基于软件体系结构的测试及其工具研究*)

叶俊民[1,2,3] 王振宇[1,3] 陈 利[2,3] 赵 恒[1,3]

(中国船舶重工集团公司第七研究院709研究所 武汉430074)[1]

(华中师范大学计算机系 武汉430079)[2] (武汉大学计算机软件工程国家重点实验室 武汉430079)[3]

Research on Testing and its Tools Based on Software Architecture

YE Jun-Min[1,2,3] WANG Zhen-Yu[1,3] CHEN Li[2,3] ZHAO Heng[1,3]

(China Shipbuilding Industry Corporation 7th Academy 709 Reaearch Institute, Wuhan 430074)[1]

(Department of Computer Center-China Normal University, Wuhan 430079)[2]

(State Key Laboratory for Computer Software Engineering, Wuhan University, Wuhan 430079)[3]

Abstract In this paper, we discuss the objective and the significance of software testing on the basis of software architecture; we propose the content of testing planing of software architecture and testing criteria; on the level of architecture, through comparison with traditional testing tools, we present testing tools in integration environment and analyze the roles of these tools. On the basis of this, we further suggest the structure of a integration testing environment.

Keywords Software architecture, Architecture testing, Testing planning, Software testing tool

1 引言

软件体系结构层测试的目的是找出体系结构的错误和缺陷，这与传统测试有很大的不同。SA 测试分为体系结构的静态测试(即体系结构分析)和体系结构的动态测试(即体系结构模拟)两个方面。前者是对体系结构的静态行为特征进行分析，如各类一致性分析等；后者是对体系结构的动态行为特征进行模拟。

SA 中的结构模式提供了一个设计和分析大型软件系统的自然框架，也为可重用的构件提供了组合成系统的规则。通过对软件体系结构的描述，使得复杂系统的测试与分析变得容易。体系结构规格说明的形式化符号系统的出现则为开发基于体系结构的测试与分析技术和相关工具提供了一种正确的工作基础。但这些以构件为基础而构成的软件，给测试以构件为基础的系统带来了特殊的挑战。迄今为止，人们很少去为以构件为基础的软件的测试做实际工作，其原因是在体系结构层上或者根本就没有代码可测，或者只有二进制代码。

在软件体系结构层上的测试与分析很有意义。最明显的是，使得研究人员将注意力集中在体系结构本身的缺陷研究与改进上，使得在软件生命周期的早期能够对这类缺陷进行测试与检查，而不是等到了实现阶段和系统集成阶段才发现这些缺陷。其次是，由于同一类 SA 经常被重用来开发同类型的、但具有多样性的软件系统，因此在 SA 层次上所付出的代价，都可以通过重用该 SA 得到偿还，最明显的作用是使得测试费用降低。

在体系结构测试方面，主要工作现状是：D. LeMetayer 对体系结构测试研究之后，提出了如下问题[1]：体系结构中构件、连接器是否是测试与分析中所必需的概念？它们的层次是否太低？是否足够抽象？M. Wermelinger[2]指出化学抽象机(Chemical Abstract Machine，记为 CHAM)是一种有效形式化描述技术，这种技术对静态、动态体系结构的分析与测试带来了方便。J. Stafford 指出[3]在高抽象层进行形式化描述软件系统，通过这一手段，可为静态分析提供基础，而相关的分析结论也可被作为软件优化、排错和测试的基础。Bertolino 等人[4]提出了利用化学抽象机来形式化描述软件体系结构，由此可导出系统的集成测试计划，以指导测试。C. Ghezzi[1]讨论了 COTS 测试，指出传统的测试技术无法保证基于构件库和用 COTS 构成的软件系统的可靠性。M. J. Harrold[5]研究了基于体系结构的进化系统回归测试问题，指出 SA 用于回归测试方面可能会比那些只注重开发测试的技术更能影响软件的费用。Charles Liu[1]研究了带回朔器的软构件，提出了相应的测试方法。M. Young[6]指出测试复杂结构的一致性关系是必不可少的。文[7～9]均注意到了软件体系结构测试方面的相关问题。

2 SA 测试研究的内容、测试准则与计划

所谓体系结构测试主要就是产生测试计划，以便对体系结构设计结论进行分析和模拟。一般地，一个能产生更好测试计划的体系结构将更有可测性。通过对体系结构测试，可以提供一种对软件生命周期早期设计结论的确认。此外，体系结构测试还是发现体系结构缺陷的有效途径。

2.1 测试内容

文[10]提出了体系结构测试应该研究以下内容：(1)体系结构可测试性。(2)基于体系结构的测试大纲。(3)体系结构仿真。(4)体系结构一致性测试。(5)体系结构发现。(6)体系结构分片。具体如下：

*)本课题得到国防科技预研基金(413150601)和武汉大学软件工程国家重点实验室开放基金(SKL(4)020)资助。叶俊民 副教授，博士研究生，主要研究方向为软件体系结构及其测试。王振宇 研究员，博士生导师，主要研究方向为新型软件测试技术。陈 利 副教授。赵 恒 博士研究生。

体系结构可测试性 系统的可测试性应尽早评估，以便能够产生出更可测试的系统。就特定的测试策略而言，某些体系结构会比另一些体系结构难以测试；因此，在进入开发工作之前重建体系结构会更好些。

基于体系结构的测试大纲 可将基于体系结构的测试计划建立成一个分层大纲，它对于测试相关的系统是有用的。例如，一个机构可以为一个体系结构风格、或一个领域专用体系结构、或一个未用体系结构，对该机构建立一个测试计划，每一个更专用的测试计划都是前一个计划的求精。

体系结构仿真 基于体系结构的测试计划可被用于评估系统体系结构。在体系结构仿真器上运行测试用例，可产生实际踪迹、与事件有关的因果历史和它们的时序。这些踪迹可以揭示动态行为中的缺陷，而这些用静态分析技术是难以发现的。

体系结构的一致性测试 重要的是评价体系结构描述与实现之间的一致性，为断定体系结构的一致性，我们不仅必须识别出体系结构应被覆盖的方面，还要规定实现的预期行为。这需要建立一个测试 oracle，它是一个将执行结果与预期结果进行比较和检查的机构。

体系结构发现 对已实现系统的测试对于发现系统的体系结构风格是有用的。这时，测试计划不是基于体系结构规格说明，而是基于系统特性测试或基于实现的集成测试。在体系结构的构件测试中所收集的踪迹可再被映射到体系结构风格上，以利用逆向工程到一个体系结构描述。

体系结构分片 体系结构的分析和集成测试二者都能揭示体系结构缺陷。为进一步诊断这些缺陷，了解潜在的、能影响有缺陷构件的那些构件是会有帮助的。此外，如果要修改一个构件，应当知道这种修改可能影响到的一些构件。这些信息可通过分片技术收集到。

除此以外，我们还要补充以下体系结构测试应该注意的内容。

2.1.1 *功能测试* 是找出软件体系结构所反映出的整体功能特征和其所对应的需求规格说明之间的不一致。由于一些构件没有实现，这会给功能测试带来困难，为此需要对体系结构层的功能测试加以定义，即体系结构层的功能测试是对全体构件以及连接件的接口的规格说明进行测试，也就是考察构件和连接件的接口所能够提供的服务/功能以及所需要的服务/功能是否不满足需求规格说明的要求。具体包括构件的内部行为以及内部变换关系；连接件的内部行为以及内部变换关系；构件与连接件之间的关系、连接件与构件之间的关系。测试手段一般都采用黑盒技术，或者黑盒技术与白盒技术（对设计的关键部分）相结合的方式。

2.1.2 *配置测试* 是最难进行的测试。配置测试的目的是测试系统中的构件和连接件之间的交互关系的不一致性。因此，配置测试实际上是针对体系结构中的动态行为方面。

为了寻找体系结构配置与其目标的差别，注意力应放在检查需求规格说明到设计的转化过程中信息可能产生的错误和遗漏上。就错误的产生及其数量而言，在整个体系结构设计生成这一步是最容易出错的，所以配置测试是一个关键的测试。在分析目标的基础上，用仿真器或者模拟器对系统的动态性进行模拟，从而得出配置测试用例。配置测试涉及以下内容：强度测试、安全性测试、性能测试和可靠性测试等。

2.2 测试准则

测试准则是测试计划研究中的一项重要任务，也是测试工具的基础。我们认为在软件体系结构层的测试准则应该包括以下内容：每一个构件本身的接口测试覆盖，包括流入、流出该构件的数据类型一致性分析；每一个连接件本身的接口测试覆盖，包括流入、流出该连接件的数据流（控制流）的匹配性分析；SA 中所有构件之间的相关数据流和控制流分析；构件-连接件组之间的连接关系分析；以及在体系结构匹配支持之下的系统集成测试路径分析与模拟。

2.3 体系结构测试计划的内容

通过对软件工程文档的分析可知：测试计划构成的核心部分是测试用例。测试用例主要由预期输入、预期输出、实际输出以及预期输出与实际输出的比较结果四个部分组成。测试计划是导致一个测试过程是否成功的决定因素。

对软件体系结构而言，测试的概念明显与代码级测试不同，从这层意义上讲，体系结构级的测试偏重对体系结构设计结论进行分析（即所谓的静态测试）与模拟（即所谓的动态测试）。因此，体系结构测试应该考虑构件测试、连接件测试和配置测试，而体系结构测试计划包括以下内容：目的、完成的标准、时间的安排、责任、测试用例库及其标准化、测试工具、硬件配置、组装方式、记录手段、纠错手段、回归测试。正如我们已经提到的：在这11项内容中，测试用例的设计是最为关键和核心的。

2.4 体系结构测试完成的标准

在测试计划中的另一项重要任务是判定测试何时结束，最常见的测试结束标准是：如果测试超过了预定的时间表时，则停止测试；或如果执行了所有测试用例但没有发现错误时，则停止测试。

3 主要的测试工具研究

完成软件体系结构的测试离不开测试工具，同时软件测试工具的开发是软件测试领域的一个重要任务。从单元测试的领域来看，已经有了相当成熟的商品工具，如：数据抓取工具、静态测量工具、动态测量工具、模拟工具、测试管理工具等。但总的来说，实时、分布式软件的测试技术和工具仍不够成熟，而涉及到软件体系结构的测试工具则更少，且存在大量问题需要研究。因此针对体系结构的测试工具远未达到适用化的程度。以下的工作是依据软件体系结构测试的内容和在对传统测试工具[11,12]进行类比的基础之上得出的相关结论。从总体上讲，测试工具应能够满足各种测试活动的需要，实现多种测试功能，完成测试人员规定的测试任务或测试步骤，实现测试目标。

开发有效的测试工具是成功进行软件体系结构各项测试的基础。测试工具根据其功能大致上可以分为两类，即静态分析工具和动态模拟工具。前者主要有：构件/连接件及其配置驱动工具、静态流分析工具、测试覆盖监控系统、构件/连接件及其配置正确性证明系统。后者主要有：符号执行系统、测试数据生成系统、环境模拟系统、执行路径分析工具。此外，为了使测试工具能够正常运行，还需要有提供保证的测试支撑环境。

3.1 静态分析工具

静态分析工具的主要构成是静态分析器。静态分析不需要对待测试的软件体系结构进行仿真和模拟，分析被测体系结构的特征，它主要通过扫描体系结构描述语言（Architecture Description Language，记为 ADL）源程序，对 ADL 描述的各种流（如各种数据流等）进行分析，也就是对于被测 ADL

描述，在不执行(模拟)ADL 描述的前提下，找出某些错误或者某些不可靠的结构，同时可以研究 ADL 的静态特征，例如死锁活锁等，以了解体系结构设计的性能。

静态分析工具主要完成的功能如下：第一，用来检查体系结构(ADL 描述)在使用了统一的记号系统方面是否与规格说明的语义描述一致。即检查构件接口或者连接件接口的一致性，以分析这些接口连接时的端口个数、类型是否一致，输入输出端口的定义与使用是否匹配；第二，检查构件和连接件中涉及的全部变元是否都有定义；第三，检查体系结构是否遵循了相关的设计规则。

静态分析器的构成通常包括4个部分，即 ADL 语言的预处理器、相关数据库、错误分析器及报告生成器。预处理器把 ADL 的词法分析和语法分析结合在一起，以识别各种语言成分。数据库中存放有体系结构的各种测试(模拟)过程和分析过程中要使用的所有表格。

要注意的是，所有静态工具都涉及可判定性问题的限制。

3.2 动态模拟(分析)工具

动态模拟工具通过选择合适的测试用例，为直接“执行”(即模拟)被测试 ADL 描述提供支持，通过将体系结构的模拟行为与设计的预期行为进行比较，验证体系结构设计的正确性和一致性，并找出体系结构设计过程中的错误。动态测试分为结构测试与功能测试。它应该完成的主要功能有：完成覆盖分析，以确定测试运行的充分性(覆盖分析器)；完成模拟过程中的跟踪与历史记录(跟踪器)等。下面将要讨论的工具主要包括：符号演算器、测试用例生成器、构件插装工具等。

3.2.1 *模拟器* 用来对动态模拟的踪迹进行解释，支持生成各种各样的图以及特定子图，同时完成约束演算，最终支持测试用例的产生。

模拟器的主要操作有语法分析、路径选择、分析符号执行路径、化简约束以及不等式求解。模拟器最主要的功能是支持分析体系结构的各种动态特征、生成测试计划和测试用例以及完成约束演算。

模拟器的工作原理是通过语法分析建立 ADL 描述的内部图，并按指定路径进行符号计算/反应；至于哪一条路径要进行分析，可以由用户在判定的基础上进行选择路径；对某一路径完成演算/反应后，约束可以进行自动化简；每一个约束经过化简器化简后，就被送入不等式求解器求解，以检查一致性，如果不一致，则该路径就不可行，如果一致的话，路径的符号会进行/反应下去，其中不等式求解器的求解技术有线性规划等方法。

3.2.2 *测试用例生成器* 是在构件/连接件以及配置测试中帮助测试者生成测试数据的工具。开发这类工具的目的在于减轻生成大量测试数据中所付出的劳动，在自动生成测试数据时还可以避免设计人员对一部分测试数据所抱有的偏见。

测试数据生成器包括：路径测试生成器、数据规格说明系统和随机数据生成器。

3.2.3 *构件/连接件以及配置插装工具* 其利用插装在 ADL 规格描述中的监控语句来收集模拟过程中的信息，以达到揭示软件体系结构描述规格说明内部行为和特征。构件/连接件以及配置插装主要是用来进行覆盖分析、监控和跟踪相关信息流的变化以及查找信息流的异常，以帮助排错。

3.2.4 *测试比较工具* 测试比较机制描述了测试执行中可以接受的行为。如果缺乏比较机制对测试结果进行判断，测试就无法完成其揭示错误和缺陷的目标，也无法对正确的系统行为进行判定，这一任务通常无法用手工完成，即使可能也会代价太高。测试比较机制可以从 SA 规格说明以及相关的测试准则中导出，并在测试过程中用于实际的指导工作。

3.3 测试支持工具

测试支持工具提供测试环境，完成测试方法的实现。主要包括测试驱动器和比较器等。

测试驱动器为构件/连接件以及配置的测试和模拟提供了一个环境，比如为了测试某一个构件，需要提供驱动构件和存根构件，驱动构件向被测试构件发送消息、服务请求或者传输数据。存根构件接受消息或者数据，回应服务请求。对连接件以及配置也一样。

测试比较器可以对比测试用例与测试结果，以分析两者的差别。比较器主要是用来保证已经作出的修改只是在特定部分完成，而对其它构件和连接件及其配置没有影响。利用比较器有助于缩小修改后的应该进行检查的范围。此外，比较器还可以对特定的比较范围进行比较，以提高相应的检查效率。

上述工具的讨论都是分开的，现将它们放在一个测试集成环境中，可得到以下结构。

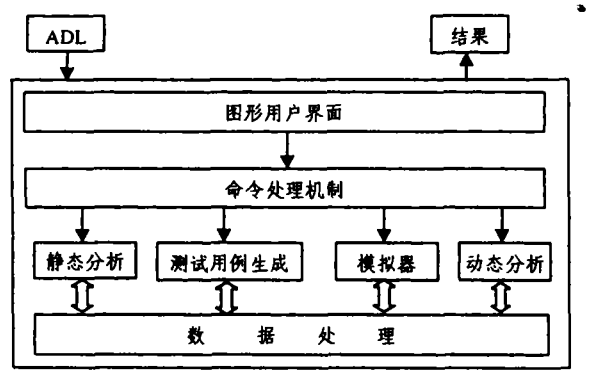

图1 一种测试集成环境原型

结论 文[4]提出了基于 CHAM 导出测试计划的方法。其基本原理是，假定在一些软件体系结构描述语言中存在一种 ADL 描述，且从这一 ADL 描述中可以导出 LTS(Labelled Transition System，记为 LTS)，该 LTS 表示了该 SA 的动态性。在此 SA 动态性的环境下，所导出的 LTS 中的节点表示了构件中的单一状态信息，而弧标号表示了系统状态的迁移信息。这样，通过标号迁移系统 LTS 来对软件体系结构的动态性方面进行建模，并可在 LTS 基础上进行抽象处理，以产生抽象标号迁移系统(Abstract Labelled Transition Systems，记为 ALTSs)。ALTSs 有趋向性地关注了待测系统的相关特性，并将我们不感兴趣的内容抽象掉。当然，在文[4]中还存在着许多待研究的问题，比如：如何从 CHAM 中导出 LTS；如何从 LTS 导出 ALTSs；如何对 ALTSs 最小化；如何形式化 obs 观察函数；如何从 ALTSs 中导出测试用例等，以及如何表示实时 LTS；如何导出实时 LTS；等等。这些内容将是我们进一步研究的方向。

参 考 文 献

1 Richardson D, Inverardi P. ROSATEA: *International Workshop on the Role of Software Architecture in Analysis E(and)Testing*. ACM SIGSOF Software Engineering Notes, 1999, 24(4): 33～42

2 Wermelinger M. Specification, Testing and Analysis of (Dynamic)

Software Architecture with the Chemical Abstract Machine. http://www.ics.uci.edu/~djr/rosatea/attendees.html. pp. 1~4
3 Stafford J A, et al. Chaining: A Software Architecture Dependence Analysis Technique: [Technical Report CU-CS-845-97]. University of Colorado, Sep. 1997. 1~12
4 Bertolino A, et al. Deriving Test Plans from Architectural Descriptions. In: ACM Proc. Int. Conf. On Software Engineering (ICSE2001), June 2000. 220~229
5 Harrold M J. Architecture-Based Regression Testing of Evolving System. http://www.ics.uci.edu/~djr/rosatea/attendees.html. pp. 1~5
6 Young M. Testing Complex Architectural Conformance Relations. http://www.ics.uci.edu/~djr/rosatea/attendees.html. pp. 1~4
7 Anderson S O, et al. Type System, Software Architecture and Testing. http://www.ics.uci.edu/~djr/rosatea/attendees.html. pp: 1~4
8 Bertolino A, et al. Reaction Graphs for the Testing and Analysis of Software Architecture. http://www.ics.uci.edu/~djr/rosatea/attendees.html. pp. 1~6
9 Rosenblum D S. Challenges in Exploiting Architectural Models For Software Testing. http://www.ics.uci.edu/~djr/rosatea/attendees.html. pp. 1~4
10 Richardson D J, Stafford J A, Wolf A L. A Formal Approach to Architecture-Based Software Testing. National Science Foundation Research Proposal. pp. 1~24
11 郑人杰. 计算机软件测试技术. 清华大学出版社, 1992
12 周芝英, 等. 计算机软件测试技巧. 清华大学出版社, 1985

(上接第140页)

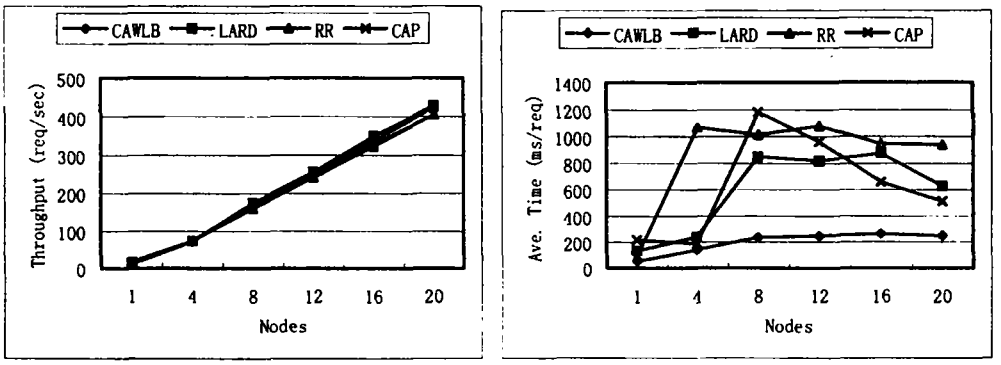

图4 电子商务型站点仿真结果

从仿真结果可以看出 LARD 与 CAWLB 算法对发布型站点在吞吐率和平均响应时间上有明显优势；对事务型或电子商务型站点，各种算法的吞吐率差异较小，但平均响应时间指标上 CAWLB 和 CAP 算法存在明显优势。综合以上仿真结果可知，CAWLB 对不同类型的站点，其吞吐率和平均响应时间均有优势。

总结 本文在分析了基于内容分配的 Web 集群算法 LARD 和 CAP 之后，提出了一种改进的负载分配算法 CAWLB。该算法根据请求内容分配后端服务器，可以获得较高的 cache 命中率，同时，根据不同请求类别的负载估算，计算后台服务器的负载量，为请求再分配提供了较准确的依据。另外，算法具有对服务器处理能力异构情况的适应性。仿真实验表明，该算法与其它相关算法相比，在系统吞吐率和平均响应时间等指标上存在优势，算法具有较好的实用价值。在实际应用中，不同负载的权重可以通过测试或分析 Web 日志文件获得。并且，可以考虑通过 ASIC 实现算法获得更快的执行速度。

参 考 文 献

1 Vivek S, Mohit A. Locality-Aware Request Distribution in Cluster-based Network Servers. In: Proc. of ASPLOS-VIII, ACM SIG-PLAN, 1998. 205~216
2 Emiliano C, Michele C. A Client-Aware Dispatching Algorithm for Web Clusters Providing Multiple Services. In: Proc. of 10th Int'l World Wide Web Conf. Hong Kong, May 2001
3 Trevor S, Steve G, Byrav R. Scalable Web Server Clustering Technologies. IEEE Network, 2000, 14(3): 38~45
4 Valeria C, Michele C, Philip S Y. Dynamic Load Balancing on Web-Server Systems. IEEE Internet Computing, 1999, 3(3): 28~39
5 Mohit A, Darren S, Peter D, Willy Z. Scalable Content-aware Request Distribution in Cluster-based Network Servers. In: Proc. of the 2000 Annual Usenix Technical Conf. San Diego, CA, June 2000
6 Ludmila C, Magnus K. Scalable Web Server Cluster Design with Workload-Aware Request Distribution Strategy WARD. In: Proc. of the 3rd Intl. Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems, Milpitas, CA USA, June 2001
7 Ludmila C. FLEX: Load Balancing and Management Strategy for Scalable Web Hosting Service: [HP Labs Technical Reports, HPL-1999-64R1, 991117]
8 Kai S, Tao Y. Cluster Load Balancing for Fine-grain Network Services. Accepted for publication at International Parallel and Distributed Processing Symposium (IPDPS'02), Fort Lauderdale FL, April 2002
9 Yong M, Rassul A. Comparison of Load Balancing Strategies on Cluster-based Web Servers. http://www.comp.nus.edu.sg
10 The Workload for the SPECweb96 Benchmark. http://www.specbench.org/osg/web96/workload.html
11 Srisuresh, Gan. Load Sharing using IP Network Address Translation (LSNAT), RFC2391, Aug. 1998
12 Brisco T. DNS Support for Load Balancing. RFC 1794, April 1995