**迭代计划（简化版）**

　　　　　　　　　　　　　制定日期：

|  |  |  |  |
| --- | --- | --- | --- |
| 组号 | 15 | 项目名称 | MY PC Logo |
| 迭代名称 | 技术原型迭代 | 计划起止日期 | 2020.10.11 – 2020.11.13 |
| 任务、进度安排和人员分配：   |  |  |  |  | | --- | --- | --- | --- | | No | 任务 | 起止日期 | 人员 | | 1 | 讨论概念建模的方案，每个人预先绘制一份草图，交流互补 | 2020.10.11-2020.10.12 | 所有人 | | 2 | 用pd绘制概念模型 | 2020.10.12 | 王执 | | 3 | 讨论制定迭代计划 | 2020.10.13 – 2020.10.14 | 所有人 | | 4 | 根据讨论结果，撰写迭代计划 | 2020.10.14 | 邹立凯 | | 5 | 论证语言、框架、工具的合理性，确定最终的技术栈 | 2020.10.15 – 2020.10.16 | 所有人 | | 6 | 寻找合适的已有开源组件，迁移到我们的框架上，减少将来的工作量 | 2020.10.17 – 2020.10.18 | 吴侃真、毛彦凯 | | 7 | 分析各个Use case模型，并找出其实体类，边界类，控制类 | 2020.10.19 – 2020.10.20 | 所有人 | | 8 | 确定编程规范 | 2020.10.21 –2020.10.22 | 所有人 | | 9 | 根据讨论结果，绘制出用例实现的时序图，通信图以及VOPC图 | 2020.10.23 –2020.10.24 | 吴侃真、邹立凯、王执 | | 10 | 整合各用例的用例实现 | 2020.10.25 | 毛彦凯 | | 11 | 根据用例实现，讨论项目的整体架构。 | 2020.10.26 – 2020.10.27 | 所有人 | | 12 | 设计数据库 | 2020.10.28 | 吴侃真 | | 13 | 评审并修改数据库设计 | 2020.10.29 | 所有人 | | 14 | 绘制软件架构设计图 | 2020.10.30 –2020.10.31 | 邹立凯，吴侃真 | | 15 | 撰写软件架构文档 | 2020.11.1 | 毛彦凯，王执 | | 16 | 讨论架构的改进 | 2020.11.2 – 2020.11.3 | 所有人 | | 17 | 搭建后端框架 | 2020.11.4 –2020.11.7 | 吴侃真 | | 18 | 绘图组件开发 | 2020.11.4 –2020.11.8 | 邹立凯 | | 19 | 根据讨论结果完善软件架构文档 | 2020.11.8 | 毛彦凯 | | 20 | 设计logo语言编译器 | 2020.11.9 – 2020.11.10 | 毛彦凯、吴侃真 | | 21 | 初步完成服务器端开发 | 2020.11.11 – 2020.11.13 | 吴侃真 | | 22 | 根据界面原型完善前端界面 | 2020.11.11 – 2020.11.14 | 邹立凯 | | 23 | 前后端通信测试 | 2020.11.15 | 吴侃真、邹立凯 | | 24 | 根据测试结果调整前后端 | 2020.11.16 | 吴侃真、邹立凯 | | 25 | 经组内评审后，完成迭代评估报告 | 2020.11.17 | 毛彦凯、王执 | | | | |
| 预期成果：  文档及模型：   1. 概念模型 2. 软件架构文档 3. 技术原型迭代评估报告 4. 架构视图 5. 关键算法设计 | | | |
| 主要的风险和应对方案：   1. 技术风险  * 美观、有良好交互性的UI绘制及turtle绘制图形的展现 * 低延迟的联机通信 * 类似IDE的语法检查与智能提示   应对方案：可参考市面上同类产品与相关设计书籍教程对UI进行设计；深入学习网络IO与并发的各种知识，尝试不同方案选取最优；实时智能语法可以参考编译原理中学到的知识。解耦各个功能模块，即使某个技术无法实现，也不会影响到主体功能的使用。   1. 架构风险  * 搭建低延迟，高并发的联机架构 * 具有高绘图性能、执行稳定的单机架构   应对方案：针对联机架构，抛弃笨重的Spring后端，使用Go语言搭建轻量且具有高并发性能的服务器；针对单机架构，深入了解不同系统的绘图API，以获得比封装函数更好的性能。   1. 进度风险  * 其余课程难度大、抢占时间，可能影响本课程项目进度 * 开发过程中遇到难题使得进度推迟，打乱预订计划   应对方案：对整个项目需要完成的任务进行归类，设定优先级，先执行优先级高的任务，在时间非常紧张的情况下只能舍弃优先级低的任务，如进阶需求、详细文档等。   1. 集成风险  * 单机联机模块对接可能产生意想不到的问题   应对方案：对模块解耦时设计良好接口，并严格遵循接口约定不轻易改变。准备多套集成方案，确保一种方案有良好的模块兼容性。   1. 需求风险  * 项目中期需求变更 * 进阶需求实现难度较大 * 由于项目经验不足，随着项目的推进才发现组内预先设定的需求不合理，需要修改   应对方案：进行良好的架构设计，尽量提高代码的可读性、可维护性，使得在需求发生变动时，能够较为快速地在原有项目基础上进行修改。 | | | |