一种基于模型的测试充分性评估方法

张 瑶[1,2] 白晓颖[1] 张任伟[3] 陆 皓[1]
(清华大学计算机科学与技术系 北京 100084)[1] (解放军总医院第一附属医院计算机中心 北京 100048)[2]
(北京大学软件与微电子学院 北京 102600)[3]

摘 要 测试充分性评估通常采用覆盖率的方法来评估测试对软件特征的覆盖充分程度。如今,传统的充分性评估方法难以满足复杂软件的测试评估需求。首先,代码覆盖准则难以准确验证软件需求;其次,软件测试还需考虑软件不同特征对系统测试充分性的不同影响。对此,提出一种基于接口的建模方法和基于该模型的综合覆盖充分性评估方法。该方法根据软件接口说明,对系统功能进行特征抽取、建模,并对接口模型的测试用例进行不同层级的充分性评估,对评估结果进行归一化处理,得到系统的综合测试充分性。通过案例表明,这种评估方法能够反映功能的测试充分性,对测试用例的设计和优化有一定指导意义。

关键词 测试充分性,基于模型的测试,测试覆盖率
中图法分类号 TP311 **文献标识码** A

Model-based Approach for Software Test Adequacy Analysis

ZHANG Yao[1,2] BAI Xiao-ying[1] ZHANG Ren-wei[3] LU Hao[1]
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)[1]
(Computer Department, No. 1 Hospital Affiliated to General Hospital of PLA, Beijing 100048, China)[2]
(School of Software and Microelectronics, Peking University, Beijing 102600, China)[3]

Abstract Test adequacy analysis usually uses coverage criteria to evaluate test design with respect to specific software characteristics. Conventional adequacy methods have following problems to address test evaluation of large software systems. First, code-based coverage cannot ensure sufficient verification and validation of software requirements. Secondly, software testing adequacy needs to take into consideration the contribution of different features. Important features deserve more test effort. The paper proposed a model-based approach for test adequacy analysis. An interface model was defined, representing executable software requirements for software components. Coverage of test case design was analyzed at two levels including service and service-compositions. Adequacy was calculated as weighted sum of coverage on various software features. Experiments were exercised to illustrate the proposed approach.

Keywords Test adequacy, Model-based testing, Test coverage

1 引言

测试充分性是软件系统是否被充分测试的度量[1],覆盖率是用于定量度量测试充分性的一种重要方法。通过分析目前主流的测试覆盖方法的实际应用,发现其存在以下不足:

1. 代码覆盖率主要用来衡量测试时程序代码的执行成分程度,不能准确反映软件系统的功能的覆盖充分性,仅仅用它来评估测试用例集的优劣不全面。

2. 在白盒测试中,通常采用代码覆盖率的方法来评估测试充分程度。黑盒测试广泛应用于系统测试和集成测试中,但缺少行之有效的覆盖率评估方法[2]。

3. 基于功能模型的软件测试是一种黑盒测试的典型方法,但目前缺乏有效的系统建模方法来描述系统功能特征,相应的评估功能模型覆盖率的方法也不多见。

4. 现有的基于模型的软件测试充分性评估多是从不同角度局部地评价测试的充分性[3],如语句、分支、路径等,缺乏从整体上评估充分性的综合评估方法。

针对上述问题,本文首先介绍了传统软件测试的概念,并列出了常见的代码覆盖率和模型覆盖准则。在此基础上,提出一种基于模型的综合测试充分性评估方法,它针对基于接口的软件系统做黑盒测试。首先进行模型特征提取,然后根据该模型进行不同层级的测试充分性评估,再以综合测试充分性的计算方法来评价系统的功能测试充分程度。最后通过案例分析,对比模型充分性与代码覆盖率,论证得出本文提出的测试充分性计算方法能够较为快速准确地评估测试用例集对系统功能的测试充分程度,并反映出不同测试用例集的优劣。

2 测试充分性评估

2.1 基本概念

软件测试是根据软件开发阶段的各种说明文档,设计测

到稿日期:2012-04-05 返修日期:2012-07-05 本文受航空科学基金自主项目(20091958005)资助。
张 瑶 女,硕士生,主要研究方向为软件工程和软件测试,E-mail:zypublic@163.com;白晓颖 女,副教授,主要研究方向为软件工程、软件测试、软件项目管理等;张任伟 男,硕士生,主要研究方向为软件工程、软件测试;陆 皓 男,硕士生,主要研究方向为软件工程、软件测试。

试用例，使用手工或自动的方式，按照测试方案和流程，使用这些测试用例运行软件系统，以检验软件系统是否满足预期需求的过程，通常可分为白盒测试和黑盒测试[4]。

测试充分性评估是对软件测试的充分程度的一种度量，通常用测试覆盖率来定量表示。根据软件测试的分类，测试充分性评估可以分为基于代码的测试充分性评估和基于功能的测试充分性评估。

基于代码的充分性评估适用于白盒测试，可评估程度代码的完备程度。在测试过程中，通过观测程序的语句、分支、路径等的执行情况，来评估测试覆盖的充分程度。

基于功能的充分性评估适用于黑盒测试，可评估软件系统对于功能需求的实现程度，基于功能模型的充分性评估是其中一种重要方法。

2.2 主要评估方法

软件的测试覆盖率是针对特定的软件特征（如语句、分支、路径等），定量地度量软件测试的充分程度，可以用如下公式来表示：

$$\text{覆盖率}=\frac{\text{已测试执行的特定特征数目}}{\text{软件的特定特征总数}}\times 100\%$$

根据不同的测试策略，可以从多个角度来计算软件测试覆盖率，以评估不同方面的软件测试充分性[5]。常用的测试覆盖准则包括：代码覆盖准则、状态机覆盖准则、UML 状态及控制流覆盖准则。

一般来说，测试用例越多，测试充分性应该得到越大的提升，覆盖率也就越高。而实际测试中，通常会有一些特征没有被覆盖到。我们不可能要求测试对所有的覆盖率指标都进行评估，但只考虑一种覆盖率标准也是不恰当的，通常需要有针对性地选用几种重要的测试指标，以取得测试用例集的大小与测试覆盖效果之间的平衡。使用少量的测试用例，达到一定程度的测试覆盖率是合理的，因此，需要测试覆盖准则来指导和控制测试用例集合的生成。下面介绍几种常用的软件测试覆盖方法。

2.2.1 常用的代码覆盖率

代码覆盖率是基于代码的测试评估的定量表述充分性的一种方法，常用的几种计算方法如下[6]：

1. 语句覆盖率（statement coverage）：在测试时运行被测试程序后，程序代码中运行到的可执行语句数量占所有可执行语句总数的比率。

2. 判定覆盖率（decision coverage）：在测试时运行被测试程序后，程序中所有判定语句的取真分支和取假分支被执行到的结果数量占所有判定结果总数的比率。

3. 路径覆盖率（path coverage）：在测试时运行被测试程序后，程序中所有被执行过的路径数占所有路径总数的比率。

2.2.2 典型的状态机覆盖准则

模型覆盖准则是针对元模型的特征定义的，用于评估测试用例集对模型的覆盖充分程度的一组规则，它可以应用于任何基于元模型的实例。模型覆盖准则应用于特定的测试模型，可以得到该模型的一组关于模型元素（状态、转移、事件等）或是模型元素组合的测试结果。状态机模型使用的典型的测试覆盖准则如下[7]：

1. 全状态覆盖（All-States）：对于状态机的每个状态，至少有一个测试用例包含这种状态。

2. 全转移覆盖（All-Transitions）：对于状态机的每种状态间的转移，至少有一个测试用例包含这种转移。

3. 全事件覆盖（All-Events）：对于每个触发状态转移的事件，至少有一个测试用例包含这个事件。

4. n 深度覆盖（Depth-n）：从初始状态出发，对于任意不大于长度 n 的运行序列，至少存在一个测试用例包含以这个运行序列为子集的序列。

5. 所有 n 深度转移覆盖（All-n-Transitions）：从任意一个状态 s 出发，对于每个不超过长度 n 的运行序列，至少有一个测试用例包含以这个运行序列为子集的序列。所有 0 深度转移覆盖即是全状态覆盖。

6. 全路径覆盖（All-Paths）：测试用例集中都应该包含状态机中所有可能的状态转移序列。这种覆盖准则无法实现。

2.2.3 UML 状态机控制流覆盖准则

UML 状态机的状态转移通常是由布尔操作符（and，or，not）连接的原子条件组成的，基于控制流状态转移条件的覆盖准则定义如下[8]：

1. 判定覆盖 DC：从任意状态 s_i 出发，对于每个状态转移判定条件 c，至少存在 2 个测试用例，当到达状态 s_j 时，分别满足条件 c 为真、c 为假。

2. 条件覆盖 CC：对于每个判定条件 c，其原子条件要求满足 DC。

3. 条件/判定覆盖 C/DC：要求测试用例集同时满足 CC 和 DC。

4. 修正的条件/判定覆盖 MC/DC：除满足 C/DC 之外，还要求在每个判定中的每个原子条件都对判定结果的取值有独立的影响。

5. 多条件覆盖 MCC：要求测试用例集满足每个判定的任意原子条件的组合。

MCC 是最强的覆盖准则，在实际使用中，如果一个状态转移条件由 n 个原子条件组成，那么满足 MCC 的测试用例集至少需要 2^n 个测试用例[9]。

3 基于模型的覆盖率评估方法

3.1 基于接口模型的测试

软件模型是对软件行为、结构进行特征抽取和抽象，并用易于理解的方式对软件系统进行描述[10]。如今软件规模庞大、结构复杂，使得软件测试和评估都十分困难，基于模型的软件测试可以根据软件行为模型和结构模型自动生成测试用例，可以有效地提高测试效率和测试自动化程度[11]。基于模型的测试评估方法对模型测试的充分性进行评价，并能指导测试用例的设计和优化。

本文定义的基于接口模型的测试充分性评估方法，其基本方式是先抽取软件系统的基本特征进行抽象，建立特征模型；然后，结合特征模型来分析测试用例集，并测算测试用例集对模型各功能需求点的测试充分程度；最后，综合考虑各功能点对系统的影响，得到定量的综合充分性评估结果。根据评估结果，对测试用例的生成策略进行优化，以重新生成更优

的测试用例集合，循环生成、评估、优化这一过程，直到得到合适的测试用例集合，用以执行测试。该评估方法的框架如图1所示。

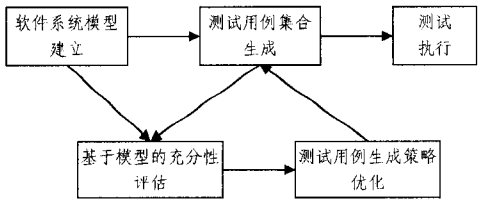

图1　基于模型的测试充分性评估方法框架

3.2　系统接口模型实例

3.2.1　服务接口描述

在定义具体的测试评估方法前，引入一个典型的接口服务作为例子，随后说明各种覆盖充分性的定义方法。机票预订系统FMS是个典型的接口服务的应用，下面以其中一个简单的接口服务GET_AIRLINE_ID为例对它进行描述（见图2）。

/* 服务GET_AIRLINE_INFO是根据航线ID，输出航线的详细信息。*/
Procedure GET_AIRLINE_ INFO
　　(AIRLINE_ID ;in String;
　　AIRLINE_INFO;out String
　　RETURN_CODE;out Integer) is
error
　当不存在航线ID等于AIRLINE_ID的数据时=>
　RETURN_CODE;=NOT_FOUND;
normal
　　AIRLINE _INFO;=AIRLINE_ID对应的航班信息
　　RETURN_CODE;=NO_ERROR;
end GET_AIRLINE_INFO;
该服务的返回值定义如下：
GET_AIRLINE_INFO

返回值	含义
NO_ERROR	成功执行
NOT_FOUND	AIRLINE_ID对应数据不存在

图2　系统FMS接口GET_AIRLINE_INFO描述

服务GET_AIRLINE_INFO的定义中，有一个输入参数AIRLINE_ID和两个输出参数AIRLINE _INFO、RETURN_CODE。服务描述中有正常和异常两种情况：当现有分区中不存在输入参数AIRLINE_ID指定的航线的数据时，认为异常，输出参数RETURN_CODE返回NOT_FOUND；否则，服务正常返回输入参数AIRLINE_ID指定的航线的详细信息，输出参数RETURN_CODE返回NO_ERROR。

本文基于标准服务接口，建立测试模型，并基于该模型的测试平台定义基于接口模型的系统综合测试充分性的分析方法和测试评估框架。

3.2.2　服务接口建模

模型驱动测试是自动化测试的一个有效途径和主要发展趋势。基于模型的测试是对待测软件系统（SUT）进行形式化的建模，通过建立一套设计测试用例的流程，对输入输出进行标准化处理，生成更可靠的测试用例。模型特征的抽取是整个方法的关键与核心，测试用例可以部分或完全地利用模型自动产生。

对该标准服务建模，将模型分为两大部分：数据（Data）部分、服务（Service）部分。数据部分用来描述数据类型、数据池等信息，服务部分用来描述服务的相关信息。

数据建模需要对每种数据创建数据池。数据池是测试数据的集合，每个数据类型或服务参数可以有一个或多个不同的数据池，每个数据池通常有若干数据分区，每个数据分区有若干条数据。数据池可以有多种分区方法，例如，对于字符串类型的参数AIRLINE_ID，可以定义valid、invalid两个数据池分区，valid分区中存放若干有效数值，invalid分区中存放若干非法数值。测试数据可以是有效数值，也可以是非法数值，测试数据的全集是无穷的，数据池只提供一部分的数值供测试使用。

对接口服务建模，主要分为服务关联、服务约束、服务行为、服务参数这4大部分。服务关联用于说明服务之间的关系，包括前置条件、后置条件，主要描述在服务之前、之后需要的环境，以及需要的参数传递关系信息。服务约束用于说明服务的各种约束条件，例如某服务要求在0.1s内执行完毕。服务行为有决策表和状态机两种描述方式，根据服务的说明文档的描述，记录服务的规则，比如服务的输入、输出参数的对应关系。服务参数包括服务的输入参数、输出参数，每个参数有其数据池。例如，对服务GET_AIRLINE_ID进行测试，其输入参数AIRLINE_ID从它的数据池valid分区中选取测试数值，从基于服务接口说明生成的决策表规则中，可以推测输出参数AIRLINE_INFO的取值应该在valid分区，RETURN_CODE的取值应为NO_ERROR。

服务及数据模型如图3所示。

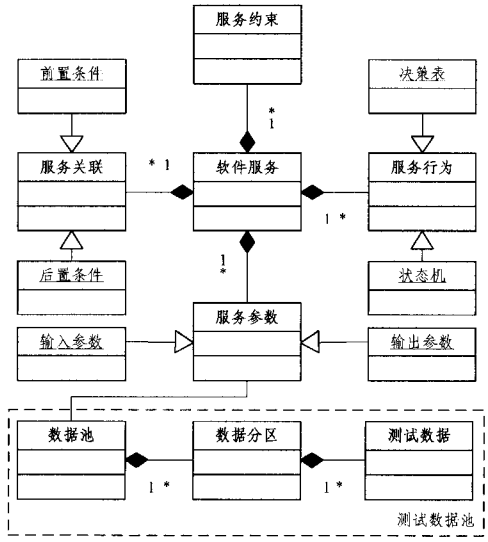

图3　服务及数据模型

按照上述服务接口的建模方法，用S表示接口服务，P表示服务参数，R表示服务行为，C表示服务约束C，W表示服务关联W，Q表示数据分区，则$S:=\langle P\{Q\},R,C,W\rangle$。

3.3　基于模型的测试充分性定义

本文将软件系统分为若干服务，这些服务集合构成软件系统提供的所有功能接口。在测试用例集T下，对软件系统

SUT 的服务集 S 进行黑盒测试，并评估软件系统的综合测试充分性。用 T 表示测试用例集。对某服务 $s_k \in S$，输入参数 $p_i \in P$，参数 p_i 的分区 $q_j \in Q$，输入输出组合关系 $r_m \in R$，服务间的组合关系 $w_n \in W$，定义测试充分性如下：

$$A = f(N, \alpha, \gamma, C, S)$$

式中，N 表示数量或次数(Number)，α 表示影响因子，γ 表示相对重要性因子，C 表示覆盖率(Coverage)，S 表示充分因子，A 表示测试充分性(Adequacy)。

3.3.1 服务参数的分区充分性

服务的参数包括输入参数、输出参数。参数的类型有简单类型和结构类型，简单类型包括 int，string，enum 等，在此将结构类型参数视为多个简单类型参数。每个参数有自己的数据池定义，根据测试数据的实际需求，将参数的测试数据进行分区。

服务 s_k 的参数 $p_i \in P(s_k)$，定义 p_i 的分区覆盖率如下：

$$C_{part}(p_i) = \frac{Part(T, p_i)}{Part(p_i)} \times 100\%$$

式中，$C_{part}(p_i)$ 为参数 p_i 的分区覆盖率，$Part(T, p_i)$ 为测试用例集 T 覆盖到的 p_i 的分区数，$Part(p_i)$ 为参数 p_i 的数据池分区总数。

然而，由于每个分区的重要程度不一样，定义参数 p_i 的分区 $q_j \in Q(p_i)$ 的影响因子为 $\alpha_{part}(q_j)$，定义 $\sum \alpha_{part} = 1$。根据参数的数据池分区的重要程度不同，在测试过程中，对应影响因子 α_{part} 越大的分区的数据在测试用例集中覆盖得越多，对测试的充分性贡献越高。因此，定义参数 p_i 的某个分区 q_j 的充分因子如下：

$$S_{part}(q_j) = \frac{N_{part}(q_j)/N_{part}(p_i)}{\alpha_{part}(q_j)}$$

规定：当 $S_{part}(q_j) \geqslant 1$ 时，$S_{part}(q_j) = 1$。

其中，$N_{part}(q_j)$ 为测试用例集 T 中的 p_i 对应分区 q_j 的测试数据的数量，则 $N_{part}(p_i)$ 为 T 中 p_i 的所有测试数据的总数量。

对于参数 p_i，它的分区充分因子是其所有分区的充分因子的加权平均，即：

$$S_{part}(p_i) = \sum \alpha_{part}(q_j) \times S_{part}(q_j)$$

最后，得到输入参数 p_i 的分区充分性为：

$$A_{part}(p_i) = C_{part}(p_i) \times S_{part}(p_i) \times 100\%$$

对于服务的接口，按照上述分区覆盖充分性的定义计算服务的参数充分性，当服务有多个输入参数时，用输入参数的分区充分性的算术平均作为服务的参数分区充分性。即，有 N_{input} 个输入参数的服务 s_k 的参数分区充分性为：

$$A_{part}(s_k) = \frac{1}{N_{input}} \times \sum A_{part}(p_i)$$

例如，将服务 GET_AIRLINE_INFO 的输入参数 AIRLINE_ID 的数据池分为 4 个分区 A, B, C, D，它们的影响因子依次是 0.4，0.2，0.3，0.1，即 $\alpha_A = 0.4$，$\alpha_B = 0.2$，$\alpha_C = 0.3$，$\alpha_D = 0.1$。测试用例集 T 中的测试数据共 100 个，即 $N_{part}(p_i) = 100$。分别从数据池分区 A, B, C, D 中选取数据的数量依次是 20，30，50，0，即 $N_{part}(q_A) = 20$，$N_{part}(q_B) = 30$，$N_{part}(q_C) = 50$，$N_{part}(q_D) = 0$。

以 s 来代表服务 GET_AIRLINE_INFO，p 来代表其参数 AIRLINE_ID，计算测试用例集 T 对服务 s 的参数 p 的分区充分性如下：

首先，对于 p 的 4 个分区，由于 T 中对分区 D 的测试数值数量为 0，即 T 覆盖到的 p 的分区只有 A, B, C 这 3 个，因此，p 的分区覆盖率为：

$$C_{part}(p) = \frac{3}{4} \times 100\% = 75\%$$

然后，计算参数 p 的各个分区的充分因子如下：

$$S_{part}(q_A) = \frac{20/100}{0.4} = 0.5, \frac{20}{100} < 0.4$$

$$S_{part}(q_B) = 1, \frac{30}{100} > 0.2$$

$$S_{part}(q_C) = 1, \frac{50}{100} > 0.3$$

$$S_{part}(q_D) = \frac{0/100}{0.1} = 0, \frac{0}{100} < 0.1$$

考虑到各分区的影响因子，参数 p 的分区充分因子为：

$$S_{part}(p) = 0.4 \times 0.5 + 0.2 \times 1 + 0.3 \times 1 + 0.1 \times 0 = 0.7$$

最后，得到参数 p 的分区充分性为：

$$A_{part}(p) = 75\% \times 0.7 \times 100\% = 52.5\%$$

即测试用例集 T 对服务 GET_PROCESS_ID 的输入参数 AIRLINE_ID 的分区充分性为 52.5%。

由于服务 GET_AIRLINE_INFO 只有 AIRLINE_ID 这一个输入参数，因此服务参数的分区充分性为：

$$A_{part}(s) = A_{part}(p) = 52.5\%$$

3.3.2 服务输入输出组合的规则充分性

服务接口的说明中给服务的输入与输出定义了具体的对应关系的规则。按照规则，对输入、输出参数进行分区，并采用决策表来描述这组规则。

服务 s_k 共有 $Rule(s_k)$ 条决策表规则，测试用例集测试到的输入、输出参数组合规则的数目为 $Rule(T, s_k)$，即测试到了 $Rule(T, s_k)$ 条决策表规则，则服务的输入输出组合覆盖率为：

$$C_{rule}(s_k) = \frac{Rule(T, s_k)}{Rule(s_k)} \times 100\%$$

然而，由于每条规则的重要程度不一样，定义服务 s_k 的规则 r_m 的影响因子为 $\alpha_{rule}(r_m)$，定义 $\sum \alpha_{rule} = 1$。在测试过程中，对应影响因子 α_{rule} 越大的规则的数据在 T 中覆盖得越多，对测试的充分性贡献越高。因此，定义服务 s_k 的规则 r_m 的充分因子如下：

$$S_{rule}(r_m) = \frac{N_{rule}(r_m)/N_{rule}(s_k)}{\alpha_{rule}(r_m)}$$

规定：当 $S_{rule}(s_k) \geqslant 1$ 时，$S_{rule}(s_k) = 1$。

其中，$N_{rule}(r_m)$ 表示测试用例集 T 对服务 s_k 的规则 r_m 所测到的次数，即规则 r_m 对应的所有输入参数 p_i 均处于 r_m 所对应的分区中的次数。$N_{rule}(s_k)$ 表示服务 s_k 的所有规则被测试用例集 T 测试到的总次数。

对所有的输入输出组合，它的规则充分因子为所有测试执行到的决策表规则的充分因子的加权平均，即：

$$S_{rule}(s_k) = \sum \alpha_{rule}(r_m) \times S_{rule}(r_m)$$

最后，得到服务 s_k 的输入输出组合充分性为：

$$A_{rule}(s_k) = C_{rule}(s_k) \times S_{rule}(s_k) \times 100\%$$

根据测试的需要，对服务 s_k 的参数的分区充分性赋予一个相对重要性因子 $\gamma_{part}\in[0,1]$，输入输出组合的规则充分性的相对重要性因子 $\gamma_{rule}\in[0,1]$，规定 $\gamma_{part}+\gamma_{rule}=1$。得到服务 s_k 的测试充分性：

$$A_{ser}(s_k)=\gamma_{part}\times A_{part}(s_k)+\gamma_{rule}\times A_{rule}(s_k)$$

对于系统中的多个服务，用所有服务的测试充分性的算术平均作为系统的服务充分性。即，有 N_{ser} 个服务的系统，其服务充分性为：

$$A_{ser}(W)=\frac{1}{N_{ser}}\times\sum A_{ser}(s_k)$$

3.3.3 服务组合的路径充分性

在实际应用中，将一个软件系统按照功能模块和运行规则划分为多个服务，这些服务之间的调用通常形成一个执行网络。为了准确描述该网络的覆盖情况，将服务的前置条件、后置条件、服务约束引入服务的运行序列。

根据服务的前置条件、后置条件、服务约束，构建服务间的执行网络，各服务作为该网络的节点，连接节点间的边即为服务间的参数传递及约束关系。考虑有限的循环，通过一定的算法将该执行序列网络分解成若干条由服务构成的执行序列。

系统的服务间的执行网络模型 W 被分解为若干执行序列 $w_n\in W$，测试用例集 T 覆盖到的路径数以 $Path(T,W)$ 来表示，系统的执行网络模型 W 的总路径数以 $Path(W)$ 来表示。定义系统的路径覆盖率为：

$$C_{path}(W)=\frac{Path(T,W)}{Path(W)}\times 100\%$$

然而，由于每条路径的重要程度不一样，定义路径 w_n 的影响因子为 $\alpha_{path}(w_n)$，定义 $\sum\alpha_{path}=1$。在测试过程中，对应影响因子 α_{path} 越大的路径的数据在 T 中覆盖得越多，对测试的充分性贡献越高。因此，定义路径 w_n 的充分因子如下：

$$S_{path}(w_n)=\frac{N_{path}(w_n)/N_{path}(W)}{\alpha_{path}(w_n)}$$

规定：当 $S_{path}(w_n)\geqslant 1$ 时，$S_{path}(w_n)=1$。

其中，$N_{path}(w_n)$ 表示测试用例集 T 对路径 w_n 所测到的次数。$N_{path}(W)$ 表示系统中所有的路径被测试用例集 T 测试到的总次数。

对所有的路径，它的路径充分因子为所有测试执行到的路径的充分因子的加权平均，即：

$$S_{path}(W)=\sum\alpha_{path}(w_n)\times S_{path}(w_n)$$

最后，得到系统的路径充分性为：

$$A_{path}(W)=C_{path}(W)\times S_{path}(W)\times 100\%$$

3.3.4 系统的综合测试充分性

根据测试的需要，为系统的路径充分性赋予一个相对重要性因子 $\gamma_{path}\in[0,1]$，服务充分性的相对重要性因子为 $\gamma_{ser}\in[0,1]$，规定 $\gamma_{path}+\gamma_{ser}=1$。得到系统的综合测试充分性：

$$A_{sys}(W)=\gamma_{path}\times A_{path}(W)+\gamma_{ser}\times A_{ser}(W)$$

4 案例分析

4.1 被测系统

在 Java 集成开发环境 Eclipse 下，载入机票预订系统 FMS(Flight Management System)，它是个标准服务接口的软件系统。使用不同数量的测试用例集对 FMS 进行测试，记录它们的测试覆盖充分程度。

使用 Eclipse 开源插件 EclEmma 软件测试工具实现对 FMS 的覆盖测试，并在测试中动态统计其代码测试覆盖率。按照测试中的统计数据绘制代码覆盖率变化曲线。

使用模型覆盖充分性的计算方法，观测给定测试用例集下 FMS 标准服务接口的数据分区覆盖充分性、决策表覆盖充分性，以计算出它的模型综合覆盖充分性。按照测试中的统计数据绘制模型综合覆盖充分性变化曲线。

4.2 测试覆盖充分性分析

模块 FMS 中，对 5 个不同的标准服务接口进行测试。

服务 A：FlightInfo getAirlineInfo(FromCity, ToCity)，查航班服务接口。该服务根据起飞城市、降落城市，输出符合要求的航班信息。

服务 B：Discount getDiscount(FlightId, FlyDate)，查折扣服务接口。该服务根据航班号、起飞日期，输出折扣信息。

服务 C：Price getPrice(FlightId, Cabin)，查价格服务接口。该服务根据航班号、舱位等级，输出机票价格。

服务 D：Result reserveTicket(Flight)，订票服务接口。该服务根据订票单详细信息进行订票，返回订票是否成功的信息。

服务 E：TicketNum getFreeTicket(FlightId, FlyDate, Cabin)，查机票余量服务接口。该服务根据航班号、起飞日期、舱位等级，输出对应舱位的机票余量。

对于以上 5 个服务，常用的订票流程如图 4 所示。

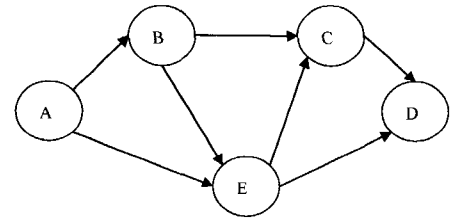

图 4 常用的订票流程

根据常用的订票流程所组成的服务执行网络图，可以将服务的执行序列分解为以下 4 条路径：

路径 1：A->B->C->D

路径 2：A->B->E->C->D

路径 3：A->B->E->D

路径 4：A->E->D

从测试用例集中随机挑选不同数量的测试用例进行测试评估，分别记录代码覆盖率和模型覆盖情况。以服务 A 为例。

服务 A 的输入参数 FromCity 和 ToCity 的数据池相同，根据城市机场的航班频次、航线承运利润等情况定义它们的分区影响因子，如表 1 所列。

表 1 服务 A 输入参数的分区影响因子

分区	影响因子	分区	影响因子
BJ	0.25	TW	0.03
SH	0.15	US	0.15
WH	0.1	AE	0.11
SZ	0.05	ES	0.03
HK	0.05	AU	0.08

服务 A 的输入输出组合的影响因子，视作在每条航线相

同，则服务 A 的 26 种输入输出组合的影响因子均为$\frac{1}{26}$。

根据定义好的影响因子，用文中的方法分析测试用例集，计算输入参数覆盖率和充分性，以及输入输出组合规则的覆盖率和充分性，得出服务的充分性，结果数据如表 2 所列。

表 2　服务 A 的模型覆盖充分性(单位：%)

测试用例数	输入参数 1		输入参数 2		输入输出组合		服务充分性
	覆盖率	充分性	覆盖率	充分性	覆盖率	充分性	
20	80	40.8	80	49.6	53.85	45.2	37.1
50	100	71	100	65	96.15	68	72.24
100	100	73	100	64	100	68.5	76.17
150	100	71.3	100	68.33	100	69.83	77.84
200	100	74	100	68.5	100	71.25	78.72
500	100	76.4	100	67.6	100	72	81.63

FMS 系统中有 5 个服务，共有 4 条不同的路径。根据路径的使用频率等，定义路径的影响因子，如表 3 所列。

表 3　服务组合的路径影响因子

路径	影响因子	路径	影响因子
1	0.3	3	0.2
2	0.4	4	0.1

根据定义好的影响因子，以不同数量的测试用例集对系统的所有服务进行模型覆盖充分性的分析和计算，测算系统的综合充分性(见表 4)。

表 4　系统的模型覆盖充分性(单位：%)

测试用例数	路径的充分因子				系统的充分性			
	路径 1	路径 2	路径 3	路径 4	路径覆盖率	路径充分性	服务充分性	综合充分性
20	50	100	50	100	100	75	60.61	67.8
50	66.67	75	100	100	100	85.42	75.83	80.62
100	90	70	95	100	100	88.75	80.14	84.44
150	86.67	61.67	100	100	100	87.08	83.41	85.25
200	93.33	57.5	100	100	100	87.71	84.64	86.17
500	90.67	60.5	100	100	100	87.8	87.79	87.79

使用不同数量的测试用例时，通过 EclEmma 得到的代码覆盖率情况如图 5 中代码覆盖率、模型充分性两条曲线所示。

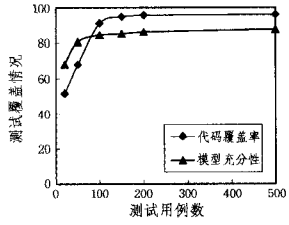

图 5　测试覆盖情况随测试用例数的变化

可以看出，当测试用例数量较少时，代码覆盖率和模型充分性取值较低。随着测试用例数的增加，代码覆盖率和模型充分性随之增长。当测试用例数增长到 200 以上时，代码覆盖率和模型充分性维持在较为稳定的取值。两种测试评估方法随着测试用例数的增加，变化趋势接近。

在实际分析中，当测试用例数量较少时，FMS 系统中的服务参数取值并没有覆盖所有的分区，也没有测试到所有的决策表规则，此时，系统的功能没有得到完全的测试，程序代码有些分支没有执行到，对应的系统模型充分性和代码覆盖率表现出较低的情况。随着测试用例数量的增加，服务参数的取值能够覆盖到更多的分区，并能覆盖更多的决策表规则，而系统的功能能被更加全面地测试，程序代码中的分支也能被更加全面地测试，此时，模型充分性和代码覆盖率表现出明显的增长。当测试用例增加到较大数量时，服务参数的取值能够覆盖到所有的分区，并能覆盖到所有的决策表规则，程序代码得以全面执行，系统的功能得到全方位的测试，因而模型充分性和代码覆盖率不再随着测试用例数的增加表现出增长，而是维持在稳定的取值。

本实验中的测试用例是随机生成的，在测试中不带有倾向性，即测试用例集不会对需要重点测试的部分提供更多的测试用例。可以看出，代码覆盖率与基于模型的测试充分性在测试过程中显示出相同的变化趋势。两种测试评估方法各有优劣，代码覆盖率侧重于评估代码的实现程度，基于模型的测试充分性侧重于评估需求实现的充分程度。代码覆盖率只能用于白盒测试，基于模型的测试充分性适用于黑盒测试。因此，在代码不可见时，可以选用基于模型的测试充分性来评估系统的测试充分程度。

结束语　基于模型的软件开发方法是软件开发领域的重大进步，也是一种流行趋势。基于接口模型的建模方法和相应的系统模型测试充分性分析方法的提出，使系统功能特征得以模型化描述，其测试的充分程度能够得到定量的计算。文中的测试充分性评估方法能准确地评估测试用例集对系统的测试充分性，对测试用例生成策略的设计和优化有一定指导意义。

参 考 文 献

[1] 伦立军，赵辰光，丁雪梅，等. 软件测试充分性研究[J]. 计算机工程与应用，2004，40(3)：60-62

[2] 万年红，李翔. 软件黑盒测试的方法与实践[J]. 计算机工程，2000，26(12)：91-93

[3] Broy M，Jonsson B，Katoen J P. Model-Based Testing of Reactive Systems：Advanced Lectures[M]. New York，Springer-Verlag Inc.，2005

[4] 朱少民. 软件测试方法和技术[M]. 北京：清华大学出版社，2010

[5] 安金霞，王国庆，李树芳，等. 基于多维度覆盖率的软件测试动态评价方法[J]. 软件学报，2010，21(9)：2135-2147

[6] 古乐，史九林. 软件测试技术概论[M]. 北京：清华大学出版社，2004

[7] Weißleder S，Schlingloff H. Automatic Model-Based Test Generation from UML State Machines[J]. Model-Based Testing for Embedded Systems，2011：77-109

[8] Utting M，Legeard，Practical. Model-Based Testing：A Tools Approach[M]. San Francisco，CA，Morgan Kaufmann Publishers Inc.，2006

[9] Chilenski J J，Miller S P. Applicability of modified condition/decision coverage to software testing[J]. Software Engineering Journal，1994，9(5)：193-200

[10] 颜炯，王戟，陈火旺. 基于模型的软件测试综述[J]. 计算机科学，2004，31(2)：184-187

[11] 马金梭，张荣华，席厚金. 模型覆盖率在嵌入式软件开发中的应用[J]. 计算机技术与发展，2006，16(10)：83-86