基于测试需求的互操作性测试用例生成方法

侯超凡 吴 际 刘 超
(北京航空航天大学计算机学院 北京 100191)

摘 要 网络化应用将成为未来软件技术发展的主导模式。为了保证网络化应用之间能够有效地协同工作,必须对其进行互操作性测试。互操作性测试具有测试需求复杂易变、测试用例设计困难的特点,因而需要消耗大量成本。为此,提出一种基于测试需求的互操作性测试用例生成方法。该方法采用模型驱动的测试思想,以测试需求模型描述互操作性测试需求,以状态图描述各被测应用的规格说明,通过两者的结合生成满足测试需求的测试用例。

关键词 互操作性测试,测试用例生成,模型驱动测试,测试需求
中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.11.032

Interoperability Test Case Generation Based on Testing Requirements

HOU Chao-fan WU Ji LIU Chao
(School of Computer Science and Engineering, Beihang University, Beijing 100191, China)

Abstract Network application will become the dominant mode of the development of software technology in the future. In the effort to ensure the effective work of the network applications, it is necessary to test interoperability among these network applications. The testing requirements of interoperability testing are complicated and inconstant, and the test cases are difficult to design, which lead to the high cost in interoperability testing. For this reason, a new interoperability test case generation approach based on testing requirements was presented in this paper. This approach uses the idea of Model-driven testing. The testing requirements are described by testing requirements model, and the specifications of the applications are presented with state diagrams. Finally, the test cases which satisfy the testing requirements are generated through the combination of the two models above.

Keywords Interoperability testing, Test case generation, Model-driven testing, Test requirement

1 引言

目前,Internet的普及和万维网(WWW)的快速发展正在引发软件技术变革,软件的发展呈现出明显的网络化趋势[1]。随着技术的进一步发展,可以预见,网络化应用将成为未来软件技术发展的主导模式[2]。互操作性[3]是网络化应用的必具特征,是实现分布式环境下,企业内及企业间多应用系统集成必须解决的问题。目前,已经存在多种实现软件间互操作性的技术,如远程过程调用(Remote Procedure Call, RPC)、通用对象请求代理体系结构(Common Object Request Broker Architecture, CORBA)、Web Service等。

在网络化应用大行其道的同时,由于其缺乏统一的规格说明指导,软硬件平台千差万别,开发语言、通信协议和数据交换的格式也不尽相同[4],导致在应用间存在互操作性问题。为保证网络化应用间能够有效地协同工作,必须对其进行互操作性测试[5]。统计资料[6]显示,测试工作开销占软件开发总成本开销的30%~50%,而互操作性测试由于以下因素,需要消耗更高成本。

1)互操作性测试的测试需求复杂,难管理。首先,因为互操作性测试的测试对象为分布式环境下的两个或两个以上应用,所以测试需求需要准确描述测试范围、各应用的部署信息以及应用间通信方式等。其次,参与互操作性测试的各应用都有可能发生变动,也有可能引入新的应用到测试中,这都会引起测试需求的变动。

2)互操作性测试用例设计复杂。互操作性测试用例的生成需要遵循各被测应用的规格说明,并准确描述应用间的互操作场景,这存在一定难度,同时也增加了测试用例设计的复杂性。

3)互操作性测试执行成本较高。互操作性测试的执行也需要在分布式环境下进行,测试环境的构建、测试用例执行等方面也必将增加测试成本。

测试工作经验表明,如果对测试需求的理解出现错误,将使测试过程产生偏差,浪费大量的人力与资金[7]。同时,测试工作的所有开销中,约40%花费在测试用例的设计上[6]。由此可见,对测试需求进行准确详尽的描述,并进行有效的管理,以指导互操作性测试用例的生成,对提高测试效率及质量,缩短网络化应用整体开发周期有重要意义。

针对上述问题,本文提出了一种基于测试需求的互操作

到稿日期:2013-07-20 返修日期:2013-08-30
侯超凡(1990—),男,硕士,主要研究方向为模型驱动测试、互操作性测试,E-mail:superhcfbuaa@gmail.com;吴 际(1974—),男,博士,副教授,主要研究方向为模型驱动开发与测试、测试自动化等;刘 超(1958—),男,教授,主要研究方向为软件测试、面向对象技术、软件开发环境等。

性测试用例生成方法。该方法采用模型驱动测试思想，以测试需求模型对互操作性测试需求进行描述和管理，以 UML 状态图模型描述各应用的规格说明，通过结合两者生成抽象互操作性测试用例。该方法能够提高互操作性测试的自动化程度，有助于提高互操作性测试效率和质量。

本文第 2 节介绍国内外研究现状及存在的问题；第 3 节简要介绍互操作性测试及其测试需求描述；第 4 节以网上购物场景为例，介绍互操作性测试用例生成方法；最后总结本文工作内容及研究价值，并提出后期可改进之处。

2 相关研究分析

互操作性测试在电信行业广泛用于对通信协议的测试。文献[8-10]等介绍了对于通信协议生成互操作性测试用例的方法。通信协议互操作性测试主要关注的是对同一协议标准，不同协议实现之间的互连通问题；而且协议的规格说明简明，且相对稳定，因此协议互操作性测试方法不完全适用于网络化应用。

近年来，对于网络化应用的互操作性测试的研究越来越多。Mang Li 等[11]提出了一种适用于 CORBA 技术的互操作性测试方法。该方法虽然对测试需求有所考虑，但并没有将其与测试用例生成结合。随着 Web 服务技术的兴起，学术界对此类应用的测试工作开展了广泛研究。Ying Yu 等[12]提出了一种基于本体的 Web 服务互操作性测试方法。该方法通过分析捕获的通信数据来发现异常，并生成新的数据，以诊断更多异常。由于其以 Web 服务间通信数据的捕获为前提，这要求被测系统已运行相当时间，因此该方法不适合开发阶段早期的测试。

2002 年，IBM、微软、BEA 等一起开发了 BPEL(Business Process Execution Language)作为描写协调 Web 服务的语言，由此出现了多种基于 BPEL 的互操作性测试用例生成方法[13,14]。BPEL 描述方法只适用于 Web 服务的测试工作，文献[15]也指出，BPEL 的抽象级别不适合直接用于业务流程的分析和设计。相反，现阶段大部分流程分析设计者更容易接受 UML 等高级别的抽象模式。Colin Smythe 在文献[16]中探讨了基于 UML 模型进行 Web 服务的互操作性测试的可行性，并介绍了基于顺序图的测试用例生成方法。该方法是对模型驱动互操作性测试方法的有效尝试，但是缺少对测试需求的考虑，而且基于顺序图也存在局限性：一张顺序图一般只能描述一个特定的场景。对于复杂网络化应用，就需要大量的顺序图来完整地描述，这必将增加工作量。

本文提出的针对网络化应用的互操作性测试用例生成方法，使用 UML 状态图描述各被测应用的规格说明，使用测试需求模型描述测试需求，两者结合实现模型驱动的互操作性测试用例自动生成。使用状态图是因其被广泛用于各种类型应用的设计、开发工作中，可以方便获得。即使不能从设计阶段直接获取，对单个应用的状态图建模也是简单易行的。由于对每个应用分别进行描述，当测试需求发生变化时，如添加新的应用到被测环境中，也无需对原有模型进行较大改动。

3 互操作性测试及其测试需求

相比于其他测试类型，互操作性测试有其特殊之处，对其测试需求描述时也需要针对性地考虑。

互操作性测试的测试对象是分布式环境下两个或两个以上相对独立的系统，主要关注各被测对象通过信息交互完成工作的过程，测试目标是验证各测试对象能否协同工作。由于这些特点，在开展互操作性测试时，需要一些针对性的工作，如划定测试范围，澄清被测系统包含哪些应用；标识各应用间的交互接口及连接方式；刻画应用间的互操作场景等。作为测试工作的基础，对互操作性测试进行需求分析时，需要充分考虑以上内容，并进行准确描述。

文献[7]详细阐述了测试需求的内涵，并提出了一种测试需求建模方法，该方法能够对测试需求进行直观清晰的描述。但该方法未考虑互操作性测试的特点，不能完整描述互操作性测试需求，为此对该方法进行改进，增强其对互操作性测试需求的描述能力。

定义 1(测试需求，Test Requirement)　TR 定义为三元组 $TR=\langle TO, TP, TC\rangle$，其中 TO、TP、TC 分别表示测试对象(Test Object)、测试目标(Test Purpose)以及测试约束(Test Constraint)。

测试对象包含被测系统实体、测试关注的系统端口及其消息类型、系统用例等内容。测试目标是从特定的测试场景，或验证某种特性的角度，来描述测试工作的目标。测试工作一般要求被测系统的所有用例都需要一个或一组测试目标进行验证。对于一个测试目标，可以手动或自动生成多个测试用例，以支持对这一目标的测试工作。测试约束包含测试特征及其约束，其中测试特征是被测系统或测试环境的可定量表达的特性；测试约束是对测试特征的取值约束。

通过定义测试需求模型，实现了对测试需求的可视化建模。为清晰准确地刻画测试需求所包含的诸多元素及元素间的关系和依赖，将测试需求划分为 4 种视图，如图 1 所示，这 4 种视图相对独立但又相互关联，四者统一起来描述测试需求的全部内容。

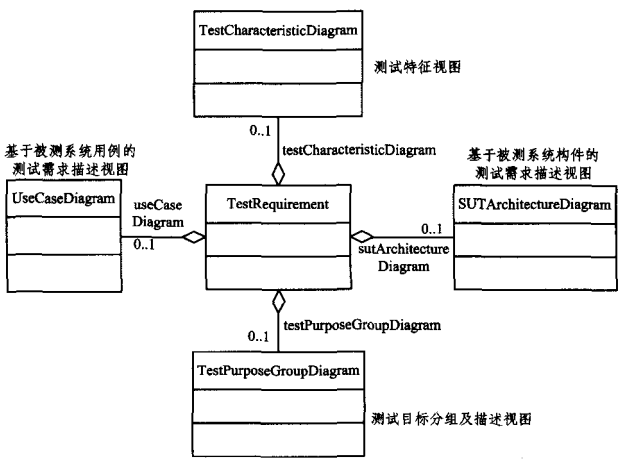

图 1　测试需求模型及其 4 种视图

测试需求中对测试对象、测试目标、测试特征等内容的描述与管理，对测试用例生成工作有直接的影响。本文引言中提到，相对于其他类别的测试，互操作性测试的测试需求内容更加复杂，也更容易发生变动。因此，借助测试需求建模方法，对互操作性测试需求进行描述及管理，以此来指导测试用例生成工作很有必要。

4 测试用例生成方法

本文提出的互操作性测试用例生成方法，其输入为测试

需求模型和各个被测应用的状态图，输出为针对测试需求模型中描述的测试目标生成的抽象测试用例。图 2 展示了该方法的基本流程。

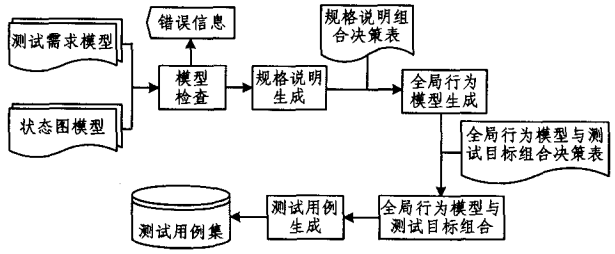

图 2　测试用例生成流程

模型验证模块负责对输入的测试需求模型和状态图模型进行检查，并反馈错误信息。规格说明模块通过提取状态图中与测试相关成分，构建应用的规格说明，以方便后续工作。然后，通过一种组合算法，将状态图组合得到全局行为模型，再将其与测试目标进行组合，在此基础上，最终得到满足测试目标的互操作性测试用例。为了方便理解，以简化的网上购物为例描述本文提出的方法。

4.1　测试需求描述

测试需求分析是测试工作的基础，在测试工作展开前首先要对测试需求进行描述，互操作性测试也不例外。

由于互操作性测试关注于多个应用间的交互，因此被测应用的部署关系及接口需要准确描述，这就是基于被测系统构件的测试需求描述视图（如图 3 所示）的意义所在。

本例中包含部署在不同节点的 3 个应用，即电子商店 eStore，电子银行 Ebank 和库存系统 Inventory System。图 3 中，应用被标记为 EUT，表明是被测实体。各应用接口上的标注表明测试工作需要关注各被测应用由这些接口所表现的行为，同时对接口允许的输入输出消息类型进行约束。消费者可以在电子商城中选择商品，建立订单。提交订单后，电子商城需要调用库存系统的查询库存服务，如果存货满足需求，还需调用电子银行实现的支付服务。

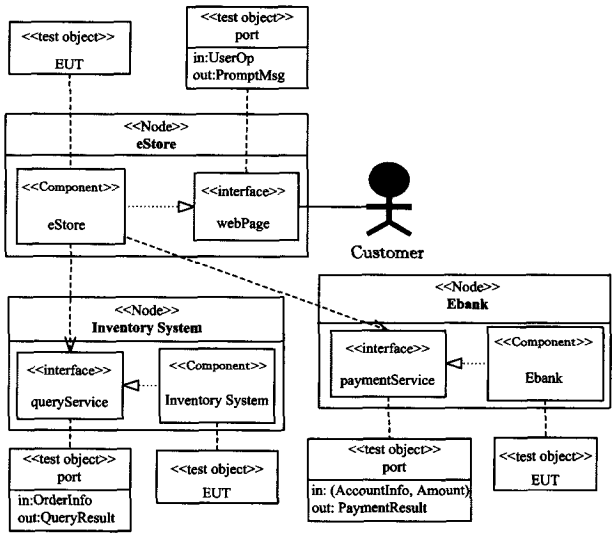

图 3　基于被测系统构件的测试需求描述视图实例

为了对测试工作进行有效管理，还需要借助测试需求模型的其他视图。本例的基于用例的测试需求描述视图，如图 4 左上所示，其中，购买商品 Purchasing Goods 用例上的标注表示针对该用例的测试目标为测试目标分组及描述视图中的 Purchasing 分组。Purchasing 分组包含了两个测试目标，其中 Successful 测试目标主要关注于成功购买商品的过程：期望测试系统模拟外部用户，通过电子商城的网页提交购买订单，若电子商城通过网页显示成功购买，则测试通过。在这两个消息中可以存在其他消息序列，包括测试系统模拟用户动作和 3 个应用间的交互消息。被测系统向测试系统反馈的消息若使用虚线图符，表示此测试目标拒绝该消息，即此测试目标对从第一个消息到此消息的消息序列并不关注。

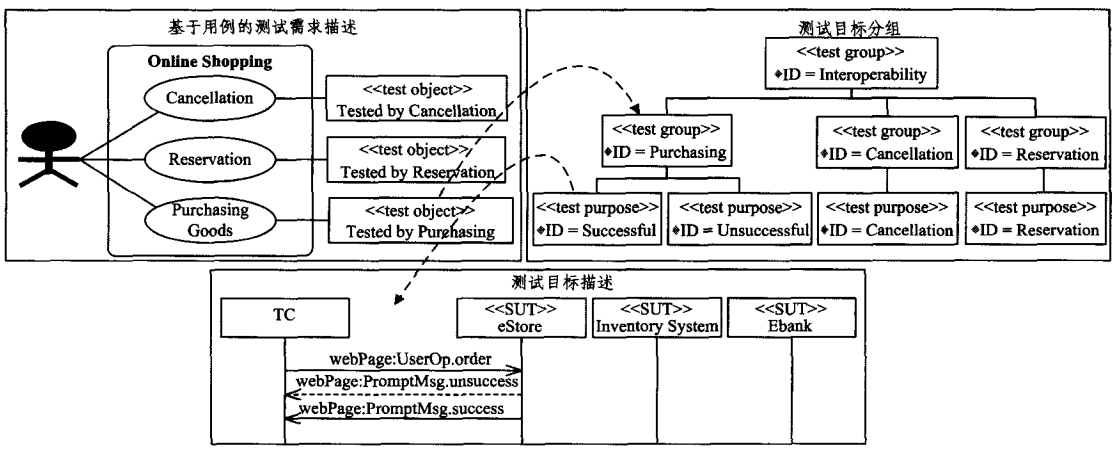

图 4　购买商品用例的测试需求描述实例

4.2　状态图描述被测应用

本文提出的方法利用了 UML 状态图，并通过借鉴通信顺序进程 CSP[17] 中的相关概念，更简明地描述应用通过接口所表现的行为。在本文中，将触发事件和动作分别认为是消息的接收和发送。接口的标识名在各应用的状态图中作为触发事件和动作的前缀，表示接收触发事件的接口和动作的输出接口。例如，迁移标记可能是：[Condition] A? Trigger / B! Action，这表示当由 A 接口接收到外部触发事件 Trigger 时，会对警戒条件 Condition 求值，若满足条件则由 B 接口向外部输出动作 Action。

本例中 3 个应用的状态图如图 5 所示。各状态图中迁移上的标记标识了组件间通过接口进行的消息传递，以及组件与外部环境进行的交互。图 5 中黑色虚线箭头，展示了 3 个应用间的 4 次消息传递。在这个例子中，应用间的连接被认为是双向的，如果需要强调连接的方向，可以通过修改基于被测系统构件的测试需求描述视图来进行描述。

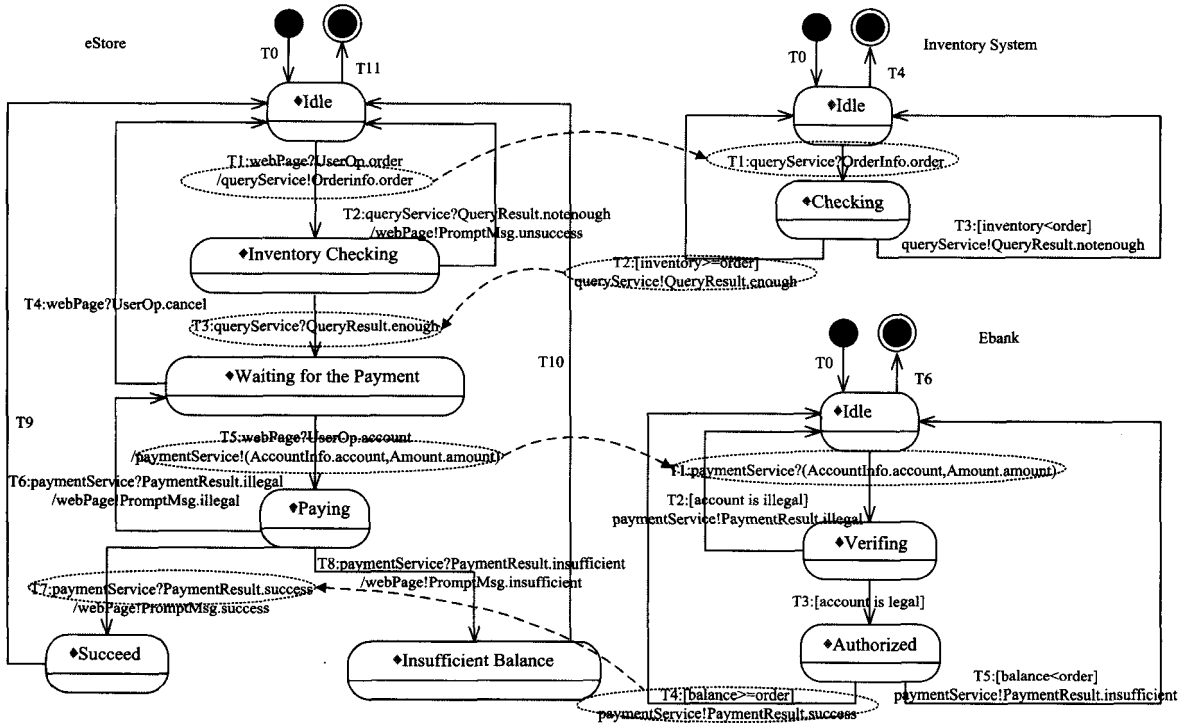

图 5　应用的状态图及其消息传递

4.3　模型检查

被测应用的状态图中，描述了应用通过接口发送/接收数据的行为，而测试需求模型的基于被测构件的视图中，对接口的输入输出数据类型进行了描述。因此，以测试需求模型为依据，可以判断构件的状态图是否存在问题。

模型检查模块通过规则 1 和规则 2 对模型进行检查，并反馈发现的问题。

规则 1：假设应用 A 的状态图中，迁移 t 上的触发事件为 e，接口为 $eport$。若接口 $eport$ 由应用 A 实现，则 e 的消息类型须与基于被测系统构件测试需求描述视图中接口 $eport$ 的输入类型一致；若接口 $eport$ 由其他应用实现，则 e 的消息类型须与接口 A 的输出类型一致。

规则 2：假设应用 A 的状态图中，迁移 t 上的动作为 a，接口为 $aport$。若接口 $aport$ 由应用 A 实现，则 a 的消息类型需与测试需求中接口 $aport$ 的输出类型一致；若接口 $aport$ 由其他应用实现，则 a 的消息类型需与接口 $aport$ 的输出类型一致。

4.4　全局行为模型建立

通过各应用的状态图组合，构建被测系统的全局行为模型。Sabnani[18] 首先提出这一方法，并将其应用于集成测试中。集成测试关心的是多个组件作为一个整体所表现出的行为，对于组件间的信息交互过程并不关注，所以原方法在组合的过程中抛弃了实体间的交互信息。而交互信息是互操作性测试重点考虑的，因此本文在该思想的基础上进行修改，在状态图组合的过程中，保留了实体间的交互信息。

由于状态图中状态间的迁移包含警戒条件、触发事件及动作等多种信息，如果将状态图直接进行组合，必将增加复杂性。状态图中会包含与测试需求无关的内容，如接口不是测试接口的触发事件和迁移以及互操作性测试不关注的内部迁移（没有触发事件和动作的迁移）。这些无关测试内容会加剧组合爆炸问题。因此，需要对各状态图进行必要处理。

本文使用输入输出标记迁移系统（IOLTS，Input-Output Label Transition System）[19] 来描述状态图处理结果，称为该应用的规格说明 S（Specification），并用其构造全局行为模型。

定义 2　输入输出标记迁移系统是一个四元组，$M=(Q^M, A^M, \rightarrow_M, q_0^M)$ 其中，Q^M 是状态的有穷非空集合，$q_0^M \subseteq Q^M$ 是初始状态。A^M 是迁移标记字母表，迁移标记由警戒条件、迁移类型、接口和消息名组成，而迁移类型 $Type \in \{SNED, RECEIVE\}$。$\rightarrow_M \subseteq Q^M \times A^M \times Q^M$ 为状态转移函数。为了标记处理过程中生成的中间状态，令 $Q_m^M \subseteq Q^M$，用以表示中间状态集合。

为便于描述，给出以下定义。假设迁移 t 由状态 s_1 指向状态 s_2，则 s_1 为 t 的源状态，s_2 为 t 的目标状态。令所有以状态 s 为目标状态的迁移集合为 $In(s)$，称为状态 s 的进入迁移。令所有以状态 s 为源状态的迁移集合为 $Out(s)$，称为状态 s 的外出迁移。

规格说明的生成过程分为以下 3 个步骤：

第 1 步　去除测试无关的触发事件和动作。对于迁移上的触发事件或动作，若其接口在测试需求中未标记为测试接口，那么该触发事件和动作是与测试无关的，应该去除。

第 2 步　去除内部迁移。经第 1 步处理后，内部迁移数量会增加。由于互操作性测试不考虑内部迁移，而内部迁移的存在会增加工作量，因此需要将其去除，同时还需保留内部迁移的警戒条件。若内部迁移 t 的源状态为 s_1，目标状态为 s_2。当 t 存在警戒条件 g 时，对 $Out(s_2)$ 中的每一个迁移 t_i，置警戒条件为 g 与 t_i 原警戒条件的逻辑与。再令 $In(s_1)=(In(s_1) \cup In(s_2)) \backslash \{t\}$，$out(s_1)=(Out(s_1) \cup Out(s_2)) \backslash \{t\}$。去除目标状态 s_2 和迁移 t。

第 3 步　添加中间状态。对于同时包含触发事件和动作的迁移：

$$s_1 \xrightarrow{[condition]port0?\ msg0/port1!msg1} s_2$$

则添加一个中间状态 s'，使得原迁移转换为：

$s_1 \xrightarrow{[condition]port0?\ msg0} s' \xrightarrow{port1!msg1} s_2$

由网上商城的状态图转换得到的规格说明，如图 6 所示。其中，标号为 1、3、5、8 和 9 的带网格状态为中间状态。

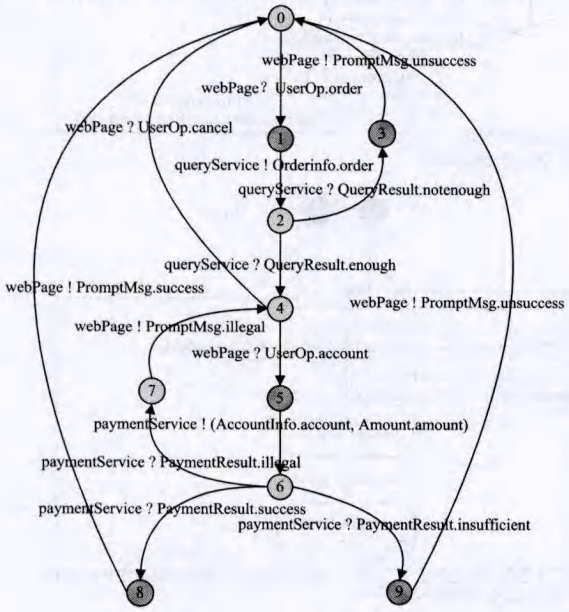

图 6　网上商城规格说明

如果规格说明 *Sa* 中某 SEND 类型迁移的动作与另一规格说明 *Sb* 中某 RECEIVE 类型迁移匹配（接口名和消息相同），则称 *Sa* 与 *Sb* 存在交互。规格说明 *Sa* 与 *Sb* 经组合运算（运算符为“#”）得到的结果为组合规格说明 *Sab*，定义如下。

定义 3　令输入输出标记迁移 $Sa=(Q^{Sa},A^{Sa},\rightarrow_{Sa},q_0^{Sa})$，$Sb=(Q^{Sb},A^{Sb},\rightarrow_{Sb},q_0^{Sb})$ 为两个存在交互的规格说明，则 $Sab=Sa\#Sb=(Q^{Sab},A^{Sab},\rightarrow_{Sab},q_0^{Sab})$。

其中，$Q^{Sab}=\{(q_a,q_b)\mid q_a\in Q^{Sa}\wedge q_b\in Q^{Sb}\}$；$q_0^{Sab}=(q_0^{Sa},q_0^{Sb})\in Q^{Sab}$；$A^{Sab}=A^{ab}\cup A^{Com}$，$A^{ab}=(A^{Sa}\cup A^{Sb})\backslash\{Sa$ 与 Sb 之间的输入输出$\}$；$A^{Com}=\{Sa$ 与 Sb 之间的交互信息$\}$；迁移类型 $Type\in\{$SEND，RECIVE，COM$\}$，COM 表示互操作消息类型；$\rightarrow_{Sab}\subseteq Q^{Sab}\times A^{Sab}\times Q^{Sab}$。

规格说明 *Sa* 与 *Sb* 的组合规格说明 *Sab* 包含 *Sa* 与 *Sb* 的所有行为。*Sab* 的状态为组合状态，它是一个二元组 (q_a,q_b)，其中，q_a 是规格说明 *Sa* 中的状态，q_b 是规格说明 *Sb* 中的状态，则 q_a 及 q_b 为组合状态 (q_a,q_b) 的角色状态。对于 *Sa* 与 *Sb* 间的交互信息，在 *Sab* 中存在与之对应的 COM 类型迁移，即两个应用间通过互操作引发的迁移。

全局行为模型可以被认为是多个规格说明的组合，其本身也是一个 IOLTS。下文以 GBM（Global Behavior Model）表示全局行为模型。为得到全局行为模型 GBM，需要进行多次组合运算。组合运算的最坏情况是输入的规格说明之间不存在互操作消息，那么输出的组合规格说明中状态（迁移）数量是输入的两个规格说明中状态（迁移）数量的乘积。为避免这种情况的发生，采取以下策略，即将所有规格说明放入容器中，每次从该容器中取出两个交互信息数最多的规格说明，计算得到的组合规格说明再放入容器中。

得到全局行为模型 GBM 的算法如算法 1 所示。

算法 1　全局行为模型 GBM 生成算法

输入：各应用的规格说明

输出：全局行为模型

将各应用的规格说明放入容器 SList 中；

WHILE SList 长度大于 1

THEN 从 SList 中取出包含交互信息数最多的两个规格说明 Sa、Sb；

分别得到 Sa 和 Sb 的初始状态 q_0^{Sa}、q_0^{Sb}，将组合状态 (q_0^{Sa},q_0^{Sb}) 放入队列 Q 中，同时 (q_0^{Sa},q_0^{Sb}) 作为组合规格说明 Sab 的初始状态；

WHILE 队列 Q 不为空

THEN 从 Q 中取出一个状态 $q=(q_a,q_b)$，按照表 1 所列规格说明组合决策表，计算 $Out(q_a)$ 与 $Out(q_b)$ 中迁移组合形成的新迁移 t 及后继状态 (q_a',q_b')。

IF Sab 中不存在状态 (q_a',q_b')

THEN Sab 添加新迁移 t 和状态 (q_a',q_b')；

IF q_a' 或 q_b' 为中间状态

THEN 将中间状态 q_a' 外出迁移 t′添加到 Sab 中；如果 Sab 中不存在 t′的目标状态 (q_a'',q_b'')，则将 (q_a'',q_b'') 加入 Sab，并放入队列 Q；

ELSE 将状态 (q_a',q_b') 放入队列 Q；

ENDIF

ELSE Sab 添加新迁移 t；

ENDIF

ENDWHILE

将 Sab 放入 SList 中；

ENDWHILE

SList 中的规格说明即为输出的全局行为模型 GBM。

表 1　规格说明组合决策表

$q_a\xrightarrow{[ca],pa,ta,ma}q_a'$ ta	pa to B	$q_b\xrightarrow{[cb],pb,tb,mb}q_b'$ tb	pb to A	ma=mb & pa=pb	结果
send	yes	recv	yes	yes	$\xrightarrow{[ca\&cb],p,com,m}(q_a',q_b')$
recv	yes	send	yes	yes	$\xrightarrow{[ca\&cb],p,com,m}(q_a',q_b')$
send	yes	send	yes	yes	—
recv	yes	recv	yes	yes	—
send/recv	yes	send/recv	yes	no	—
send/recv	no	send/recv	no	*	$\xrightarrow{[ca],pa,ta,ma}(q_a',q_b)$ $\xrightarrow{[cb],pb,tb,mb}(q_a,q_b')$
send/recv	no	send/recv	yes	*	$\xrightarrow{[ca],pa,ta,ma}(q_a',q_b)$
send/recv	yes	send/recv	no	*	$\xrightarrow{[cb],pb,tb,mb}(q_a,q_b')$
com	*	send/recv	yes	*	$\xrightarrow{[ca],pa,com,ma}(q_a',q_b)$
send/recv	yes	com	*	*	$\xrightarrow{[cb],pb,com,mb}(q_a,q_b')$
com	*	send/recv	no	*	$\xrightarrow{[cb],pb,tb,mb}(q_a,q_b')$ $\xrightarrow{[ca],pa,com,ma}(q_a',q_b)$
send/recv	no	com	*	*	$\xrightarrow{[ca],pa,ta,ma}(q_a',q_b)$ $\xrightarrow{[cb],pb,com,mb}(q_a,q_b')$
com	*	com	*	*	$\xrightarrow{[ca],pa,com,ma}(q_a',q_b)$ $\xrightarrow{[cb],pb,com,mb}(q_a,q_b')$

4.5　测试用例生成

本文生成的测试用例为测试系统与各被测实体间以及被测实体之间通信消息构成的消息序列。在得到全局行为模型后，便可通过测试需求模型中描述的测试目标生成定制的测试用例。测试目标描述也同样需要转化为输入输出标记迁移系统，称为测试目标 TP。

测试目标 $TP=(Q^{TP},A^{TP},\rightarrow_{TP},q_0^{TP})$。其拥有两种陷阱

状态：$Accept^{TP} \subseteq Q^{TP}$，$Refuse^{TP} \subseteq Q^{TP}$。$Accept^{TP}$ 表示状态被测试目标所接受，如果达到此状态，则测试判定成功，测试通过；$Refuse^{TP}$ 表示状态在此测试目标下不被关注，测试判定未定义。行为字母表 A^{TP} 与全局行为模型的行为字母表相同。由测试目标描述视图可以方便地得到测试目标 TP，由图 4 中测试目标描述得到的测试目标 TP 如图 7 所示。

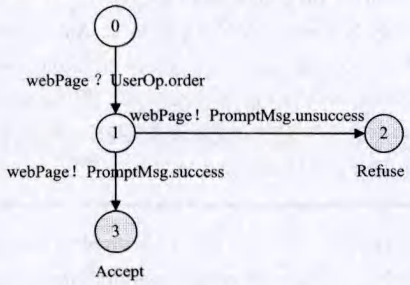

图 7　测试目标实例

为生成测试用例，需将全局行为模型 GBM 与测试目标 TP 组合，组合结果同样也是一个 IOLTS 系统，称为 GBMTP。该过程可利用算法 1 得到，其输入为全局行为模型 GBM 和测试目标 TP，但需将表 1 替换为全局行为模型与测试目标组合决策表，如表 2 所列。

表 2　全局行为模型与测试目标组合决策表

$q_g \xrightarrow{[cg],pg,tg,mg} q_g'$	$q_{tp} \xrightarrow{ptp,ttp,mtp} q_{tp}'$		结果
$pg=ptp=p_0$	$tg=ttp=t_0$	$mg=mtp=m_0$	
yes	yes	yes	$\xrightarrow{[cg],p_t,t_t,m_t}(q_g', q_{tp}')$
else			$\xrightarrow{[cg],pg,tg,mg}(q_g', q_{tp}')$

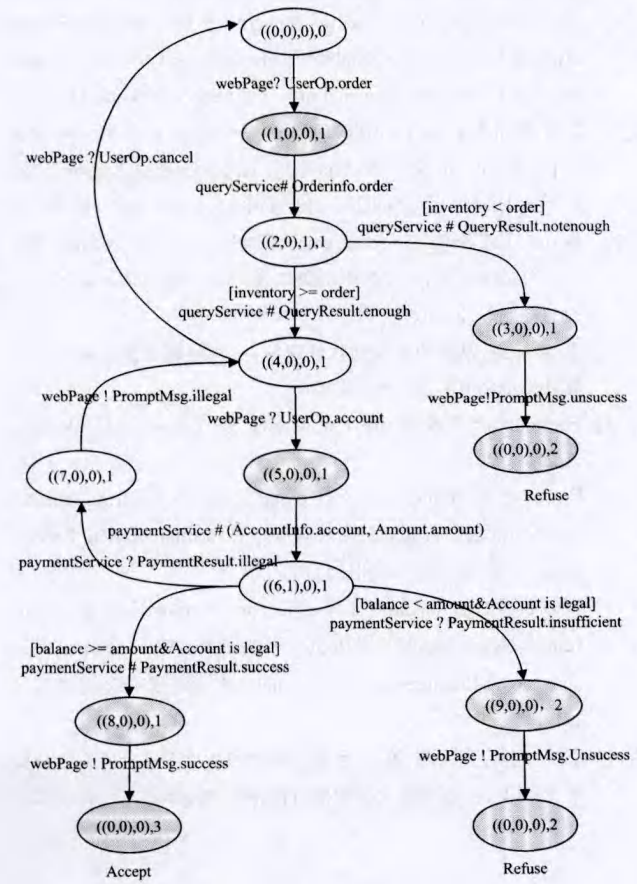

图 8　全局行为模型 GBM 与测试目标 TP 组合结果

图 8 展示了得到的 GBMTP，它包含两个 Refuse 节点和一个 Accept 节点。以初始节点为起点，搜索 GBMTP 中可到达 Accept 节点的所有基本路径，即可得到与测试目标对应的测试用例集。图 9 展示了得到的一个测试用例。

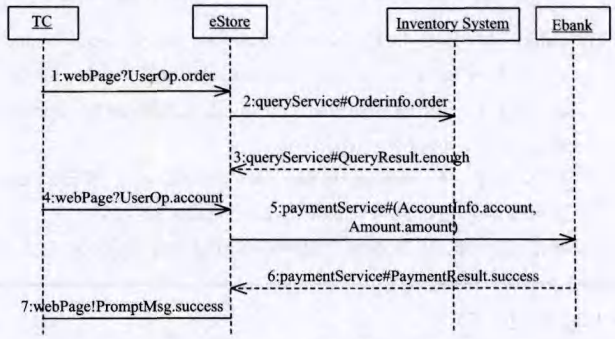

图 9　抽象测试用例

结束语　网络化应用间的互操作性测试，具有测试需求复杂、难以管理，测试用例设计难度大，测试执行成本高等特点。这导致互操作性测试需要消耗较高成本。针对此问题，本文提出了一种基于测试需求生成互操作性测试用例的方法，使得测试工作能够很好地契合测试需求，及时响应测试需求的变动，降低了测试用例设计的开销，提高了测试执行的效率。该方法目前还未涉及测试需求中测试特征描述，在后续工作中，将对此进行相关研究。

参 考 文 献

[1] 马于涛，何克清，李兵，等. 网络化软件的复杂网络特性实证[J]. 软件学报，2011，22(3)：381-407

[2] 怀进鹏. 对未来网络化软件技术的几点认识[J]. 中国计算机学会通讯，2008，4(1)：19-26

[3] 苏森，唐雪飞. 开放系统中的互操作性[J]. 计算机应用，1997，17(6)：4-7

[4] 罗玲，白晓颖. Web 服务技术的分析[J]. 计算机科学，2004，31(4)：19-23

[5] Walter T，Plattner B. Conformance and interoperability-A critical assessment[R]. Technical Report，Computer engineering and networks laboratory (TIK)，Swissfederal institute of technology Zurich，1994

[6] 朱少民. 软件测试方法和技术[M]. 北京：清华大学出版社，2005

[7] 毕考. 基于被测系统模型的测试需求描述及其可视化[D]. 北京：北京航空航天大学计算机学院，2011

[8] Seol S，Kim M，Chanson S T，et al. Interoperability test generation and minimization for communication protocols based on the multiple stimuli principle[J]. IEEE Journal on Selected Areas in Communications，2004，22(10)：2062-2074

[9] Wang Z，Wu J，Yin X. Towards interoperability test generation of time dependent protocols：a case study[C]// Global Telecommunications Conference，2004 (GLOBECOM'04). IEEE，2004，2：589-594

[10] Hao R，Lee D，Sinha R K，et al. Integrated system interoperability testing with applications to VoIP[J]. IEEE/ACM Transactions on Networking (TON)，2004，12(5)：823-836

[11] Li M，Puder A，Schieferdecker I. A test framework for CORBA interoperability[C]// Proceedings. Fifth IEEE International Enterprise Distributed Object Computing Conference，2001

(EDOC'01). IEEE, 2001:152-161

[12] Yu Y, Huang N, Ye M. Web services interoperability testing based on ontology[C]// The Fifth International Conference on Computer and Information Technology, 2005(CIT 2005). IEEE, 2005:1075-1079

[13] Baldoni M, Baroglio C, Martelli A, et al. A priori conformance verification for guaranteeing interoperability in open environments[C]// Service-Oriented Computing(ICSOC 2006). Springer Berlin Heidelberg, 2006:339-351

[14] Li Xi-tong. A framework for interoperability of BPEL-based workflows[J]. High Technology Letters, 2008, 4:18

[15] 王雷,徐立臻. BPEL 建模工具中业务流程模型到 BPEL 程序的转换算法[J]. 计算机与数字工程,2009,37(9):20-23

[16] Smythe C. Initial Investigations into Interoperability Testing of Web Services from their Specification Using the Unified Modelling[C]// International Workshop on Web Services-Modeling and Testing (WS-MaTe 2006). 2006:95-119

[17] Hoare. Communicating sequential processes [J]. Communications of the ACM, 1978, 21(8):666-677

[18] Sabnani K K, Lapone A M, Uyar M U. An algorithmic procedure for checking safety properties of protocols [J]. IEEE Transactions on Communications, 1989, 37(9):940-948

[19] 郝瑞兵,吴建平. 一种形式化的协议互操作性测试方法[J]. 计算机学报,1997,20(4):305-359

(上接第 140 页)

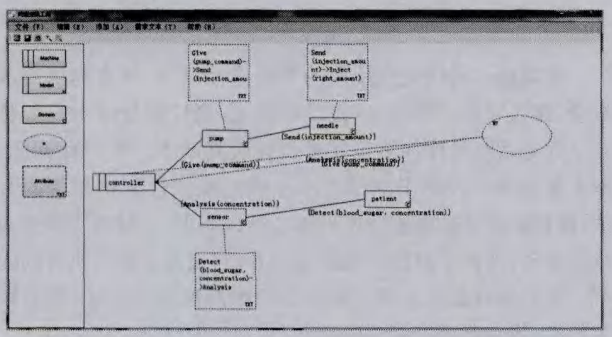

图 8 胰岛素泵子问题 1 渐变步骤 3:运用转换视角规则

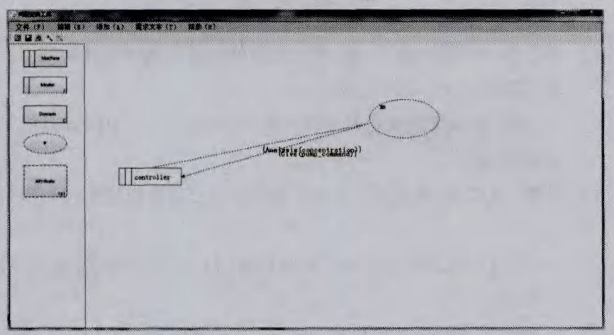

图 9 胰岛素泵子问题 1 渐变步骤 4:运用删除简化规则

结束语 在本文中我们介绍了一种基于问题框架建模变换的软件需求分析辅助支持工具,它是本文通信作者长期从事问题框架研究的结果[8,12-14]。目前,该工作的一个早期原型版本可以从网站 http://www.se.gxnu.edu.cn/tooldemo 中下载和试用。未来的工作是继续完善本 CASE 工具,并在时机成熟时采用经验软件工程的研究手段对其进行评估和评价。

参 考 文 献

[1] Jackson M. Software requirements and specifications: a lexicon of principles, practices and prejudices [M]. Boston: Addison-Wesley, 1995

[2] Jackson M. Problem frames: analyzing and structuring software development problems[M]. Boston: Addison-Wesley, 2001

[3] Hall G H, Rapanotti L, Jackson M. Problem-oriented software engineering: a design-theoretic framework for software engineering, 2007[C]// Proceedings of the 5th IEEE International Conference on Software Engineering and Formal Methods. Los Alamitos: IEEE CS Press, 2007:15-24

[4] Hall G H, Rapanotti L, Jackson M. Problem-oriented software engineering: solving the package router control problem [J]. IEEE Transactions on Software Engineering, 2008, 34(2):226-241

[5] Strunk E A, Knight J C. The essential synthesis of problem frames and assurance cases[J]. Expert Systems, 2008, 25(1):9-27

[6] Mannering D, Hall J G, Rapanotti L. Towards normal design for safety-critical systems[C]// Fundamental Approaches to Software Engineering. Springer Berlin Heidelberg, 2007:398-411

[7] Yin B, Jin Z, Li Z. Reliability concerns in the Problem Frames Approach and system reliability enhancement patterns[J]. Jisuanji Xuebao(Chinese Journal of Computers), 2013, 36(1):74-87

[8] Li Z, Hall J G, Rapanotti L. On the systematic transformation of requirements to specifications [J]. Requirements Engineering, 2013, (doi:10.1007/s00766-013-0173-8), online first article

[9] Berry M D. Software requirements and design: the work of Michael Jackson[J]. ACM SIGSOFT Software Engineering Notes, 2011, 36(2):39-40

[10] 李智,金芝. 从用户需求到软件规约:一种问题变换的方法[J]. 软件学报,2013,24(5):961-976

[11] Sommerville I. Software Engineering 9th Edition[M]. Boston: Addison-Wesley, 2011

[12] Rapanotti L, Hall G J, Li Z. Deriving specifications from requirements through problem reduction[J]. Journal of IEE Proceedings-Software, 2006, 153(5):183-198

[13] Li Z, Hall G J, Rapanotti L. On the construction of specifications from requirements[C]// Procs of the 14th Workshop on Requirements Engineering. Rio de Janeiro, Brazil: BDBComp, 2011:431-442

[14] 李智,庞柳,刘国源,等. 一种模型驱动的软件需求分析方法及技术支持[J]. 广西师范大学学报:自然科学版,2013,31(2):19-26