代码规则检查工具评析

赵 伟
(江苏自动化研究所,江苏 连云港 222061)

摘 要:软件测试是保障软件质量最有效的手段,而代码规则检查能够透过征兆直接看到问题本质,快速找到缺陷,发现30% ~70% 的逻辑设计和编码缺陷,是所有测试手段中最高效的。由于此方法要求测试人员熟悉软件编程,因此对测试人员有较高要求。且读懂对方代码,也是一项耗时、费力的工作,严重影响测试效率。代码规则检查工具实现了对代码是否符合国军标相关标准的规则检查,能按照代码规则自动、快速验证代码与相关标准的复合性,同时检查代码的可读性、代码与设计的一致性、代码逻辑表达的正确性及代码结构的合理性等。通过对几种主流的代码规则检查工具的对比分析,探讨了各个工具的优缺点及选择的诸多因素,对于软件测试人员如何选择合适的该类测试工具,具有实际参考价值。

关键词:软件测试;代码规则检查;MISRA;缺陷
中图分类号:TP301 **文献标识码**:A **文章编号**:1673-629X(2015)07-0193-05
doi:10.3969/j.issn.1673-629X.2015.07.043

Evaluation and Analysis of Code Inspection Tools

ZHAO Wei
(Jiangsu Automation Research Institute, Lianyungang 222061, China)

Abstract: Software testing is the most effective means of software quality assurance. But code rule checking can directly see the essence through signs. Finding the defects quickly and finding 30% -70% logic design and coding defects is the most efficient in all test means. But this method requires the tester to be familiar with software programming, so it has higher requirements. And reading opposite code is a time-consuming and laborious work, it can seriously affect the efficiency of the test. Code checking tool can check if the code conform to the relevant standard and it also can quickly and automatically check if the code follow the relevant standard according to the code rule. At the same time, it can check the consistency of code and design, the readability of the code, the correctness of code logical expression and the rationality of the code structure. Through several mainstream contrastive analysis of code rule checking tools, discuss the advantages and disadvantages of each tool and various factors for the selection. It has actual reference value for software staff how to select suitable testing tool of this kind.

Key words: software testing; code rule checking; MISRA; defect

0 引 言

近年来,随着计算机应用领域的迅速扩大,人们对软件质量提出了新的、更高的要求。早在 60 年代软件危机初期,人们就认识到了软件复杂度高,开发周期长,可靠性差,开发和维护费用大等问题。其中可靠性差就是软件质量问题的集中表现,而软件质量差又是软件维护费用大的主要因素之一。在一些特殊领域中,软件质量往往关系到用户的生命安危。例如在国防领域,这类称为安全性第一的软件具有高质量要求、高复杂度、高开发代价的特征。因此必须通过软件测试确保软件产品的可靠性。软件测试中最直接最有效的错误检测方法莫过于代码规则审查。代码规则检查看到的是问题本身而非征兆,能快速找到缺陷,发现30% ~70% 的逻辑设计和编码缺陷,比动态测试更有效率。作为保障软件质量的重要手段,代码规则检查已成为软件从设计到实装过程中必不可少的一个环节。军标 Z141、DO-178B 标准、GJB5000A 认证都要求强制进行代码规则检查。而 GJB5369 则更进一步规定了 C 语言编程规范。但当前做代码规则检查主要面临以下困境:

(1)代码规则检查需要付出很繁重的劳动:重新理解代码,即使非常专业的测试人员,理解别人写的代

收稿日期:2014-08-24 **修回日期**:2014-11-27 **网络出版时间**:2015-06-23
基金项目:中国人民解放军总装备部"十二五"装备预先研究项目(51319080202)
作者简介:赵 伟(1983-),男,工程师,研究方向为软件测试和软件工程化。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20150623.1028.019.html

码也是一项很繁琐的工作,并且容易出错。

(2)时间和资源的限制:由于当前全球及周边环境不稳定,国家非常重视国防建设。对装备质量及研制周期都提出了很高要求,往往要求在有限的时间内,利用有限的经费,来做高可靠性的软件测试。

因此,在软件测试中使用代码规则检查工具来提高测试效率与测试质量是非常有必要的。目前市场上主流的代码规则检查工具主要有 QAFramework(PR 公司)、CodeWizard(Parasoft 公司)、PcLint(GIMPEL SOFTWARE 公司)、C++Test(Parasoft 公司),文中对上述代码规则检查工具进行评析。

1 代码规则及代码规则检查的意义

1.1 代码规则的含义

代码规则也称为编码规则,是程序开发人员编码的行为规范,主要包括排版、注释、命名、可读性、变量、函数和过程、可测性、程序效率,具体如下:

(1)排版:规定源程序中各种语句的排版布局,包括层级缩进、语句换行等;

(2)注释:规定源程序中注释语句内容、位置、数量等;

(3)命名:规定源程序命名规则;

(4)可读性:规定避免使用难以理解或含二意性的文字;

(5)变量:规定变量的使用规则。如禁止使用同名变量、禁止变量未初始化就使用、禁止变量定义未使用、禁止变量不合理赋值等;

(6)函数、过程:规定函数规模、功能及使用规则。如函数规模尽量小于等于 200 行、尽量避免设计多参函数、尽量减少函数的递归调用等;

(7)可测性:规定编码之前应设计好程序调试与测试的方法和手段。如打印函数;

(8)程序效率:规定编码时尽量避免使用降低代码效率的代码。如多重循环中将最忙循环放在最内层、尽量减少循环嵌套次数、尽量将判断语句放置循环体之外等。

软件测试要求在同一个开发团队中使用相同的代码规则,可以形成这个开发团队统一的开发风格,产品个性;同时,遵循一定的代码规则,可以提高模块的可移植性和可维护性。代码规则检查是提高代码质量最有效、最直接的手段。

1.2 代码规则的相关标准

目前代码规则相关标准有很多,比如 MISRA1998、MISRA2004、MISRA C++ 2008、DERA C、EADS C/C++、GJB5369 等。在民品行业中应用最广泛的是 MISRA 2004,即《运输器械软件 C 语言使用规则》,而在国内军品行业中则主要遵循 GJB5369,即《航天型号 C 语言安全子集》。下面着重介绍 MISRA 组织、MISRA2004 标准和 GJB5369。

MISRA:全称 The Motor Industry Software Reliability Association,中文名-汽车工业软件可靠性联会,1994 年成立,总部坐落于英国。福特汽车、利兹大学和福特 VISTEON 汽车系统公司、MIRA 公司、TRW 汽车电子、AB 汽车电子、宾利汽车、捷豹汽车、路虎公司、Lotus 公司、罗孚汽车、Ricardo 公司都是其成员。它最根本的任务是服务并协助汽车工业,为客户提供高效、安全、值得信赖的嵌入式软件。MISRA 历经 4 年筹划,1998 年发布了《汽车专用软件的 C 语言编程指南》,这是一个为汽车工业量身定做的 C 语言编程规范,全文共有 127 条规则。英文名:Guidelines for the Use of the C Language in Vehicle Based Software,称为 MISRAC:1998。随着 MISRAC 编程规范深入人心,MISRAC:1998 在汽车工业中逐渐成长为最权威的 C 语言安全性规范[1]。2004 年,该规范进行了更新。新版本将标准应用对象扩大到所有具有高质量需求的软件系统。MISRA 规则现已被航空、通信、医疗器械等行业采用,作为确保代码完整性的基础。

MISRA2004:MISRA2004 是 MISRA 最出名的成果,MISRA2004 删除了 15 条旧规则,删除后的规则共包括 121 条强制规则,20 条推荐规则,共计 141 条,分 21 类[2]。具体见表 1。

表 1 规则类型统计表

分类	强制规则	推荐规则
常量	1	0
声明和定义	12	0
注释	5	1
开发环境	4	1
运行失败	1	0
类型	4	1
语言外延	3	1
Switch 语句	5	0
结构体和联合体	4	0
算术类型转换	6	0
指针和数组	5	1
表达式	9	4
指针类型转换	3	2
控制流	10	0
初始化	3	0
函数	9	1
标准库	12	0
标识符	4	3
预处理命令	13	4
字符集	2	0
控制表达式	6	1

所有对应 MISRAC:2004 而编制的软件都必须完全遵守这 121 条编码规范,而且尽可能遵守准则中推荐的 20 条规范。一般情况下,若严格按照该规范编码,那么这个 C 语言程序就是可靠性强、容易阅读、方便维护、易于重用。最近越来越多的软件开发者都以这个规范来约束编程风格,例如鼎鼎大名的 uC/OS-II 就号称自己接近 100% 地遵循此标准。

GJB5369:是 2005 年由航天科工集团提出,由航天科工集团二院 706 所起草的《航天型号软件 C 语言安全子集》,该标准是参照 MISRA 1998 和 LDRA(Liverpool Data Research Associates)2000 年的《MISRA C Checking》,并结合航天型号软件的特点,经过裁剪和补充而形成的[3]。规定了 C 语言的编程准则,与 MISRA 非常相似,GJB5369 将准则分为两种,即推荐类型和强制类型。推荐类型不必强制执行[4],仅作为参照,而强制类型则必须执行。是目前国内军工行业代码规则审查的主要参照依据。

1.3　为什么要进行代码规则审查

相对于其他看得见摸得着的产品,软件产品并不能仅仅从功能上评价它的好坏,并不是说完成功能就是一个好的软件。对于软件来说,衡量它好坏的指标很多,比如是否容易阅读、是否容易维护、是否容易移植、是否安全可靠。其中安全可靠是非常重要的一条指标,特别针对高安全性需求的系统,例如飞机、火车、武器装备生产中。一个小小的差错就可能给国家、人民带来巨大的灾难。作为健壮的装备不仅要有安全稳固的硬件架构(主要体现在环境适应性、电磁兼容性),更要有一个成熟的大脑,即软件程序[5]。

然而很少有程序员能全面地考虑软件安全问题。很多软件产品只是仅仅局限于功能实现,表面上能干活,但存在大量的安全隐患。由于 C/C++是易于入门但难以精通的语言、是一门复杂多变的语言,它的语法繁多,而且编程方式灵活,这对于一个初学者可谓是陷阱密布。并且,C/C++语言并未全部定义完毕,即使世界权威的 C/C++语言标准,也有不少定义未覆盖的情况。想使全部开发人员都精通 C/C++语言是脱离实际的。最佳途径就是制定编程安全性准则,让开发人员有据可循。

C 语言开发中,5 种情况最可能造成软件风险:编程者犯错、编程者对语言错误认识、编程者对编译器错误认识、编译器出错和运行时错误。

编程者犯错最为常见。很多程序员犯下的错误可以被编译器及时地纠正(如变量未定义等),但是也存在编译器无能为力的时候。似乎很多编程者都有过将判等号“==”误写成赋值号“=”的经历。编译器并不会发现“if($a = b$)”是程序员的失误而顺利通过编译,造成程序 BUG[6]。

C/C++灵活多变,它赋予编程者很大的自由发挥空间。但凡事都具有双面性,越自由也就越容易犯错。假若有的 BUG 是编程人员无意之中犯下的错误,那么由于编程者对 C 语言或编译器理解错误产生的 BUG 就是“明知故犯”了。C/C++的理论概念中有一部分比较难,编程人员容易产生错误认识,表达式计算就是其一。除此之外,两种编译器针对相同程序可能有不同的处理方式。例如不同编译器规定长整型变量的长度也不尽相同。这就对编程者有了更高要求,编程者必须在了解 C/C++本身特性的基础上,对编译器有所了解,有不小难度。另外,有的错误是编译器本身造成的,这些错误往往难以发现,有可能会留存到程序中。

对于编程者来讲,编程规范提供给他们一些编程思路,让他们渐渐培养好的编程习惯,逐渐将那些不健康的编程习惯丢弃,编写更加易读、易用、可靠性高的代码。现今,大多数软件开发者都不重视软件质量度量,在编写软件时不考虑扇入/扇出、圈复杂度、注释行数等软件质量度量指标,这种程度无论在可读性和运行效率方面都不尽人意,为将来软件的后期维护和重用带来隐患。

因此,鉴于编程规范的重要性,无论是国内还是国外相关组织、行业或企业都制定了相应的软件编码规则。这些规则限制了编程随意性,加强了代码的安全性,大大提高了软件代码的质量。

2　代码规则检查工具

代码规则检查工具实现了对代码是否符合国军标相关标准的规则检查,能按照代码规则自动、快速判断与相关标准的复合性,还能检测代码是否与设计相符及代码的可读性、代码逻辑表达的正确性以及代码结构是否合理。具体作用如下:

(1)代码自动检查,辅助各企事业单位普及编程标准;

(2)软件开发实时自查,在开发阶段发现代码问题;

(3)代码级评测,作为代码评审依据。

2.1　QAFramework

QAFramework 是英国 PR 公司综合应用多种近年来最先进的静态分析技术开发的一款代码规则检查工具。该工具支持 GJB5369 语言编程规范的工具、完全支持嵌入式 C 语言编程规范 MISRA 2004 的工具。采用领先的分析器技术(EDG),对 C/C++代码进行静态分析,软件内嵌 C 语言规则和 C++规则各一千余条,共计两千余条[7],涵盖了 C 语言使用的各个方面,并且支持规则定制,能够完全实现对各种 C 编程规范的

支持,支持最新的 C++标准:MISRA C++ 2008。

QAFramework 对软件中与规则不符的部分予以报告,涵盖代码危险用法,存在的安全隐患,未定义就使用。同时提供基于函数、基于文件和类的上百种软件静态度量[8]。还能够给出软件的结构分析。它不只是代码规则检查工具,还提供代码缺陷管理与趋势分析功能。通过对不同版本代码分析结果的记录,跟踪代码违反规则数的变化趋势,给出多个版本之间的对比。

2.2 CodeWizard

CodeWizard 是美国 Parasoft 公司推出的一款代码规则检查工具。能够对源程序直接进行自动扫描、分析和检查,一旦发现违例,产生信息告知与哪条规则不符并做出解释[9]。CodeWizard 内置超过 500 条以上行业相关的编码准则,自动识别编译器未检测到的危险代码构造。CodeWizard 可以选择对当前工程执行哪些编码标准,并能容易地通过 RuleWizard 功能创建新定制的准则,或者抑制用于定制分析的准则,能与 VC++紧密集成,安装完毕后 C++中有 CodeWizard 工具条,操作便捷。

2.3 PcLint

PcLint 是 GIMPEL SOFTWARE 公司推出的一款代码规则检查工具。PcLint 可以检查编译器不易发现的错误,能够对 100 多个 C 库函数进行检查,可以发现标准 C/C++代码中的 1 000 多个常见错误[10]。不但能对程式进行全局分析、识别没有被适当检验的数组下标、报告未被初始化的变量、警告使用空指针连同冗余代码,还能够有效地提出程序在空间利用、运行效率上的改进点。通过配置可将 PcLint 与 Visual Studio 集成在一起。

2.4 C++Test

C++Test 与 CodeWizard 出自同一家公司(美国 Parasoft 公司)。C++Test 是一个 C/C++单元级测试工具,内置 800 余条 C/C++混合在一起的编码规则,可进行自动化编程规范检查[11]。可使用图形化的 RuleWizard 编辑器自定义规则,将 API 使用标准化并预防单个错误发现后类似错误重复出现。虽然 C 和 C++语法有些相似,但两者还是有本质的区别,两者的编程标准也不尽相同,C++Test 将两种语言使用同一套标准考核,无法做到专业、准确。

2.5 工具对比

在多个测试项目中反复实践,对 QAFramework、CodeWizard、PcLint、C++Test 的分析结果比较如表 2 所示[12-14]。

从上述分析不难得出,代码规则检查工具在提高软件效率与质量方面非常有用,但没有一个工具是完全没有短板的,不同工具采用的分析手段及集成的规

表 2 工具对比

特征	QAFramework	CodeWizard	PcLint	C++Test
MISRA2004 支持	作为 MISRA 协会的主要成员,目前对该规则支持最为全面	支持,但不完全	支持,但不完全	通过 RuleWizard 自定义模块支持 MISRA 规则,只支持部分 MISRA C/C++规则
报告代码潜在问题	1 700 多种 C 和 1 300 多种 C++规则	C/C++共 500 多种	C/C++共 1 000 多种	C/C++共 800 多种
复杂度的度量	支持 44 种复杂度度量,包括圈复杂度,静态路径统计	不详	不支持	不支持
程序结构分析	提供软件调用结构图与模块内部逻辑流图	不支持	不支持	不支持
度量结果输出	静态统计功能,提供文本图形报告	不支持	不支持	不支持
规则的在线帮助	支持	支持	无规则解释	不支持
用户定制规则	提供 Post Analyses 接口用于规则定制。用户自己可方便地定制本单位编程规范	支持正则表达式检查	不支持	提供 Rule Wizard 模块,基于 GUI 的规则定制,可视化开发,学习需要一定的时间
用户定制报告	支持	不详	不支持	提供 20 种报告模版,不支持定制
分析速度	采用先进的 EDG 解决方案,分析器性能优越,分析速度快,报告准确。当前主流硬件配置下万行代码不超过 10 s	快	很快	私有分析器技术,分析速度中等,但打开自定义规则分析时速度较慢
与 IDE 集成	提供大多数 IDE 的集成插件	支持	提供大多数 IDE 的集成插件	可充分集成于 Wind River Workbench 和 ARM RVDS
支持平台	Solaris, HPUX, Linux, Windows NT, Windows ® 95,98 & 2000	大多数 Windows 与 Unix 平台	大多数 Windows 与 Unix 平台	大多数 Windows 与 Unix 平台

则数量决定了它们都只能发现部分缺陷。为了保证和提高软件测试的质量,笔者认为软件代码规则检查工具在软件静态分析时是行之有效的,但同时也要意识到测试工具的不足之处。

软件代码规则检查工具优点包括:

(1)开发早期发现软件编码规则错误,易于修改,降低开发成本;

(2)代码分析使用与开发的任何阶段,不需要牵扯太多其他因素;

(3)不需要设计测试用例,不需要代码插装,节约时间;

(4)发现问题时直接指引到问题所在行,易于修改;

(5)开发早期发现错误,有助于开发人员发现个人编码风格的缺点。

软件代码规则检查工具缺点包括:

(1)有可能会测试不全,造成测试质量无法 100% 保证;

(2)需要辅以人工判断;

(3)各家算法不尽相同,存在分析结果上的差异。

3　结束语

软件代码规则检查工具可以在软件开发的任何阶段使用,有助于降低软件成本和开发时间。随着此类工具分析技术及集成规则数量的不断发展,测试质量不断提高,渐渐受到软件开发人员、测试人员的钟爱。但无论如何,工具自身存在误报、漏报及算法片面性的因素,只是通过代码规则检查工具来进行检查是远远不够的,还需要在软件开发阶段实施动态测试。动态测试与静态分析相结合,黑盒测试与白盒测试相结合,进而提升测试质量。不同测试工具都有自身不同的特点,必须结合自身需求,挑选趁手的兵器。

参考文献:

[1] 王雅文,宫云战,杨朝红. 软件测试工具综述[J]. 北京化工大学学报:自然科学版,2007,34(A01):1-4.

[2] 黄茂生. 软件自动化测试工具的评估与选择[J]. 电子质量,2007(12):22-25.

[3] 杨　宇,张　健. 程序静态分析技术与工具[J]. 计算机科学,2004,31(2):171-174.

[4] 邓青华. 软件自动化测试工具研究[J]. 软件导刊,2011,10(1):57-59.

[5] 周　涛. 航天型号软件测试[M]. 北京:宇航出版社,1999.

[6] 鞠秀娟,赵　明. 软件自动化测试概述及应用工具分析[J]. 计算机应用,2007,27(B06):317-318.

[7] 周伟明. 软件测试实践[M]. 北京:电子工业出版社,2008.

[8] 韩　柯,杜旭涛. 软件测试[M]. 北京:机械工业出版社,2003.

[9] 王　倩. 软件自动化测试工具的分类与选择[J]. 玻璃,2008,35(8):51-53.

[10] 亢　勇,陈自力,李　鹏,等. 面向对象的软件测试[J]. 测试技术学报,1999,13(2):80-88.

[11] 罗　娜,林和平,袁福宇. 面向对象软件测试的方法研究[J]. 东北师大学报:自然科学版,2004,36(1):39-45.

[12] 高艳霞. 软件测试过程要因分析[J]. 中原工学院学报,2004,15(4):73-75.

[13] 王　云. 基于软件测试的软件质量分析研究[J]. 警察技术,2013(2):40-41.

[14] 赵振宇. 嵌入式软件测试技术研究及应用[D]. 北京:北京邮电大学,2011.

(上接第 192 页)

[4] Morgan W B, Prison S J. Theeffect of fractional wettability on the archie saturation exponent[C]//Proc of fifth annual logging symposium. Midland, TX:[s. n.],1964.

[5] 张海传,刘钟阳,许东卫,等. 基于 RBF 神经网络模型的臭氧浓度软测量研究[J]. 大连理工大学学报,2010,50(6):1020-1023.

[6] 刘军霞,阳春华,王雅琳. 螺旋分级过程数学模型研究及应用[J]. 计算机工程与应用,2010,46(4):230-232.

[7] 周　勇,胡中功. RBF 神经网络理论及其在控制中的应用[J]. 武汉科技学院学报,2007,20(5):40-42.

[8] 董长虹. Matlab 神经网络与应用[M]. 北京:国防工业出版社,2005.

[9] Deng Julong. Spectrum mapping in grey theory[J]. The Journal of Grey System,2000(2):116-124.

[10] Deng J L. Grey forecasting model[M]//Grey system. Beijing: China Ocean Press,1988:54-69.

[11] Deng J L. Properties of the grey forecasting model GM(1,1)[M]//Grey system. Beijing: China Ocean Press, 1988: 70-78.

[12] 赵振勇,王　力,王保华,等. 遗传算法改进策略的研究[J]. 计算机应用,2006(S2):189-191.

[13] 王　玮,蔡莲红. 基于粗集理论的神经网络[J]. 计算机工程,2001,27(5):65-67.

[14] 郝丽娜,徐心和. 粗糙集神经网络系统在故障诊断中的应用[J]. 控制理论与应用,2001,18(5):681-685.

代码规则检查工具评析

作者： 赵伟， ZHAO Wei
作者单位： 江苏自动化研究所,江苏 连云港,222061
刊名： 计算机技术与发展 ISTIC
英文刊名： Computer Technology and Development
年，卷(期)： 2015(7)