**迭代计划（简化版）**

制定日期：2020-10-12

|  |  |  |  |
| --- | --- | --- | --- |
| 组号 | 14 | 项目名称 | PC Logo |
| 迭代名称 | 技术原型迭代 | 计划起止日期 | 2020.10.12-2020.11.11 |
| 任务、进度安排和人员分配：   |  |  |  |  | | --- | --- | --- | --- | | No | 任务 | 起止日期 | 人员 | | 1 | 编写迭代计划 | 2020.10.12 | 苏浩然 | | 2 | 小组讨论，确定最终选择的语言框架工具和架构风格 | 2020.10.13 | 全体成员 | | 3 | 小组讨论，确定概念类图中需要实现的概念类 | 2020.10.14 | 全体成员 | | 4 | 画概念类图 | 2020.10.15-2020.10.16 | 吴泽川 | | 5 | 根据界面原型迭代答辩时助教的建议，对界面原型进行改进，小组讨论改进的方法 | 2020.10.16-2020.10.17 | 全体成员 | | 6 | 小组讨论，对概念类图进行分析评审 | 2020.10.17 | 全体成员 | | 7 | 根据讨论结果，对概念类图进行最终修改、提交 | 2020.10.18 | 吴泽川 | | 8 | 根据小组对界面原型讨论的结果，着手对界面原型进行初步修改 | 2020.10.18-2020.10.19 | 张哲溢、吴泽一 | | 9 | 设计软件物理视图 | 2020.10.19-2020.10.20 | 苏浩然 | | 10 | 设计解释器算法 | 2020.10.19-2020.10.20 | 吴泽川 | | 11 | 根据小组对界面原型讨论的结果，着手对界面原型进行进一步更细致的修改 | 2020.10.20-2020.10.21 | 张哲溢、吴泽一 | | 12 | 设计软件逻辑视图 | 2020.10.21-2020.10.22 | 苏浩然 | | 13 | 小组内部对修改后的界面原型进行评审 | 2020.10.22 | 全体成员 | | 14 | 根据小组对界面原型评审的结果，着手对界面原型进行最终修改 | 2020.10.23-2020.10.24 | 张哲溢、吴泽一 | | 15 | 实现基本的解释器算法 | 2020.10.24-2020.10.25 | 吴泽川 | | 16 | 小组内部对物理视图和逻辑视图进行评审、修改 | 2020.10.24-2020.10.25 | 全体成员 | | 17 | 小组讨论，确定软件总共需要设计哪些视图 | 2020.10.26-2020.10.27 | 全体成员 | | 18 | 对解释器算法进行进一步改进 | 2020.10.26-2020.10.27 | 吴泽川 | | 19 | 设计软件架构中的其它视图 | 2020.10.28-2020.10.29 | 苏浩然 | | 20 | 撰写编程规范 | 2020.10.28 | 张哲溢 | | 21 | 撰写软件架构文档 | 2020.10.30 | 苏浩然 | | 22 | 小组内部对编程规范和软件架构文档进行评审修改 | 2020.10.31-2020.11.01 | 全体成员 | | 23 | 运用设计模式对软件类图进行详细设计 | 2020.11.02-2020.11.03 | 苏浩然 | | 24 | 对软件其它相关技术的选用进行小组讨论 | 2020.11.04-2020.11.05 | 全体成员 | | 25 | 实现语音识别技术 | 2020.11.06-2020.11.07 | 吴泽一 | | 26 | 小组内部对软件详细设计进行评审、修改 | 2020.11.08 | 全体成员 | | 27 | 小组讨论并测试技术原型是否达到技术要求 | 2020.11.09-2020.11.10 | 全体成员 | | 28 | 对技术原型进行进一步的测试 | 2020.11.10-2020.11.11 | 张哲溢、吴天龙 | | 29 | 根据测试结果进行改进 | 2020.11.11-2020.11.12 | 吴泽川 | | 30 | 小组内部对迭代进行评审 | 2020.11.14-2020.11.15 | 全体成员 | | 31 | 编写迭代评估报告，撰写评审记录、测试记录和开发总结 | 2020.11.16-2020.11.17 | 吴泽一 | | | | |
| 预期成果：  文档与模型：《Vision文档》、use-case模型、《软件需求规约》、《软件架构文档》、编程规范、UML分析与设计模型  代码：技术原型涉及到的代码  迭代：《迭代计划》、《迭代评估报告》 | | | |
| 主要的风险和应对方案：   1. 进度风险。项目有可能因为时间问题难以在截止日期前完成迭代要求。缓解应急措施：加强团队沟通与交流，及时召开小组会议，及时制订迭代计划，估算团队开发速率和项目进度，及时修正迭代计划，严格按照迭代计划执行。当进度风险十分急迫时，可以采取增加工作时长的方法来应急。 2. 技术风险。小组成员有可能对相关技术了解不够充分，导致软件开发出现困难。缓解应急措施：提前学习相关技术，对团队成员的技术能力进行充分的预判，在使用不熟悉的技术前进行充分的学习、调试。在遇到难以自己克服的困难时，及时向助教、老师或者其他小组成员寻求帮助 3. 质量风险。项目有可能无法达成某项质量指标要求。缓解应急措施：严格执行符合要求的软件开发流程，认真进行code review，及时对产品代码进行评审，进行全方面的严格的测试，及时与需求方沟通并确认质量要求。在质量问题难以改善时，尽早采取其他架构，减少沉没成本。 4. 需求风险。小组成员有可能在最初工作开始时，对需求没有理解充分，需求有可能因为种种原因在项目开发过程中发生变更。缓解应急措施：及时地召开小组会议对产品需求进行评估，认真评估项目相关的需求文档，加强小组成员之间的沟通，对项目需求充分理解。采用合理的软件开发流程，采用演化模型等相关方式，以更好地应对需求变更。 | | | |