项目测试计划书

项目名称： 流程图编辑器（Flowchart Editor）测试实践

课程名称： 软件质量与评测技术

团队名称： [the stone roses]

提交日期： 2025年11月26日

---

1. 引言

1.1 编写目的

本测试计划旨在规划对"流程图编辑器"项目的全面测试活动。通过本项目测试实践，验证软件是否满足《流程图编辑器需求文档》中规定的功能需求和性能要求，发现潜在的软件缺陷。

1.2 项目背景

被测项目为往届学生团队（utnubu组）开发的基于Qt框架的流程图编辑器。该软件主要用于业务流程建模和软件开发逻辑设计，支持图元绘制、连线管理、文件操作及属性编辑等核心功能。

1.3 测试范围

本次测试将覆盖以下内容：

- **功能测试：** 绘图工具、连接管理、编辑功能、属性编辑、文件操作（导入/导出/保存）等。
- **非功能测试：** 易用性测试、用户文档检查。
- **智能化测试：** 适配并应用智能Agent进行静态代码扫描与自动化用例生成。

---

2. 测试环境与工具

2.1 硬件环境

- 处理器：Intel Core i7/i9
- 内存：16GB 及以上
- 操作系统：Windows 10/11 (64-bit)

2.2 软件环境

- **开发框架：** Qt 6.10.0
- **编译器：** MinGW 64-bit (Debug模式)
- **IDE：** Qt Creator 10.0+
- **被测源代码：** `diagramscene_ultima` 文件夹

2.3 测试工具

- **手动测试工具：** 截图工具、录屏软件。

- **智能化测试工具：** 智能测试Agent（Qt适配版）。
    - 该工具基于LLM构建，计划针对C++/Qt特性进行Prompt工程优化和接口适配。

---

3. 测试内容与策略

3.1 静态测试策略

- **代码审查：** 重点审查 `ZigLine` 折线算法（64种情形散列编码）的逻辑覆盖率，以及 `UndoStack` 撤销重做机制的内存管理。
- **文档检查：** 对比《需求文档》与《用户文档》，验证功能描述的一致性，特别是"错误自检"和"协同开发"功能的实际实现情况。

3.2 动态测试策略 (黑盒测试)

将采用等价类划分、边界值分析和错误推测法设计测试用例，重点关注：

- **核心绘图：** 图元拖拽、缩放、旋转及图层调整（前置/后置）的流畅度。
- **连线交互：** 测试不同象限下折线算法的自动寻路准确性，以及连线吸附功能的灵敏度。
- **文件系统：** `.fcproj` 工程文件的序列化/反序列化稳定性，以及 SVG/PNG 导出的保真度。
- **快捷键响应：** 验证 Ctrl+C/V/Z/Y 等快捷键在不同焦点下的响应情况。

3.3 易用性测试

- 评估界面布局（工具栏、属性栏、菜单栏）的合理性。
- 依据《用户文档》进行全流程操作，验证帮助文档的准确性和指导意义。

---

4. 智能化测试工具应用方案

为响应课程关于"自行开发软件测试智能化工具"的要求，团队将对既有的 Agent 进行 **"测试专用化"** 改造，剔除无关功能，专注于**测试生成**。

4.1 工具定义：智能测试生成 Agent

该 Agent 旨在解决传统单元测试编写成本高、覆盖率低的问题。其核心功能包括：

1. **黑盒测试用例生成 (Test Case Generation)：**
    - **输入：** 需求文档中的功能描述（如"图元连接管理"）。
    - **处理：** Agent 分析业务逻辑，利用思维链（CoT）推理出正常流、异常流及边界条件。
    - **输出：** 结构化的测试用例表格（包含前置条件、测试步骤、预期结果）。
2. **白盒单元测试脚本生成 (Unit Test Code Generation)：**
    - **输入：** 核心类代码（如 `DiagramItem` 或 `DeleteCommand`）。
    - **处理：** 识别代码中的关键路径和分支（如 `ZigLine` 的 switch-case 分支）。
    - **输出：** 基于 **QTest** 或 **Google Test** 框架的可执行 C++ 单元测试代码，自动构造 Mock 对象和断言。

4.2 实施步骤

1. **Prompt 工程优化：** 定制针对 Qt C++ 特性的 Prompt 模板，教导 Agent 理解 Qt 的信号槽机制和图形项（QGraphicsItem）属性。
2. **批量生成：** 将 `diagramscene_ultima` 中的关键算法类输入 Agent，批量生成单元测试文件。
3. **人工验证与执行：** 将生成的测试代码加入工程进行编译运行，记录通过率。

---

5. 进度安排

根据课程关键时间节点要求，制定如下进度表：

阶段	时间	主要任务	对应课程节点
启动与计划	11.25 - 11.26	搭建环境，提交《测试计划书》，进行计划陈述	**11.26 项目测试计划陈述**
功能测试执行	11.27 - 12.10	执行黑盒测试用例，记录 Bug，完成《项目测试实践》主体测试工作	-
智能工具研发	12.01 - 12.14	适配 Agent，生成并运行单元测试代码，验证智能化效果	-
中期汇报	12.15	汇报测试进度及目前发现的主要缺陷	**12.15 项目中期进展汇报**
工具展示与冲刺	12.16 - 12.20	**展示智能化工具成果**；完成个人单元测试实践任务	**12.17 智能化工具汇报**
报告撰写	12.21 - 12.25	提交个人单元测试报告；撰写团队项目文档	**12.21 提交单元测试报告**
回归与收尾	12.26 - 12.30	对修复后的版本进行回归测试，美化答辩 PPT	-
最终交付	12.31 / 01.04	进行最终答辩，提交所有项目文档及工具源码	**12.31 答辩 / 01.04 提交文档**

---

6. 风险评估与管理

- **环境风险：** Qt 版本不一致可能导致构建失败。
  - *对策：* 严格按照《QT项目启动》文档配置 MinGW 6.10.0 环境，确保路径无中文。
- **技术风险：** Agent 生成的代码可能存在幻觉（Hallucination）。
  - *对策：* 所有 AI 生成的测试代码和修复建议必须经过人工 Code Review 后方可采纳。

---

7. 交付成果

1. 项目测试计划书
2. 项目测试缺陷清单 (Bug List)
3. 单元测试实践实验报告
4. **软件测试智能化工具开发报告** (含 Agent 源码、设计文档及运行结果)
5. 项目测试总结报告
