面向 TTCN-3 性能测试的负载生成方法

徐 珞[1),2)] 吴 际[1)] 刘 超[1)]

[1)](北京航空航天大学计算机学院 北京 100191)

[2)](华北计算技术研究所 北京 100083)

摘 要 TTCN-3(Testing and Test Control Notation version 3)是一种面向黑盒测试的测试描述与实现语言.随着 TTCN-3 语言的广泛应用,用户对使用 TTCN-3 进行性能测试的需求日益强烈.然而,TTCN-3 语言没有提供有效的负载描述和产生机制.目前,在使用 TTCN-3 产生性能测试的负载时,通常需要依靠大量的人工编码.该文提出了一种模型驱动方法以更加有效地支持面向 TTCN-3 的负载生成.在该方法中,负载指标模型用于刻画负载指标及约束关系;负载剖面模型则能够定义指标的取值及指标值随时间变化的情况.基于这些模型,该文提出的算法能够完成从模型到 TTCN-3 测试系统的自动转换.TTCN-3 测试系统可在负载控制点的支持下得以执行,从而模拟出满足模型描述的负载场景.该文通过案例分析验证了上述方法的有效性和所模拟负载场景的准确性.

关键词 TTCN-3;性能测试;模型驱动测试;负载建模;负载生成

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.00985

A Workload Generation Method for TTCN-3 Performance Testing

XU Luo[1),2)] WU Ji[1)] LIU Chao[1)]

[1)](*School of Computer Science and Engineering, Beihang University, Beijing* 100191)

[2)](*North China Institute of Computer Technology, Beijing* 100083)

Abstract TTCN-3 (Testing and Test Control Notation version 3) is a test specification and implementation language to define test procedures for black-box testing. As TTCN-3 has been widely accepted and applied in many fields of testing, the demand of using TTCN-3 in performance testing is emerged and increased rapidly. In performance testing, how to define and generate workload is an important issue. However, because of TTCN-3 language's insufficient support, testers usually need to make a great effort to manually write workload generation code in TTCN-3. This paper proposes a model-driven method to facilitate workload generation in TTCN-3 performance test. According to the method, the LQM (Load Quantity Model) is used to character the workload quantities and the constraints between the quantities, while the LPM (Load Profile Model) could define the value of each quantity and its changes with respect to time. This paper also presents a series of algorithm to automate the transformation from the model to the TTCN-3 test system. Finally, together with a set of control points the authors designed in this paper, the TTCN-3 test system could be executed in order to generate the expected workload. A case is used to demonstrate the capability of the method and validate the accuracy of the generated workload.

Keywords TTCN-3; performance testing; model-driven testing; workload modeling; workload generation

收稿日期:2010-10-20;最终修改稿收到日期:2011-05-04.本课题研究得到国家"八六三"高技术研究发展计划项目基金(2009AA01Z145)、软件开发环境国家重点实验室基金项目(SKLSDE-2010ZX-15)资助.**徐 珞**,男,1976 年生,博士研究生,高级工程师,主要研究方向为软件测试.E-mail:xuluo@sei.buaa.edu.cn.**吴 际**,男,1973 年生,博士,副教授,主要研究方向为软件测试、软件可靠性分析.**刘 超**,男,1958 年生,教授,博士生导师,中国计算机学会高级会员,主要研究方向为软件工程、软件测试.

1 引 言

TTCN-3(Testing and Test Control Notation version 3)是由欧洲电信标准化协会(ETSI)于2000年制定和推动的测试规范与测试实现标准,并且已被国际电讯联盟(ITU)等国际标准化机构所采纳.经过了十余年的发展,TTCN-3已经被成功应用在通信和软件领域的一致性和功能测试中.各种采用TTCN-3制订的标准测试集也被陆续发布出来.随着TTCN-3的应用日益广泛,用户对使用TTCN-3进行非功能测试的需求也越来越强烈,其中,基于TTCN-3的性能测试已经成为了一个研究热点.

性能测试通过加载负载于被测系统(System Under Test,SUT),观察SUT在给定负载下的性能表现,并对各项性能数据进行分析与评估,从而得出对SUT性能的整体评价.为了实现性能测试,如何有效地模拟系统的负载成为了一个关键问题.相关研究表明,性能测试的好坏很大程度上取决于负载被准确理解并真实模拟的程度[1].然而,TTCN-3语言没有提供有效的负载描述和产生机制.目前,在使用TTCN-3定义性能测试的负载时,通常需要测试人员手工编写大量的代码,同时由于TTCN-3语言自身的局限性,某些负载的产生还需要依靠对外部函数的使用[2].这些既增加了测试开发的工作量,影响了测试的效率,也提高了测试代码的复杂度,增加了引入人为错误的风险,使测试系统的质量难以得到保证.另一方面,现有的TTCN-3性能测试研究多集中在对语言的实时性扩展[2-4]、测试系统设计[5-7]和针对特定系统的性能测试[8-10]方面,对负载描述和产生问题的研究相对较少.

与传统的性能测试方法相比,基于TTCN-3的性能测试对负载描述和产生方面也有其特殊的需求,主要体现在以下4个方面:

(1)在负载建模方面,现有的负载模型多是针对特定类型的SUT(如Web系统、通信设备等),其中的负载指标及指标关系已经固化在负载模型中,难以进行调整和扩展.然而,TTCN-3是一种通用的测试语言,其应用范围和SUT类型是不固定的,因此,面向TTCN-3的负载模型必须提供面向任务的建模能力,即能够根据测试任务和SUT类型,由用户定义相适应的负载模型,刻画负载指标及指标间关系.

(2)在负载描述能力方面,特定类型的负载模型通常只关注某方面的负载特征.例如,在Web系统性能测试中,相关的负载模型就分为两类,分别针对网络特征(如SpecWeb99[11])和用户行为特征(如CBMG[12]).然而,由于TTCN-3使用范围广泛,所面对的负载特征也较为复杂.因此,在进行负载建模时应将多个方面的负载特征纳入统一的负载指标体系,加以综合考虑,从整体上反映出SUT的负载情况.同时,在负载刻画中还必须要考虑这些特征之间的约束关系.

(3)在负载产生方面,现有的负载模型通常依赖于专有的测试工具来产生负载,如LoadRunner、OpenSTA等.然而,TTCN-3性能测试系统必须依赖标准化的测试平台.TTCN-3标准[13]规定了测试平台的结构和接口.因此,面向TTCN-3的负载产生程序必须能够在相关标准的约束下,使用标准化的接口来实现对各类负载指标的控制,以模拟出符合负载模型定义的负载场景.

(4)负载产生方面的另一重要问题是,由于TTCN-3性能测试要求负载模型具备面向任务的建模能力,其中的负载指标可由用户进行定义和调整;但另一方面,TTCN-3负载产生程序又要受到TTCN-3标准的约束,可使用的负载控制点和控制方式相对固定.因此,如何依靠有限的控制点对由用户定义的多样化的负载指标进行控制,使其产生出的负载能符合相关指标的描述,是TTCN-3负载产生中必须要关注和解决的问题.

针对TTCN-3性能测试对负载建模和生成的需求,本文提出了一种模型驱动的TTCN-3负载产生方法.其基本思想是通过模型刻画负载,再依靠自动转换算法将模型转换为TTCN-3测试系统,从而避免了人工进行编码,同时也提高了对负载的描述和模拟能力.该方法以一组负载模型为核心,其中,负载指标模型描述了负载指标的组成和指标之间的约束,负载剖面模型定义了负载指标的取值以及指标值随时间的变化.所建立的负载模型能够在控制规划算法的支持下将其中的指标值映射成TTCN-3负载控制参数,并在本文所提出的负载控制点的支持下得以执行,从而模拟出满足模型描述的负载场景.本文针对一个Web应用系统开展了实验验证,对该方法的建模能力和负载模拟能力进行了分析,并与相关方法进行了对比,从而验证了该方法的可行性和有效性.

本文的第2节是模型驱动的负载生成方法的概述,给出该方法的基本过程;第3节介绍主要测试模

型及其构造方法；第 4 节论述负载产生方法；第 5 节给出使用本文方法进行的实验，并对实验结果进行分析；第 6 节介绍相关研究，并与本文的工作进行比较；最后在第 7 节对本文工作进行总结和展望.

2　模型驱动的 TTCN-3 负载生成方法

模型驱动的负载生成过程（如图 1 所示）包括负载建模和负载生成两个阶段. 在建模阶段，本文针对现有负载模型的局限性和 TTCN-3 性能测试对负载模型的需求，提出了一种分层次的建模方法，具体包含 3 个层次：

（1）面向测试类型的建模. 其建模制品是负载指标模型. 负载指标模型用于定义面向某一类 SUT（如 Web 系统）的常用负载指标集合及这些指标之间的约束关系. 负载指标是对负载某方面特征的定量描述. 对于同一类系统的测试，应使用相同的负载指标模型.

（2）面向测试任务的建模. 其建模制品是任务指标集. 在一个具体测试任务中，测试人员通常不会使用所有指标来定义负载，而会根据任务要求，从负载指标模型中选取一组重点指标，检验 SUT 在这些指标下的性能表现. 这些指标是从负载指标模型中选取的一个子集，称作“任务指标集”.

（3）面向负载场景的建模. 其建模制品是负载剖面模型. 为了对 SUT 的性能进行充分检验，并确定性能随负载变化的情况，测试人员需定义一系列的测试场景，在每个场景中对负载指标的取值会有所不同. 本文使用负载剖面模型定义测试场景. 每个负载剖面模型中定义了任务指标集中所有负载指标的取值以及其取值随时间变化的情况.

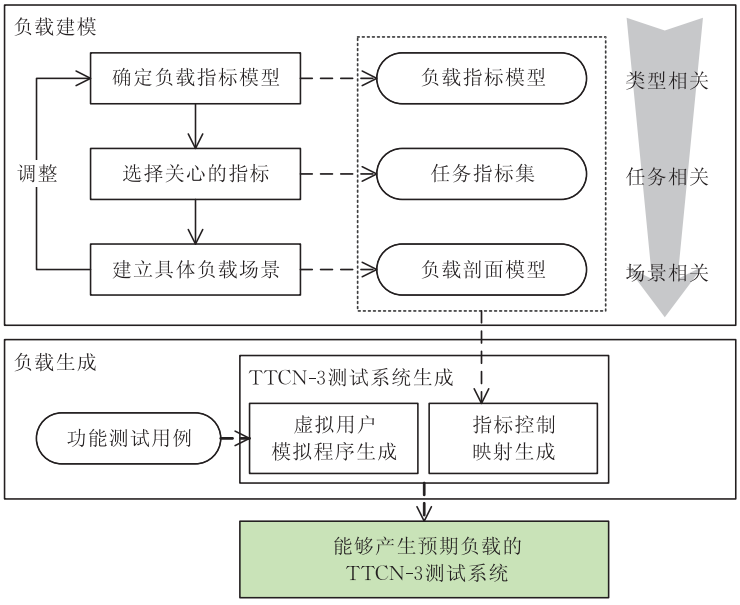

图 1　模型驱动的负载生成过程

通过将负载模型划分为 3 个层次，满足了 TTCN-3 性能测试对负载建模灵活性和可扩展性方面的要求. 负载指标模型可允许根据不同的 SUT 类型定义不同的负载指标及指标关系；任务指标集和负载剖面模型则允许根据不同的测试任务选择所关注的指标范围并确定指标的取值. 上述 3 类模型既相互联系，又位于不同层次. 按其抽象程度可从高到低依次排列. 其中，高层次的模型具有一定的通用性，可在不同测试项目中进行重用；低层次的模型包括与测试任务和负载生成相关的细节，可方便地转换为 TTCN-3 测试代码或控制参数. 负载建模过程就是一个对模型不断转化和细化的过程.

在负载生成阶段，其主要目标是根据负载模型生成一个 TTCN-3 测试系统，该系统的执行能够产生出模型所定义的负载场景. 这个 TTCN-3 测试系统由两个部分构成：一组虚拟用户模拟程序和一组负载控制点. 其中，虚拟用户模拟程序定义了测试系统与 SUT 的一个交互过程；负载控制点则控制着虚拟用户模拟程序的启动时机、并发数量、执行过程等，通过对这些因素进行不断的调整，能够模拟出负载模型所规定的负载场景. 与这两个部分对应，负载生成也包括了两方面的内容：

（1）虚拟用户模拟程序的生成. 由于 TTCN-3 功能测试用例往往是从用户角度与 SUT 进行交

互,具有和虚拟用户模拟程序相似的设计原则.因此可利用这种相似性,定义一套转换规则,应用这些规则将已有的功能测试用例自动转化为虚拟用户模拟程序.

(2)负载控制点的控制关系生成.采用的方式是建立负载模型中各项指标与 TTCN-3 负载控制点之间的映射关系,定义每项指标由哪些控制点进行控制,进而将负载指标的取值分解为相应控制点的控制参数.

在上述两个方面的生成中,前者主要采用了我们在文献[14]提出的方法,这里不作详述.本文则重点对后者进行讨论,包括定义 TTCN-3 负载控制点和提出一套用于自动建立指标与控制点映射关系的控制规划算法.

3 负载模型及建模方法

下面,本文将对负载建模阶段的主要模型以及各模型的构造方法进行详细论述.

3.1 负载指标模型

负载指标模型给出了面向特定类型 SUT 的常用负载指标.这些指标能够从用户和环境的角度反映出 SUT 在实际使用中的负载特征以及它们之间的关系.本文定义的负载指标模型如下.

定义 1. 负载指标模型. $LQM^{\Theta}=\langle Q,D,P,CG\rangle$,其中,$Q=\{q_1,q_2,\cdots,q_n\}$,$q_i$ 为针对 Θ 类 SUT 的一个负载指标,每个负载指标可赋予一个具体值(记作 $value(q_i)$),代表了在负载产生时对这些指标的期望值;$D=\{d_i\mid 1\leq i\leq|Q|\}$,$d_i$ 为指标 q_i 的值域(记作 $domain(q_i)$);$P=\langle C,R\rangle$,代表了指标分类,其中 C 为类别集合,$R=\{\langle q_i,c_j\rangle\mid q_i\in Q,c_j\in C\}$,$\langle q_i,c_j\rangle$ 是一个二元关系,表示指标 q_i 隶属于类别 c_j,指标 q_i 的类别记作 $category(q_i)$;$CG=\{cg_1,cg_2,\cdots,cg_m\}$,$cg_i$ 为两个或多个负载指标之间的一个约束组,每个约束组包含了有关指标之间的一系列约束.根据约束的类型不同,约束组可分为两类:值约束组和区间约束组.

LQM 中的负载指标并不是孤立存在的,相互之间存在着联系和制约关系.定义 1 将这种关系表示为约束组 *CG*.由于这些约束组对负载建模和负载生成具有重要影响,因此我们还必须对它们进行更加具体的刻画.下面分别给出了两类约束组的形式化定义.

定义 2. 值约束组. $vcg=\langle Q',F\rangle$,其中 $Q'=\{q_i\mid q_i\in LQM^{\Theta}.Q\}$,定义了该约束组所约束的指标集合,$Q'\subseteq LQM^{\Theta}.Q$;$F=\{f_1,f_2,\cdots,f_n\}$,并且有

$$(\forall q_y)(q_y\in Q')\rightarrow(\exists f_i)(\exists q_{x1})\cdots(\exists q_{xm})$$
$$(f_i\in F\wedge q_{x1},\cdots,q_{xm}\in Q'\wedge$$
$$f_i(q_{x1},\cdots,q_{xm})\rightarrow q_y).$$

其中:f_i 是一个多元函数,定义了负载指标 $q_y(q_y\in Q')$ 的取值($value(q_y)$)与一个或多个其它指标取值($\{value(q_{xj})\mid q_{xj}\in Q'-\{q_y\}\}$)之间的函数关系,这意味着当所有 q_{xj} 的取值确定后 q_y 的取值可由函数 f_i 确定.此外,还要求对 Q' 中的每一个指标都存在至少一个函数与之对应.

定义 3. 区间约束组. $dcg=\langle Q',DC\rangle$,其中 $Q'=\{q_i\mid q_i\in LQM^{\Theta}.Q\}$,定义了该约束组所约束的指标集合,$Q'\subseteq LQM^{\Theta}.Q$;$DC=\{dc_1,dc_2,\cdots,dc_n\}$,$dc_i$ 定义了负载指标 $q_y(q_y\in Q')$ 的取值区间($domain(q_y)$)与一个或多个其它指标取值($\{value(q_{xj})\mid q_{xj}\in Q'-\{q_y\}\}$)之间的约束关系,这意味着当所有 q_{xj} 的取值确定后 q_y 将得到一个新的取值区间 d'_y,并且有 $d'_y\subset LQM^{\Theta}.d_y$.在下文中,我们使用 $domain'(q)$ 表示指标 q 在区间约束下的取值区间,当不存在对 q 的区间约束时,$domain'(q)=domain(q)$.

LQM 模型中不仅包括了指标集合和指标约束,还给出了指标的分类.对指标进行分类不仅有助于对指标进行管理,同时对于确定指标的控制方式也具有重要的作用,在负载产生时对于同一类别的指标通常可采用相同的负载控制方式.目前,本文定义了 6 种指标类别,如表 1 所示.

为了能够确定负载指标的控制方式及其与控制点的映射关系,*LQM* 模型对各指标与类别的隶属关系做出如下约束:

$$\begin{cases}(\forall q)(q\in LQM^{\Theta}.Q)\rightarrow\\ \quad(\exists\langle q,c\rangle)(\langle q,c\rangle\in LQM^{\Theta}.R),\\ (\langle q,c_1\rangle\in LQM^{\Theta}.R\wedge\langle q,c_2\rangle\in LQM^{\Theta}.R\rangle)\rightarrow\\ \quad(c_1=c_2).\end{cases}$$

负载指标模型是与 SUT 的类型 Θ 相关的.对于不同的 SUT 类型,测试使用的负载指标也会不同.例如,对于 Web 系统测试而言,并发用户数和请求间隔时间是常用的负载指标;而对于数据存储系统的测试,数据吞吐量则成为了一个重要的指标.在实验部分,本文给出了 *LQM* 模型针对 Web 应用测试时的实例及其分析.

表 1　负载指标的主要类别

类别	描述	负载指标示例
c_1访问分布	定义了在负载产生时各类用户或消息的比例，或者访问不同对象或资源的比例	用户类型分布 消息类型分布 资源类型分布
c_2访问规模	定义对 SUT 访问的总体数量，可作为负载产生的停止条件	用户总数 事务总数
c_3并发规模	定义在负载产生时必须达到的并发访问数量	并发用户数 并发连接数
c_4访问时间	定义了与时间有关的因素，又可具体分为间隔时间类和持续时间类	消息到达时间 连接持续时间
c_5数据规模	定义了 SUT 的收发数据规模，或者单位时间内的数据流量	访问文件大小 数据吞吐量
c_6其它	不能列入上述类别的负载指标，这些指标通常没有直接对应的控制点，因而需要采用间接控制方式进行模拟(见下文)	会话平均长度

3.2　任务指标集选取

在一个具体的测试项目中，测试人员通常不会使用 LQM 中的全部指标来定义负载，而会根据测试任务的要求，从 LQM 中选取部分指标，重点检验 SUT 在这些指标所描述的负载下的性能表现. 例如，在进行 Web 系统压力测试时，并发用户数和会话发生间隔就是主要关注的指标. 本文将根据测试任务从 LQM 中选取的指标集合定义为任务指标集(Task oriented Quantity Set，TQS)：

$$TQS^{\Phi} = \{q_i \mid q_i \in LQM^{\Theta}.Q\}$$

每个 TQS 都是和测试任务 Φ 相关的，其中的指标取值($value(q_i)$)可以由测试人员根据任务 Φ 的需要进行设置.

但由于 LQM 中存在着值约束，导致负载指标之间的取值不是完全独立. 在某些情况下，对非独立的指标进行设置会造成冲突. 例如，当已经设置了会话内请求间隔和会话长度之后，再对会话持续时间进行设置就会破坏如下约束，从而导致取值冲突.

会话持续时间＝会话内请求间隔×(会话长度－1)

这种冲突的出现会对测试执行阶段的负载控制造成影响，导致控制不收敛，从而无法产生出测试人员所期望的负载场景.

为了避免取值冲突的出现，本文提出了一个指标选取算法，来辅助测试人员从 LQM 中选择相互独立的负载指标以生成 TQS.

算法 1.　TQS 选取算法.

输入：

　LQM^{Θ}　　//负载指标模型

输出：

　TQS^{Φ}　　//任务指标集

　RQS^{Φ}　　//关联指标集

初始数据：

　$CS = LQM^{\Theta}.Q$，$RQS^{\Phi} = \varnothing$，$TQS^{\Phi} = \varnothing$

```
begin
  while CS≠∅
  begin
    GetUserInput()→in
    if in∈CS then //用户从 CS 中选择了一个指标
      CS−{in}→CS
      TQS^Φ ∪ {in}→TQS^Φ
      //针对新增加的指标查找可能的相关指标
      for each vcg∈LQM^Θ.CG ∧ in∈vcg.Q′
      begin
        if (∃q)(q∈vcg.Q′ ∧ q∉TQS^Φ ∪ RQS^Φ ∧
            vcg.Q′−{q}⊆TQS^Φ ∪ RQS^Φ)
          then SelectAsRQS(q)
      end
    else if in=finish then //用户完成了选取
      return TQS^Φ with RQS^Φ
  end
  //CS 中已经无指标可选
  return TQS^Φ with RQS^Φ
end

SelectAsRQS(in) //选择相关指标
begin
  CS−{in}→CS
  RQS^Φ ∪ {in}→RQS^Φ
  for each vcg∈LQM^Θ.CG ∧ in∈vcg.Q′
  begin
    if (∃q)(q∈vcg.Q′ ∧ q∉TQS^Φ ∪ RQS^Φ ∧
        vcg.Q′−{q}⊆TQS^Φ ∪ RQS^Φ)
      then SelectAsRQS(q)
  end
end
```

TQS 选取算法首先建立了 3 个集合，分别是候选指标集 CS、任务指标集 TQS 和相关指标集 RQS. 其中，CS 初始时包含了 LQM 中的所有待选指标，每次测试人员选中的指标都会从 CS 中去掉；TQS 初始时为空集，随后每次选中的指标都会加入到该集合；RQS 包括了可能与 TQS 中的指标产生取值冲突的负载指标. 这些指标的选取由本算法自动完成，并将随 TQS 一起返回. 本算法选取 RQS 指标的规则是，若一个指标与本次显式选中的指标之间存在值约束，并且除该指标外约束组中的其它指标均已属于 TQS 或 RQS，则将该指标加入 RQS. 具体的选取过程由函数 SelectAsRQS 完成.

TQS 和 RQS 中的指标具有如下性质：

(1) TQS 中所有指标的取值相互独立，因而可在不违背其值域(或区间约束)的前提下任意设置；

(2) *RQS* 中的指标与 *TQS* 中的部分指标之间存在着取值冲突,因而不能对其中的指标进行设置;

(3) 当 *TQS* 中的全部指标取值确定后,*RQS* 中的全部指标取值可根据相应的值约束函数计算得出.

3.3 负载剖面模型

在确定了 *TQS* 之后,就可以设置其中指标的取值,以描述负载场景.在一个测试任务中,通常需要不同的负载场景,其中的指标取值应有所变化,从而检验 SUT 在多种负载水平下的性能表现.例如,为了测试 Web 系统对并发用户的承受能力,测试人员应对并发用户数指标设置多组期望值,以模拟并发用户数量从低到高的不同水平,从而检查 SUT 的性能变化并发现其承受极限.为此,本文定义了负载剖面模型,以描述任务指标集中所有负载指标的取值以及其取值随时间变化的情况.该模型的形式化定义如下.

定义 4. 负载剖面模型 $LPM=\langle I,T\rangle$,其中,$I=\{\langle q_i,v_i\rangle \mid q_i\in TQS^{\Phi},v_i\in domain'(q_i),1\leq i\leq |TQS^{\Phi}|\}$,并且$(\forall q)(q\in TQS^{\Phi})\rightarrow(\exists\langle q,v\rangle)(\langle q,v\rangle\in LPM.I)$,$\langle q_i,v_i\rangle$是一个二元关系,表示将指标 q_i 的值设置为 v_i,v_i 既可以是确定值也可以是随机变量,对后者要给出其分布及参数,集合 I 代表了 *LPM* 模型中的全部指标初始值,必须保证对 TQS^{Φ} 中每一个指标在 I 中均有对应的$\langle q,v\rangle$;$T=\{\langle q_x,v_x\rangle_{\Delta t(j)} \mid q_x\in TQS^{\Phi},v_x\in domain'(q_x),1\leq j\leq m\}$,$T$ 是一个时间序列,$\Delta t(j)$代表一个时间片段,$\langle q_x,v_x\rangle_{\Delta t(j)}$表示从第 $j-1$ 个时间点开始经过了 $\Delta t(j)$时间之后,指标 q_x 的取值将被修改为 v_x.m 是指标变化时间点的个数.

负载剖面模型 *LPM* 从两个方面定义了对负载场景的期望,进而对测试系统的负载生成进行指导.一方面 *LPM.I* 定义各项负载指标的初始期望值,这代表了在测试初始时测试系统所要产生出的负载水平;另一方面 *LPM.T* 定义在这之后指标随时间变化的情况,测试系统应据此对负载水平进行调整.由于有 *LPM.T* 的存在,*LPM* 可以更好地反映真实环境下的负载特征,同时也能够检验 SUT 在负载动态变化过程中的表现,即 SUT 对负载变化的适应能力.在相邻两次变化之间(即 Δt 内),*LPM* 中的指标值保持不变,测试系统也将稳定在相应的负载水平上.

在 *LPM* 建立以后,不仅 *TQS* 中的所有指标取值已经确定,而且由于值约束关系的存在,相关 *RQS* 中的指标取值也可以由此确定.基于这两组指标数据,本文将通过负载控制点和指标控制映射,实现 *LPM* 所描述的负载场景的产生.

4 负载产生方法

LPM 模型定义了性能测试中的负载场景.在测试执行阶段,测试系统需要依据这些模型产生出期望的负载.TTCN-3 测试系统采用基于虚拟用户的负载产生方式,其具体测试系统构成如图 2 所示.

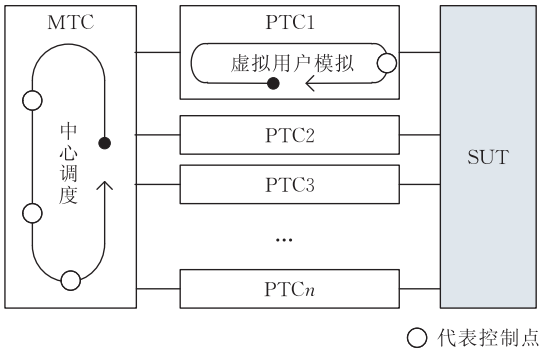

图 2 TTCN-3 测试系统构成

该系统包括了两类 TTCN-3 测试构件:MTC (Main Test Component) 和 PTC (Parallel Test Component).根据 TTCN-3 标准[13]的规定,在一个 TTCN-3 测试系统中有且仅有一个 MTC,该构件在测试用例启动时自动创建.PTC 用来描述并发测试行为,可根据测试的需要动态创建.同一时刻存在于系统中的多个测试构件(包括 MTC 和 PTC)将并行执行.在性能测试中,每个 PTC 代表一个虚拟用户,用于模拟对 SUT 的一个典型交互过程.MTC 则作为中心调度程序,用于控制各个 PTC 的启动、并发和执行过程,从而产生出满足负载模型定义的负载场景.这些控制的具体实现依赖一组负载控制点 (Control Point,CP),每个控制点能够以某个负载指标值作为参数,对 MTC 或 PTC 的某方面行为进行精确调控,从而使其产生出的负载能够达到相应指标的要求.

如上文所述,负载生成的作用是产生 PTC 上的虚拟用户模拟程序以及指标与控制点的控制关系.对于前者,我们在文献[14]中已有论述.本文则重点讨论如何依据负载模型动态选择各指标的控制方式,规划负载指标与控制点的映射关系.下面,首先给出负载控制点的概念,然后给出控制点规划算法.

4.1 负载控制点

本文将控制点定义为:TTCN-3 测试系统中,能

够依据 LPM 中负载指标的取值，对 MTC 或 PTC 上的测试行为进行干预，从而使测试系统所呈现的外部行为满足 LPM 负载指标要求的程序.

从定义可以看出，控制点是与 LPM 模型中的指标关联在一起. 但是，由于在本文提出的负载模型中，指标不是固定的，而是可根据 SUT 类型进行调整的，因而不能根据具体的指标来设计控制点. 为此，本文采用了根据指标类型设计控制点的方法. 由于同类指标具有相同的特性，因此可采用相同的控制方法. 此外，由于 TTCN-3 是一种标准化的系统，因此，负载控制点的定义还需要满足 TTCN-3 标准对测试系统结构和接口的约束. 基于这些约束，本文为不同类别的指标设计了 6 种负载控制点：

① 用户选择控制点. 这是针对访问分布类指标(c_1)的控制点，位于 MTC 中，以外部函数[13]的形式实现. 该控制点的作用是，以负载指标给出的分布作为指导，选择要在 PTC 上创建的用户类型，并将选出的用户类型返回给 MTC 调度程序.

② PTC 创建控制点. 该控制点用于控制创建 PTC 的时机，位于 MTC 中，以 TTCN-3 函数的形式实现. 该控制点可依据两类指标控制 PTC 的创建，一种是依据并发规模类指标(c_3)进行控制，另一种是依据时间间隔类指标(c_4)进行控制.

③ 全局发送控制点. 这是针对访问时间类指标(c_4)的控制点，位于 TRI 接口的 send 函数[13]内. 该控制点的作用是，根据负载指标给出的时间值，控制测试系统向 SUT 发送消息的间隔. 该控制点不区分消息来自哪个 PTC，只从全局的角度控制消息的发送时机.

④ PTC 延迟控制点. 这是针对访问时间类指标(c_4)的控制点，位于 PTC 中，利用 TTCN-3 的计时器机制实现. 该控制点的作用是，根据负载指标给出的时间值，在虚拟用户的两个相邻活动之间增加延迟，从而对 SUT 模拟出时间间隔或持续时间的效果.

⑤ 数据模拟控制点. 这是针对数据规模类指标(c_5)的控制点，位于 PTC 中，以外部函数的形式实现. 该控制点可根据指标给出的数据大小或流量，向 SUT 发送相应的数据流，从而模拟出负载指标规定的数据规模.

⑥ 负载停止控制点. 这是负载产生中必不可少的一类控制点，位于 MTC 中，以 TTCN-3 函数的形式实现. 该控制点根据访问规模类指标(c_2)中给出的总量，决定 MTC 何时终止负载产生.

本文设计的这 6 种控制点全部符合 TTCN-3 标准的规定，每种控制点都可根据负载产生的需要在 TTCN-3 测试系统中设置任意数量的实例. 但其前提是，这些控制点的引入不会违反控制约束. 控制约束是指一个控制点的引入会对另一个控制点产生影响，导致另一个控制点(或者二者都)无法模拟出对应指标所要求的负载. 例如，请求发生间隔和会话内请求间隔是对 Web 系统负载建模时常使用的两个指标. 二者的作用都是对 HTTP 请求的发送时间进行控制. 如果我们在一次负载产生中同时引入全局发送控制点和 PTC 延迟控制点，来对这两个指标进行控制，那么就会违背控制约束，引发控制冲突. 由于二者试图同时对一个请求的发送时间进行调整，其结果必然是该消息的发送时间无法满足任何一个指标的要求.

4.2　负载指标与控制点映射

在定义了负载模型之后，就需要建立模型中的各项指标与控制点之间的映射关系，确定每个指标由哪个或哪些控制点进行控制. 这种映射关系的建立需要分 3 个步骤完成：

(1) 当 LQM^{Θ} 建立之后，需要为 $LQM^{\Theta}.Q$ 中的每个指标确定一个能够对其进行直接控制的控制点. 这些控制点总体上可依据指标类型决定，但对于那些对应关系不唯一或控制点使用方式不固定的指标，则需要由测试人员根据经验指定. 例如，访问时间类指标(c_4)既可以由 PTC 延迟控制点控制，也可以由全局发送控制点控制，同时对于前者在 PTC 中插入的位置也要根据虚拟用户的具体行为决定，因此，这类指标的控制点就必须要依靠人工确定. 表 2 给出了不同类型指标的控制点确定方式.

(2) 在确定了直接控制点之后，测试人员还要根据经验分析不同指标的控制点之间是否存在控制约束，指出哪些指标的控制点不能同时使用，从而建立一个控制约束集合.

(3) 当 TQS^{Φ} 和 LPM 建立之后，以步骤 1 中确立的直接控制关系为基础，针对 TQS^{Φ} 中具体使用到的指标，检验这种控制关系是否能够成立. 有如下 3 种情况会导致可能无法使用这种直接控制：① 指标没有直接对应的控制点. 例如，隶属于“其它”类别的负载指标就没有直接的控制点，因此，无法建立起上述对应关系；② 多个指标使用同一个控制点. 由于每个控制点只能根据一个指标进行控制，因此，这些指标无法同时通过该控制点进行控制；③ 多个指标所依赖的控制点之间存在控制约束. 在这种情况

下,这些指标在负载产生时无法同时进行控制,否则可能会引发控制冲突.对于无法直接进行控制的情况,就要尝试采用间接控制方式.本文提出了一种控制点规划算法,为 TQS^{Φ} 中的每个指标自动选择适合的控制方式,通过合理的规划,充分利用有限的控制点,完成对多指标的负载控制.

4.3 控制点规划算法

针对负载指标没有直接对应控制点的问题,本文提出了一种间接控制方式:若一个指标 q 无法直接进行控制,但存在 q 的一个值约束,并且其中除 q 外的其它指标均有直接控制点,那么可通过对这些指标进行控制,使 q 达到预期值.例如,虽然我们不能对会话长度进行直接控制,但可以通过控制虚拟用户类型分布来使会话长度的均值达到预期.

针对多个指标共享控制点及存在控制约束的问题,可通过合理规划,使得两个指标能够同时使用.例如,对于请求发生间隔和会话内请求间隔这两个指标,虽然它们的控制点存在控制约束,无法同时进行直接控制,但我们可以选择对后者直接控制,而对前者间接控制的方式,令这两个指标能够被同时满足.

表 2 不同类型指标的控制点确定方式

指标类别	可能的控制点	使用方式	确定方式
c_1	用户选择控制点	固定	自动
c_2	负载停止控制点	固定	自动
c_3	PTC 创建控制点	固定	自动
	PTC 创建控制点	固定	
c_4	全局发送控制点	固定	人工
	PTC 延迟控制点	不固定	
c_5	数据模拟控制点	不固定	人工
c_6	无	N/A	无控制点

综上所述,当确定了任务指标集 TQS 之后,需要提供一种机制,在给出 LQM 中各指标的直接控制点和控制约束的条件下,规划 TQS(及 RQS,其指标可用于间接控制)中各个指标的控制方式,从而建立指标控制映射集.在能够找到多个方案的情况下,应选择控制点个数少的作为优选方案.为此,本文提出了一种控制点规划算法.下面,本文首先给出指标控制映射集的定义,然后再介绍该规划算法.

定义 5. 指标控制映射集是一个二元关系 $Q2C=\{\langle q_i,CP_j\rangle \mid q_i\in TQS^{\Phi},CP_j\subseteq CPS\}$,其中,$\langle q_i,CP_j\rangle$ 定义了测试任务 Φ 的任务指标集中的指标 q_i 与 TTCN-3 负载控制点集合的子集 CP_j 之间的映射关系,在负载生成阶段 q_i 的产生需依赖 CP_j 中的控制点进行控制.

用于生成 $Q2C$ 的控制点规划算法如下所示,具体包括 3 个步骤:

(1)对 TQS 和 RQS 中的指标进行分类,其结果为 3 个类别:直接控制指标集 DS、间接控制指标集 IS 和未确定控制方式指标集 PS.具体分类规则为,对于独占一个直接控制点且不存在控制约束的指标加入 DS,对于不可能进行直接控制的指标(如属于"其它"类别的指标)加入 IS,其它指标则加入 PS.RQS 中的指标全部加入 PS.

(2)确定 PS 中指标的具体控制方式.本文采用约束优化方法,首先建立约束系统,然后再对约束进行求解(该解使目标函数最小化),从而确定各指标的最优控制方式.约束系统由一系列约束条件构成,一个可行解必须满足其中的所有约束条件.本文所定义的约束条件是从控制点数量的角度描述的,其建立具体遵循了如下 3 条规则.

规则 1. 对于所有涉及未确定指标的值约束组 vcg,构造如下约束条件:

$$\sum \#cp(q_i)\le |\ vcg.Q'\ |-1.$$

其中,$q_i\in vcg.Q'$,$\#cp(q_i)$ 是一个 0-1 函数,定义了与指标 q_i 相关的直接控制点个数,当 q_i 有直接控制点时,其值为 1,反之为 0.该约束条件的含义是不能对一个值约束组中的所有指标都进行直接控制,以免出现控制冲突.

规则 2. 对于所有控制约束 cc,构造如下约束条件:

$$\sum \#cp(q_i)\le 1.$$

其中,$q_i\in cc$,其含义是一个控制约束中至多只能存在一个直接控制的指标,其它都需要间接控制或不控制.

规则 3. 对于所有已经确定和可能的间接指标 q_x,为其所在值约束组集合 $G=\{vcg_y\mid q_x\in vcg_y.Q'\}$ 构造如下约束条件:

$$\bigvee_{vcg_j\in G}\Big(\sum \#cp(q_i)=|\ vcg_j.Q'\ |-1\Big),\ \#cp(q_x)=0.$$

该约束条件是一组约束等式的析取.其中,每个等式对应 G 中的一个值约束组 vcg_j,在该等式中 $q_i\in vcg_j.Q'$.整个约束条件的含义是,如果指标 q_x 采用间接控制方式,则与 q_x 有关的所有值约束组中至少有一个能够满足上文提出的间接控制条件.

(3)按上述方式构造的约束系统属于 GCOP (Generalized Constrained Optimization Problem)约束系统,可采用文献[15]提出的方法对其进行求解,

所使用的目标函数为 $\min(\sum \sharp cp(q_i)), q_i \in PS$，即将控制点最少的方案作为最优解. 若找到最优解，可据此确定 PS 中指标的控制方式，并建立指标控制映射. 若找不到解，则证明在当前的测试平台上测试人员选择的 TQS 无法执行，应该重新调整 TQS.

算法 2.　控制点规划算法.

输入：

TQS^{Φ}　　//任务指标集

RQS^{Φ}　　//关联指标集

输出：

$Q2C$　　//指标控制集

初始数据：

$DS=\varnothing$, $IS=\varnothing$, $PS=\varnothing$, $Q2C=\varnothing$

```
begin
  //对 TQS 中的指标进行分类
  for each q∈TQS^Φ
  begin
    if (∃cp)(cp∈CPS∧cp --直接控制--> q) then
      if q∈控制约束 then
        PS∪{q}→PS      //未确定控制方式
      else
        DS∪{q}→DS      //确定为直接控制
    else
      IS∪{q}→IS   //确定为间接控制
  end
  //将 RQS 中的指标全部加入 PS
  PS∪RQS^Φ→PS

  //建立约束系统并进行约束求解
  BuildCS()→constraints
  FindSolution(constraints) → solution

  //根据最优解确定 PS 中指标的控制方式
  for each q∈PS∧q∈TQS^Φ
  begin
    if solution(q)=1 then
      PS−{q}→PS
      DS∪{q}→DS
      ⟨q,cp_r⟩∪Q2C → Q2C   //建立直接控制映射
    elseif solution(q)=0 then
      PS−{q}→PS
      IS∪{q}→IS
      ⟨q,CP_r⟩∪Q2C→Q2C   //建立间接控制映射
  end
end
```

5　案例分析

为了对本文提出的负载模型及负载产生方法进行验证，我们以一个开源 Web 应用作为 SUT 开展了实验. 该 Web 应用是一个网上购书系统，基于 JSP 实现主要业务逻辑，后台采用 MySQL 数据库. 本文为该 Web 应用定义了四类虚拟用户，分别对应四种典型的使用场景，如表 3 所示. 同时，我们基于 Eclipse 平台开发了一个负载建模和生成原型工具，并基于 TTworkbench TTCN-3 平台[①]实现了所有控制点. 前者生成的 TTCN-3 测试系统可在后者中运行，从而产生出测试所需的负载.

以该 Web 应用作为 SUT，我们设计了 3 组实验来检验本文所提出的模型及算法的有效性. 其中：第 1 组实验重点检验了负载指标的建模和 TTCN-3 控制点对单项指标的控制能力；第 2 组实验则用负载剖面模型定义了更加复杂的测试场景，并在该场景下对控制点规划和指标间接控制能力进行了验证. 最后，在前两组实验的基础上，我们还进行了本文方法与 LoadRunner 的对比实验.

表 3　虚拟用户及对应场景

虚拟用户	使用场景	会话长度
vu_1	仅浏览书店主页	2
vu_2	进行新用户注册	3
vu_3	查询一本特定图书	4
vu_4	购买已选择的图书	6

5.1　第 1 组实验

在该实验中，本文首先建立了面向 Web 系统的负载指标模型. 现有的 Web 负载模型主要关注了两个方面的特征：网络特征和用户行为特征. 我们为了保证负载指标的全面性，并检验 LQM 模型的描述能力，本文从两个方面分别选取了一组典型指标，建立了面向 Web 系统的负载指标模型 LQM^{Web}，其中所包含的指标如表 4 所示.

表 4　面向 Web 系统测试的负载指标

Q	指标	类别
q_1	虚拟用户类型分布	c_1访问分布
q_2	请求类型分布	c_6其它
q_3	会话长度	c_6其它
q_4	会话持续时间	c_6其它
q_5	会话内请求间隔	c_4访问时间
q_6	并发用户数	c_3并发规模
q_7	请求发生间隔	c_4访问时间
q_8	会话发生间隔	c_4访问时间
q_9	思考时间	c_4访问时间
q_{10}	虚拟用户总数	c_2访问规模

上述指标之间存在着本文所提出的两类约束关

① http://www.testingtech.de/products/ttwb_intro.php

系.例如,会话持续时间(q_4)、会话内请求间隔(q_5)和会话长度(q_3)之间存在着值约束关系:$value(q_4)=value(q_5)\times(value(q_3)-1)$.又如,会话内请求间隔($q_5$)和思考时间($q_9$)之间存在着区间约束关系:$domain'(q_9)=[0,value(q_5))$.通过对 LQM^{Web} 的负载指标进行分析,并且通过测试进行验证,我们定义了指标值约束组和区间约束组,如表 5 所示.

表 5 面向 Web 系统测试的指标约束

CG	Q'	F/DC
vcg_1	$\{q_3,q_4,q_5\}$	$f_1: value(q_3)=value(q_4)/value(q_5)+1$ $f_2: value(q_4)=value(q_5)\times(value(q_3)-1)$ $f_3: value(q_5)=value(q_4)/(value(q_3)-1)$
vcg_2	$\{q_4,q_6,q_8\}$	$f_1: value(q_4)=value(q_6)\times value(q_8)$ $f_2: value(q_6)=value(q_4)/value(q_8)$ $f_3: value(q_8)=value(q_4)/value(q_6)$
vcg_3	$\{q_3,q_7,q_8\}$	$f_1: value(q_3)=value(q_8)/value(q_7)$ $f_2: value(q_7)=value(q_8)/value(q_3)$ $f_3: value(q_8)=value(q_7)\times value(q_3)$
vcg_4	$\{q_3,q_5,q_6,q_7\}$	$f_1: value(q_3)=1/(1-value(q_6)\times value(q_7)/value(q_5))$ $f_2: value(q_5)=value(q_6)\times value(q_7)\times value(q_3)/(value(q_3)-1)$ $f_3: value(q_6)=value(q_5)/value(q_7)\times(value(q_3)-1)/value(q_3)$ $f_4: value(q_7)=value(q_5)/value(q_6)\times(value(q_3)-1)/value(q_3)$
dcg_1	$\{q_5,q_9\}$	$dc_1: domain'(q_9)=[0,value(q_5))$

为了检验本文提出的 TTCN-3 负载控制点对单项指标的控制能力.我们为 LQM^{Web} 中每个可直接控制的指标,分别定义了代表不同负载水平的取指,并使用相应的负载控制点进行了负载产生实验.在负载产生后,我们通过分析被测 Web 应用的日志信息,比较了从 SUT 侧实际观察到的指标值与期望值的差别,并分析了控制的误差.具体实验结果如表 6 所示.

表 6 单指标控制实验结果

Q	设置值	实测值	平均误差/%
q_1	vu_1:25%, vu_2:25% vu_3:25%, vu_4:25%	vu_1:25%,vu_2:25.01% vu_3:25%,vu_4:24.99%	0.032
	vu_1:35%,vu_2:10% vu_3:30%,vu_4:25%	vu_1:34.98%,vu_2:10% vu_3:30%,vu_4:25.02%	
q_5	2.5s 19.8s	2.506s 19.815s	0.158
q_6	15vu 300vu	14.87vu 298.54vu	0.667
q_7	0.5s 15.2s	0.501s 15.213s	0.143
q_8	1.0s 25.1s	1.004s 25.108s	0.216

其中,误差的计算公式为

$$\text{误差}=\frac{|\text{实测值}-\text{设置值}|}{\text{设置值}}\times 100\%.$$

从表 6 中可以看出,在这些指标中,虚拟用户类型分布的误差最低.时间类指标(如会话内请求间隔和会话发生间隔)比前者要稍高一些.这除了测试系统本身的控制误差外,也由于这是基于 SUT 的日志进行的观测,引入了额外的网络和 Web 系统处理时延,因此观测值不如前者准确的缘故.总体而言,本文所提出的 TTCN-3 控制点的控制误差大多小于或接近 0.2%,控制精度较高.唯一的例外是并发用户数.其原因是在负载产生的开始和结束阶段,并发用户数分别有一个逐步提高和归零的过程,我们在计算平均并发用户数时也包括了这两个阶段,因而导致了误差的加大.

5.2 第 2 组实验

在第 2 组实验中,我们使用 LPM 模型描述了两个测试场景.每个场景中均对多个指标的取值及其变化进行了定义.然后,检验在多指标约束下的控制点规划和负载控制能力.

5.2.1 TQS 选择和 LPM 模型

以 LQM^{Web} 为基础,我们选择了两组任务指标.所选出的指标如表 7 所示.其中,第 1 组指标主要体现了访问的整体特征,而第 2 组指标则兼顾了用户行为和整体两个方面的特征.该任务指标的选择是基于 TQS 选取算法完成的,该算法的作用是保证选出的指标取值独立.检验指标取值是否独立的充要条件是:TQS 中的任意指标之间不存在可成立的值约束关系,而可能与它们发生取值冲突的指标则被移到 RQS 中.对照表 5 中的约束组,对表 7 中的选择结果进行检查,我们可以判断其选择结果满足独立性要求.

表 7 任务指标选择结果①

组	TQS	RQS
1	q_1虚拟用户类型分布 q_6并发用户数 q_7请求发生间隔	q_3会话长度 q_4会话持续时间 q_5会话内请求间隔 q_8会话发生间隔
2	q_1虚拟用户类型分布 q_5会话内请求间隔 q_7请求发生间隔	q_3会话长度 q_4会话持续时间 q_6并发用户数 q_8会话发生间隔

对于 Web 系统而言,用户类型的分布是系统使用的一个关键特征,因此,在测试时通常会给予特别的关注.针对这个特点,我们在实验中分两个步骤构建了 LPM 模型.首先,分析了表 3 中 4 类虚拟用户

① 本文的实验均以虚拟用户总数(q_{10})做为停止负载生成的条件,因此 q_{10} 也是 TQS 和 LPM 中的组成指标之一.此处为论述的简洁,从表中略去.同时,由于该指标在负载生成中不会出现控制误差,因此下文也不对其进行误差分析.

访问 SUT 的比例，从而得到指标 q_1 的取值；之后，再根据任务要求的不同负载水平，分别确定其它指标的多组取值；最后，将这些不同的取值作为 *LPM* 中的多个阶段，组成一个完整的 *LPM* 模型. 针对第一个 *TQS* 所建立的 *LPM* 模型如图 3 所示.

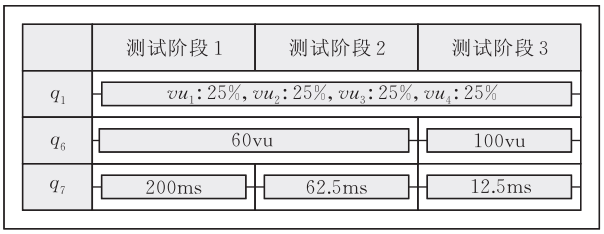

图 3　建立的负载剖面模型

在这个 *LPM* 模型中，根据指标取值的变化测试被分为 3 个阶段. 其中，用户类型分布的取值将在各阶段中保持不变，从而贯穿测试始终；其它指标在不同阶段将采用不同的期望值，以模拟不同的负载水平，例如，请求发生间隔在 3 个阶段的期望值分别为 200ms、62.5ms 和 12.5ms. 这个值的变化说明我们在测试中希望以渐进的方式不断提高用户访问的频度.

5.2.2　控制点规划

在实验中，我们采用控制点规划算法，建立了上述两组任务指标与 TTCN-3 负载控制点之间的映射关系. 其中，由于第 1 组中的指标 q_1、q_6、q_7 均有对应的 TTCN-3 控制点，并且它们之间没有控制约束，所以可全部采用直接控制方式，3 个指标分别由用户选择控制点、PTC 创建控制点和全局发送控制点进行控制. 但对于第 2 组指标，由于其中的指标 q_5 和 q_7 之间存在上文所述的控制约束，因此，需要利用本文提出的控制点规划算法建立起一种间接控制方法. 该规划算法首先根据指标之间的关系建立约束系统. 针对第 2 组 *TQS* 及相关的 *RQS*，所建立的约束系统如下所示：

$$\begin{cases} q_5+q_6+q_7+1\leq 3 \\ q_7+q_8+1\leq 2 \\ q_6+q_8\leq 1 \\ q_5+q_7\leq 1 \\ (q_7=1)\vee((q_7=0)\wedge((q_5+q_6+1=3)\vee \\ \qquad (q_8+1=2))) \\ (q_5=1)\vee((q_5=0)\wedge(q_6+q_7+1=3)) \end{cases}$$

对上述约束系统进行求解，该算法最终规划出指标控制映射关系，如表 8 所示.

表 8　第 2 组指标的控制映射关系

Q	控制方式	对应控制点
q_1	直接控制	用户选择控制点
q_5	直接控制	*PTC* 延迟控制点
q_7	间接控制	用户选择控制点(对应 q_1)，*PTC* 创建控制点(对应 q_8)

5.2.3　负载控制误差分析

基于所建立的 *LPM* 模型和指标控制映射关系，我们对网上购书系统进行了测试，并采用与第 1 组实验相同的方法进行了日志分析. 两组 *LPM* 测试的具体观测结果和误差如表 9 和表 10 所示，其中，*LPM* 模型中各阶段的数据是分别统计的(*LPM* 1 中有 3 个测试阶段，在 *LPM* 2 中我们定义了两个阶段).

表 9　*LPM* 1 的控制结果

Q	阶段	设置值	实测值	平均误差/%
q_1	全部	vu_1:25%, vu_2:25% vu_3:25%, vu_4:25%	vu_1:25%, vu_2:24.98% vu_3:25.02%, vu_4:25%	0.04
q_6	1	60vu	59.815vu	0.49
	2	60vu	59.787vu	
	3	100vu	99.192vu	
q_7	1	200ms	200.094ms	0.176
	2	62.5ms	62.565ms	
	3	12.5ms	12.547ms	

在表 10 中，我们不仅列出了 *LPM* 2 中所有指标的实测值，同时也列出了参与间接控制的指标的实测值. 从中可以看出，间接控制指标的误差具有累积效应，与参与控制的控制点个数(及约束关系)有关. 在本实验中，请求发生间隔(q_7)采用了间接控制方式，其误差值比其它几个直接控制指标表现出明显的升高. 通过这个观察我们可进一步得出，为了更好地控制误差，控制点规划时除了要依据总体控制点数量外，还应关注每个间接控制指标的控制点个数和参与控制点的精度，尽可能优选控制点数量少、精度高的方案.

表 10　*LPM* 2 的控制结果

Q	阶段	设置值	实测值	平均误差/%
q_1	全部	vu_1:35%, vu_3:30%, vu_2:10%, vu_4:25%	vu_1:35%, vu_3:29.98% vu_2:10%, vu_4:25.02%	0.037
q_5	1	11.8s	11.815s	0.108
	2	9.4s	9.409s	
q_7	1	1.6s	1.595s	0.457
	2	0.9s	0.906s	
用于间接控制 q_7 的指标				
q_8	1	5.92s	5.924s	0.123
	2	3.33s	3.336s	

最后，本文对第 1 组实验和第 2 组实验的控制误差进行了对比，如图 4 所示. 通过对比分析可以看

出,在需要多指标协同控制的复杂负载场景下,直接控制指标的误差与单指标控制实验中的误差大致相同,大多数指标的误差也维持在 0.2% 以内.这说明并没有因为需要对多个指标进行协同控制,而导致误差的增大.但间接控制指标的误差稍高,接近 0.5%,这主要是由参与控制的控制点数量及误差累积所决定的.

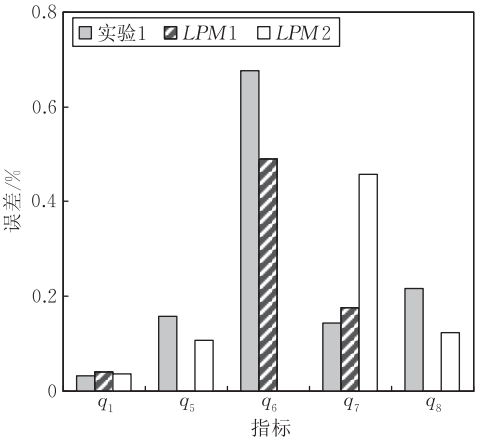

图 4 两组实验的控制误差对比

通过两组实验可以看出,本文所提出的负载建模方法能够根据 SUT 的类型建立由多种指标构成的指标体系,并通过负载剖面模型描述由不同指标所约束的负载场景.所提出的负载生成方法能够较为精确地模拟出模型描述的负载场景,其中所有指标的控制误差均小于 0.7%,主要指标的控制误差小于 0.25%.

5.3 对比实验

本文还进行了与其它负载产生方法的对比实验.LoadRunner① 是目前技术较为成熟、使用较为广泛的一种性能测试工具,其采用的负载描述和产生方法也在测试工程实践中得到了大量应用.本文选择 LoadRunner 9.5 版作为比较的对象,从负载描述能力、负载控制能力和方法的适用性 3 个方面与本文提出的方法进行了对比.

5.3.1 负载描述能力

LoadRunner 支持通过多种指标来刻画负载,并为不同类型的 SUT 定义了不同的指标集合.对于 Web 应用测试而言,表 4 中列出的大部分指标 LoadRunner 均能支持,并能够灵活地设置指标取值(包括指标值随时间变化的情况).但该工具在指标的使用上存在限制.具体来说,LoadRunner 提供了如下两种负载描述方法,每种方法中允许使用的指标有所不同.

(1) Manual Scenario.通过手工设置多个可被 LoadRunner 直接控制的指标来定义负载,这些指标包括虚拟用户分布(q_1)、并发用户数(q_6)、思考时间(q_9)、虚拟用户总数(q_{10})等;

(2) Goal-Oriented Scenario.以单一指标为目标的负载定义方法,其目标指标通常是 LoadRunner 无法直接控制的指标,因而需要控制其它指标来间接产生能够满足目标要求的负载,目标指标包括每秒点击数(q_7的倒数)、每秒会话数(q_8的倒数)等.

在上述两种描述方法中,前者可支持使用多指标刻画负载,但所使用的指标仅限于 LoadRunner 能够直接控制的指标;后者虽然可以使用间接控制指标,但每次只能指定一个指标作为目标,且目标指标也不能与其它直接或间接控制指标(除 q_1 外)同时使用.由此可见,与本文提出的方法相比,LoadRunner 在多指标负载描述方面的能力有所不足.正是因为这个原因,本文第 2 组实验中给出的两种负载场景均无法使用 LoadRunner 完整描述.其中,在第 1 场景中,由于 q_7 是间接控制指标,需要采用 Goal-Oriented Scenario 方式进行定义,因而不能与 q_6 同时定义;在第 2 场景中,除 q_7 的问题外,LoadRunner 也不支持对会话内请求间隔(q_5)进行定义.

5.3.2 负载控制能力

针对上述两种负载描述方法,LoadRunner 提供了两种与之对应的负载控制方法.对于 Manual Scenario,该工具依据每个给定的指标值直接控制负载;对于 Goal-Oriented Scenario,该工具采用与本文间接控制类似的方法,依据目标指标给出的期望值,通过改变并发用户数、会话间隔时间等直接控制指标来动态调整负载,直到满足该目标指标的要求.

为了与本文的负载控制效果进行对比.本文使用 LoadRunner 在同样的实验环境下,针对 *LPM* 1 中的负载指标,采用 3 组不同的取值进行了实验.由于 LoadRunner 不能同时对 q_6 和 q_7 进行定义,因此,我们在实验中仅保留了 q_1 和 q_7 两个指标,其中,前者属于直接控制,后者为面向目标的间接控制.具体指标取值和观测结果如表 11 所示.从中可以看出,q_7 的控制误差与本文方法在实验 2 中的数值接近,q_1 的误差略高.总体而言,LoadRunner 与本文方法的负载控制误差处于相当水平.此外,表 11 中还给出了 q_6 的观测值.由于 LoadRunner 无法对其进行控制.因此,该值在实验中的变化幅度较大.

① http://en.wikipedia.org/wiki/HP_LoadRunner

表 11　LoadRunner Goal-Oriented Scenario 控制结果

Q	阶段	设置值	实测值	平均误差/%
q_1	1	vu_1:25%,vu_2:25% vu_3:25%,vu_4:25%	vu_1:25.89%,vu_2:25% vu_3:25%,vu_4:24.11%	1.588
	2	vu_1:13%,vu_2:20% vu_3:27%,vu_4:40%	vu_1:13.33%,vu_2:20% vu_3:26.67%,vu_4:40%	
	3	vu_1:60%,vu_2:20% vu_3:10%,vu_4:10%	vu_1:60%,vu_2:20% vu_3:11.1%,vu_4:8.9%	
q_7	1	125ms	126.103ms	0.474
	2	125ms	125.047ms	
	3	125ms	125.628ms	
q_6	1	N/A	56vu	N/A
	2		61vu	
	3		45vu	

表 12　使用指标约束后的控制结果①②

Q	阶段	设置值	实测值	平均误差/%
q_1	1	vu_1:25%,vu_2:25% vu_3:25%,vu_4:25%	vu_1:25%,vu_2:25% vu_3:25%,vu_4:25%	0.674
	2	vu_1:13%,vu_2:20% vu_3:27%,vu_4:40%	vu_1:13.33%,vu_2:20% vu_3:26.67%,vu_4:40%	
	3	vu_1:60%,vu_2:20% vu_3:10%,vu_4:10%	vu_1:59.3%,vu_2:20.3% vu_3:10.2%,vu_4:10.2%	
q_6	1	60vu	60vu	0.556
	2	60vu	60vu	
	3	60vu	59vu	
q_7	1	125ms	124.735ms	0.108
	2	125ms	125.047ms	
	3	125ms	125.094ms	
用于间接控制 q_7 的指标				
q_9	1	11293ms	N/A	N/A
	2	10480ms		
	3	13130ms		

我们还通过实验表明了使用本文提出的指标约束关系可以改进 LoadRunner 的多指标控制能力. 根据 vcg_4 给出的关于 q_3、q_5、q_6 和 q_7 之间的函数关系，本文转而采用 Manual Scenario 方式，通过直接控制 q_1（通过 q_1 可推算出 q_3）、q_6、q_9（在 LoadRunner 中无法直接控制 q_5，因此我们通过思考时间 q_9 加上一个估计延迟近似模拟 q_5）3 个指标间接影响 q_7，从而达到同时控制 LPM 1 中全部指标的目的. 其中，q_1 和 q_6 属于约束变量，需要根据 LPM 1 的定义进行设置，q_9 属于自由变量. 在负载产生中，我们通过调整 q_9 使 q_7 的值达到期望目标. 本实验的结果如表 12 所示. 与表 11 相比，在 q_1 和 q_7 控制误差没有显著提升的情况下，q_6 的观测值已明显收敛到期望值 60. 该实验表明：本文提出的指标间关系不仅可用于 TTCN-3 负载控制，对于其它方法也具有普适作用. 我们认为，运用负载指标之间的关联关系，并结合多目标控制算法，可作为一种对 LoadRunner 的负载控制算法进行改进的思路.

5.3.3　方法适用性

LoadRunner 的负载控制方法具有一定通用性，可适用于不同类型 SUT 的性能测试. 但是，其虚拟用户模拟脚本的生成采用了“录制与回放”的方式，因此，只能针对该工具可识别的 SUT 类型进行测试. 虽然经过多年的发展，LoadRunner 目前已经可以支持诸如 Web 应用、Web 服务、数据库、流媒体等常见的 SUT 测试，但对于新出现的 SUT 类型（如新型通信协议），该工具可能无法支持.

本文提出的也是一种通用的负载生成方法，其虚拟用户模拟与负载控制是基于 TTCN-3 测试系统的. 由于 TTCN-3 具有标准化、通用化和独立于 SUT 的特性，因此，本文的方法具有较好的适用范围，可适用于不同类型 SUT 的测试. 但是，针对不同 SUT 开发相应的 TTCN-3 虚拟用户模拟程序和适配器程序是应用本方法时测试人员必须要完成的工作. 文献[14]中的方法可以用来提高这项开发的效率并减轻工作量.

6　相关工作讨论

性能测试是 TTCN-3 研究中的热点之一. 近年来得到国内外的广泛关注. 文献[3]针对 TTCN-3 语言在时间特性描述方面的不足，研究了 TTCN-3 的实时扩展方案，提出了一种 Timed TTCN-3 语言，其中引入了面向非功能测试的判决、绝对时间、实时属性的在线和离线评估等机制. 文献[4]则分析了 TTCN-3 在并发测试控制方面的不足，并通过引入信号量等机制增强了测试构件之间的同步能力. 文献[2]分析了 TTCN-3 在测试判定、时间机制、同步机制等方面的局限性，并从核心语言的性能扩展、测试系统自身的性能评估和外部函数的重新设计 3 方面探讨了基于 TTCN-3 的性能测试描述方法，针对如何使用 TTCN-3 进行性能测试给出了建议. 上述研究所提出的扩展机制使得 TTCN-3 能够更好地为性能测试提供支持. 在性能测试系统设计方面，文献[8]提出了一套基于 TTCN-3 的性能基准测试系统，用于检验 IMS(IP Multimedia System)系统在呼叫密度和并发用户量方面的承受能力. Schieferdecker 等人在文献[9]中提出了一种自动化的 TTCN-3 测试框架，该框架可用于 Web Service 的功能和性能测试. 文献[10]也研究了使用 TTCN-3 进行服务化软件性能测试的方法. 蒋凡等人则在一

① 在本实验中用 q_9 近似模拟 q_5 时，估计延迟设为 20ms.
② 思考时间 q_9 在实验中无法观测，因此表中相应位置为 N/A.

致性测试框架的基础上，设计了可用于端到端性能测试的系统——TTPerf[7].

有一些研究关注了基于 TTCN-3 的负载测试问题，文献[5]研究了分布式环境中 TTCN-3 负载测试问题.作者指出 TTCN-3 具备了进行负载测试的能力，可以通过定义并发的测试构件来模拟负载场景，并通过物理分布的测试节点来执行这些构件所定义的行为.但在该文献中，作者的主要关注点是如何对异构的物理测试节点进行调度，从而保证硬件资源的利用率，而并没有关注负载建模的问题.文献[6]也研究了 TTCN-3 负载测试中的调度问题.作者认为负载测试所面临的挑战之一是如何利用比真实系统更少的资源模拟出期望的负载.为此，提出了一种基于有限状态机理论的模型与算法，用于优化对“虚拟线程”的调度，从而提高资源利用率.

综上所述，目前的 TTCN-3 性能测试研究多集中在对语言的扩展、测试系统设计、针对特定系统的测试方面.对于负载建模的研究相对较少.在这些研究所提出的方法中，负载描述还需要测试人员通过繁琐的手工编码实现.而如何通过模型驱动的方式提高 TTCN-3 负载描述和生成的能力及自动化程度正是本文的研究重点之一.此外，上述部分研究还涉及了负载产生的问题，提出了一些用于负载测试的 TTCN-3 系统设计.这为本文的负载控制点设计打下了基础.但这些研究多是针对具体 SUT，所设计的负载生成方法不具备通用性，也无法满足模型驱动方法对负载生成提出的要求.

另一方面，负载建模是传统性能测试研究中的一个重要问题.特别是关于 Web 系统负载模型的研究，目前已经涌现出一定数量的研究成果. SpecWeb99[11]是一套针对 Web 支撑平台的性能基准测试，主要用于对硬件平台和 Web 服务器的性能进行评价与比较.该测试采用了一种基于资源序列的负载模型，将负载视作对 Web 系统中各类资源的访问及其频度，在测试负载产生时，按照访问频度的大小，依次选择相应的资源序列作为工作负载. TPC-W、WebStone、Webbench 等也采取了与之类似的思想[1].这些方法能够从整体特征方面对 SUT 的工作负载进行刻画，但忽略了用户的具体行为对整体负载的影响.为了更好地刻画用户行为特性，文献[12]研究了使用 CBMG (Customer Behavior Model Graph)模型刻画负载的方法.该模型从用户的行为特征出发，将会话模式作为负载特性刻画的主要手段，一个会话模式主要由 3 个特征分量组成：会话长度、会话持续时间、会话期间访问的页面类型，上述特征可通过对 Web 应用服务器的日志文件进行统计分析而得出.文献[16]则从用户提交表单和系统活动之间的迁移关系入手，提出了一种面向表单的随机模型，并使用该模型刻画 Web 系统负载.该模型的主要特点是能够描述历史活动对用户行为的影响.

上述负载研究分别关注了 Web 系统在网络和用户行为两个方面的特征.但对于一个真实的负载而言，这些方面的特征既具有密切的联系又相互补充.因此，一个更加全面的负载模型应该能够从多个方面来描述负载，从整体上反映 SUT 的负载情况.为此，本文提出的负载模型包括了一个可扩展的指标体系，并支持对指标之间的多种约束关系进行定义.同时，为了实现对多指标约束下的负载生成，本文还提出了控制点规划算法，以支持对多类负载指标进行协同控制.

7 结束语

随着 TTCN-3 应用范围的不断扩大，基于 TTCN-3 的性能测试成为了一个重要的问题.但目前 TTCN-3 在描述和生成性能测试所需的负载方面存在明显的不足.本文针对这个问题，提出了一种模型驱动的 TTCN-3 负载产生方法.该方法首先建立负载指标模型以描述面向 SUT 类型的负载指标体系和指标约束关系，然后，通过负载剖面模型刻画了随时间变化的负载场景.所建立的模型能够在转换算法的支持下转换为可执行的 TTCN-3 测试系统，并在本文提出的负载控制点的支持下得以执行，从而模拟出满足模型描述的负载场景.本文通过 3 组实验对所提出的模型的描述能力和负载控制方法进行了验证，并与相关方法进行了对比.

在今后的研究中，我们拟从两个方面进一步完善本文提出的方法.一方面将加强负载指标模型的约束表达能力和负载剖面模型的场景刻画能力.在本文中，我们主要关注的是一些具有确定值的指标，下一步将加强对随机指标及其约束的建模；另一方面将继续完善控制点的设计和控制点的规划算法.主要目标是通过提高控制点的控制精度以及控制规划的合理性，使得所产生的负载能够更好地体现测试人员所关注的真实负载的特征.

参 考 文 献

[1] Deng Xiao-Peng, Xing Chun-Xiao, Cai Lian-Hong. Progress

in testing for Web applications. Journal of Computer Research and Development, 2007, 44(8): 1273-1283(in Chinese)
(邓小鹏，邢春晓，蔡莲红. Web 应用测试技术进展. 计算机研究与发展，2007，44(8)：1273-1283)

[2] Lou Hao, Zeng Hua-Xin. Some recommendations for performance testing using TTCN-3//Proceedings of the 2009 International Conference on Electronic Computer Technology (ICECT 2009). Macau, China, 2009: 249-253

[3] Dai Z R, Grabowski J, Neukirchen H. Timed TTCN-3—A real-time extension for TTCN-3//Proceedings of the 14th IFIP International Conference on Testing of Communicating Systems (TestCom 2002). Berlin, Germany, 2002: 407-424

[4] Song Bo, Li Xiu-Feng, Jiang Chao-Zhe. Extension to TTCN-3 synchronization. Journal of Southwest Jiaotong University, 2005, 40(1): 39-43(in Chinese)
(宋波，李秀峰，蒋朝哲. 对 TTCN-3 同步的扩展. 西南交通大学学报，2005，40(1)：39-43)

[5] Din G, Tolea S, Schieferdecker I. Distributed load tests with TTCN-3//Proceedings of the 18th IFIP International Conference on Testing of Communicating Systems (TestCom 2006). New York, NY, USA, 2006: 177-196

[6] Bozoki F, Csondes T. Scheduling in performance test environment//Proceedings of the 16th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2008). Split-Dubrovnik, Croatia, 2008: 404-408

[7] Jiang Fan, Wan Xiao-Fei. End to end performance testing system using TTCN-3. Computer Science, 2006, 33(11): 29-31(in Chinese)
(蒋凡，万小飞. 使用 TTCN-3 的端到端性能测试系统. 计算机科学，2006，33(11)：29-31)

[8] Din G. An IMS performance benchmark implementation based on the TTCN-3 language. International Journal on Software Tools for Technology Transfer (STTT), 2008, 10(4): 359-370

[9] Schieferdecker I, Din G, Apostolidis D. Distributed functional and load tests for Web services. International Journal on Software Tools for Technology Transfer (STTT), 2005, 7(4): 351-360

[10] Shan Min, Wang Xian-Rong, Zhao Li-Jun, Guo Li-Li. Using TTCN-3 in performance test for service application//Proceedings of the 7th ACIS International Conference on Software Engineering Research, Management and Applications (SERA'09). Haikou, China, 2009: 253-258

[11] Godbole A, Kim Seung-Yun, Guzman R, et al. Performance evaluation studies of client-server models using SpecWeb99 benchmarks//Proceedings of the 2003 IEEE International Conference on Information Reuse and Integration (IRI 2003). Las Vegas, NV, USA, 2003: 406-414

[12] Ballocca G, Politi R, Russo V, et al. Benchmarking a site with realistic workload//Proceedings of the 5th IEEE Workshop on Workload Characterization (WWC-5). Austin, TX, USA, 2002: 14-22

[13] ETSI ES 201 873-1 V4.2.1: Methods for testing and specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language. Sophia-Antipolis, France: European Telecommunications Standards Institute (ETSI), 2010

[14] Qiu Shuo, Wu Ji, Jin Mao-Zhong. TTCN-3 performance testing based on test suite reuse. Computer Science, 2008, 35(11, Special Issue): 117-122, 151(in Chinese)
(邱硕，吴际，金茂忠. 基于测试集复用的 TTCN-3 性能测试. 计算机科学，2008，35(11，专刊)：117-122，151)

[15] Liu Jun-Xiang, Wang Yong-Ji, Wang Yuan, Xing Jian-Sheng, Zeng Hai-Tao. Real-time system design based on logic OR constrained optimization. Journal of Software, 2006, 17(7): 1641-1649(in Chinese)
(刘军祥，王永吉，王源，邢建生，曾海涛. 基于逻辑"或"约束优化的实时系统设计. 软件学报，2006，17(7)：1641-1649)

[16] Lutteroth C, Weber G. Modeling a realistic workload for performance testing//Proceedings of the 12th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2008). Munich, Germany, 2008: 149-158

XU Luo, born in 1976, Ph. D. candidate, senior engineer. His main research interests focus on software testing.

WU Ji, born in 1973, Ph. D., associate professor. His main research interests include software testing and software reliability analysis.

LIU Chao, born in 1958, Ph. D., professor, Ph. D. supervisor, senior member of China Computer Federation. His main research interests include software engineering and software testing.

Background

TTCN-3 (Testing and Test Control Notation version 3) is a language developed and standardized by ESTI (European Telecommunications Standards Institute) to specify and implement black-box testing. With over 10 years of evolution, the language has been widely accepted and successfully applied in many testing fields. Besides the functional testing, the demand for TTCN-3 based performance testing continues to grow. However TTCN-3's ability of implementing per-

formance testing is limited since insufficient support on workload description and generation by TTCN-3 language itself and its supporting tools. Current TTCN-3 performance testing projects, in practice, usually depend on test engineers to manually develop TTCN-3 code to generate workload, which is time consuming and error prone. In conclusion, research in the area of TTCN-3 performance testing is important both for testing itself and improving TTCN-3 language.

This paper proposes a model-driven method for workload generation. The basic idea of our method is to describe workload by model and automatically transfer the model to TTCN-3 test system by proposed algorithms. The results we obtained from a case study show that, the method could significantly improve the capability of workload description and generation, as well as reduce the amount of manual coding.

This work is supported by the Chinese National 863 Target-oriented project (Grant No. 2009AA01Z145) and the Foundation of the State Key Lab for Software Development Environment (Grant No. SKLSDE-2010ZX-15). The goal of these projects is to investigate model-driven testing method and develop supporting tools for Web application testing. Software Engineering Institute of BeiHang University takes charge of these projects and divides the research into three parts, including the core platform, which defines the core meta-model representing the common concepts of Web application testing field and implements basic services for model-driven testing, the test modeling and generation platform, which supports testers to build test model and generate executable TTCN-3 test suite from the test model by model transformation algorithms, and the test execution platform which provides TTCN-3 runtime support for distributed test execution. As an important part of these projects, this research focuses on the performance testing of Web application system and dedicates to utilize model-driven testing method to improve test capability and reduce test cost.