的同学们装上，正常进行开发活动

**Week11**

**1108讨论**

**上周TODO**

1. 操作粘性：该问题后置
2. 工作流挖掘
   1. 已经调研的：多模态（后续有需要再继续了解）
   2. 贺品嘉：调研相关研究内容 LSW LYH
3. 思考数据集的收集和构建
   1. 插件要保证开发完成
   2. 插件的发布
   3. 如何找到合适的参与者（后续考虑）

**插件开发（本周重点） PZP**

1. 当前的选择操作会记录选择的内容，需要优化
   1. 改为记录当前选择内容所包含的artifact
2. 需要**增加**一些操作的记录
   1. 创建文件夹 / 删除文件夹
   2. 键盘操作
   3. 运行命令
   4. 滚轮操作
   5. 搜索 / 替换
   6. **识别到代码地图更改时将其记录到最近的一个操作中**

（有现成的生成代码地图的方案） LYH

* 1. etc. (参考MSR'18,IEEE'21查漏补缺)

1. 发布：虽然现在插件功能还不不完整，可以先试试能不能打包出来并发布

**模型实现**

1. 先需要数据，后端建模

**课上讨论**

1. 通过实际操作IDE，找能检测哪些事件
2. 终端输出文本的选中复制
3. 实验refrence的收集，整个repo的搜索会不会爆炸，可行性
4. 推进准备收集数据的流程，参考MSR18

**Week12**

**1115讨论**

**插件开发**

1. 插件实现不用太过考虑数据预处理，对于冗余或不合规的数据可以在收集后进行处理
2. 本周应该能初步开发完插件，尝试打包

最近计划：

1. Week12基本完成插件开发
2. Week13收集数据
3. Week14开始尝试模型训练

**课上讨论**

1. IDE内terminal 和 console的输入数据和输出数据收集
2. 程序员在IDE的关注区域，以及区域内的信息，能用它来干什么
3. 在代码地图（group2）中查找 reference，目前是字符串匹配的形式查找 reference
   1. 目前先这样，等待第二组
4. 鼠标悬停看api文档
5. 确保能够获取到开发者通过 button 还是 terminal 进行的交互
   1. 优先获取通过 button 进行的行为
   2. 其次是输出内容
   3. 最后是输入
6. 自顶向下，程序员使用 IDE 能进行的行为列表
7. 增量式训练