

SPLINT\_V可视化静态代码检测工具

2015-5-1

作者：亚航



需求规格说明书

目录

[目录 1](#_Toc16443)

[1前期调研 4](#_Toc31925)

[1.1背景 4](#_Toc3463)

[1.2基本概念 4](#_Toc16190)

[1.3可行性分析 5](#_Toc6339)

[2编写目的 5](#_Toc11944)

[3需求规定 5](#_Toc31304)

[3.1功能需求 5](#_Toc20841)

[3.2性能需求 6](#_Toc14389)

[3.4开放性和可扩展性需求 6](#_Toc12473)

[4开发工具 6](#_Toc11304)

[5运行环境 6](#_Toc2010)

[5.1硬件环境 6](#_Toc21491)

[5.2软件环境 6](#_Toc13965)

[6风险管理 7](#_Toc19647)

[6.1需求风险 7](#_Toc22552)

[6.2质量风险 7](#_Toc29816)

[6.3技术风险 7](#_Toc11142)

[6.4项目组成员能力素质风险 7](#_Toc19326)

[6.5进度风险 8](#_Toc7962)

[7. 面向对象分析与建模 8](#_Toc16619)

[7.1层次图 8](#_Toc8189)

[7.2代码分析流程图 8](#_Toc9704)

[7.3 USD用例图 9](#_Toc6319)

[8 时间安排 10](#_Toc10483)

[8.1前期准备 10](#_Toc7116)

[8.2时间安排 10](#_Toc32767)

[8.3形成文件 11](#_Toc26141)

[9 参考文献 12](#_Toc24782)

# 1前期调研

## 1.1背景

随着信息社会的发展，网络的不安全问题越来越严重，许多安全问题都是由于软件本身存在安全漏洞引起的，并且造成了不少的经济损失。原因诸多，有程序员编写程序时的疏忽，也有些是因为相关语言没有提供完整的安全机制。在众多的语言中，c语言是被认为最容易收到攻击的语言，它不像 java一样有自己的运行虚拟机，在可能存在一些诸如缓冲区溢出攻击的潜在危险时会自动检查。 所以代码的安全检查显得十分重要，但是由于现代软件工程越来越复杂，致使传统的检查方法在准确度和效率上都不能满足要求，所以代码的静态安全检测应运而生。

## 1.2基本概念

静态方法是指不运行被测程序本身，仅通过分析或检查[源程序](http://baike.baidu.com/view/546605.htm" \t "http://baike.baidu.com/_blank)的语法、结构、过程、接口等来检查程序的正确性。对需求规格说明书、[软件设计](http://baike.baidu.com/view/575391.htm" \t "http://baike.baidu.com/_blank)说明书、[源程序](http://baike.baidu.com/view/546605.htm" \t "http://baike.baidu.com/_blank)做[结构分析](http://baike.baidu.com/view/551724.htm" \t "http://baike.baidu.com/_blank)、[流程图](http://baike.baidu.com/view/7289.htm" \t "http://baike.baidu.com/_blank)分析、符号执行来找错。静态方法通过程序静态特性的分析，找出欠缺和可疑之处，例如不匹配的参数、不适当的[循环嵌套](http://baike.baidu.com/view/7144415.htm" \t "http://baike.baidu.com/_blank)和分支嵌套、不允许的递归、未使用过的[变量](http://baike.baidu.com/view/296689.htm" \t "http://baike.baidu.com/_blank)、空指针的引用和可疑的计算等。静态测试结果可用于进一步的查错，并为[测试用例](http://baike.baidu.com/view/106882.htm" \t "http://baike.baidu.com/_blank)选取提供指导。而静态代码检测工具则是通过静态分析程序中的语法结构以及借口等来判断程序中有的错误或者潜在的危险。

静态代码检测的优缺点如下：

1、静态分析的主要优点是，它使消除软件瑕疵的成本大幅减少。错误发现得越早，修改所需的成本越小。静态分析无需依赖你正在使用的编译器和被编译程序的执行环境。这使你可以察觉到某些可能经过数年时间才能表现出来的隐藏错误，比如那些未定义的行为错误。当转换到其他编译器版本或使用其他代码优化开关时，此类错误才会表现出来。

1. 静态分析对内存泄露和并发错误的诊断较差。要检查此类错误，你需要虚拟化执行部分程序。静态分析工具会警告你碎片的存在。事实上，代码是正确的。这种现象称为误报。等等。

## 1.3可行性分析

1. 技术可行性

在splint开源软件上做出功能扩展，需要一些技术。在原有软件上包装并作出扩展，首先需要找出其的数据输入输出流，这样可以进行包装。其次分析splint的结果，提取多种结果，经过处理后进行多种方式的显示。

1. 操作可行性

本splin\_v软件需要如下的运行环境：安装有Windows7/8/8.1操作系统中的其中一种。因此，从操作可行性来看，只要用户的硬件和操作系统满足以上条件，即可运行。

1. 经济可行性

该项目的核心程序等已较为成熟，而该项目应用非常广泛，有很多成熟但未开源的商业软件。一旦完成具有相当光明的前景,如果开源出去会有很好的效果。

# 2编写目的

编写本报告是为了进一步明确splin\_v系统的结构，目的包括：明确系统开发历史、现状及可行性分析；明确系统开发环境及工具；确定系统需求，包括功能，性能，数据管理，数据库存储，数据输入输出，结果分析等；前期准备：安排分工，用图表确立时间节点；确定系统的风险管理，解决方案；用E-R图，用例图，顺序图，框架图，类图等进行面向对象分析；最终形成代码和可执行文件。

# 3需求规定

## 3.1功能需求

1. 打开一个.c或者导入一个c工程文件。
2. 在界面上显示分析结果。
3. 点击结果中的错误时，在上方会高亮显示。
4. 能够将错误进行分类，并导出相应的分析报告。
5. 版本控制，分析结果存储在数据库中，进行数据持久化。
6. 导出图表，将数据进行分析显示在图表中。

## 3.2性能需求

1. 准确度。软件运行的正确，能够分析单独的文件和大型的项目工程，准确的检测错误并显示结果，并将分析结果准确的显示在图表和报表中。

本系统支持：

单独的.c文件；

整个c工程项目。

2.效率。由于处理的是c文件或者工程，要求运行效率高，较多的文件可以在一定时间内运行完毕；由于导出的是多种图表和报表，需要此部分功能能够高效完成。

## 3.4开放性和可扩展性需求

如果条件允许最好可以将该以开源软件为基础的扩展性软件进行开源；在需求上，最好可以将版本实现在本地存储，这样对于系统中没有数据库的用户可以方便配置环境。

# 4开发工具

VS2012、SQL Sever数据库

# 5运行环境

## 5.1硬件环境

装有vs的 windows系统即可。

硬件要求不严格。

## 5.2软件环境

应用服务器：不支持远程终端服务，因此不能用于服务器，只能通过本地控制台使用软件；

操作系统：支持Vista/Windows7/Windows8/Windows8.1，包括32位和64位版本；

# 6风险管理

## 6.1需求风险

需求已经成为项目基准,但需求还在继续变化;需求定义欠佳,而进一步的定义会扩展项目范畴;添加额外的需求;在需求分析中客户参与度不够;预防这种风险的办法是建立有效的需求变化管理，设置工作人员密切关注项目需求的变化，及时做出相应的改动和调整。

## 6.2质量风险

本系统对产品质量、程序优化方面，良好的设计模式有很高的要求，并且项目组成员对此类型项目的开发经验不足，需要密切关注项目的质量风险。预防这种风险的办法是广泛调查获取人们的需求、采用符合要求的开发流程、认真组织对阶段成果的检查和评审、计划和组织严格的独立测试等。

## 6.3技术风险

项目组需要本着项目的实际要求，选用合适、成熟的技术，不能无视项目的实际情况而选用一些虽然先进但并非项目所必须而自己又不熟悉的技术，也应该避免为了设计模式而设计模式，在一些不必使用设计模式的地方强加一些设计模式。Splint\_V静态代码检测系统对检测效率和正确结果的要求很高，还需要具有良好的扩展性，需要重点关注该风险因素。预防这种风险的办法是选用项目所必须的技术，充分学习设计模式并为系统设计一个良好的模式，方便以后扩展功能，让软件更加符合用户的使用。

## 6.4项目组成员能力素质风险

团队成员的能力（包括业务能力和技术能力）和素质，对项目的进展和质量具有很大的影响，项目组组长在项目的建设过程需要实时关注该因素。预防这种风险的办法是在用人之前先选对人，把相应的人分配不同的工作，各有所长，才能有效地防止一些风险。

## 6.5进度风险

大量的纸面工作导致进程比预期的慢;前期的质量保证行为不真实,导致后期的重复工作;太不正规(缺乏对软件开发策略和标准的遵循),导致沟通不足,质量欠佳,甚至需重新开发;过于正规(教条地坚持软件开发策略和标准),导致过多耗时于无用的工作;采用敏捷模型开发，可能对文档的撰写不是很严格;风险管理粗心,导致未能发现重大的项目风险。为了预防进度风险，需要增加项目监控的频度和力度，增加团队成员之间的沟通交流，确保项目进度处于可控状态，简化不必要的繁琐工作，保证质量、避免返工。

# 面向对象分析与建模

## 7.1层次图



## 7.2代码分析流程图



## 7.3 USD用例图



# 8 时间安排

## 8.1前期准备

安装相关工具，学习相关工具的使用方法。配置windows下的splint环境，学会使用该工具，寻找splint输入输出流，调试代码，分析其流程和具体功能。

## 8.2时间安排

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 第1周 | 小组成员确定 | 搜集资料，了解静态检测，下载软件 | 配置linux下splint，通过manual使用手册学习使用 | | 寻找到llmain主函数 | 使用Enterprise Architect逆向分析该软件 | |
| 第2周 | 下载windows下的splint，并学习配置环境 | | 分析代码的注释，获取信息 | 汇报项目进度，听取他组内容，取长补短 | 完成配置windows下的环境 | 调试程序，寻找数据输入输出流 | |
| 第3周 | 确定需求，进行分工，分析和设计并行进行 | 获取输出流，设计界面 | 制作界面，实现导出文件和项目 | 汇报项目进度，听取他组内容，取长补短 | 分析代码检测流程，制作相应的流程图 | | 深入分析parse部分 |
| 第4周 | 完成分析文档，进行设计模式设计 | | 设计报告内容和代码结构 | 设计图表结构和数据接口，研究规则库 | | 设计版本控制的逻辑和代码结构 | |
| 第5.6周 | 使用观察者模式，写代码 | 完成报表的导出 | 数据库和版本控制连接，完成版本部分 | | 完成图表生成和导出 | 高亮显示代码，研究如何潜入一个编辑器 | |
| 第7周 | 实现潜入一个编辑器 | | 美化界面 | 综合讨论学习规则库设计 | | 尝试加入规则库 | |
| 第8.9.10周 | 需求、概要、详细设计、测试、使用说明书文档书写 | | | 收尾汇报 | 根据老师建议修改 | | |

## 8.3形成文件

需求分析说明书、设计说明书、可以运行的原型系统，整体配置文件和说明书。

# 9 参考文献

[1].饶坤，彭四伟.C／C++源代码静态检测系统的设计和实现[J].电子设计工程.2011,19(12):28-31.

[2].刘春燕,赖晓晨.基于关系语法树的C／C++代码静态检测方法研究[J].2013电子科技大学.2011,32(8):2879-2900.

[3].梁婕，张淼.基于静态分析技术的源代码安全检测模型[J].计算机应用研究.2008,25(9):2703-2705.