嵌入式软件测试开发环境的框架设计

章 亮 刘 斌 陆民燕

(北京航空航天大学 工程系统工程系, 北京 100083)

摘 要: 采用软件设计上的复用思想, 对嵌入式软件仿真测试环境进行框架的设计, 可以使测试环境在不做大幅度修改的情况下适应不同的被测软件, 节省设计的时间和费用. 论述了由框架开发软件的过程, 并把此过程应用于测试环境; 对嵌入式软件仿真测试环境进行总体分析, 确定了它的3个主要组成部分; 归纳出作为测试环境重要组成部分的测试开发环境的基本功能, 并使用专门针对框架开发的UML-F建模语言设计了一个框架; 对已开发成功的测试开发环境框架, 提出了具体的框架适配方案. 本设计已在实际工程中得到了应用.

关 键 词: 嵌入式软件; 软件测试; 测试环境; 框架

中图分类号: TP 311

文献标识码: A 文 章 编 号: 1001-5965 (2005) 03-0336-05

Framework design of embedded software testing development environment

Zhang Liang Liu Bin Lu Minyan

(Dept. of System Engineering of Engineering Technology, Beijing University of Aeronautics and Astronautics, Beijing 100083, China)

Abstract: By adopting the idea of software design reuse, framework design for an embedded software simulation testing environment can offer an efficient solution to construct testing environment without largely changing the architecture of testing environment when testing different embedded software, and hence reduce development time and cost. The software development process using ideas of framework design was discussed and then applied to the testing environment's development. Based on the general analysis of existing embedded software simulation testing environment, three main components were identified to be the basic parts of testing environment. The basic function of testing development environment was defined, which was an important part of the testing environment. A concrete framework was designed by using the unified modeling language profile for framework UML-F, a specific language for framework design. An adaptation scheme was put forward for a testing development environment that was successfully developed. The framework studied has been applied in a practical project.

Key words: embedded software; software test; testing environment; framework

根据IEEE给出的定义, 软件测试是使用人工或自动手段来运行或测定某个系统的过程, 其目的在于检验软件是否满足规定的需求或是弄清预期结果与实际结果之间的差别. 采用软件仿真测试环境进行软件测试是实现软件自动测试的一个重要途径, 特别是对嵌入式软件的测试.

软件测试环境是面向软件测试的计算机系统, 测试人员可以根据被测软件的要求, 通过对被测系统的各种资源进行配置, 组织被测软件的输入, 来驱动被测软件运行, 同时接收被测软件的输出结果, 从而对软件进行闭环测试. 软件测试环境通过自动进行软件测试和分析, 能够克服由于手工测试带来的巨大的人力、物力和财力的消耗, 具有较强的适应能力[1].

嵌入式软件是运行于嵌入式系统中的起核心控制作用的软件, 它需要在特定的硬件环境下才

收稿日期: 2003-08-28

作者简介: 章 亮 (1979-), 男, 江西南昌人, 硕士生, brightchan@163.com.

能运行. 嵌入式软件仿真测试环境是一个面向嵌入式软件测试的测试环境. 由于嵌入式软件的运行与硬件有着密切的联系, 因此嵌入式软件仿真测试环境需要根据被测软件的改变而改变, 但是重新设计开发或者进行大范围修改测试环境都将浪费大量的人力和物力, 效率也得不到保证. 对软件进行设计上的复用, 可以减少测试环境的设计时间, 提高效率, 有利于解决上述问题. 框架是一种可复用的体系结构, 属于设计上的复用, 根据框架来适配应用软件, 最终能够减少相似软件的代码量, 提高软件的可复用性和适应性. 本文从框架体系结构入手, 研究嵌入式软件仿真测试开发环境中的固有结构和变化点, 获得测试环境在设计上的复用.

1　框架概念及开发过程

框架是一种研究和构造软件体系结构的方法, 它是关于应用领域的已建立的系统结构, 也称为应用框架. 从设计模式的角度看, 应用框架是一个给定应用领域中的完全的软件体系模式, 是大颗粒度的可复用部件[1]. 框架是从实际应用领域中抽象出来的具有领域特征支撑性功能体. 它可以是一组系统级对象, 也可以是由一组程序模块组成的模板.

应用框架可以分为冷点 (frozen spots) 和热点 (hot spots). 冷点定义了软件系统的完整结构, 包括它的基本部件和部件的关系, 在应用框架的任何实例中它们都是保持不变的. 热点描述了应用框架中的那些随具体应用需求而不同的部分[2]. 热点的设计应考虑到一般化和通用性, 要易于适配成具体的应用程序. 框架的适配是把框架调整为具体的应用程序的过程, 它意味着两种情况: 其一是对框架类中作为热点的方法或者接口进行重载; 其二是采用框架中已有的程序模块装配新的程序模块.

框架随着在各个具体领域中软件的应用而得到飞速发展. 在各个领域里面, 大多都有对体系框架的研究. 比如业务应用的软件框架、电力系统的框架以及石油应用系统的框架, 也有对软件测试领域内的框架研究[3] 等等.

要开发成熟的、可方便重用的软件框架和组件, 应该采用设计模式[4]. 设计模式描述了如何利用面向对象的基本概念和机制解决可扩展的软件设计中经常出现的问题, 针对反复出现的设计问题给出可复用的解决方案. 设计模式的提出标志着整个软件设计技术和水平的逐渐成熟. IBM 公司的 San Francisco 框架就采用了大量的已有软件的设计模式.

框架开发需要付出很多的开发精力, 要求框架开发者对软件所处领域进行深入的了解, 甚至有可能已经设计实现过该领域中的应用软件系统. 由于框架开发是一个随着对领域认识逐渐深入而反复抽象的过程, 框架的开发通常要循环进行以下 3 个过程: 结构定义、框架实现以及对框架的评估反馈. 框架开发的进化过程如图 1. 本文将采用框架的进化过程进行测试开发环境框架的开发.

在结构定义的阶段, 要求框架的开发者能够根据在特定领域中的已有知识和已有应用软件来抽象出软件结构, 识别出特定领域的应用软件的冷点和热点, 目的是提炼出这个软件结构具有领域中应用软件的内在一致的结构, 确定框架的总体结构、控制流和数据流.

在框架实现阶段, 伴随着具体应用系统的实现, 得到框架的实例, 其中包括与应用有关部件的程序模板.

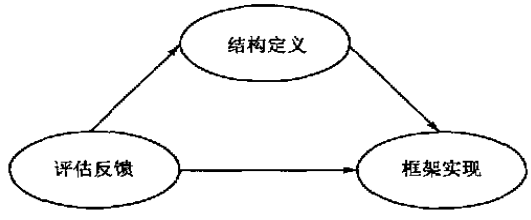

图 1　框架开发的进化过程

以提高对领域框架的可重用性和通用性为目的, 鉴于对领域的认识是逐步深入的过程, 还需要对框架进行评估和验证, 并给框架开发人员反馈更加全面的信息. 框架的实例可以提供给用户使用, 并在用户的使用过程中得到评价、验证和反馈. 之后, 框架开发人员根据反馈, 对框架的结构进行更进一步的抽象和修改. 如此反复, 正好构成了框架开发的进化过程.

2　软件测试开发环境分析

嵌入式软件仿真测试环境可以分为 2 层, 分层体系结构如图 2. 软件测试开发环境是一个基于测试的集成开发环境. 系统仿真建模环境是对目标系统的交联设备进行仿真建模的环境; 测试环境的实时内核是处理测试时实时数据的程序模块.

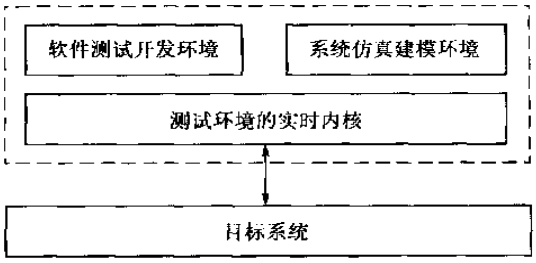

图 2 软件仿真测试环境的分层体系结构

通常测试环境包括以下功能:

1) 初始化;

2) 测试准备;

3) 测试执行;

4) 测试结果分析处理;

5) 测试文档管理;

6) 日常维护管理.

软件测试开发环境是构成软件测试环境的一部分, 主要是软件测试环境中和测试人员交互的软件系统, 是一个基于测试的集成开发环境. 测试开发环境具有的功能有:

1) 提供一个良好的用户界面;

2) 配置和管理测试, 用户管理以及针对嵌入式软件测试的时间特性的配置;

3) 生成测试脚本;

4) 适应不同的测试脚本;

5) 具有配置 I/O 和仿真模型的功能. 通过配置 I/O, 可以适应不同的 I/O 接口, 支持 I/O 种类的扩展; 配置仿真模型则是为了适应不同的目标系统的交联环境;

6) 可以开发显示面板, 供测试人员在测试期间进行观看测试的回收数据;

7) 具有日志管理的功能.

以上 7 个功能, 是在抽象的基础上获得的程序骨架, 在利用这几个功能部件搭建测试开发环境时, 必须把它们分别实例化. 构成的原理主要就是针对测试开发环境的热点进行抽象, 以取得软件的灵活性, 同时对它的冷点进行建模, 分析出固有的部件和部件之间的关系, 符合框架中处理问题的原理.

3 构建软件测试开发环境框架

具体地开发框架, 需要有一个可以对框架设计、开发和适配过程进行建模的语言. UML-F 就是这样的一种语言. 有了这种建模语言, 可以更好地对框架设计进行交流.

UML-F 确定了 UML 的一个子集, 专门针对框架开发和适配[5]. UML-F 采用了 UML 中的 3 种视图: 类图、对象图和顺序图. 另外, UML-F 为描述框架提供了额外机制. 在 UML-F 中, 所有类都是实现类, 方法调用是考虑对象之间交互的主要形式, 对象组合以关联为基础. 对于测试开发环境, 本文也采用 UML-F 来对框架进行建模.

本文采用了 UML 的用例图对框架的需求进行了分析. 图 3 是采用测试开发环境的用例图.

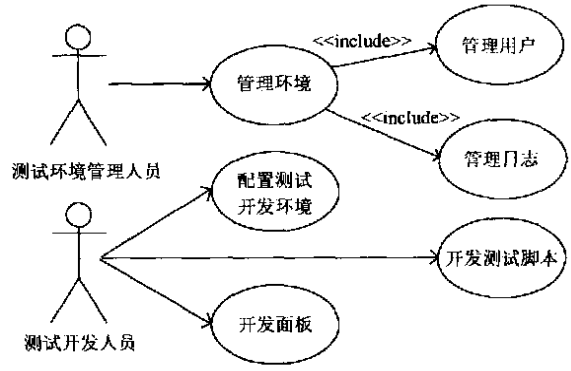

图 3 测试开发环境用例图

结合前面提到的测试开发环境的功能构成, 框架体系主要由以下几个部分构成:

1) 开发环境的用户界面 (GUI): 是一个由通用的图形界面开发库构成的操作界面, 由开发库里面的固定类所构成. 界面程序可以随着开发库的类的扩展而得到扩展, 它在测试开发环境中体现为包 (package), 下面每一个类和子系统都需要依赖于 GUI;

2) 测试用户类 (CTestUser): 它是用户界面中的热点, 是为了适应不同的用户, 如测试开发人员、测试人员和测试环境管理人员, 体现为测试相关人员的一个抽象类;

3) 分级管理机制类 (CTestHiberarchy): 主要就是定义本测试开发环境中所需要的分级管理的深度和每一层的名称及各层次之间关系. 具体来说, 分级管理可以分为 3 层: 测试项目、测试工程和测试脚本; 它们之间的关系为包含关系, 测试项目包含测试工程, 测试工程包含测试脚本. 它创建了日志管理对象, 采用了 Factory method 设计模式;

4) 日志管理类 (CTestLog): 它依赖于测试的分级管理, 也是一个抽象类, 定义了日志所需要的几个基本的元素, 如时间和事件. 由它继承下来的子类的数目不多于测试开发环境中的分级管理中定义的层数, 它依赖于其它各类;

5) 环境配置类 (CEnvConfig): 它对整个环境进行配置, 比如测试开发环境为了适应多种测试脚本, 就要配置所使用的测试脚本的类型; 另外, 测试环境需要引入相应的测试脚本解释器, 因此类里面还定义了方法用来引用相应的脚本解释器; 此外还可以配置被测嵌入式软件的最小运行周期和平台的调度周期; 开发人员也可以在环境配置类中添加属性和方法来扩充新的子系统, 使得环境具有一定的灵活性和可扩充性;

6) I/O 配置类 (CIOConfig): I/O 配置需要一个外来的 I/O 列表文件, 通过读取 I/O 列表文件, 得到用户给测试开发环境提供的 I/O 类型. I/O 配置类定义了 I/O 初始种类链表、选择后的种类链表以及 I/O 和仿真模型的关联链表;

7) 仿真模型配置类 (CModelConfig): 同 I/O 配置类相同, 它也需要外来的一个模型列表, 列表的节点对应于仿真模型开发环境提供的特定格式的仿真模型, 是适应模型的一个热点. 通过读取模型列表, 测试开发环境确定测试中的仿真模型. 仿真模型配置类定义了模型初始种类链表和选择过后的模型链表, 并且通过引用 I/O 和仿真模型的关联链表, 实现和 I/O 配置类的关联;

8) 测试脚本抽象类 (CTestScript): 测试脚本抽象类定义了测试脚本的名称和路径属性, 具有确定的测试脚本类型; 测试脚本可以由颗粒度更小的任务组成, 方便以后单个脚本在不同节点上分布. 测试开发人员通过测试脚本开发程序生成测试脚本, 把测试脚本的抽象类实例化成具体的测试脚本;

9) 测试脚本开发程序包 (ScriptDev): 它包含了几种测试生成方法, 例如文本和图形方式. 它依赖于环境配置类, 环境配置类提供了测试脚本的类型. 通过脚本类型的配置, 用已经配置的测试脚本的语法来生成测试脚本;

10) 测试显示面板的开发程序包 (PanelDev): 引用几个常用控件来组织一个面板, 可以用于观看实时的测试结果数据. 要实现测试显示面板的可扩充性, 需要定一个控件类型列表, 每一个列表元素都可以对应一种控件. 要增加测试的显示类型, 可以通过增加控件, 然后把控件类型添加到类型列表, 得到新的扩充后的面板控件集合, 它依赖于测试脚本.

图 4 是框架关系的类图. CTestUser 实现用户管理, CTestHiberarchy 实现分级管理, CTestLog 实现日志管理, ScriptDev 和 PanelDev 分别是测试脚本和面板的开发程序. 针对嵌入式软件的测试, 本测试开发环境首先在 CEnvConfig 类中配置用于嵌入式软件测试的测试脚本类型, 引入相应的脚本解释器, 配置被测嵌入式软件的最小运行周期 (比如某系统的周期为 25ms)、测试环境的任务的种类、任务调度周期和各任务的优先级等. ScriptDev 根据配置的环境变量、仿真模型、被测嵌入式软件的 I/O, 生成测试脚本, 在实时内核中运行的脚本解释器对 CTestScript 进行实时解析, 最终分发到仿真模型中进行解算, 驱动被测嵌入式软件的运行.

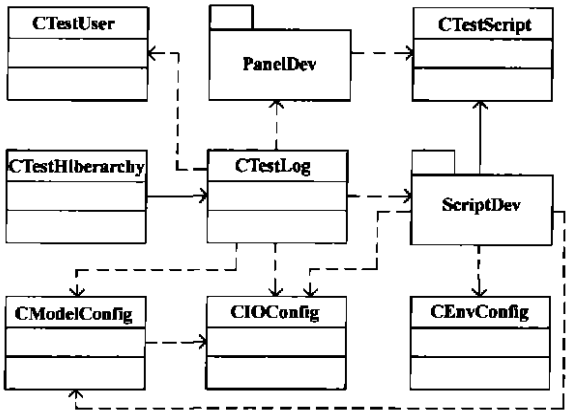

图 4　测试开发环境框架的类图

图 5 是 CTestLog 和 CTestHiberarchy 之间采用了 Factory Method 设计模式的类图. 有 "..." 标记的代表相关图的描述是不完整的, 忽略了一部分内容[5]; "Adapt-static" 标记表明子类为静态适配.

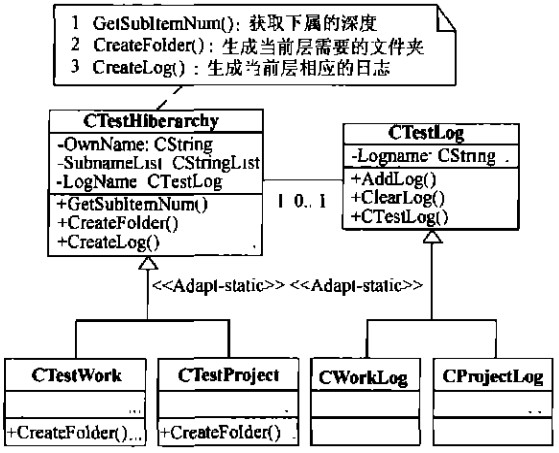

图 5　CTestLog 和 CTestHiberarchy 的类图

测试分级管理机制类和日志管理对象之间的交互图如图 6 所示. 首先 CTestHiberarchy 对象创建一个文件夹, 用来存放所需的文件, 然后使用 CreateLog 方法创建 CTestLog 的对象, 最终得到一

个相应的日志实例.

测试开发环境的框架结构在某些地方还可以划分成颗粒度更小的软件部件，使得软件框架可以更好地得到设计上的复用，这正好符合软件框架的进化开发过程.

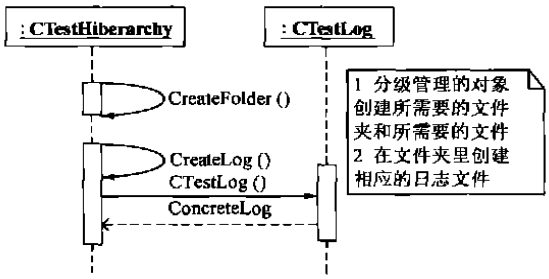

图 6 分级管理机制类和日志管理类的交互图

4 应用与结论

本测试开发环境采用 Visual C++ 进行开发，已在某嵌入式软件仿真测试单机平台中得到了验证. 嵌入式软件仿真测试单机平台已经成功地对航空某型号的航电系统软件进行了测试；另外，在保持框架的固定结构的基础上，测试环境通过对测试开发环境框架功能变化点的适配，还可以方便地进行扩充，不但能够测试本航电系统，而且可以适配成为针对其他嵌入式软件系统的测试环境.

本文通过对测试开发环境的框架的分析设计，提出了框架中的功能热点，并结合设计模式给出了它的作为冷点的部件之间的关系. 对于测试开发环境的应用开发来说，开发者只需要在此框架内，把上文定义的各个类实例化，把框架的功能块和相互之间的交互关联体现到具体应用中. 具体适配嵌入式软件测试开发环境的做法是：修改框架的热点、增加新的子类或者实现具体的接口来适应不同的嵌入式目标软件. 这种做法节省了测试环境的开发和准备时间，同时提高了测试的效率. 从框架研究的本身来看，本框架的研究并没有到此结束，还需要对本领域有兴趣的人员的积极参与、不断改进和优化，使框架实现最佳的设计上的复用.

参考文献 (**References**)

[1] 刘 斌. 嵌入式软件可靠性仿真测试环境的研究 [D]. 北京：北京航空航天大学工程系统工程系，2000
Liu Bin. Study of embedded software reliability simulater on testing environment [D]. Beijing: Dept of Systems Engineering, Beijing University of Aeronautics and Astronautics, 2000 (in Chinese)

[2] 万建成，卢 雷. 软件体系结构的原理、组成与应用 [M]. 北京：科学出版社，2002
Wan Jiancheng, Lu Lei. Principle, structure and practical of software architecture [M]. Beijing: Science Press, 2002 (in Chinese)

[3] Tsai W T, Na Y, Paul R, *et al*. Adaptive scenario-based object-oriented test frameworks for testing embedded systems [A]. In: Proceedings of 26th Annual International on Computer Software and Applications [C], 2002. 321~326

[4] 伽 马，赫尔姆. 设计模式可复用面向对象软件的基础 [M]. 李英军等译. 北京：机械工业出版社，2000
Gamma E, Helm R. Design patterns elements of reusable object oriented software [M]. Translated by Li Yingjun, *et al*. Beijing: China Machine Press, 2000 (in Chinese)

[5] 方塔纳，朴 利，卢姆普. 框架体系结构的 UML 档案 [M]. 张 罡等译. 北京：机械工业出版社，2003
Fontoura M, Pree W, Rumpe B. The UML Profile for framework architectures [M]. Translated by Zhang Gang, *et al*. Beijing: China Machine Press, 2003 (in Chinese)