

性能测试设计与执行

软件质量保障与测试课程 Lab7 课程作业（第 9 组）

Tian, Jiahe^{*} Hu, Xiaoxiao[†] Huang, Jiani[‡] Liu, Jiaxing[§]
Shi, Ruixin[¶] Wu, Chenning^{||} Zhang, Cenyuan^{**}
Zhang, Yihan^{††} Wang, Chen^{‡‡}

2020 年 5 月 14 日

^{*}Equal Contribution, Fudan University, 17307130313 (tianjh17@fudan.edu.cn)

[†]Equal Contribution, Fudan University, 17302010077 (xxhu17@fudan.edu.cn)

[‡]Equal Contribution, Fudan University, 17302010063 (huangjn17@fudan.edu.cn)

[§]Equal Contribution, Fudan University, 17302010049 (jiaxingliu17@fudan.edu.cn)

[¶]Equal Contribution, Fudan University, 17302010065 (rxshi17@fudan.edu.cn)

^{||}Equal Contribution, Fudan University, 17302010066 (cnuwu17@fudan.edu.cn)

^{**}Equal Contribution, Fudan University, 17302010068 (cenyuanzhang17@fudan.edu.cn)

^{††}Equal Contribution, Fudan University, 17302010076 (zhangyihan17@fudan.edu.cn)

^{‡‡}Equal Contribution, Fudan University, 16307110064 (wangc16@fudan.edu.cn)

性能测试设计与执行

软件质量保障与测试课程 *Lab7* 课程作业

摘要

本次作业为软件质量保障与测试课程的 Lab7 课程作业，需要我们以小组为单位完成对出题系统的静态检查。本文档分为五小节。第一小节介绍了本小组进行 Sonar 检查的结果情况；第二小节介绍了本小组利用 p3c 工具进行检查的检查结果情况；第三小节介绍了本小组利用 Jslint 工具进行静态检查的检查结果情况；第四小节介绍了本小组对上述不同检查工具之间的异同点、优缺点分析的情况；第五小节介绍了本小组对本次静态检查的测试总结。

关键词

系统与软件工程；系统与软件质量要求和评价；测试文档

目录

摘要	2
关键词	2
1 Sonar 工具静态测试	4
1.1 测试结果完整报告	4
1.2 测试结果核心内容截图	4
1.2.1 前端检测报告	4
1.2.2 后端检测报告	7
1.3 测试结果分析	7
1.3.1 sonar 测试报告特点	7
1.3.2 扫描效果	10
2 p3c 工具静态测试	10
2.1 测试结果完整报告	10
2.2 测试结果核心内容截图	10
2.3 测试结果分析	10
2.3.1 阿里巴巴 JAVA 开发手册	10
2.3.2 扫描效果	12
2.3.3 插件使用	12
3 jshint 工具静态测试	13
3.1 测试结果完整报告	13
3.2 测试结果核心内容截图	13
3.3 测试结果分析	13
3.3.1 配置项	13
3.3.2 扫描效果	15
3.3.3 工具使用	15
4 不同工具之间的对比分析	15
5 测试总结	16
参考文献	17



图 1: 前端-总览

1 Sonar 工具静态测试

1.1 测试结果完整报告

“出题系统”前端 (Javascript) 检测报告见附件 2020-05-24-test-maker-fore-report

“出题系统”后端 (Java) 检测报告见附件 2020-05-24-my_project-report

1.2 测试结果核心内容截图

sonar 检测报告网页版视图中，分别以总览、问题、安全热点、指标、代码来记录对代码的分析结果，下面将分别截图这些部分。

sonar 检测报告的文档版的内容与网页版一致，具体报告在附件中可以查看。

1.2.1 前端检测报告

sonar 前端检测报告网页版截图见图 1-5，分别为总览、问题、安全热点、指标、代码的网页显示。

The screenshot shows the SonarQube interface for the 'test-for' project. The left sidebar contains a '过滤器' section with various filters applied. The main area displays a list of code smells found in different files. Each item in the list includes a summary, severity, type, status, and a link to the code.

文件	问题描述	严重程度	状态
assets/s/admin/admin-login.js	1 duplicated blocks of code must be removed. 为何是问题?	次要	未分配
	Remove this commented out code. 为何是问题?	次要	未分配
	Refactor this function to use "return" consistently. 为何是问题?	次要	未分配
	Unexpected string concatenation. 为何是问题?	次要	未分配
assets/s/admin/page/page-generate.js	1 duplicated blocks of code must be removed. 为何是问题?	次要	未分配
	Remove the declaration of the unused 'selected' variable. 为何是问题?	次要	未分配
	Unexpected string concatenation. 为何是问题?	次要	未分配
	Remove this useless assignment to variable "qlanguages". 为何是问题?	次要	未分配
	Remove unused function '_postData'. 为何是问题?	次要	未分配
assets/s/admin/page/page-preview.js	Jump statements should not be used in this context. 为何是问题?	次要	未分配
	String literals should not be concatenated. 为何是问题?	次要	未分配

图 2: 前端-问题

The screenshot shows the SonarQube interface for the 'test-for' project, focusing on security hotspots. The left sidebar lists various security categories. The main area shows a specific hotspot in the 'users.js' file, detailing the issue, code snippet, risk level, and remediation steps.

Review this potentially hardcoded credential.

Authentication

状态: 需要复审

什么是风险? 是否处于风险中? 如何修复?

Code Snippet:

```

39     createUserForm.find('input[type="checkbox"]').bootstrapSwitch();
40
41     createUserForm.validate({
42       rules: {
43         password: 'required',
44         passwordConfirm: {
45           equalTo: "#user-password"
46         }
47       }
48     });
49   });

```

In the past, it has led to the following vulnerabilities:

- CVE-2019-13466

图 3: 前端-安全热点

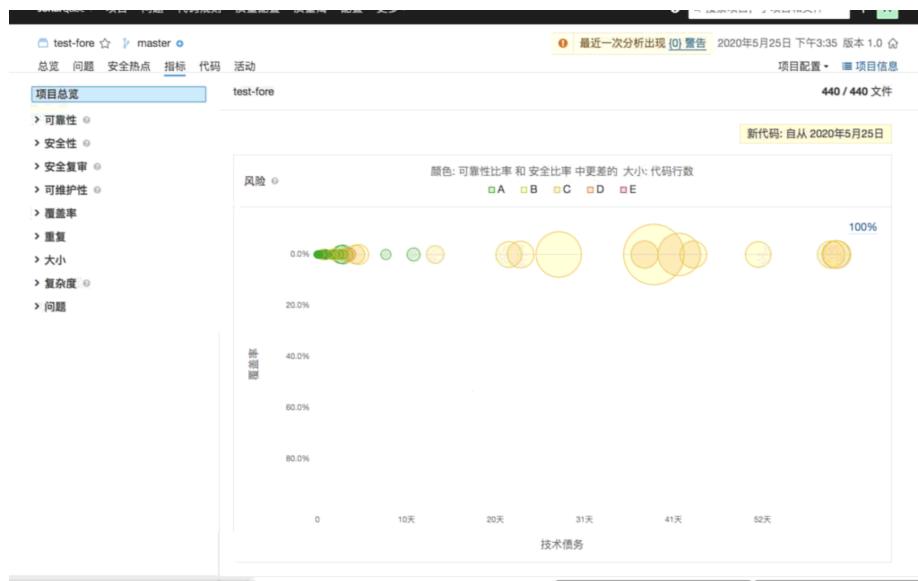


图 4: 前端-指标



图 5: 前端-代码



图 6: 后端-总览

1.2.2 后端检测报告

sonar 前端检测报告网页版截图见图 6-10，分别截取了总览、问题、安全热点、指标、代码的网页显示。

1.3 测试结果分析

Sonar 是一个用于代码质量管理的开源平台，用于管理源代码的质量。

1.3.1 sonar 测试报告特点

sonar 的问题（图 2，图 7）部分分析的粒度为类型、严重程度、处理方式、状态、标准、新问题、语言、规则、标签、目录、文件、负责人、作者。严重程度划分为 blocker, critical, major, minor, info, 对应致命（阻断），关键（严重），主要，微小（次要），未知（提示）。可以便于之后对跟踪缺陷修改和变化的跟进。

sonar 的安全热点（图 3，图 8）中会给出一些 CVE 中已收录漏洞在当前代码中的检测，以避免未来可能造成的安全问题。

sonar 的指标部分（图 4，图 9）给出对于代码的可靠性，安全性，可维护性，覆盖率，重复，复杂度等特征的评级，用于帮助优化代码以及判断修复各种缺陷可能需要的开销。

类型	数量	优先级
Bug	29	次要
漏洞	11	次要
异味	520	主要

最近一次分析出现 10 警告 2020年5月24日 下午5:08 版本 1.0 项目配置 项目信息

批量修改 选择问题 浏览 1 / 560 问题 7天 0小时 工作

图 7: 后端-问题

最近一次分析出现 10 警告 2020年5月24日 下午5:08 版本 1.0 项目配置 项目信息

搜索项目、子项目和文件 + A

条件 分配给我 全部 状态 需要复审 全部代码 安全热点复审 0.0%

需要复审 11 安全热点

复审优先级: 高

Authentication 1

'password' detected in this expression, review this potentially hard-coded credential.

状态: 需要复审
安全热点需要进行复审, 以确定代码是否存在风险。

src/.../cstqb/exam/testmaker/services/impl/UserServiceImpl.java

```

243 User firstUser = userDao.first();
244     if (firstUser == null) {
245         Config builtInUser=configContext.getConfig().getConfig("application.built-in");
246         logger.debug("UserServiceImpl.findFirstUser: built-in: {}", builtInUser);
247         firstUser = new User(builtInUser.getString("username"), builtInUser.getString("password"));
248         logger.info("There is no user yet. Creating first user as admin, username={}", firstUser.getEmail(builtInUser.getString("email")));
249         firstUser.setPhone(builtInUser.getString("phone"));
250         firstUser.setAdmin(true);
251         firstUser.setFullName(builtInUser.getString("fullName"));
252         userDao.create(firstUser);
253     }
  
```

什么是风险? 是否处于风险中? 如何修复?

Because it is easy to extract strings from an application source code or binary, credentials should not be hard-coded. This is particularly true for applications that are distributed or that are open-source.

In the past, it has led to the following vulnerabilities:

- CVE-2019-13466
- CVE-2018-15389

Credentials should be stored outside of the code in a configuration file, a database or secret management service.

图 8: 后端-安全热点

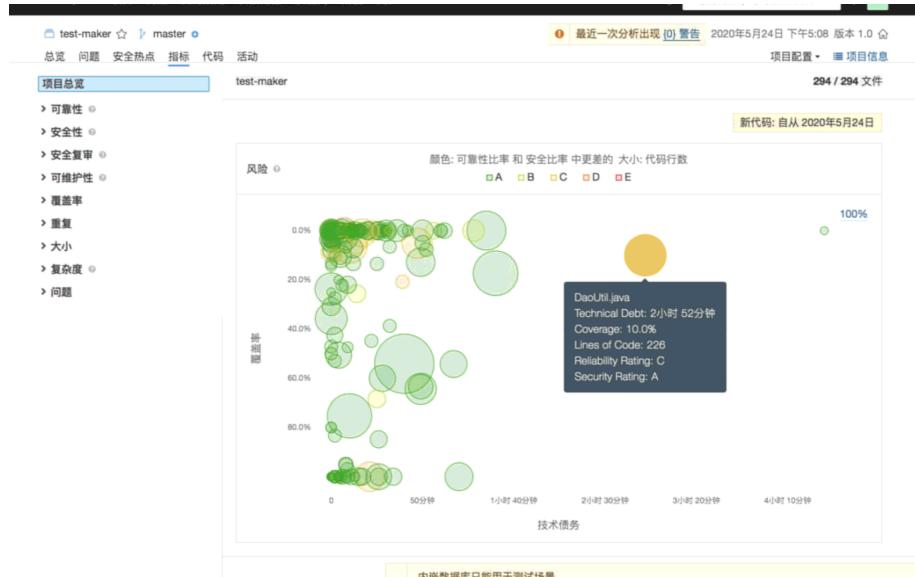


图 9: 后端-指标

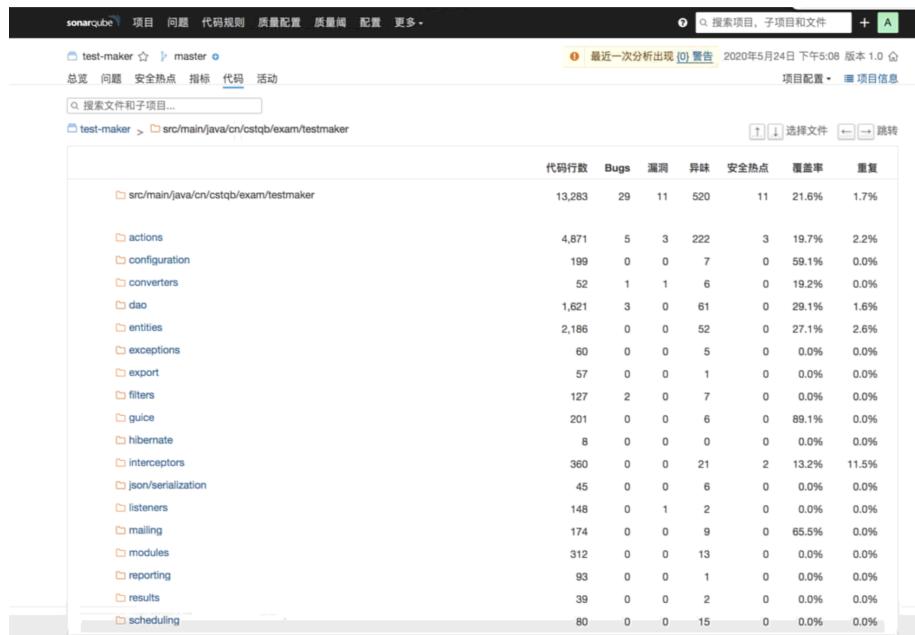


图 10: 后端-代码

sonar 的代码（图 5，图 10）部分向开发者展示代码源文件和扫描出的缺陷，包含了行数、问题数、单元覆盖率、重复度、代码近期提交信息等。用户可以看见每个缺陷的位置，同时还可以看到此类缺陷系统给出的修改提示和为什么这会是个缺陷的详细解释。

1.3.2 扫描效果

选择的是 sonar 默认的扫描规则，效果非常细致，相对来说扫描过程用时也比较久。sonar 将问题分为 bug, code_smell 和 vulnerability，严重程度划分为 minor,major,blocked,critical,info。bug 类型的问题多涉及到比如比较判断符错误使用，空指针重定向，条件分支不可达等，code_smell 类型涉及到的一般是坏的代码习惯造成的缺陷，例如方法返回值不能一直不变，方法体不能为空需要附加注释等，vulnerability 类型则是涉及引用了已知有漏洞的函数的代码。sonar 扫描的问题比较全面，也可以做到持续的代码检查跟进，具有高可用性和较短的反馈循环。提供了多种语言检测支持。整体上，sonar 扫描的比较全面，还可以与版本控制，开发人员关联起来，有比较好的控制反馈功能，可视化结果也比较直观

2 p3c 工具静态测试

2.1 测试结果完整报告

请通过网址[第 9 组 P3C 报告网站](#)来查看本小组的 P3C 完整报告。

2.2 测试结果核心内容截图

如图 11 和 12 所示，可分为 Blocker、Critical、Major 三部分。

2.3 测试结果分析

P3C 是阿里巴巴推出的《阿里巴巴 Java 开发规约》扫描插件，目前在 IDEA 和 Eclipse 都有较好的支持。P3C 扫描结果文档给出了项目的 bug，并根据 bug 的严重程度分为三个级别展示。三个级别分别是：Blocker, Critical, Major，严重程度由高到低。

2.3.1 阿里巴巴 JAVA 开发手册

根据《阿里巴巴 JAVA 开发手册》版本 1.3.0，我们可以对扫描结果进行分析。该版本对 JAVA 开发的规约含有编程规约、异常日志、单元测试、



图 11: p3c 插件静态扫描结果截图 1

IntelliJ IDEA inspection report:

Inspection tree:	Problem description:
<input checked="" type="checkbox"/> 'InspectionViewTree' project 172 blockers 99 criticals 628 majors <ul style="list-style-type: none"> <input type="checkbox"/> Blocker 172 blockers <input checked="" type="checkbox"/> Critical 99 criticals <input type="checkbox"/> Major 628 majors 	Select a problem element in tree

图 12: p3c 插件静态扫描结果截图 2

安全规约等六大类，而本项目的扫描结果重点体现的是编程规约。编程规约在开发手册共分为 9 类：

- 命名规范
- 常量定义
- 代码格式
- OOP 规约
- 集合处理
- 并发处理
- 控制语句
- 注释规约
- 其他

2.3.2 扫描效果

根据扫描结果可以发现：同一类型的编程规约由于其严重性不同可以划分为不同级别的 bug。比如注释规约中，抽象方法的 javadoc 注释属于 Major，而枚举字段的注释属于 Critical；命名规范中也有类似的例子。另外，bug 的严重程度分类比较合理。注释规约、命名规范的约定内容严重程度较低，并发处理、控制语句的约定内容严重程度较高。报告中给出了命名风格错误（类名 UpperCamelCase，方法名、变量名 lowerCamelCase）、缺少注释（javadoc 注释，类创建信息注释等）、魔法值常量、缺少注解（重新方法注解）、类方法行数过多、集合初始化为指定数量、线程使用高风险、控制语句语句块包括等问题。整体上，P3C 扫描插件发现的问题比较基础，它侧重 JAVA 编程细节可能导致的系统失效。

2.3.3 插件使用

P3C 插件有以下的优点：

- 基本满足代码规范检测的需求。
- 能够检测出细致和易忽略的问题，可以提高开发过程中对代码细枝末节的注意。
- Quick fix，检测出问题后可以快速查看 bug 位置、解决方案并一键替换。
- 中文提示，解释内容与开发手册一致。

此外 P3C 插件也存在平台限制、功能不成熟、扫描能力有限等问题。

Code	Line	Column	Evidence	Reason
<code>/webapp/assets/js/ajax-utils.js</code>				
W097	1	1	'use strict';	Use the function form of "use strict".
W033	60	11	});	Missing semicolon.
W033	61	2	}	Missing semicolon.
W117	12	16	<code>var data = _.isNull(param) _.isUndefined(param) ? null : param;</code>	'_' is not defined.
W117	12	35	<code>var data = _.isNull(param) _.isUndefined(param) ? null : param;</code>	'_' is not defined.

图 13: jshint 工具静态扫描结果截图 1

Code	Line	Column	Evidence	Reason
<code>/webapp/assets/js/ajax-utils.js</code>				
W097	1	1	'use strict';	Use the function form of "use strict".
W033	60	11	});	Missing semicolon.
W033	61	2	}	Missing semicolon.
W117	12	16	<code>var data = _.isNull(param) _.isUndefined(param) ? null : param;</code>	'_' is not defined.
W117	12	35	<code>var data = _.isNull(param) _.isUndefined(param) ? null : param;</code>	'_' is not defined.

图 14: jshint 工具静态扫描结果截图 2

3 jshint 工具静态测试

3.1 测试结果完整报告

请通过网址[第 9 组 JSHint 报告网站](#)来查看本小组的 JSHint 完整报告。

3.2 测试结果核心内容截图

如图 13 和 14 所示所示，共发现 89 个 failures，0 个 error 和 89 个 warnings。其中每一项都标明了具体的行数、问题代码和问题原因。

3.3 测试结果分析

JSHint 是由 Anton Kovalyov 基于 JSLint 的代码实现的开源项目，JSHint 与 JSLint 相比较之下，更友好，也更容易配置，所以发展很快并得到了众多 IDE 和编辑器的支持。

JSHint 是一个 JavaScript 语法和风格的检查工具，但检查不出逻辑问题。它可以根据配置参数扫描 JavaScript 代码，分析其中的语法与风格从而给出代码质量报告。

3.3.1 配置项

JSHint 工具使用的关键是配置项。如果不设置配置项，那么可能会有很多“假”错误或警告。比如自定义全局变量，不同脚本上下文的符号引用。



图 15: 项目配置项

这些内容既不算错误的语法，也不算差劲的风格，可是 JSHint 依旧把他们认错了。JSHint 有一些常见的配置项：

- “strict”: true 严格模式
- “asi”: true 允许省略分号
- “bitwise”: true 禁止使用位运算符，比如经常把 `&&` 写错 `&` 规避此错误
- “eqeqeq”: true 禁止使用 `==` 和 `!=` 强制使用 `===` 和 `!==`
- “undef”: true 禁止使用不在全局变量列表中的未定义变量
- “curly”: true 循环或者条件语句必须使用花括号包住
- “devel”: true 定义用于调试的全局变量: `console,alert`
- “jquery”: true 定义全局暴露的 jQuery 库（可以去掉）
- “browser”: true 暴露浏览器属性的全局变量如 `window document`
- “globals”: {“\$”:true,“require”:true,”_”:true}

JSHint 的配置项一般是放入项目根目录下的 `.jshintrc` 文件中，JSHint 工具在扫描的时候就会运用这些配置项。配置项的作用一方面是明确项目的编程规范，约束开发人员的行为。另一方面是避免某些规范，减轻开发人

员的负担（比如允许省略分号等）。

3.3.2 扫描效果

不同的配置项扫描结果是不尽相同的。项目最初未定义配置项中的全局变量，这导致了非常多的未定义错误或警告。这显然不是我们需要的扫描结果。完成配置项后，扫描结果报告逐渐明晰，其中报告了出现了缺少分号、使用 `==` 和 `!=` 错误、缺少 `{` 或 `}` 错误和未定义变量等错误。正如上面所言，JSHint 可以分析出 JavaScript 代码的语法和风格，但是它无法识别出项目的逻辑错误，比如冗余代码、死循环、无效代码等问题。它的分析功能是非常基础和有限的。

3.3.3 工具使用

JSHint 工具不可否认具有一定的优点：

- 有了很多参数可以配置
- 支持配置文件，方便使用
- 支持了一些常用类库（如 jquery 等）
- 支持了基本的 ES6

但与其他功能强大的 Web 项目静态扫描工具比较而言，它具有不支持自定义规则、无法根据错误定位到对应的规则和不提供快捷的修正方式等缺点。不支持自定义规则自然让它只能检测出很基础的预定义规则，无法根据错误定位到对应的规则使得扫描结果不易阅览，不提供快捷的修正方式（不能跳转到指定代码位置和不能提供修正方案）自然无法让开发人员方便的处理扫描结果。

4 不同工具之间的对比分析

- sonar：定位是代码质量平台，本身不进行代码分析，但可以集成各个静态分析工具以及其他软件开发测试工具，并基于集成工具的结果数据按照一定的质量模型，对软件的质量进行评估。甚至可以选择接入 p3c 规范进行代码扫描。sonar 基于扫描规则进行扫描，因此扫描的问题可以比较全面，本身是代码质量平台，可以做到持续的代码检查跟进，具有高可用性和较短的反馈循环。提供了多种语言检测支持。生成的检测报告也比较详细，数据可视化好。是几个工具中专业性最高的。

- p3c 是一套自动化的 IDE 检测插件（主要是 IDEA、Eclipse）该插件在扫描代码后，将不符合《手册》的代码按 Blocker/Critical/Major 三个等级显示在下方，根据错误定位到对应的规则，在 IDE 中提供快捷修正方式，但 p3c 扫描检测的问题较为基础，侧重 JAVA 编程细节可能导致的系统失效，之后仍需要类似 FindBugs 的插件再次扫描检测 bug。
- JSHint 是由 Anton Kovalyov 基于 JSLint 的代码实现的开源项目，是一个 JavaScript 语法和风格检查的命令行工具，不能检查出逻辑问题。它可以根据配置参数扫描 JavaScript 代码，分析其中的语法风格从而给出代码质量报告。相比于 sonar 它不支持自定义规则、相比于 p3c 它无法根据错误定位到对应的规则，也不提供快捷的修正方式，分析功能也非常基础和有限，是并不算强大的 web 静态扫描工具。

5 测试总结

在我们选取的三个测试工具中，sonar 定位于代码质量平台，集成各种静态分析工具和其它测试工具，提供持续代码检测跟进，是最强大和专业的静态测试工具。而 p3c 和 JSHint 分别是针对 Java 和 JS 开发的基于编码及设计实践的静态检测工具，p3c 做成了 IDE 检测插件，JSHint 则是命令行工具，由于二者都偏重于代码编写格式，及是否符合编码规范的检验，内置编程规范比较基础，对代码 bug 发现功能较弱。此次整体感受是静态代码分析工具能够在代码构建过程中帮助开发人员快速、有效的定位代码缺陷并及时纠正代码缺陷，从而极大地提高软件可靠性并节省软件开发和测试成本。

参考文献

International Organization for Standardization. 2014. *Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE*. International Organization for Standardization. Vol. 2014. <https://www.iso.org/standard/64764.html>.

中国国家标准化管理委员会. 2016. GB/T 25000.51-2016《系统与软件工程系统与软件质量要求和评价 (SQuaRE) 第 51 部分：就绪可用软件产品 (RUSP) 的质量要求和测试细则》. 系统与软件工程系统与软件质量要求和评价 (SQuaRE). Vol. 51. 中国国家标准化管理委员会. <http://openstd.samr.gov.cn>.

———. 2017a. GB/T 25000.12-2017《系统与软件工程系统与软件质量要求和评价 (SQuaRE) 第 12 部分：数据质量模型》. 系统与软件工程系统与软件质量要求和评价 (SQuaRE). Vol. 12. 中国国家标准化管理委员会. <http://openstd.samr.gov.cn>.

———. 2017b. GB/T 25000.24-2017《系统与软件工程系统与软件质量要求和评价 (SQuaRE) 第 24 部分：数据质量测量》. 系统与软件工程系统与软件质量要求和评价 (SQuaRE). Vol. 24. 中国国家标准化管理委员会. <http://openstd.samr.gov.cn>.

———. 2018. GB/T 25000.40-201《系统与软件工程系统与软件质量要求和评价 (SQuaRE) 第 40 部分：评价过程》. 系统与软件工程系统与软件质量要求和评价 (SQuaRE). Vol. 40. 中国国家标准化管理委员会. <http://openstd.samr.gov.cn>.

———. 2019. GB/T 25000.23-2019《系统与软件工程系统与软件质量要求和评价 (SQuaRE) 第 23 部分：系统与软件产品质量测量》. 系统与软件工程系统与软件质量要求和评价 (SQuaRE). Vol. 23. 中国国家标准化管理委员会. <http://openstd.samr.gov.cn>.