360技术嘉年华-测试之美

代码质量管理系统

Ensuring Code Quality

团队介绍

自动编译

真机远程租用

代码质量管理

流程管理

用例管理

- 来自360 QA
- ONES DEVOPS, 规范流程
- PC，Android自动化测试技术
- g-qa-dev@360.cn

ONES DEVOPS

企业级研发管理方案

360旗下所有客户端产品
正在使用ONES DEVOPS

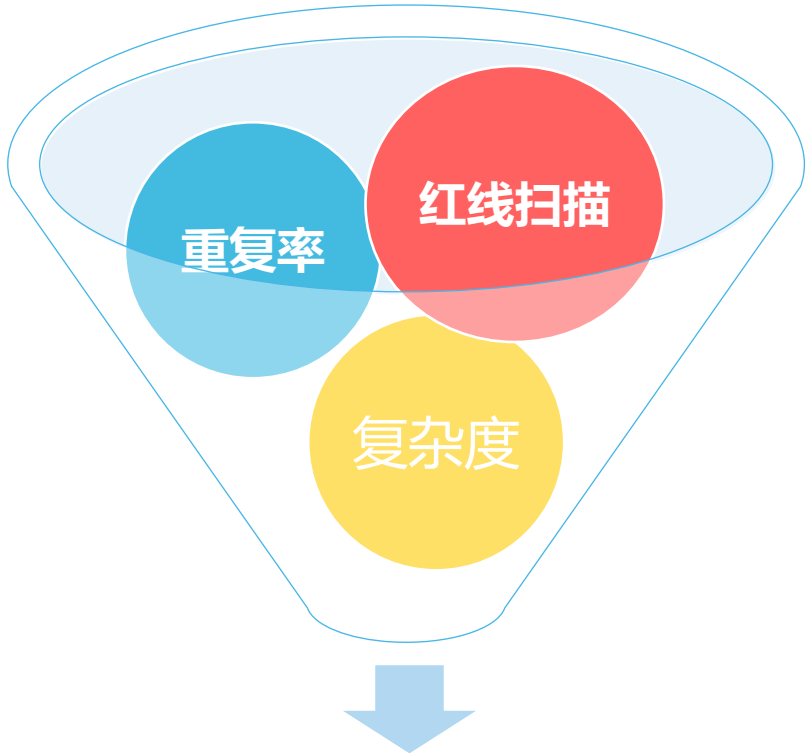
CONTENTS

目录

01 代码质量管理系统介绍

系统概况

系统介绍

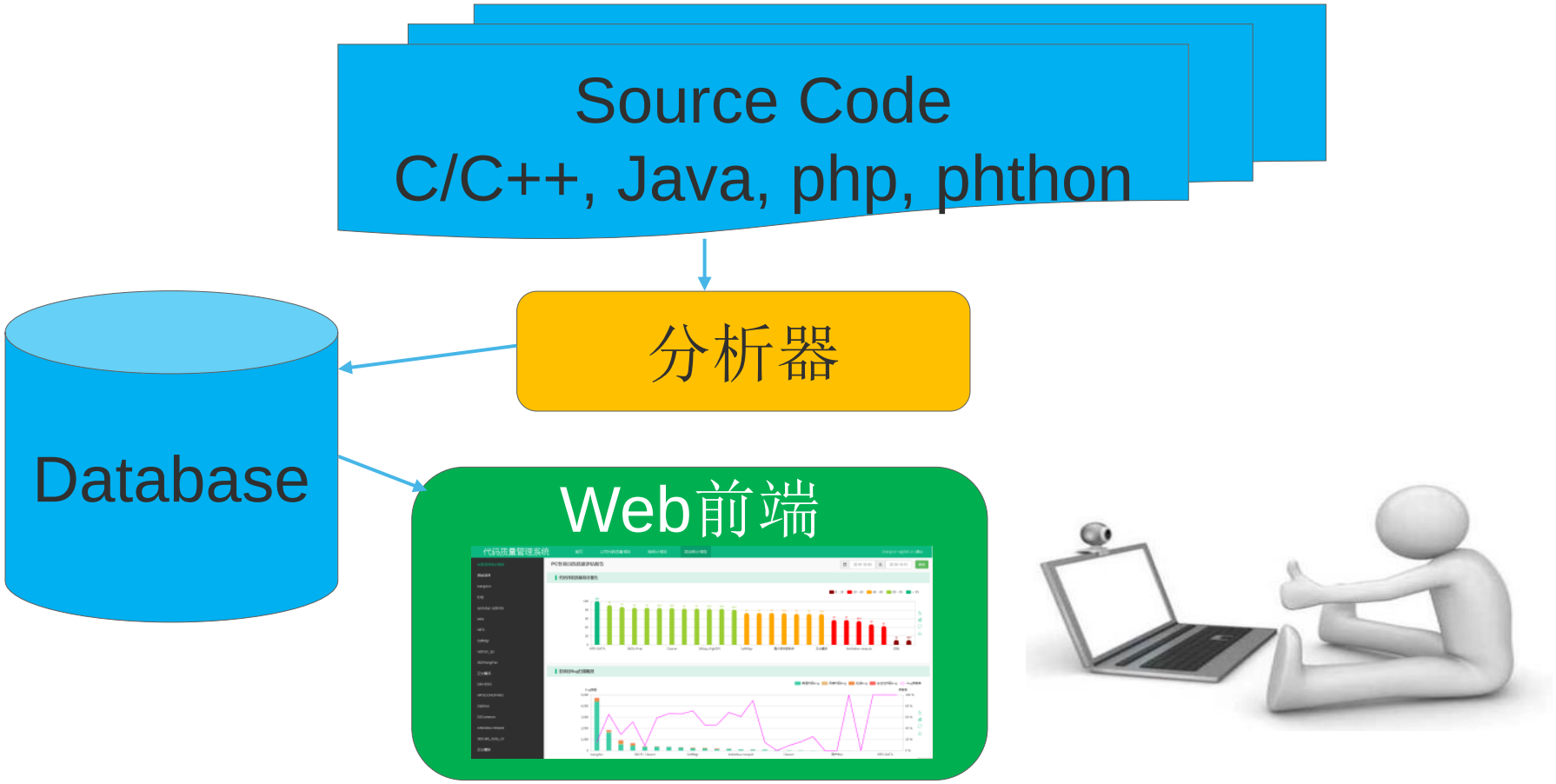


- 红线扫描检查代码错误和漏洞，降低产品风险。
- 代码重复率检查代码冗余度，便于代码维护。
- 文件复杂度检查代码复杂度，提高代码健壮和易测试。

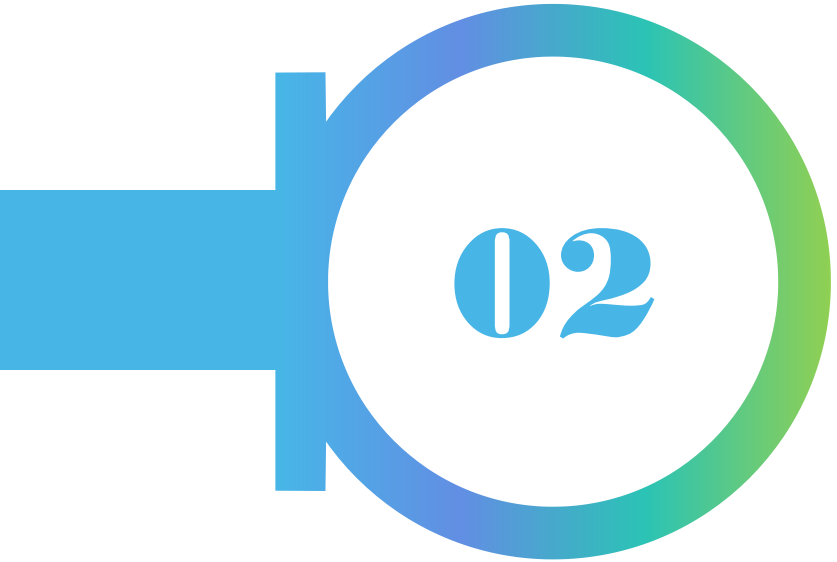
系统介绍 — 流程


有问题则驳回修复
红线扫描
bug数为0
代码质量管理系统
代码质量分析
提交
开始
提测
开发部门
QA
产品代码
编译系统
审计系统
自动化回归测试
新功能手工测试
版本迭代
发布等流程

系统介绍 — 组成



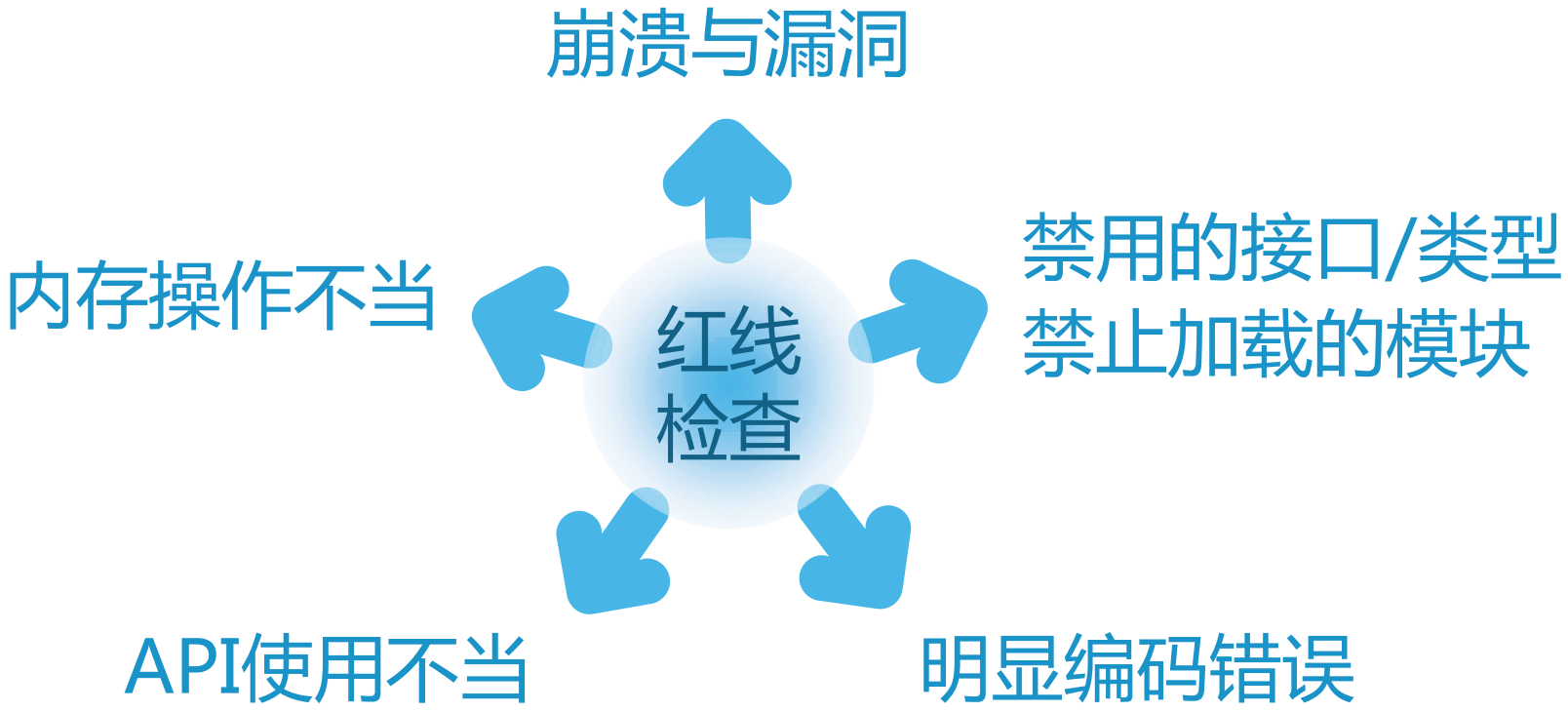
支持语言包括：C/C++、java、python、html/css、 javascript、object-c/c++等



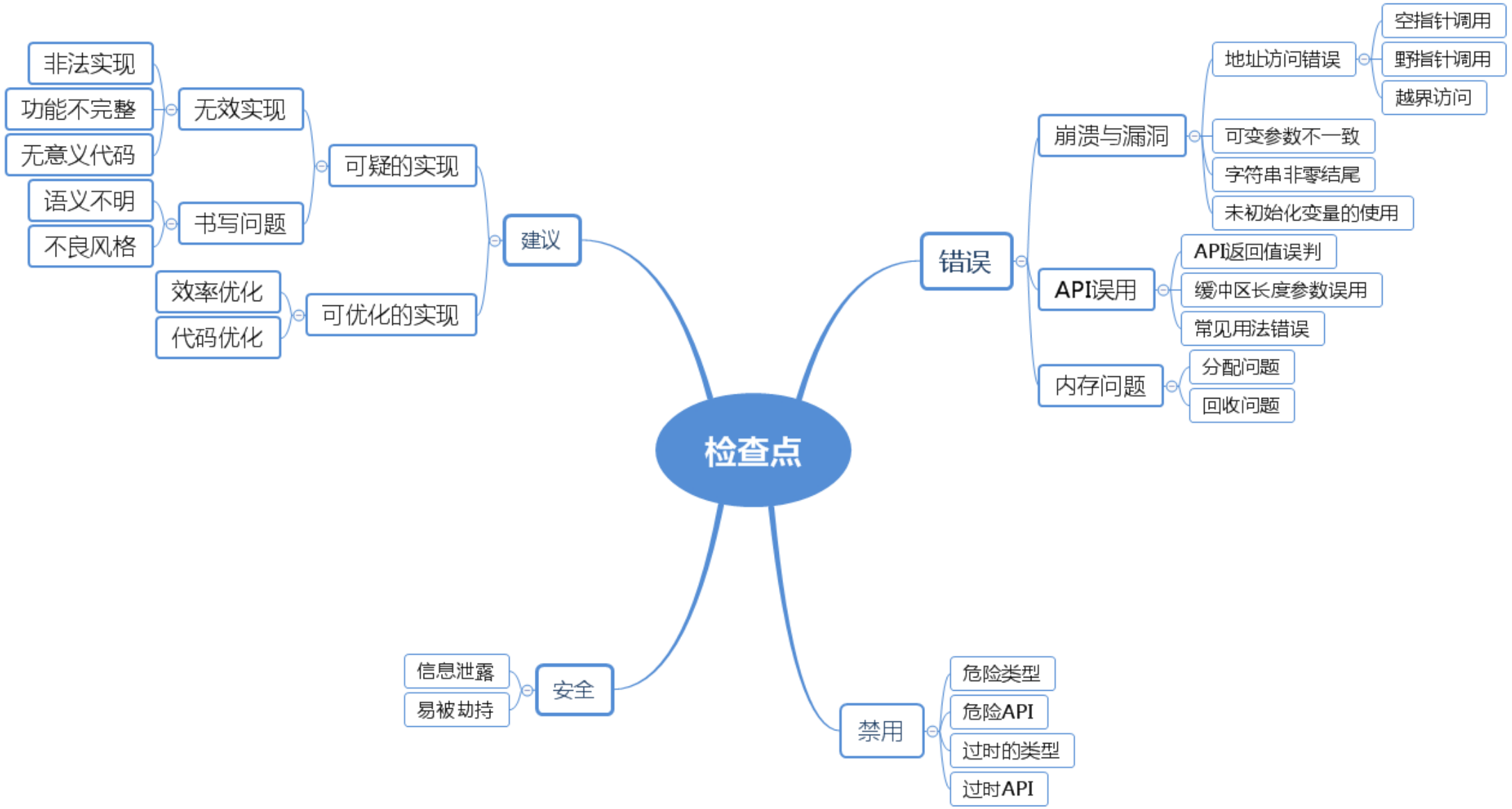
代码规范的制定

红线扫描漏洞及代码质量评分规则

代码规范的制定 — 编码红线



代码规范的制定 — 部分红线检查点



代码规范的制定 — 代码扫描过程

01 预处理

- 头文件展开
- 宏定义展开

02 符号化

- 以关键字、标识符、运算符为“符号”
- 生成符号之间的线性关系序列

03 语法树

- 寻找符号之间的语义关系
- 生成符号之间含有语义信息的树型结构

04 检查输出

- 按规则检查各结构化数据
- 生成检查报告

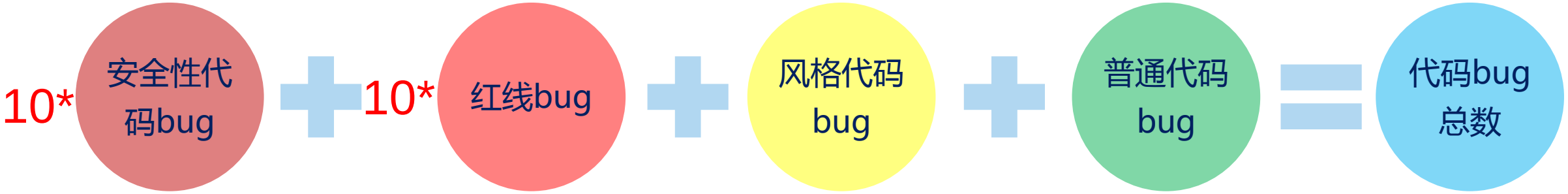
代码规范的制定 — 问题代码举例

```
1  #include <windows.h>
2
3  void fun(LPCTSTR lpctFilePath)
4  {
5      HANDLE hFile = CreateFile(lpctFilePath,
6                                GENERIC_READ, FILE_SHARE_READ,
7                                NULL, OPEN_EXISTING, NULL, NULL);
8      if (hFile)
9      {
10         DWORD dwRead;
11         DWORD dwFileSize = GetFileSize(hFile, NULL);
12         UCHAR *pBuffer = (UCHAR*)malloc(dwFileSize);
13         ReadFile(hFile, pBuffer, dwFileSize, &dwRead, NULL);
14         DoSomeThingInBuffer(pBuffer, dwRead);
15     }
16 }
```

代码规范的制定 — 修复后的代码

```
1  #include <windows.h>
2
3  void fun(LPCTSTR lpctFilePath){
4      HANDLE hFile = CreateFile(lpctFilePath,
5                                GENERIC_READ, FILE_SHARE_READ,
6                                NULL, OPEN_EXISTING, NULL, NULL);
7      if (hFile != INVALID_HANDLE_VALUE){
8          LARGE_INTEGER fileSize = {};
9          if (GetFileSizeEx(hFile, &fileSize) && !fileSize.HighPart){
10             DWORD dwRead = 0;
11             BYTE *pBuffer = (BYTE*)malloc(fileSize.LowPart);
12             if (pBuffer && ReadFile(hFile, pBuffer, fileSize.LowPart, &dwRead, NULL)){
13                 return DoSomeThingInBuffer(pBuffer, dwRead);
14             }
15         }
16     }
17     /* 非预期情况处理.....*/
18 }
```

代码规范的制定 — 红线扫描



公司千行代码bug数标准是低于**15**个

千行代码bug数=代码bug总数/新增代码行数*1000

代码规范的制定 — 重复率

公司代码重复率标准是低于**10%**

代码重复率= 文件重复行数/文件代码总行数

文件块重复率详情

Show 100 entries

有效重复行	文件路径	开始行号	结束行号
227	ideo/utils/IntentParser.java	12	480
	utils/IntentParser.java	12	483
201	ount/utils/IntentParserTool.java	276	872
	/ideo/utils/IntentParser.java	246	842
146	ebView.java	86	347
	iew.java	78	339

代码规范的制定 — 复杂度

公司代码复杂度标准是低于**15**

代码复杂度 = 函数复杂度总和/函数个数

```
  3  void HandleTest::Example(st_msgdata&data)
  4  {
  5      switch (data._calltype){
  6      case     CALLBACK_CLEANEND:{
312      break;
313      case CALLBACK_INITENGINE:{
316      break;
317      case     CALLBACK_SCANBENGIN:{
320      break;
321      case     CALLBACK_SCANEND:{
488      break;
489      case     CALLBACK_SCANCATEBENGIN:{
497      break;
498      case     CALLBACK_SCANCATEEND:{
514      break;
515      case     CALLBACK_ADDITEM:{
555      break;
556      case     CALLBACK_UPDATEITEM:
557      case     CALLBACK_UPDATEITEM_SINGLE:{
569      break;
570      case CALLBACK_UPDATITLE:{
583      break;
584      case     CALLBACK_PROGRESS:{
587      break;
588      case     CALLBACK_POPMSG:{
591      break;
592      case     CALLBACK_CLEANBEGIN:{
595      break;
596      case CALLBACK_TIP_CLOSEPROCESS:{
615      break;
616      case     CALLBACK_CLEANCATEBEGIN:{
630      break;
631      case     CALLBACK_CLEANCATEEND:{
654      break;
655      case CALLBACK_HANDLE_DEALING:{
665      break;
666      case CALLBACK_UPDATEITEM_SELECTALL:{
696      case     CALLBACK_HANDLEEND:{
753      break;
754      case     CALLBACK_HANDLECATEEND:{
765      break;
766      default:
767          break;
768      }
769  }
```

代码质量评分

代码质量分数根据代码质量的3个标准加上权重进行计算 。

代码质量分数(M)	0.8A+0.1B+0.1C		
各项标准分数	千行代码bug数（A）	代码重复率（B）	代码复杂度（C）
权重	0.8	0.1	0.1

03 本系统如何提高代码质量

优化与提高

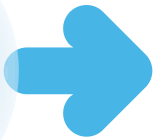
红线扫描

优化方向

- 自定义规则实时更新
- 深入具体业务场景
- 督促开发执行红线规范
- 强制修复红线bug

重复率

优化方向



- 优先使用公共库代码
- 删除冗余的代码和文件
- 提取公共代码

复杂度

优化方向

- 优化逻辑，精简分支
- 适当拆分，明确功能
- 不易过长，不超200

代码质量报告展示

结果反馈

代码质量报告-总体概括

任务xxxxxxx的扫描详情

总体概括 | 红线扫描 | 重复率 | 复杂度

质量概况

代码质量评分：83 分 良好

扫描项	扫描结果	得分
扫出的问题总数	xxxxxxxxx	100 分
新增代码行	xxxxxxxxx	
代码行重复率	33.75 %	1 分
文件平均复杂度	32.45	25 分

优化建议

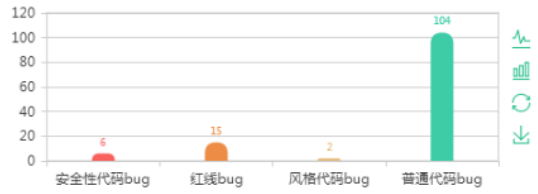
1、安全性bug和红线bug是致命问题，影响发布，必须修复。请督促开发修复本次的 21 个严重问题。

2、重复率 >= 20% 的文件具有极高风险。请督促开发优化本次的 750 个文件，**提取公共代码、删除重复文件**，避免代码过度冗余。

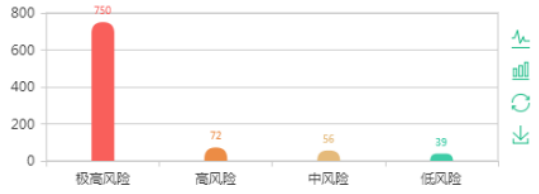
3、复杂度 >= 90 的文件具有极高风险，代码测试困难。请督促开发优化本次的 116 个文件，**减少文件函数数量、拆分函数、避免过多的分支语句**，提高代码可读性，降低测试难度。

各扫描项概况

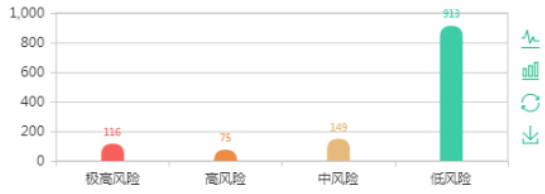
红线扫描bug数量



代码重复率



文件复杂度



任务概况

提测人	xxxxxxxxx@360.cn	
扫描路径	https://host.test/svn/test/test	xxxxxxxxx

代码质量报告-红线扫描

任务xxxxxxx的扫描详情

总体概括 | 红线扫描 | 重复率 | 复杂度

红线扫描概况(原始报告)

得分: 100 分 优秀		新增代码行数: xxxxx	本次扫描出问题数: xxxxx	本次修复的问题数: 0
安全性bug数: xx	红线bug数: xx	风格bug数: xx	普通bug数: xx	未修复的bug数: xxx

本次扫出的问题详情(xx 个)

文件路径	所在行	问题级别	问题类型	Bug归属人	修复状态	代码路径	Bug版本	操作
/projectname/test.cpp	137	红线问题	improperNullTermination	xxxxxx@360.cn	未修复	https://host/test	xxxxx	加白
/projectname/test1.cpp	293	红线问题	potentialNullPointer	xxxxxx@360.cn	未修复	https://host/test	xxxxx	加白
/projectname/test2.cpp	292	红线问题	potentialNullPointer	xxxxxx@360.cn	未修复	https://host/test	xxxx	加白
/projectname/test3.cpp	263	红线问题	potentialNullPointer	xxxxxx@360.cn	未修复	https://host/test	xxxx	加白
/projectname/test4.cpp	264	红线问题	potentialNullPointer	xxxxxx@360.cn	未修复	https://host/test	xxxx	加白
/projectname/test5.cpp	27	红线问题	potentialNullPointer2	xxxxxx@360.cn	未修复	https://host/test	xxxx	加白
/projectname/test6.cpp	609	红线问题	improperNullTermination	xxxxxx@360.cn	未修复	https://host/test	xxxx	加白
/projectname/test7.cpp	419	红线问题	improperNullTermination	xxxxxx@360.cn	未修复	https://host/test	xxxx	加白
/projectname/test8.cpp	261	红线问题	improperNullTermination	xxxxxx@360.cn	未修复	https://host/test	xxxx	加白
/projectname/test9.cpp	263	红线问题	potentialNullPointer2	xxxxxx@360.cn	未修复	https://host/test	xxxx	加白
/projectname/test9.cpp	650	红线问题	wrongLengthArgument	xxxxxx@360.cn	未修复	https://host/test	xxxx	加白

代码质量报告-重复率

任务xxxxxxx的扫描详情

总体概括 | 红线扫描 | 重复率 | 复杂度

重复率总体概况

得分: 1 分 极差		代码行重复率: 33.75%	文件重复率: 34.75%
重复行总数: 203143	重复块总数: 2129	重复文件总数: 917	重复阈值: 10
扫描行总数: xxxxxx	有效行总数: xxxxxxx	扫描文件总数: xxxxxxx	报告时间: 2018-10-31 14:59:10

文件行重复率详情

Show 10 entries

文件路径	重复率(%)	重复行数	文件行数	重复行信息	SVN路径	SVN版本
/peojectname/value.h	100	1069	1069	第 1 行 -- 1069 行	codepath	xxxxxx
Public /publicname/value.h	100	1069	1069	第 1 行 -- 1069 行	publiccodepath	xxxxxxxxx
/peojectname/test.h	100	603	603	第 1 行 -- 603 行	codepath	xxxxx
Public /publicname/test.h	100	603	603	第 1 行 -- 603 行	codepath	xxxxx

Showing 1 to 10 of 917 entries

Previous 1 2 3 4 5 ... 92 Next

文件块重复率详情

代码质量报告-复杂度

任务xxxxxxx的扫描详情

总体概括 | 红线扫描 | 重复率 | 复杂度

复杂度总体概况

得分: 25 分 极差		文件平均复杂度: 32.45	函数平均复杂度:3.71
代码总复杂度: 40660	文件总数: 1253	函数总数: 10966	文件平均函数量: 8.75

文件复杂度详情

Show 10 entries

文件路径	文件复杂度	有效代码数	函数数量	SVN路径	SVN版本
test.cpp	948	3570	136	xxxxxxxxxxxxxxxxxxxx	xxxxxxx
test1.cpp	704	2778	81	xxxxxxxxxxxxxxxxxxxx	xxxxxxx
test2.cpp	657	3441	108	xxxxxxxxxxxxxxxxxxxx	xxxxxxx
test3.cpp	651	3038	110	xxxxxxxxxxxxxxxxxxxx	xxxxxxx
test4.cpp	650	2892	73	xxxxxxxxxxxxxxxxxxxx	xxxxxxx
test5.cpp	631	2779	95	xxxxxxxxxxxxxxxxxxxx	xxxxxxx
test6.cpp	594	2404	108	xxxxxxxxxxxxxxxxxxxx	xxxxxxx
test7.cpp	509	1546	293	xxxxxxxxxxxxxxxxxxxx	xxxxxxx
test8.cpp	465	1581	75	xxxxxxxxxxxxxxxxxxxx	xxxxxxx
test9.cpp	454	1866	68	xxxxxxxxxxxxxxxxxxxx	xxxxxxx

Q

A