软件质量保证工具分析

共　　页

（包括封面）

拟　制 _____

审　核 _____

批　准 _____

修改记录

文件编号	版本号	拟制人/修改人	拟制/修改日期	更改理由	主要更改内容（写要点即可）
注 1：每次更改归档文件时，需填写此表。 注 2：文件第一次归档时，“更改理由”、“主要更改内容”栏写“无”。					

目　　录

1 概述

1.1 编写目的

本文对多种软件质量保证工具进行简要的介绍，并对它们进行综合比较和评价。以提供各种类型的软件单元测试时选用测试工具及其组合的参考。

软件质量保证工具安装指导、详细介绍和使用方法请参考相应的文档，比如厂家资料、使用说明、工具向导等。

本文不关注网络事业部<软件单元测试工作规程>的 8 个活动的前四个活动,只关注后四个活动.

1.2 研究软件质量保证工具的目的

使用软件质量保证的根本目标是提高测试质量和提高测试效率。

常用软件软件质量保证工具的测试可分为代码分析工具和软件测试工具。

1.2.1 代码分析工具

采用工具进行的代码检查主要包括：

1. 规则检查；
2. 代码特性分析；

对于规则检查希望具有以下全部或部分功能：

1. 能够检查使用的编程规范中所有规则；
2. 能够定位违反规则的代码和违反规则细节；
3. 能够输出报告；
4. 使用方便、直观；

对于代码特性分析希望具有以下全部或部分功能：

1. 给出分支结构图；
2. 给出状态图，或状态、转换表；
3. 给出调用和被调用者关系；
4. 给出软件质量指标数据；
5. 使用方便、直观；

1.2.2 软件(单元)测试工具

测试工具又可分为两类：支持测试用例的生成与加载、运行信息的收集与输出。

1.2.3 测试用例的生成与加载

支持测试用例的生成与加载，即提供测试框架或者测试床和用例管理，希望达到以下全部或部分目标：

1. 减少生成测试用例的工作量和难度；
2. 方便修改、添加、屏蔽、删除测试用例；
3. 方便生成和定制桩；
4. 方便加载执行测试用例；

5. 方便选择测试用例运行；
6. 方便测试结果的判别；
7. 强大的测试用例管理功能；
8. 方便进行回归测试；
9. 测试用例的导入和导出；

1.2.4 运行信息的收集与输出

具体而言希望达到以下全部或部分目标：

1. 提供覆盖率数据，入代码覆盖，分支覆盖，路径覆盖等；
2. 提供代码运行错误报告，如内存、指针错误，变量使用错误，内存使用统计等；
3. 故障定位功能，便于分析排除故障；
4. 使用方便、直观，如 GUI 界面；
5. 输出测试结果、报告；

2 代码分析

2.1 代码规则检查

能够进行代码规则检查的工具主要有：

1. Verilog 公司 Logiscope 的 RuleChecker 工具；
2. Parasoft 公司的 CodeWizard 或 C++Test 的 RuleWizard 工具；
3. 中兴通讯股份有限公司 的 CodeClear；

CodeClear 是我们公司开发的用于 C/C++源代码编码程序风格的审核工具，CodeClear 可以对 C/C++的源代码进行扫描，给出源代码和已经定义的程序风格摸板的差异和修改报告，经过代码的早期审核，既有利于编码中错误的早期发现，又有利于源代码的管理和后期维护。

只要能够检查软件编程规范中列举的编程规则（要求用工具检查部分），使用哪一种工具应该都是可以的。由于我们公司购买了 logiscope 的 RuleChecker 工具的 license，所以我们可以采用该工具进行编程规则的检查（也可采用 CodeClear 检查的），而且目前检查的规则很少，这里就不再进行比较了。

2.2 代码特性分析

代码分析工具主要有：

1. Verilog 公司 Logiscope 的 WinViewer 工具或 Audit 工具；
2. SourceInsight；
3. Visicode；

其中，SourceInsight 以及一些编辑工具只是提供了函数调用关系，远不是代码分析工具，由于目前公司采用了 Logiscope 系列工具，而且代码分析的工作要到下一步才作，因此暂时也不做详细的比较，下面仅简要描述一下 Audit 的功能。

2.2.1 Audit 的功能

Audit 以 ISO9126 模型作为质量评价模型的基础。质量评价模型描述了从 Halstead、McCabe 的度量方法学和 Verilog 引入的质量方法学中的质量因素（可维护性、可重用性、等）和质量准则（可测试性、可读性等）。

Audit 将被评价的软件与规定的质量模型进行比较，用图形形式显示软件质量。因此，质量人员可以把精力集中到需要修改的代码部分，对与质量要素和质量模型不一致的地方给予解释和找出纠正的办法。具体的图形表示法有以下几种：

- 整个应用的体系结构：显示部件之间的关系，评审系统设计。

- 具体部件的逻辑结构：通过控制流图显示具体部件的逻辑结构，评审部件的可维护性。
- 评价质量模型：通过度量元对整个应用源代码进行度量，并作出 Kiviat 图显示分析结果，对可维护性作出评判。

Telelogic Tau Logiscope Viewer 窗口可以报告七种统计结果，包括：

- Component List
- Call Graph
- Average Kiviat
- Average Table
- Component Distribution per Metric
- Component Distribution per Criterion
- Report

2.2.1.1 查看总体质量报告

总体质量报告中，以饼图的形式给出被分析的文件中了 Excellent（卓越）、Good（优秀）、FAIR（差）、POOR（极差）四个等级的 Component（函数）的百分比。这是个等级是根据 Logiscope 内建的质量模型，对函数的各个质量指标进行评估，再加权平均得出的。其质量模型可以通过修改质量模型文件（Quality Model file）调整。

有两类总体质量报告，一类是函数可维护性 (function MAINTAINABILITY) 质量报告；另一类是相关调用可维护性(relativeCall MAINTAINABILITY)质量报告。

2.2.1.2 查看每个质量度量的合格/不合格分布

2.2.1.3 查看判定结果的分布

2.2.1.4 查看被测函数质量列表

函数质量列表中可以选择显示质量度量选项，将这个列表按某个度量值排序，以便找出该度量值在一定范围内的所有函数，如圈复杂度大于 10 的函数或扇入数大于 6 的函数，这些函数在代码走查中需要被特别关注。

2.2.1.5 查看调用关系图

2.2.1.6 查看控制图

3 软件(单元)测试工具

用于软件(单元)测试的工具主要有：

1. Rational 公司的 RTRT；
2. Rational 公司的 PurifyPlus（purify、pureCoverage、pureQuantity）；
3. Verilog 公司 Logiscope 的 TestChecker；
4. Parasoft 公司的 Insure++、C++Test、C++Test for Embedded system、 Jtest，
5. NuMaga 公司的 BoundsChecker；
6. Xunit；
7. AMC 公司的 CodeTest；

这些测试工具从主要功能上中主要分为以下几类：

1. 仅提供测试代码框架，不能统计覆盖率和捕获运行错误，如 Xunit；
2. 仅能够统计覆盖率，但不能捕获运行错误，如 logiscope（TestChecker）；
3. 不能统计覆盖率，但能够捕获运行错误，如 Insure++、BoundsChecker；
4. 能够统计覆盖率，能够捕获运行错误，如 CodeTest、purifyPlus；
5. 能够提供测试框架，能够统计覆盖率，能够捕获运行错误，如 RTRT（及其辅助工具）；

这些工具从适用的平台上讲可分为：

1. 适用于 Windows 环境（对 C/C++一般与 VC 集成），如 PurifyPlus、Insure++、BoundsChecker、C++Test 等；
2. 适用于 RTOS 环境，如 Codetest，C++Test for Embeddedsystem、Purify 的 RTOS 版本；
3. 可用于 windows，也可用于 RTOS，如 RTRT、logiscope 、Xunit（部分）；
4. 其它

对于在 Windows（VC）平台上捕获运行错误的测试工具的功能基本差不多，Insure++、BoundsChecker 等并不比公司购买了 license 的 PurifyPlus 系列有明显的优势，因此本文也不作比较了。大家对于 Windows（VC）下的覆盖率和捕获运行错误的测试可以直接使用 Purify 系列工具。

下面先简单列举主要工具的优点和缺点：

3.1 C++TEST

3.1.1 主要功能

C++Test 是 Parashoft 公司推出的自动化 C/C++单元测试和代码标准分析产品。

C++Test 功能主要有：

1. 代码分析
2. 单元测试：自动实时单元测试，支持白盒测试设计、黑盒测试设计和回归测试

3.1.2 优点

功能：

1. 测试粒度可大可小，测试单元大小可以为函数、单个文件、多个文件、project。
2. 自动生成测试用例；
3. 自动加载测试用例。不用编写脚本来加载测试用例；
4. 方便地修改、添加、屏蔽、删除测试用例，可以设置修改预置条件、输入参数和预期结果；
5. 自动生成和定制桩，测试代码调用函数是用桩还是源码可选；
6. 导入导出测试用例和测试对象；
7. 各种测试覆盖率统计；
8. 自动回归测试；
9. 用户定制测试设置；

10. 代码抑制；
11. 支持黑盒测试用例设计
12. 生成测试报告；
13. 结合insure++检测运行错误；
14. GUI界面，操作、查看信息方便直观；
15. 可与VC集成（windows版本）；
16. 输出直观，输出结果和源码可以相互链接；

优点点评：

1. GUI 界面，操作、查看信息方便直观；
2. 测试用例管理十分清晰直观。适用于单元测试，也适用于集成测试和回归测试；
3. 可以完成规则检查、单元测试、覆盖率统计、捕获运行错误（与 insure＋＋结合）；

3.1.3 缺点

缺点点评：

1. 对于接口参数和环境条件复杂的代码其自动生成的测试用例可用度不大。一般需要修改测试用例，甚至编写测试函数间接测试。不过对于大多数测试软件自带例程这样的接口简单的代码，自动生成的测试用例还是可用的，而编写测试用例是每个单元测试工具所必须的，C++Test 可以用对测试用例的测试来间接测试被测代码；
2. RTOS 版本功能没有这么强，一般对上层软件的单元测试还是移植到 VC 中进行测试。

3.1.4 综合评价

C++Test 是个很不错的单元测试工具，其测试用例管理和支持回归测试的功能让人心动。但是需要购买 license，不菲的价格可能影响大规模的使用。

3.2 RTRT

3.2.1 主要功能

RTRT 单元测试工具是 Rational 公司推出的一套软件测试工具，它还包括四种辅助功能：

（1）PurifyLT：内存分析
（2）QuantifyLT：性能分析
（3）Coverage：覆盖率分析
（4）Trace：动态跟踪

3.2.2 优点

功能：

1. 被测单元既可以是单独的模块，也可以是几个模块的集成系统；
2. 自动生成测试用例模板；
3. 提供丰富的脚本语言，用于测试用例的构造；

4. 自动生成无差错的测试驱动模块和桩模块；
5. 测试执行过程的实时监测；
6. 可视化的测试结果报告；
7. 回归测试的自动化；
8. 直观的图形界面；
9. 捕获运行错误，如内存错误报告；
10. 性能统计，包括函数执行时间和调用次数统计；
11. 统计覆盖率，包括语句、条件、循环覆盖率结果；
12. 可直接在目标机上运行；

优点点评：

1. GUI 界面，操作、查看信息方便直观；
2. 直接完成单元测试（测试结果判别）、覆盖率统计、捕获运行错误；
3. 真正的 RTOS 单元测试工具；
4. 公司购买了 license；

3.2.3 缺点

缺点点评：

1. 需要用脚本语言编写测试用例，脚本写错比较难检查；
2. 测试用例管理能力较弱；

3.2.4 综合评价

虽然有人对 RTRT 需要编写测试脚本颇有微辞。但 RTRT 是个很不错的适合于 RTOS 软件的单元测试工具，其对 RTOS 的支持能力，以及同时收集覆盖率、捕获运行错误的能力，为大多数单元测试工具所不及。

公司已经购买了 RTRT 的 license，因此建议大家使用该工具进行单元测试。

3.3 Xunit

3.3.1 主要功能

Xunit 家族包括 Junit、CppUnit、Cunit、Dunit、EmbUnit 等，分别支持 Java、C++、C、Delphi、RTOS 等。

3.3.2 优点

功能：

1. 开放的源码，无须license；
2. 自动加载测试用例；
3. 支持回归测试；

4. 测试用例（驱动）和桩编写灵活；

优点点评：

1. 开放的源码，提供单元测试框架，使用十分灵活；
2. 支持回归测试；

3.3.3 缺点

缺点点评：

1. 需要编写测试用例；
2. 不能统计覆盖率，不能捕获运行错误；
3. 虽然可以在目标机上执行测试，但是主机不能收集目标机的测试结果（没有 target 和 Host）。
4. 测试用例管理能力较弱。
5. 测试结果输出简单；

3.3.4 综合评价

Xunit 是个不错的单元测试软件，使用简单方便。但是测试用例管理功能较弱，统计覆盖率、捕获运行错误需要其它软件的帮助。

3.4 logiscope

3.4.1 主要功能

Logiscope 是 Verilog 公司推出的自动化 C/C++单元测试和代码标准分析产品。

Logiscope 主要有以下一些工具：

1. Audit，－－用于代码质量评价
2. RuleChecker，－－用于编程规则检查
3. TestChecker，－－用于覆盖率统计

Audit和RuleChecker在代码评审里讨论了，这里只讨论TestChecker。

3.4.2 优点

功能：

1. 可以统计语句覆盖、判定覆盖、MC/DC（条件组合覆盖）和基于应用级的PPP覆盖；
2. 支持对嵌入式系统的覆盖率分析，支持VxWorks 、pSOS 、VRTX等；

3.4.3 缺点

缺点点评：

1. 只能统计覆盖率，不能捕获运行错误；

3.4.4 综合评价

logiscope 的 TestChecker 能够统计嵌入式系统的覆盖率，这是不少单元测试工具所没有的，可以用它结合其它单元测试工具来进行单元测试。

3.5 codeTest

3.5.1 主要功能

CodeTest 是 AMC 公司推出的嵌入式软件开发、测试工具，既适用于开发人员开发调试，也适用于测试人员进行测试。

C++Test 功能主要有：

1. 捕获运行错误；
2. 统计覆盖率；
3. 性能评估；

3.5.2 优点

功能：

1. 支持嵌入式系统软件测试；
2. 捕获运行错误，如捕获内存指针错误，定位故障；
3. 统计覆盖率；
4. 性能评估，如统计函数的调用次数、执行时间，评估代码性能，定位代码执行效率瓶颈；
5. 使用Probe硬件设备，直接在地址总线、数据总线上捕获硬件信号，运行效率高，对被测系统影响小；
6. 界面直观；

3.5.3 缺点

缺点点评：

1. 购买 Probe 硬件设备，需要目标机提供硬件接口，且设备价格昂贵；
2. 单独 CodeTest 软件的性能和一般软件测试工具相同；

3.5.4 综合评价

CodeTest 是专业的嵌入式软件开发测试工具，带有硬件 Probe 的 CodeTest 是笔者到目前为止见到的最好的嵌入式软件测试工具。没有硬件配套的 CodeTest 大打折扣，但是仍然具有与 Tornado 的集成，使用方便直观的优势，与 RTRT 的辅助工具的功能相当。

3.6 Purify 系列

3.6.1 主要功能

Purify 系列是 Rational 公司推出的软件测试工具，主要有三部分：

1. Purify，捕获运行错误；
2. Coverage，统计覆盖率；
3. Quantify，性能评估；

Purify 系列软件有多个版本，如 Windows（VC）版本、RTOS 版本等。功能齐全使用方便；

3.6.2 优点

功能：

1. 无需被测应用程序的源代码或特殊的工作版本，就能检测应用程序的代码以及所有链接到该应用程序的构件代码；
2. 特有的PowerCheck 功能，可以按模块逐个调整所需的检查级别；
3. 可选择对整个应用程序或仅仅对选择的模块执行测试；
4. 使用方便，输出结果直观，生成测试报告；
5. 支持最多的平台和开发环境；

3.6.3 缺点

缺点点评：

1. 捕获动态测试运行信息，不提供单元测试的测试用例的生成和加载的支持，结合 RTRT 就基本覆盖了单元测试所需的所有功能；

3.6.4 综合评价

Purify 系列是很不错的测试工具，特别是 PurifyPlus 真是必不可少的软件（非 RTOS 软件）测试工具，集成在 RTRT 中的辅助工具也是很好的 RTOS 软件测试工具。

4 测试工具的搭配

规则检查和代码分析部分前面做了简单的比较。这里主要比较动态测试的工具。

从以上各个测试工具的分析可知，有些测试工具只支持测试用例的生成与加载，如 Xunit；有些工具只收集运行信息，如 Purify 系列、Logiscope（TestChecker）、CodeTest；有些测试工具两个兼备，如 RTRT（及其辅助工具）、C++Test（Insure++），其实它们也是一个公司将两个或多个工具集成在一起使用。

同样，我们也可以将不同公司的测试工具组合起来使用，已达到既支持测试用例的生成和加载，也支持运行信息的收集，同时方便使用。

根据各个工具的功能和特点，提出以下几种搭配关系，提出以下搭配关系时部分考虑了 license 和价格的因素，对于已经购买 license 的或免费的测试工具将增加其推荐权重。

4.1 基于目标机的测试工具搭配方案一

针对目标机（基于 RTOS）软件测试可以直接使用 RTRT 测试工具，但是采用以下的工具搭配也未尝不是一件值得尝试的事情。

EmbUnit（Cunit）＋ RTRT 的四个辅助工具 ＋Logiscope(RuleChecker)

能够实现的功能：

1. 编程规范检查；
2. 测试用例编写方便（C/C++语言编写，而非脚本语言）；
3. 测试用例的加载、执行；
4. 统计覆盖率；
5. 捕获运行错误；

6. 性能分析；
7. 收集测试结果；
8. 无须增加购买 license

利用 EmbUnit（Cunit）替代 RTRT 编写和加载测试用例，这样就不需要用 RTRT 脚本编写测试用例。用源代码语言（语言 C）直接编写测试用例，使得测试用例的编写、调试和加载十分方便，其测试组件和测试用例的组织也更直观。EmbUnit（Cunit）是开放的源代码，无须 license。

利用 RTRT 可以将测试码下载到目标板运行，并收集目标机上测试用例的执行情况和结果。

RTRT 的四个辅助工具仍然可以得到充分发挥，purifyLT 捕获内存错误、统计内存使用情况；QuantifyLT 分析代码执行性能，如函数执行时间、调用次数等；Coverage 统计覆盖率，包括语句覆盖率、条件覆盖率和循环覆盖率数据；

利用 RuleChecker 进行代码规则检查；

4.2 基于目标机的测试工具搭配方案二

EmbUnit（Cunit）＋ CodeTest ＋Logiscope(RuleChecker)

能够实现的功能：

1. 编程规范检查；
2. 测试用例编写方便（C/C++语言编写，而非脚本语言）；
3. 测试用例的加载、执行；
4. 统计覆盖率；
5. 捕获运行错误；
6. 性能分析；
7. 收集测试结果；

利用 EmbUnit（Cunit）编写和加载测试用例，用源代码语言（语言 C）直接编写测试用例。利用 CodeTest 将测试码下载到目标板运行，并收集目标机上测试用例的执行情况和结果。

利用 CodeTest 的功能来捕获内存错误、统计内存使用情况；分析代码执行性能，如函数执行时间、调用次数等；统计覆盖率；

利用 RuleChecker 进行规则检查；

4.3 基于主机的软件测试工具搭配方案一

基于主机（基于 Windows 平台）软件测试可以可以直接使用 RTRT 测试工具，但是采用以下的工具搭配也未尝不是一件值得尝试的事情。

CppUnit＋PurifyPlus ＋ RuleChecker

能够实现的功能：

1. 编程规范检查；
2. 测试用例的代码框架；
3. 测试用例的加载、执行；
4. 统计覆盖率；
5. 捕获运行错误；
6. 性能分析；
7. 收集测试结果；
8. 无须增加购买 license

利用 CppUnit（Cunit）编写和加载测试用例，方便测试用例的编写、调试和加载。

PurifyPlus 测试 Windows 平台下 VC 开发的软件十分方便，功能也很强大，同时对 Java 应用的测试也一样的方便实用。利用 PurifyPlus 的 purify 工具来捕获内存错误、统计内存使用情况；利用 Quantity 来分析代码执行性能，如函数执行时间、调用次数等；利用 PureCoverage 来统计覆盖率；

利用 RuleChecker 进行规则检查；

4.4 基于主机的软件测试工具搭配方案二

针对公司和事业部常用的 Windows 平台软件的测试可以采用如下的搭配：

C++Test ＋ Insure++（或 PurifyPlus）

能够实现的功能：

1. 编程规范检查；
2. 测试用例的代码框架；
3. 测试用例的加载、执行；
4. 统计覆盖率；
5. 捕获运行错误；
6. 收集测试结果；

利用 C＋＋Test（对于 Java 软件实用 J++Test）自动生成测试用例和加载测试用例，C++Test 的测试用例管理功能十分直观、方便，对于集成测试、回归测试特别有用。C++Test 自动生成测试用例使得不需要编写测试脚本。对于接口较为复杂的软件单元，建议编写测试用例，C++Test 测试“测试用例”来间接测试软件单元。C++Test 本身集成了规则检查和覆盖率测试。

Insure＋＋测试 Windows 平台下 VC 开发的软件十分方便，功能也很强大，利用它来捕获内存错误、统计内存使用情况，检测变量非法使用等；

缺点是需要购买 license。

4.5 基于主机的软件测试工具搭配方案三

针对公司和事业部常用的 Windows 平台软件的测试可以采用如下的搭配：

JUnit＋PurifyPlus ＋RuleChecker

能够实现的功能：

1. 测试用例的代码框架；
2. 测试用例的加载、执行；
3. 统计覆盖率；
4. 捕获运行错误；
5. 性能分析；
6. 收集测试结果；
7. 无须增加购买 license；

参见 Cunit＋PurifyPlus＋RuleChecker 的搭配，使用方法是一样的，只不过是用 Junit 来编写组织和加载测试用例。