浙 江 大 学

硕士学位论文开题报告

**（专业学位）**

论文题目：基于Web的投保人医疗保险信息系统的性能测试与

分析

姓 名： 王艳灵

学 号： Z134325199

专 业： 软件工程

院 别： 软件学院

导 师： 李善平

2014 年 11月

**目 录**

[1.课题来源及类型 - 1 -](#_Toc120507617)

[2.课题的意义及国内外现状分析 - 1 -](#_Toc120507618)

[2.1投保人医疗保险信息系统概述 - 1 -](#_Toc120507619)

[2.2软件性能测试概述 - 2 -](#_Toc120507620)

[2.3国内软件测试面临的问题 - 3 -](#_Toc120507621)

[2.4投保人医疗保险信息系统性能测试的提出 - 3 -](#_Toc120507622)

[3. 课题的研究目标、研究内容和拟解决的关键问题 - 4 -](#_Toc120507623)

[3.1 课题研究目标 - 4 -](#_Toc120507624)

[3.2课题研究内容 - 5 -](#_Toc120507625)

[3.3 拟解决的关键问题 - 5 -](#_Toc120507626)

[4. 课题的研究方法、设计及试验方案，可行性分析 - 6 -](#_Toc120507627)

[4.1课题的研究方法 - 6 -](#_Toc120507628)

[4.1.1风险因素识别方法 - 6 -](#_Toc120507629)

[4.1.2风险分析方法 - 7 -](#_Toc120507630)

[4.1.3风险评估方法 - 8 -](#_Toc120507631)

[4.1.4风险管理方法 - 8 -](#_Toc120507632)

[4.2课题设计方案 - 9 -](#_Toc120507633)

[4.3课题可行性分析 - 10 -](#_Toc120507634)

[5.课题计划进度和预期成果 - 10 -](#_Toc120507635)

[5.1计划进度 - 10 -](#_Toc120507636)

[5.2预期成果 - 11 -](#_Toc120507637)

[5.2.1政府办公自动化项目风险分析 - 11 -](#_Toc120507638)

[5.2.2风险评估系统原型建设 - 11 -](#_Toc120507639)

[5.2.3风险管理的过程模型 - 11 -](#_Toc120507640)

# 1.课题来源及类型

随着软件行业的发展，计算机软件的使用规模越来越大。整个社会对软件质量的要求越来越高，软件测试成为了提高软件质量的关键保障。软件自动化测试也成为了软件领域无法逾越的发展阶段。基于此，本文结合投保人医疗保险信息系统的特点以及软件测试的管理，研究如何在web项目的开发过程中进行性能测试，以此提高软件开发的整体质量。【重点应该突出—>性能测试，点明本文的主要内容，而且可以增加关于Web开发与性能测试】

# 2.课题的意义及国内外现状分析

## 2.1基于web的投保人医疗保险信息系统概述

随着社会的发展，人们对健康越来越关注，医院和药店的业务不断增长，处理越来越复杂，简单的人工经营管理，已无法满足医院和药店的需要, 成为医院和药店发展的瓶颈， 人们对医疗相关的各项服务的要求也越来越高。传统的手工操作模式,病人帐目繁杂，病人查帐困难, 职工的医疗保健系统和报销的问题出现错误现象严重且无法追踪, 人工传送各种记帐单容易漏、错帐，人工划价收费差错难以避免，因此,如何提高服务水平、提高工作效率，成为迫待解决的问题。因此医疗保险信息系统恰好满足了这些需求。

医疗保险信息系统（MIIS：Medical insurance information management system）是为了使病人获得基本的医疗保险服务，又能给医院和药店提供快速便捷的信息交互功能。这个系统涉及到的部门非常广，如参保单位、银行、税务部门、定点医疗机构、定点零售药店等，各部门的密切配合，使资源得到充分合理的利用，通过与定点医疗机构、定点零售药店建立网络连接，来实现投保人在就诊和购药时，使用医保卡就能实时支付的功能。最后通过建立计算机管理信息系统，实现医保业务处理计算机化。本系统是医院管理信息系统的子系统。在本系统中要实现的系统模块是投保人医疗保险信息系统，主要包括投保人的信息、投保药品、药店、和投保人理赔参数的设置等，达到一个提取信息的功能。另外，由于基于Web服务器的应用系统由于提供浏览器界面而无须安装，大大降低了系统部署和升级成本，得以普遍应用。目前，很多企业的核心业务系统均是Web应用，用户可以享用更丰富的功能和体验，所以，在这里研究软件测试的结合项目是基于web的投保人医疗保险信息系统。

## 2.2软件性能测试概述

性能是指表明软件系统或构件对于其及时性要求符合程度的指标。属于产品的特性范畴，可以用响应时间、吞吐量、每秒点击数等参数指标来进行衡量。性能测试本质上是一项规范，它利用有关产品、人员和过程的信息来减少应用程序、升级程序或修补程序部署中的风险。Web性能测试是借助测试工具来模仿真实用户对Web系统进行施压，在正常和疲劳情况下检测系统的各项指标，预测系统是否满足特定的场景和发现系统瓶颈的过程。主要目的是在并发环境下，评估系统的承受力、发现系统弱点，系统优化和保证系统长时间运行的可靠、安全和稳定。

性能测试可以从侠义和广义两方面考虑，狭义的性能测试是通过模拟生产运行的业务压力或用户使用场景来测试系统的性能是否满足生产性能的需求。广义的性能测试则是压力测试、负载测试、强度测试、并发测试、容量测试、配置测试、可靠性测试等。

1. 压力测试

压力测试是持续加压，直到最大负荷系统没有崩溃的前提下，获取系统性能指标的峰值。压力测试的手段是一点点的增加负荷，直至显现出性能测试的反方向变化，从而确定系统性能最大的承受值。压力测试用工具模拟实现，也可认为是暴力破坏的测试。

1. 负载测试

负载测试与压力测试很相似，也是寻找系统性能指标的极大值。例如500用户点击提交按钮，服务器内存使用率达到98%，可以初步认为内存是瓶颈，为下一步详细判断提供依据。压力测试偏重于施压，负载测试则侧重于连续测试的时间。性能测试时，总是把两种测试方法结合起来。

1. 强度测试

强度测试总是在最大负荷之外对系统施压，检测系统的抗压程度。以预防以外事件的发生，检测系统稳定性与可扩展性。例如，执行50000个用户同时在线，观测资源占用情况，服务器是否崩溃等。疲劳性测试也是一定负荷长时期执行，观察系统反应情况。例如，执行三天三夜的不间断运行。

1. 并发测试

大量的虚拟用户同时间对系统施压，检测多线程条件下软件可能出现的各类问题。并发是性能测试的核心思想，只要是性能测试，基本都和并发有着关系，而且测试工具在并发测试中是不可或缺的。

1. 海量数据测试

对于一些特殊要求的系统进行单用户海量数据测试或者并发的海量数据测试。包括使用混合业务、独立业务、疲劳性和批处理来综合起来完成。海量数据是必须的，例如执行100条数据某个算法运行正常，但是运行100000条数据由于算法缺陷或其他原因，响应时间无法接受。测试时可以借助于自动化测试工具来生产和管理数据。

1. 配置测试

配置测试主要是合理分配系统的软硬件资源使性能最大化来消除瓶颈。如果在某台服务器上有缺陷产生而在另外服务器上无同类缺陷，就很有可能是配置问题了。可以更换系统设备或者控制参数来实现，配置测试需要综合分析和多次试验才能完成。

1. 可靠性测试

在特定的软硬件环境下，特定负载、特定时间允许系统，检测系统是否可靠稳定。长时间运行系统可以发现很多潜在的缺陷，可以集中压力测试，也可以部署后真实环境的测试和破坏性测试。可靠性测试包括软件可靠性和硬件可靠性测试。

各种性能测试方法是相互联系并且可以相互转化的，在实际测试过程中往往是结合一种或几种测试策略综合分析。所以，在性能分析时不可以隔断各种测试方法的相互联系，才能准确、真实、快速的实施性能分析。

其实，性能测试是一种信息的收集和分析过程，过程中收集的数据用来预测怎样的负载水平将耗尽系统资源。性能测试保证程序具有良好的性能，它考察在不同的用户负载下，Web服务对用户请求作出的响应情况，以确保将来系统运行的安全性、可靠性和执行效率。性能测试的主要 目的是为维护系统的性能找到有效的改善策略。性能测试在软件的质量保证中起着重要的作用，其测试内容丰富多样，主要包括三个方面的内容：应用在客户端性能的测试，应用在网络上性能的测试和应用在服务器端性能的测试。通常情况下，三方面有效、合理的结合就可以达到对 系统性能全面的分析和瓶颈的预测。

应用在客户端性能的测试 应用在客户端性能测试的目的是考察客户端应用的性能，其测试的入口是客户端。它主要包括并发性能测试、疲劳强度测试、大数据量 测试等，其中并发性能测试是重点。以下将对这些测试进行介绍。

(1)并发性能测试的过程是一个负载测试和压力测试的过程，即逐渐增加负载，直到系统出现瓶颈或到达不能接受的性能点，通过综合分 析请求响应数据和资源监控指标来确定系统并发性能的过程。 负载测试是确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统组成部分的相应输出项，如吞吐量、响应时间、CPU负 载、内存使用等来决定系统的性能。负载测试是一个分析软件应用程序和支撑架构、模拟真实环境的使用，从而来确定能够接受的性能过程。 压力测试是通过确定一个系统的瓶颈或者不能接受的性能点，来获得系统能提供的大服务级别的测试。进行压力测试的目的是破坏一个 Web服务来观察测试系统的反映。压力测试用来测试系统的限制和故障恢复能力，也就是测试Web应用系统会不会崩溃，在什么情况下会崩溃。 并发性能测试的目的主要体现在三个方面:以真实的业务为依据，选择有代表性的、关键的业务操作设计测试案例，以评价系统的当前性 能；当扩展应用程序的功能或者新的应用程序将要被部署时，负载测试会帮助确定系统是否还能够处理期望的用户负载，以预测系统的未来性 能；通过模拟成百上千个用户，重复执行和运行测试，可以确认性能瓶颈并优化和调整应用，目的在于寻找到瓶颈问题。

(2)疲劳强度测试可以被当作是一个长期的负载或压力测试，它是采用系统稳定运行情况下能够支持的大并发用户数，持续执行一段时 间业务，通过综合分析交易执行指标和资源监控指标来确定系统处理大工作量强度性能的过程。疲劳强度测试可以采用工具自动化的方式进 行测试，也可以手工编写程序测试，其中后者占的比例较大。一般情况下以服务器能够正常稳定响应请求的大并发用户数进行一定时间的疲 劳测试，获取交易执行指标数据和系统资源监控数据。如出现错误导致测试不能成功执行，则需要及时调整测试指标，如降低用户数、缩短测 试周期等。还有一种情况的疲劳测试是对当前系统性能的评估，用系统正常业务情况下并发用户数为基础，进行一定时间的疲劳测试。

(3)大数据量测试可以分为两种类型：针对某些系统存储、传输、统计、查询等业务进行大数据量的独立数据量测试；与压力性能测试、 负载性能测试、疲劳性能测试相结合的综合数据量测试方案。大数据量测试的关键是测试数据的准备，可以依靠工具准备测试数据。

应用在网络上性能的测试 应用在网络上性能测试重点是利用成熟先进的自动化技术进行网络应用性能监控、网络应用性能分析和网络预测。网络应用性能分析的目的 是准确展示网络带宽、延迟、负载和TCP端口的变化是如何影响用户响应时间的。利用网络应用性能分析工具，例如Application Expert，能够发现 应用的瓶颈，同时可知应用在网络上运行时在每个阶段发生的动作行为，并且可以分析应用的问题，因此可用来解决多种问题：客户端是否对数 据库服务器运行了不必要的请求；当服务器从客户端接受了一个查询，应用服务器是否花费了不可接受的时间联系数据库服务器；在投产前预测 应用的响应时间。Application Expert能够让你快速、容易地仿真应用性能，根据终用户在不同网络配置环境下的响应时间，用户可以根据自己的 条件决定应用投产的网络环境。在系统试运行之后，需要及时准确地了解网络上正在发生的事情：什么应用在运行，如何运行；多少PC正在访问 LAN或WAN；哪些应用程序导致系统瓶颈或资源竞争。这时网络应用性能监控以及网络资源管理对系统的正常稳定运行是非常关键的。应用在服务器上性能的测试 对于应用在服务器上性能的测试，可以采用工具监控，也可以使用系统本身的监控命令，如Tuxedo中可以使用Top命令监控资源使用情 况。实施测试的目的是实现服务器设备、服务器操作系统、数据库系统、应用在服务器上性能的全面监控。

性能验证是性能测试中最重要的一个内容， 在本项目中，性能测试的最主要目的之一就是检 测系统当前所处性能水平，验证其性能是否可以 满足未来的应用需求．性能测试关注的是系统的 整体，它和通常所说的强度、压力／负载测试测试 有密切关系．针对本系统的性能测试主要包括执 行效率测试、资源消耗测试、容量测试、网络测试 和稳定性测试等． 1)执行效率测试 主要测试在特定应用的业务逻辑、用户界面、 功能下事务的响应时间，包括服务器事务处理平均响应时间、每秒请求数等指标考察系统在各种情况下的性能表现． 2)容量测试 主要指在事务响应时间可以接受的最低限度 的情况下，系统可以承载的最大业务并发用户数． 3)资源消耗测试 资源消耗测试是借助测试工具对系统在各种 负载下的业务处理时间和系统响应时间进行大数据量的并发和压力测试，依据需求分析和设计文 档提出的性能指标验证软件的符合性，并据此对 系统的性能做出全面的评价． 4)网络效率测试 网络效率通常由网络吞吐量指标来衡量．通过对网络吞吐量的监控，将网络占用带宽和目前网络的带宽比较，可以判断当前网络对整体性能 是否存在瓶颈． 5)稳定性测试 考察系统在一定负载下长时间运行时，系统 的健壮程度．

性能测试类型包括负载测试，压力测试，容量测试等。

负载测试（Load Testing）：负载测试是一种主要为了测试软件系统是否达到需求文档设计的目标，譬如软件在一定时期内，最大支持多少并发用户数，软件请求出错率等，测试的主要是软件系统的性能。

压力测试（Stress Testing）：强度测试也就是压力测试，压力测试主要是为了测试硬件系统是否达到需求文档设计的性能目标，譬如在一定时期内，系统的cpu利用率，内存使用率，磁盘I/O吞吐率，网络吞吐量等，压力测试和负载测试最大的差别在于测试目的不同。

容量测试（Volume Testing）：确定系统最大承受量，譬如系统最大用户数，最大存储量，最多处理的数据流量等。

基准测试

基准测试的关键是要获得一致的、可再现的结果。可再现的结果有两个好处：减少重新运行测试的次数；对测试的产品和产生的数字更为确信。使用的性能测试工具可能会对测试结果产生很大影响。假定测试的两个指标是服务器的响应时间和吞吐量，它们会受到服务器上的负载的影响。服务器上的负载受两个因素影响：同时与服务器通信的连接（或虚拟用户）的数目，以及每个虚拟用户请求之间的考虑时间的长短。很明显，与服务器通信的用户越多，负载就越大。同样，请求之间的考虑时间越短，负载也越大。这两个因素的不同组合会产生不同的服务器负载等级。记住，随着服务器上负载的增加，吞吐量会不断攀升，直到到达一个点。

注意，吞吐量以稳定的速度增长，然后在某一个点上稳定下来。

在某一点上，执行队列开始增长，因为服务器上所有的线程都已投入使用，传入的请求不再被立即处理，而是放入队列中，当线程空闲时再处理。

注意，最初的一段时间，执行队列的长度为零，然后就开始以稳定的速度增长。这是因为系统中的负载在稳定增长，虽然最初系统有足够的空闲线程去处理增加的负载，最终它还是不能承受，而必须将其排入队列。

当系统达到饱和点，服务器吞吐量保持稳定后，就达到了给定条件下的系统上限。但是，随着服务器负载的继续增长，系统的响应时间也随之延长，虽然吞吐量保持稳定。

注意，在执行队列开始增长的同时，响应时间也开始以递增的速度增长。这是因为请求不能被及时处理。

为了获得真正可再现的结果，应该将系统置于相同的高负载下。为此，与服务器通信的虚拟用户应该将请求之间的考虑时间设为零。这样服务器会立即超载，并开始构建执行队列。如果请求（虚拟用户）数保持一致，基准测试的结果应该会非常精确，完全可以再现。

您可能要问的一个问题是：“如何度量结果？”对于一次给定的测试，应该取响应时间和吞吐量的平均值。精确地获得这些值的唯一方法是一次加载所有的用户，然后在预定的时间段内持续运行。这称为“flat”测试。

与此相对应的是“ramp-up”测试。

ramp-up测试中的用户是交错上升的（每几秒增加一些新用户）。ramp-up测试不能产生精确和可重现的平均值，这是因为由于用户的增加是每次一部分，系统的负载在不断地变化。因此，flat运行是获得基准测试数据的理想模式。

这不是在贬低ramp-up测试的价值。实际上，ramp-up测试对找出以后要运行的flat测试的范围非常有用。ramp-up测试的优点是，可以看出随着系统负载的改变，测量值是如何改变的。然后可以据此选择以后要运行的flat测试的范围。

Flat测试的问题是系统会遇到“波动”效果。

注意波动的出现，吞吐量不再是平滑的。

这在系统的各个方面都有所体现，包括CPU的使用量。

注意，每隔一段时间就会出现一个波形。CPU使用量不再是平滑的，而是有了像吞吐量图那样的尖峰。

此外，执行队列也承受着不稳定的负载，因此可以看到，随着系统负载的增加和减少，执行队列也在增长和缩减。

注意，每隔一段时间就会出现一个波形。执行队列曲线与上面的CPU使用量图非常相似。

最后，系统中事务的响应时间也遵循着这个波动模式。

注意，每隔一段时间就会出现一个波形。事务的响应时间也与上面的图类似，只不过其效果随着时间的推移逐渐减弱。

当测试中所有的用户都同时执行几乎相同的操作时，就会发生这种现象。这将会产生非常不可靠和不精确的结果，所以必须采取一些措施防止这种情况的出现。有两种方法可以从这种类型的结果中获得精确的测量值。如果测试可以运行相当长的时间（有时是几个小时，取决于用户的操作持续的时间），最后由于随机事件的本性使然，服务器的吞吐量会被“拉平”。或者，可以只选取波形中两个平息点之间的测量值。该方法的缺点是可以捕获数据的时间非常短。

性能规划测试

对于性能规划类型的测试来说，其目标是找出，在特定的环境下，给定应用程序的性能可以达到何种程度。此时可重现性就不如在基准测试中那么重要了，因为测试中通常都会有随机因子。引入随机因子的目的是为了尽量模拟具有真实用户负载的现实世界应用程序。通常，具体的目标是找出系统在特定的服务器响应时间下支持的当前用户的最大数。例如，您可能想知道：如果要以5秒或更少的响应时间支持8,000个当前用户，需要多少个服务器？要回答这个问题，需要知道系统的更多信息。

要确定系统的容量，需要考虑几个因素。通常，服务器的用户总数非常大（以十万计），但是实际上，这个数字并不能说明什么。真正需要知道的是，这些用户中有多少是并发与服务器通信的。其次要知道的是，每个用户的“考虑时间”即请求间时间是多少。这非常重要，因为考虑时间越短，系统所能支持的并发用户越少。例如，如果用户的考虑时间是1秒，那么系统可能只能支持数百个这样的并发用户。但是，如果用户的考虑时间是30秒，那么系统则可能支持数万个这样的并发用户（假定硬件和应用程序都是相同的）。在现实世界中，通常难以确定用户的确切考虑时间。还要注意，在现实世界中，用户不会精确地按照间隔时间发出请求。

于是就引入了随机性。如果知道普通用户的考虑时间是5秒，误差为20%，那么在设计负载测试时，就要确保请求间的时间为5×（1 +/- 20%）秒。此外，可以利用“调步”的理念向负载场景中引入更多的随机性。它是这样的：在一个虚拟用户完成一整套的请求后，该用户暂停一个设定的时间段，或者一个小的随机时间段（例如，2×（1 +/- 25%）秒），然后再继续执行下一套请求。将这两种随机化方法运用到测试中，可以提供更接近于现实世界的场景。

现在该进行实际的容量规划测试了。接下来的问题是：如何加载用户以模拟负载状态？最好的方法是模拟高峰时间用户与服务器通信的状况。这种用户负载状态是在一段时间内逐步达到的吗？如果是，应该使用ramp-up类型的测试，每隔几秒增加x个用户。或者，所有用户是在一个非常短的时间内同时与系统通信？如果是这样，就应该使用flat类型的测试，将所有的用户同时加载到服务器。两种不同类型的测试会产生没有可比性的不同测试。例如，如果进行ramp-up类型的测试，系统可以以4秒或更短的响应时间支持5,000个用户。而执行flat测试，您会发现，对于5,000个用户，系统的平均响应时间要大于4秒。这是由于ramp-up测试固有的不准确性使其不能显示系统可以支持的并发用户的精确数字。以门户应用程序为例，随着门户规模的扩大和集群规模的扩大，这种不确定性就会随之显现。

这不是说不应该使用ramp-up测试。对于系统负载在一段比较长的时间内缓慢增加的情况，ramp-up测试效果还是不错的。这是因为系统能够随着时间不断调整。如果使用快速ramp-up测试，系统就会滞后，从而报告一个较相同用户负载的flat测试低的响应时间。那么，什么是确定容量的最好方法？结合两种负载类型的优点，并运行一系列的测试，就会产生最好的结果。例如，首先使用ramp-up测试确定系统可以支持的用户范围。确定了范围之后，以该范围内不同的并发用户负载进行一系列的flat测试，更精确地确定系统的容量。

渗入测试

渗入测试是一种比较简单的性能测试。渗入测试所需时间较长，它使用固定数目的并发用户测试系统的总体健壮性。这些测试将会通过内存泄漏、增加的垃圾收集（GC)或系统的其他问题，显示因长时间运行而出现的任何性能降低。测试运行的时间越久，您对系统就越了解。运行两次测试是一个好主意——一次使用较低的用户负载（要在系统容量之下，以便不会出现执行队列），一次使用较高的负载（以便出现积极的执行队列）。

测试应该运行几天的时间，以便真正了解应用程序的长期健康状况。要确保测试的应用程序尽可能接近现实世界的情况，用户场景也要逼真（虚拟用户通过应用程序导航的方式要与现实世界一致），从而测试应用程序的全部特性。确保运行了所有必需的监控工具，以便精确地监测并跟踪问题。

峰谷测试

峰谷测试兼有容量规划ramp-up类型测试和渗入测试的特征。其目标是确定从高负载（例如系统高峰时间的负载）恢复、转为几乎空闲、然后再攀升到高负载、再降低的能力。

实现这种测试的最好方法就是，进行一系列的快速ramp-up测试，继之以一段时间的平稳状态（取决于业务需求），然后急剧降低负载，此时可以令系统平息一下，然后再进行快速的ramp-up；反复重复这个过程。这样可以确定以下事项：第二次高峰是否重现第一次的峰值？其后的每次高峰是等于还是大于第一次的峰值？在测试过程中，系统是否显示了内存或GC性能降低的有关迹象？测试运行（不停地重复“峰值/空闲”周期）的时间越长，您对系统的长期健康状况就越了解。

【性能测试概述，概述啊啊，竟然写了12页，这也太多了，有些内容与课题研究方法章节重复，可以删除；有些内容可以挪到下面章节】<--一个字“删”

## 2.3国内软件测试面临的问题

1. 没搜到，不会写。哇哇哇……呜呜呜

## 2.4投保人医疗保险信息系统性能测试的提出

目前，作为和每个人切身利益相关的医疗保险信息系统被大家尤为关注，那么这其中，投保人的医疗保险的各项信息的整合和管理就变得比较复杂，传统的手工模式根本无法应付这项工程，于是，投保人医疗保险信息系统就自然而然的出现在人们的面前。随着各种信息管理系统web应用的兴起，基于 B/S结构的软件日益强劲发展，正在成为未来软件模式的趋势，所以，这里的投保人医疗保险信息系统也是基于这种模式的。然而，当一个web应用被开发并展现在用户、供应商或合作伙伴的面前时，尤其是即将被部署到实际运行环境之前，用户往往会疑问：这套web应用能否承受大量并发用户的同时访问？系统对用户的请求响应情况如何？在长时间的使用下系统是否运行稳定？系统的整体性能状况如何？如果存在性能瓶颈，那么是什么约束了系统的性能？而这些正是web性能测试解决的问题，如何有效进行web性能测试，目前并没有一个系统和完整的回答。那么，如何对一个基于web的医疗保险信息系统有效的进行性能测试，更是没有一个系统而完整的答案。此外，由于紧凑的开发计划和复杂的系统架构，web应用的测试经常是被忽视的，即使进行了测试，其关注点也主要放在功能测试上。但是，近年来web性能 测试越来越引起重视，成为web系统必不可少的重要测试内容。本文的研究就是基于这种需求，进行基于目标的web性能测试方法。

软件界已经认识到，开发的软件质量有好有坏，参差不齐，如果想对这些软件的质量进行把关，或者说想尽可能的提高软件的质量，就需要在软件工程过程中对软件进行测试。因此一些企业或者软件开发人员会对软件进行一些功能测试，但功能测试只是对系统的基本功能做相关的测试，并没有考虑系统性能相关问题，一旦因为性能问题系统崩溃，将会给用户带来很不愉快的上网体验，也会因此给相关机构和企业带来损失。

基于web的投保人医疗保险信息系统，针对此类系统性能测试的专门研究又不是很系统，因此，本文为基于web的投保人医疗保险信息系统的质量，而尝试进行一些性能测试的研究是很有意义的。【感觉第二段很突兀，第一段最后一句已经提出了“web的性能测试”，而且最后一段可以合并到第一段】

# 3. 课题的研究目标、研究内容和拟解决的关键问题

## 3.1 课题研究目标

本文以jmeter为测试工具，对基于web的投保人医疗保险信息系统进行性能测试，以提高其质量。

研究目标是：结合这个基于web的投保人医疗保险信息系统的特征，不仅用测试工具去运行一些测试脚本来证明产品是否可以达到性能指标，更关键是要发现产品性能上的缺陷，并解决定位问题，这才是软件性能测试的真正目的。其目标可分为以下两种级别：

（1）性能测试总体目标。找出这个投保人医疗保险信息系统可能存在的性能瓶颈或者软件缺陷，确认其是否可以达到用户的需求。收集测试结果并分析产生缺陷原因，提交总结报告，让软件开发方对此系统进行性能改进。

1. 性能测试具体目标。此目标又可以分为：
2. 确定此系统的总体性能参数，包括所支持的最大并发用户数、事务处理成功率、请求相应的往返延迟等；
3. 确定在各个级别的负载及压力测试下服务器输出的具体性能参数；
4. 系统耐力参数。

这些测试目标驱动了整个测试过程的进行，因而在此系统性能测试中起着至为关键的核心作用。

## 3.2课题研究内容

本文主要是研究的是使用jmeter针对基于web的投保人医疗保险信息系统性能进行测试，找出系统中需要改进的地方，使系统尽可能的完美。

本文主要内容包括：

第一步、熟悉系统应用，根据系统开发的逻辑来了解整个系统的架构，这样才能知道测试中需要模拟的应该是什么样的请求。

第二步、测试需求的了解和分析，确定测试业务场景和和性能测试指标。

确定测试业务场景：

根据用户需求中所描述的系统架构来分析性能测试的重点表现在哪几个方面，如：并发访问的性能、批处理业务执行效率、系统处理的稳定性等，同时结合系统架构分析可能存在的性能瓶颈，这个也是选择测试业务场景的基础。

根据用户需求并结合实际，尽量做到对不同业务类型和操作类型的覆盖，如；对数据库的读操作（简单查询、复杂查询）、数据库的写操作（如：插入、删除和更新），选取一些具有代表性的业务操作作为测试业务场景。

分析性能测试指标：

1）根据用户需求和所选取的典型业务场景，首先选取性能指标项，如：并发用户数、交易响应时间、每秒交易数（TPS）、服务器资源占用率、网络吞吐量等；

2）在选取性能指标项后，根据用户需求或总体设计文档对性能指标进行分析，获得各个性能指标项值，如：用户需求中说明了某个业务使用的用户数为N，但没有说明并发用户数，可以按照10%N~20%N来设置并发用户数；有的文档中会提到一年会处理多少业务量，那么可以按照28原则计算出每秒交易数（TPS

）

第三步、搭建测试环境：

1、测试环境搭建尽量与真实运行环境大致一样，这样测试结果才有参考价值。

2、营造独立的测试环境，被测系统在预期性能测试执行期间应保证其资源独占性，即测试过程中要确保我们的测试环境独立，避免测试环境被占用，影响测试进度及测试结果，比如设备连网后，如果其他开发组或测试组也在共用，这样就可能影响我们的测试结果。有时开发人员为确定问题会使用我们的测试环境，这样会打乱我们的测试活动，更严重的是影响测试进度。因此需要为本次测试搭建独立的测试环境。

3、构建可复用的测试环境，项目实际执行过程中，测试环境是经常变化，比如测试软件版本更新、测试人员流失等等，需要随时跟踪和改进，尽量将可控的资源进行分类整理。可控资包括：测试环境配置手册、测试硬件信息、环境变更记录等等，目的是尽量将测试环境进行备份，方便出现未知问题时快速的还原。当刚搭建好测试环境，安装测试软件之前及测试过程中，对操作系统及测试环境进行备份是必要的，这样一来可以为我们下轮测试时直接恢复测试环境，避免重新搭建测试环境花费时间，二来在当测试环境遭到破坏时，可以恢复测试环境，避免测试数据丢失，重现问题。

第四步、测试数据准备，在实施性能测试时，需要运行系统相关业务，这时需要一些数据支持才可运行业务，这部分数据即为初始测试数据，数据准备和清理的工作量是非常大的，需要在测试前提前考虑。为更加真实的模拟现实运行环境，我们在测试过程中，应尽可能准备与真实业务执行相一致的初始数据，如系统用户数据、业务数据、辅助数据等。

1)系统用户数据：登陆系统使用的帐户名-口令等，数量与虚拟用户数一致；

2)业务数据：每个虚拟用户模拟真实用户进行操作时使用到的数据；

3)辅助数据：为保证业务操作的正常进行而设置的基本信息资料。此外，测试数据可分可重用和不可重用数据：

4)可重用数据：如客户信息等查询类的数据，此类数据只需一次准备

即可；

5)不可重用数据：此类数据为一次性消耗数据，不可重用，一般应用

在数据增加或修改类业务交易，此类数据如增加客户标识、帐户标

识等。

第五步、测试环境配置。

第六步、录制、调试和执行脚本。

第七步、分析测试结果，获得执行脚本过程中，所收集的性能指标值的分析报告，可以通过分析报告中的数据，得知系统的性能情况。

最后，本文在总结性能测试结果，并分析性能瓶颈，得出建议优化措施，进而提高该系统的性能，使软件质量得到进一步的保障。

## 3.3 拟解决的关键问题

这个不知道？？？？。

# 4. 课题的研究方法、设计及试验方案，可行性分析

## 4.1课题的研究方法

### 4.1.1强度测试

一些软件项目的研究学者已经尝试总结威胁软件开发成功率的因素。例如，Alter识别出的八大风险因素：不合作的用户、多个用户或者实现单位、在多个参与者之间交接、无法指定目标或者用途、无法缓冲对其他人的冲击、缺少支持、缺少经验以及技术或投入的效力问题。又如McFarlan将他所提出的三大角度映射到软件项目开发中，即项目大小、应用技术的经验和项目结构。而Boehm则提出十大优先级排序的软件风险项：人员短缺、不现实的进度和预算安排、开发错误的软件功能、开发错误的用户界面、需求等的镀金、不停的需求改变、外部提供的组件不足、外部执行的任务有缺点、实时性能的缺点以及计算机科学技术上的限制。再如，由Roger S.Pressman编写的软件工程经典著作――1999年出版的《Software Engineering，A Practitioner’s Approach，Fourth Edition》中，将风险分为产品规模、商业影响、客户特性、过程定义、开发环境、建造技术和人员数目及经验七大风险种类，每个风险种类又有八到二十个风险项。

除了以上这些概括性的风险因素外，一些学者还针对大型的国家项目，进行了更加细致具体的研究，罗列出十分详尽的风险因素。例如，Charette 根据美国空军系统管理手册AFSCP800-45总结出一张风险表，概括了一般软件项目开发中的风险，从技术、进度、成本、操作性、支持度五个角度，列举了拥有三到六个属性的24个风险因素。该手册中将风险严重程度的划分为低、中、高三个级别，并对每一因素每一属性在不同级别的表现进行了描述。

比较成体系的还有Dale Karolak于1996年推出的SERIM（Software Engineering RiskModel）方法。SERIM的理论体系主要基于如下概念：风险元素、因素、指标和活动，SERIM的分析模型反应了这几个概念之间相辅相成的关系。Karolak认为软件风险体现在三个方面：技术、成本和进度。其中技术方面与性能、可用性等相关，应该尽早识别这个方面的风险；成本则包括预算、盈利等；进度包括进度表的灵活度、现实性等，贯穿于整个开发周期。SERIM操作的数据来源是81个问题，这些问题覆盖了软件风险的十大因素：组织、预测、监控、开发方法学、工具、风险文化、可用性、正确性、可靠性和人员。风险因素对风险在技术、成本和进度三方面的影响用高、中、低评估，不同的影响程度可以给予一个确定的权值，以便于定量的计算。

软件过程描述的是项目之间的共通点，而风险管理描述的是项目与众不同的地方。因此，风险管理中十分重要的是不要陷于习惯，不要只列出每个项目都类似的风险，而应该识别出特定项目所独有的风险。所以，我们研究政府办公自动化软件的特点，提出其中一些典型的风险。

以上研究成果对软件项目风险因素的总结都各有自己的关注域和侧重点。政府办公自动化系统本质上也是软件项目，自然也受到这些风险因素的影响。因此本文参考这些软件研究学者已总结的风险因素资料，结合头脑风暴、专家判断等识别方法，结合我们在大榭区政府办公自动化项目开发过程中的经验，识别影响政府办公自动化项目建设成败的风险因素变量。

### 4.1.2负载测试

问卷调查和专家评分法是一种最常用、最简单的定性与定量相结合的分析方法。它的应用分成两个步骤：首先，识别出某一特定项目可能遇到的所有风险，列出风险调查表；其次，利用专家经验和调查统计结果，对所列风险的重要性进行评价，最后综合成整个项目的风险。本文采用这种方法进行风险分析，并结合大榭区政府办公自动化项目的实际情况进行了改进。

我们在对软件项目风险管理和政府办公自动化项目进行初步研究，有了一定认识后，拟定出两份风险调查问卷，以文字讨论形式为主，分别面向开发单位和使用单位，各自列举了双方可能导致项目失败的不恰当做法，以及在项目开发中可能遇到的影响项目正常开展的阻碍。然后，我们通过当面访问讨论和邮件咨询的方式，对6位有相关经验的开发人员、2位主要从事政府OA软件开发的公司负责人和2位使用办公自动化系统的单位领导进行了调查，倾听他们对政府OA项目开发的心得体会，了解他们对这些风险因素影响项目成败程度的观点，还有他们自己独到的风险解释。

接着，我们依据调查反馈信息，参考已识别出的风险因素，重新拟定出两份风险因素列表，分别面向开发单位和使用单位，列举双方面对这些风险时是否采取了积极的行动。例如，与“项目经理的经验”相对应的问题是“项目经理拥有足够的项目开发管理经验吗？”。对问题的回答（即风险值）则反映了这一积极行动的完成度，用0-10之间的数表示。例如，对上一个问题的回答可以用10表示该项目经理对相似类型和规模的项目管理极富经验，完全胜任；用6表示项目经理有一定的经验，但不够丰富等。得分越高表示对风险项的把握越充足，从而项目风险比较低；反之，分数越低表示项目风险项的关注程度不够，没有达到一定的要求，从而项目风险比较高。我们采用简洁的平均值方法汇总分析这些风险分值后，得出了这些风险因素变量各自的风险值。

风险值仅仅代表了风险因素在项目中可把握的程度，而风险等级代表风险因素对项目整体风险的影响程度。因此，在第二次问卷调查中特别添加了风险等级的评定，针对结果显示出风险值低而同时等级高的因素变量，我们需要加倍关注。为了简化问卷填写并标准化答案，我们采取区间单位法来评定风险等级，分为可以忽略、不太重要、不确定、比较重要和十分重要五大区间，这是参考了Borda的方法。Borda将风险等级划分为五级：十分严重、严重、中等、微小和可忽略。不同的级别代表了风险事件发生时造成的成本增加、进度滞后、需求无法完全实现的程度。而在进行数学计算的时候，我们将Borda的五个级别转换为分别对应0.9、0.7、0.5、0.3和0.1的数值。

最后，我们综合每项风险因素变量的风险值和等级评分，得出风险评判值（也在0～10之间），评判值越高，表示该因素存在的风险越大。删除评判值在1以下的一些风险因素变量，并参考政府办公自动化项目案例，修正后列出了具有研究意义的风险因素变量，作为我们进行风险评估的依据。

### 4.1.3压力测试

本文采用多层分类法，从具体的风险变量入手，通过分类和讨论其表现形式，提出一个系统的综合的评估政府办公自动化项目风险的模型――GOAREM（Government Office Automation Risk Evaluation Model）。

GOAREM的思想来源于Dale Karolak提出的SERIM（Software Engineering Risk Model，软件工程风险模型）方法。该方法的特点在于从多个角度对软件开发中的风险因素进行评估，并可以在开发周期的任何时候根据评估结果监控风险，因此受到IEEE的推荐。SERIM从技术和商业两个角度对软件风险管理进行剖析，考虑的问题涉及开销、进度、技术性能等。它还提供了一些指标和模型来估量和预测风险，由于这些数据来源于大量的实际经验，因此具有很强的说服力。

但是，SERIM中作为评估来源的81个问题主要面对软件开发人员，没有考虑软件使用方的情况，有些问题也过于追求技术细节。而且，在国内政府办公自动化系统建设中很少使用风险管理的现状下，将风险文化作为一大风险因素也不太妥当。因此，SERIM的内容并不适用于本文研究的政府办公自动化项目风险分析，但其框架和思路对我们提出的GOAREM有很大的启发作用。

### 4.1.4风险管理方法

风险管理的基本概念是：项目管理小组通过风险识别、风险分析和风险评估，并以此为基础合理地使用多种管理方法、技术和手段对项目活动涉及的风险实行有效的控制，采取主动行为，创造条件，尽量扩大风险事件的有利结果，妥善处理风险事故造成的不利后果，以最少的成本保证安全、可靠地实现项目的总目标。

参考经典的风险管理模型有：Barry Boehm模型、SEI提出的持续风险管理模型CRM 、丹麦Aalborg大学提出的Leavitt模型和美国Maryland大学提出的Riskit方法。

从以上的风险管理模型可以看出，风险应该先分析后管理，这几乎已经成为共识。而对于分析和管理中应以何种粒度进一步划分步骤、应突出和侧重哪一个措施等问题，则各有千秋，不尽相同。大的指导思想和框架基本一致，具体实施措施有些差异，很难绝对地判断这些模型的优劣。因此，本文也沿用公认的风险管理步骤，添加步骤之间的信息流，从而得到政府办公自动化项目的风险管理模型。

## 4.2课题设计方案

首先通过问卷调查和专家评分的方法，深入调研当前政府办公自动化项目建设过程中遇到的问题，分析其产生的原因。然后，参考SERIM模型分类评估的框架和思路，提出针对政府办公自动化项目的风险评估模型――GOAREM（如下图4.1）和风险管理过程模型（如下图4.2），使得项目的风险情况能够进行定量分析，以便有目的地制定项目风险控制计划。最后结合实际项目案例，讨论在项目开发不同阶段的风险控制侧重点和风险缓解方法。









**………**









**……**

**P(T)**

**P(S)**

**P(C)**

#### P

指标层

准则层

指标子集层

目标层

**图4.1 GOAREM模型流程图**

其中（i＝1，2，…，32）表示影响项目成败的风险因素变量的评判值，（j＝1，2，…，9）表示风险类别的评估值，P(T)、P(S)、P(C)和P分别代表三种风险表现形式和整个项目的最终风险评估值。

**图4.2 政府OA系统的风险管理模型**

问题标识

情况变化

修改后的风险状态

期望的结果与

实际项目数据

选中的行为

优先级别高的风险

风险列表

启动

风险管理定义

风险识别

风险监控

风险分析

风险控制

风险控制计划

情况变化

项目权威

风险管理基础设施

风险承担人、期望值

方法、工具、指导

## 4.3课题可行性分析

基于以上分析结果，我们认为本文的研究方法和设计方案切实可行，可操作性高，研究结果也具有很高的实用性，课题具有一定的研究价值。

# 5.课题计划进度和预期成果

## 5.1计划进度

|  |  |  |
| --- | --- | --- |
| **开始时间** | **结束时间** | **主要工作内容** |
| 2005年03月 | 2005年05月 | 查阅文献资料，编写课题开题报告 |
| 2005年05月 | 2005年06月 | 进行第一次问卷调查工作 |
| 2005年06月 | 2005年07月 | 进行第二次问卷调查工作 |
| 2005年08月 | 2005年09月 | 依据GOAREM模型对已完成项目进行评估验证，并开发项目风险评估系统的原型 |
| 2005年10月 | 2005年11月 | 撰写论文正文 |

## 5.2预期成果

总结全文，本文预期取得的主要研究成果如下：

### 5.2.1政府办公自动化项目风险分析

①分析政府办公自动化项目风险因素的特点，总结它们之间的关系，以及对项目实施成败的影响；

②根据两次问卷调查结果，应用加权平均分析方法，得出风险因素的定量评估，并参考SERIM模型提出政府办公自动化项目的风险评估模型――GOAREM。该模型可以根据风险因素变量、风险类别和风险表现，系统地多层次对一个政府办公自动化项目进行评估，定量表现项目风险程度。

### 5.2.2风险评估系统原型建设

根据风险评估模型――GOAREM，我们采用UML进行系统分析设计，以Visual Basic.NET和Microsoft Access 2000为开发工具，使用面向对象的开发思想实现风险评估系统的一个原型建设。

### 5.2.3风险管理的过程模型

①总结当前几种经典的项目风险管理模型，指出其各自的优点、侧重面和欠缺的地方；

②结合政府建设和使用办公自动化的特点，提出针对政府办公自动化项目的风险管理过程模型，并建立管理中使用的文档模板。最后以大榭区政府办公自动化项目的风险管理过程为例，讨论在各个开发阶段风险管理的侧重点。