6.2 性能测试组织

性能测试并不是一个人或者一个部门可以完成的工作，相对于功能测试来说，性能测试更为复杂，所以一个公司成立一个性能测试部门并不难，但是需要将这个部门的作用发挥出来就不是那么简单的事情了。

6.2.1 性能测试团队

性能测试作为测试团队中的一部分，有其特殊性，所以对于性能测试团队的管理和组织架构和普通的功能测试工作在流程和组织上有所不同，这里给出一个在整个项目研发过程中性能测试相关角色的矩阵图，如图 6.87 所示。

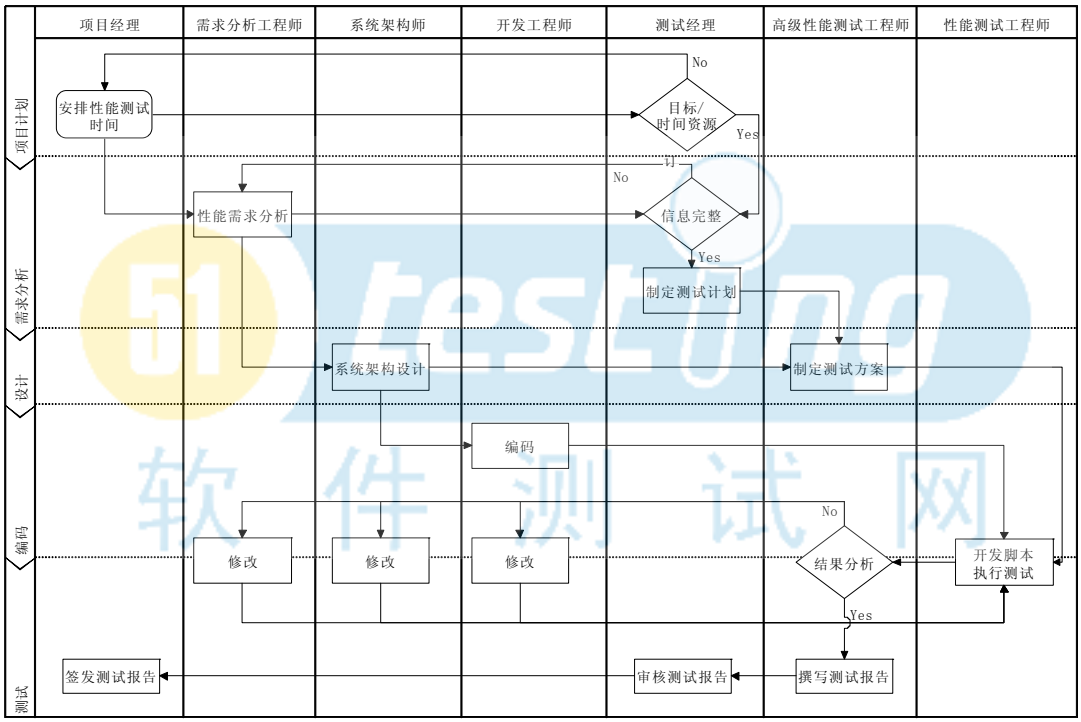

图 6.87 性能测试团队工作流程

- PM 项目经理

项目经理作为整个项目最关键的人，必须对性能测试有所了解，这样才能确保性能测试被重视，并且有足够的资源支持。

- 性能需求分析工程师

确定系统的性能需求，除了传统的功能需求，我们需要得到更加准确的性能需求，例如系统的容量、响应时间等明确指标。这些需求需要在开发前就明确，这样才能在一定的约束下进行开发，确保在设计初期就考虑了性能方面的需求。高级性能测试工程师可以辅助需求分析工程师对性能需求进行定义和明确。

- 架构设计师

系统的性能在需求分析工程师和架构设计师经过分析后确定。架构设计师根据用户的性能需求来决定系统的架构特点并提供足够的可扩展性。后期做的性能测试只是让系统更接近于系统设计时的性能而已，但是无法超越系统架构的设计，也就是说性能是做出来的而不是

测出来的，待到后期通过性能测试发现问题时，调优的成本和难度也非常大了。

在大多数情况下，系统的架构性能都是由架构设计师独自测试分析的，如果有性能测试工程师介入，也可以提供一定的技术支持和建议。

- 开发工程师

开发工程师的主要职责是在架构上进一步编码，受到个人能力的影响，某些开发工程师能够很好利用架构在不降低性能的情况下实现功能，而也存在编码不合理导致效率降低的情况。

- 测试经理

作为一名测试经理，更多的工作是协调沟通，即组织和协调资源确保性能测试的有效进行。

- 高级性能测试工程师

负责测试计划、测试策略和分析工作。

- 性能测试工程师

根据用户的需求，完成脚本的开发并形成系统负载，将性能问题表现出来，而结果的分析由高级性能测试工程师和相关人员共同完成。

6.2.2 性能测试流程分工

在性能测试的整个周期上，不同阶段各个部门的工作内容是不同的，通过图 6.88 可以了解到在每个过程中，需要哪些部门成员的介入。

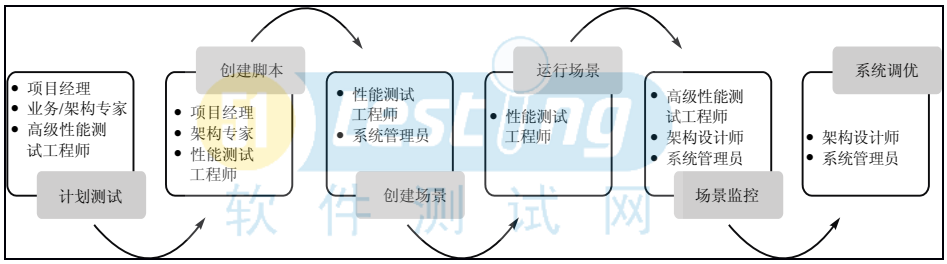

图 6.88 性能测试流程分工

① 测试计划

性能测试的计划是由项目经理、业务/架构专家和高级性能测试工程师共同完成的，项目经理需要提供时间的计划安排和资源，而业务/架构专家需要提供架构设计的目标以及系统的相关信息，最后由高级性能测试工程师编写测试计划。

② 脚本开发

在开发脚本的过程中，一方面需要项目经理的协调，而另一方面架构专家提供脚本开发中的一部分技术支持，包括测试环境的搭建和早期性能测试的介入。

③ 场景设计

根据性能需求分析工程师指定的相关需求设计场景。系统管理员提供一些软硬件平台搭建上的支持。

④ 场景运行

完成场景的运行设置、负载生成规则和执行策略，开始执行场景。

⑤ 负载下的监控分析

在场景运行的过程中，作为一个性能测试工程师并不是简单地等待数据，还需要系统管理员和架构设计师在负载过程中对一些数据进行监控，随时确认瓶颈产生的原因。

⑥ 系统调优阶段

架构设计师和系统管理员根据前面得出的数据进行系统调优，当调优工作完成后，重新进入新一轮的性能测试，整个过程按照 PDCA 循环持续，直至达到目标。

如何进行调优请参考各个应用的 tuning 手册和相关书籍。

6.2.3 配置管理

为什么要进行性能测试的配置管理呢？对于性能测试来说，作为测试用的脚本也会随着项目的进展被不断地修改，多次的场景运行后会带来大量场景数据，对应的大量有待进行基准测试分析的数据，而如果没有有效的配置管理，会带来性能测试工作的混乱。另一方面开发性能测试脚本和场景也是开发的一种，同样也需要对其进行有效的权限划分和版本追溯。为了确保性能测试中的相关数据和代码能够有效地进行管理，实行配置管理是非常有必要的。

HP 也考虑到这些问题，推出了 Performance Center 和 Quality Center 两大系统平台。其中 Performance Center 提供了对性能测试后场景运行管理和场景结果数据的统一管理和横向比较功能，而 Quality Center 提供了版本控制和脚本运行的控制，LoadRunner 本身提供了对 PC 和 QC 的插件支持，但这两个工具部署复杂并且价格不菲，从性价比的角度来说，并不太适合小公司。如果公司已经使用了 QC 来管理缺陷流程，那么建议继续使用。对于没有以上条件的公司，其实可以通过配置管理工具和一定的流程来实现同样的功能。

管理内容

- 性能测试脚本及参数化文件
- 性能测试场景
- 性能测试场景运行后的结果
- 性能测试分析及报告

达到目标

对性能测试中使用的所有内容进行版本管理，实现对性能测试脚本和场景的基线化操作，对每个基线代码形成有效的性能测试数据记录，并能够简便地实现同场景下多结果的配置及基准测试数据的性能比较。

这里推荐使用业界流行的 Subversion 作为性能测试的配置管理工具，客户端使用 TortoiseSVN。

在整个性能测试过程中，需要配置的内容主要包括以下几点：

- 脚本中的代码
- 脚本中的参数化数据
- 脚本的 Run-time Setting
- 场景的设计
- 场景运行后得到的数据
- Analysis 分析的报告
- 相关的文档和报告

对于性能测试的配置管理可以使用以下的目录结构：

```
/
/Trunks/
/Trunks/PT/
/Trunks/PT/
/Trunks/PT/Document
/Trunks/PT/Document/Plan
/Trunks/PT/Document/Report
/Trunks/PT/Script
/Trunks/PT/Script/Param
```

```
/Trunks/PT/Scenario
/Trunks/PT/Res
/Trunks/PT/Analysis
```

其中，/Trunks 是 SVN 中的主干目录；/PT 是性能测试的英文缩写，说明该目录下的内容都是性能测试；Document/目录下主要保存相关性能测试的文档，如性能测试需求分析、性能测试计划、性能测试报告；Script/目录主要管理 VuGen 脚本。默认情况下，VuGen 脚本由以下几个文件组成：

```
Action.c
Vuser_end.c
Vuser_init.c
Globals.h
Default.cfg
Default.usp
Scenario.usr
```

通过前面的学习我们知道前 3 个.c 文件和一个.h 文件是脚本文件用来存放 Action。Scenario.usr 文件是 VuGen 打开脚本的主文件，文件内包含了一些 VuGen 的通用设置和运行属性。Default.cfg 和 default.usp 是配置文件，其中 runtime setting 存放在 default.usp 文件中，所以这些文件必须要添加到配置管理工具中。

当脚本被运行后，会得到以下内容：

```
Combined_scm_.c
Scm.c
Pre_cci.c
Logifle.log
Mdrv.log
Mdrv_cmd.txt
Options.txt
Output.txt
```

result1 脚本运行后的 result 目录

其中 Mdrv.log 和 output.txt 文件是运行脚本所产生的 log 文件。Mdrv_cmd.txt 是命令行启动 mdrv 的参数写法，使用 mdrv.exe 启动脚本的命令方式参考第 3.5.5 节中的相关内容。三个.c 文件是运行脚本所需要的系统函数库，这些内容都无须进行配置管理。

脚本中使用的参数化内容存放在 Script/Param 目录下以便于管理。参数化后脚本中会添加一个.prm 文件，该文件记录了参数化列表中的取值规则，需要进行配置管理。

对于.bak/.log 和 result1 目录这种无须进行配置管理的文件，可以使用 TortoiseSVN 中的添加忽略列表功能将这些内容从 commit 操作中去除，如图 6.89 所示。

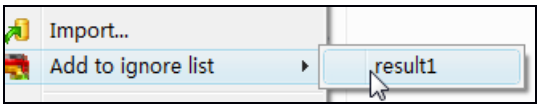

图 6.89　TortoiseSVN 中设置过滤文件

通过这种方法可以轻松地管理每次脚本运行或修改脚本的过程，只需在运行或修改后单击 commit 按钮就可以将所有的操作通过配置管理库进行集中管理，甚至可以进行多人并行

开发。

所以 Scripts 目录下需要管理的内容主要有：

```
*.c
*.prm
*.usr
*.usp
*.cfg
*.h
```

param 目录下的所有.dat 文件

当然也可以通过 VuGen 中的 export script 功能将脚本打包导出进行管理，不过需要再解包进入配置管理，相对麻烦了一点。

场景的设计文件都保存在 Scenario 目录下，场景文件是由.lrs 文件管理，该文件内保存了所有场景设置的内容。只需要对该文件进行配置管理即可。

场景运行后的 results 文件是整个性能测试分析的基础，该数据需要单独保存。

在场景运行前，先设置场景运行结果的存放地址。将场景的 results 文件存放在 res 目录下，并设置存放 res 数据的目录名称为场景名称，选中自动创建新的 results 目录为每一次场景运行。

例如：我们的场景名为 scenario1，那么在场景中设置的结果目录就应该为..\res\scenario1，如图 6.90 所示。

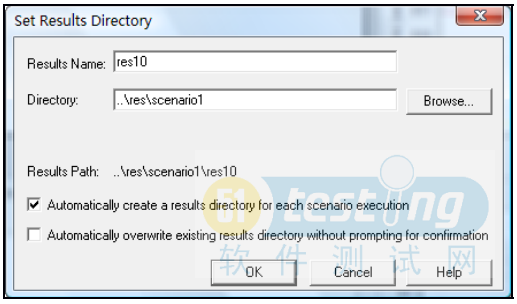

图 6.90　修改场景运行结果地址为相对路径

这样，当场景运行结束后，相关的内容就会被保存在\res\scenario1\res0 目录下，只需要对 res 目录进行管理即可。

对于 Analysis 报告建议保存的文件名为“场景名_场景数据目录名.lra”，例如前面的场景名为 scenario1 而得到的 result 目录为 res0，如果对该场景下的这个结果目录进行分析，那么报告名应该为 scenario1_res0.lrs。.lrs 文件保存在 Analysis 目录下。

通过 Analysis 生成的报告文档存放在 document/report 目录下。

权限规则表

每个公司研发部门的规模都不尽相同，而对应的配置管理目录权限也不同，这里给出一个简单的参考方案，如表 6.25 所示。

表 6.25　性能测试配置管理权限矩阵表

	初级性能测试工程师	高级性能测试工程师	项目经理	性能需求分析工程师
Document	R	RW	R	RW
Script	RW	RW	/	/
Scenario	/	RW	/	/
Res	/	RW	/	/
Analysis	R	RW	/	/
	架构设计师	开发工程师	测试经理	DBA/Sys Admin
Document	RW	/	RW	R
Script	R	/	R	/
Scenario	R	/	R	R
Res	R	/	R	R
Analysis	R	/	R	R

R=read　　W=write　/=none

文件命名规则

每个公司的命名规则不同，这里给出参考方案。

文档命名规则

文档以 PT 开头+文档类型+编号，例如：性能测试需求 PT_RQ_0001.doc、性能测试报告 PT_Report_0001.doc。

脚本命名规则

脚本根据实现的功能进行描述，即“PT+功能说明”，例如：注册用户脚本 PT_RegUser.usr。

场景命名规则

场景名称使用“脚本名+场景类型+场景编号”，例如：注册用户目标场景（Goal Scenario）PT_RegUser_GS_0001.lrs、注册用户行为 500 个用户在线手工场景（Manual Scenario）PT_RegUser_MS500_0001.lrs。

编号可以使用文档管理，来说明该场景下的一些负载生成规则。

Analysis 命名规则

使用“场景名+场景运行”编号，例如：注册用户目标场景分析结果 PT_RegUser_GS_0001_Res01.lra。

基线化规则

当性能测试脚本和场景确定后，script 脚本目录和场景文件将被标记为 tag 进入基线状态。当使用该基线性能测试场景对某个软件基线版本进行性能测试后，相关的数据也需要进行基线化处理并且标明该性能测试结果的数据来自于哪个开发基线版本，从而实现性能测试与开发的版本配对。

6.2.4　性能测试自动化

各大论坛中现在谈得最多的就是测试自动化、单元测试自动化、自动化测试，通过每日构建我们可以大规模地降低测试成本并及早发现和定位问题。通过合适的流程同样也可以实现性能测试的自动化，将性能测试加入每日构建。

首先来看看每日构建的常见步骤和过程。

① 对被测系统的标记进行分支操作。

② 将被测系统下载至指定目录。
③ 对被测系统进行编译及白盒静态测试。
④ 对被测系统进行打包发布。
⑤ 下载安装包并进行安装测试。
⑥ 对系统进行自动化测试。
⑦ 对系统进行手工冒烟测试。
⑧ 提交测试报告。

性能测试是介于自动化测试和手工冒烟测试之间的步骤，这里需要解决的问题有以下几点：

- 测试代码及测试环境的自动下载
- 场景的命令行启动
- 性能测试结果的自动签入

如果希望实现性能测试的自动化，那么配置管理是最基本的要求，当完成了第 6.2.3 节中的配置管理后，就可以通过使用 Shell 脚本来解决以上问题，大家可以根据自己的爱好选择 Python、Perl、Ruby 等语言进行编写。

基于 SVN 的性能测试自动化要点主要包括：

① 确保当前目录为工作目录，同时 SVN 命令在系统 PATH 环境变量中。
② 使用 SVN 的 cp 参数创建 Tag。

例如现在要为以下目录创建分支：

```
/Trunks/PT/Script
/Trunks/PT/Script/Param
/Trunks/PT/Scenario
```

可以编写以下命令：

```
Svn cp /Trunks/PT/Script  /Tags/20090301
Svn cp /Trunks/PT/Scenario  /Tags/20090301
```

这样就把当前场景和脚本数据定义为一个基线存放到 Tags 下的目录中。

③ 通过 Svn checkout 命令检出 Tag。

```
Svn checkout /Tags/20090301
```

将服务器上的/Tags/20090301 目录下的所有内容下载到本地目录，这里包含了所有待执行的脚本和场景文件。

④ 运行场景。

以命令行方式启动场景。进入 LoadRunner 安装目录下的 bin 启动以下命令：

```
Wlrun.exe -TestPath g:\Scenario\scenario1.lrs –Run
```

上面这个命令说明调用 wlrun.exe 启动场景，通过-TestPath 参数启动 g:\scenario\scenario1.lrs 场景文件，-Run 参数说明启动该场景。

更多命令及 Wlrun 支持的参数如表 6.26 所示。

表 6.26　场景执行命令表

命令参数	说　明
TestPath	Path to the scenario, for example, 这里可以填写场景的物理地址 C:\LoadRunner\scenario\Scenario.lrs This argument can also be used for a scenario residing in a Quality Center database. For example, 也可以填写场景在 TD/QC 上的地址 "[TD]\Subject\LoadRunner\Scenario1" If the path includes blank spaces, use quotation marks. 如果路径中存在空格，那么使用引号
Run	Runs the scenario, dumps all output messages into 运行场景，场景结束后将所有信息输出到以下地址并关闭 Controller res_dir\output.txt and closes Controller
InvokeAnalysis	Instructs LoadRunner to invoke Analysis upon scenario termination. If this argument is not specified, LoadRunner uses the scenario default setting. 当场景结束时自动调用 Analysis，如果该参数没有被定义，则使用场景中的默认设置。
ResultName	Full results path. 自定义场景结果的完整地址（包括路径和名称） For example, "C:\Temp\Res_01"
ResultCleanName	Results name. 自定义结果的名称 For example, "Res_01"
ResultLocation	Results directory. 自定义结果的路径 For example, "C:\Temp"

进行性能测试自动化时还经常遇到一个麻烦的问题就是测试环境的变动，为了进行基准测试，经常需要在不同的环境下进行同负载的测试，这时对于代码的可维护性就有了较高的要求，这里可以通过命令参数的形式来解决。

在前面我们谈到过如何使用命令行启动脚本，那么在场景中同样也可以通过命令行导入环境信息，首先将所有的服务器地址信息替换成一个参数如{serverinfo}，然后通过r_save_string(lr_get_attrib_string("sinfo"),"serverinfo")函数将命令输入的服务器地址信息载入再保存到参数中去。这样在执行场景的时候只需要通过修改启动场景的配置参数就可以实现场景运行支持环境的变化。

这里需要先设计好场景，在场景中打开自动生成测试数据目录功能，当然也可以使用命令行自行定义。脚本运行后会得到 res 目录，然后通过添加的方式将该数据添加到 SVN 服务器上。

⑤ 使用 Svn add 命令将性能测试数据提交至服务器。

例如我们将场景执行后 res 目录下的场景数据提交至服务器，输入命令 Svn add \res\，最后通过 Svn commit 命令提交数据。

在 Windows 平台下可将上述操作添加进 BAT 批处理文件，通过 AT 命令或者计划任务

设置为定时启动即可。

这里主要提出了一个简单性能测试自动化的方式，每日构建需要相当成熟的研发团队和开发流程，切勿生搬硬套。

关于配置管理的原理和 SVN 工具的使用及 Shell 脚本的开发请参考相关文档。

小结

本章详细介绍了进行性能测试的流程步骤及要点。可以看到，性能测试的分析并不是一件很神秘的事情，但确实是一件枯燥、单调甚至有些压抑的工作，进行性能测试并不困难，但深入性能测试工作却一路坎坷。

本章需要掌握的重点：

- 性能测试的流程
- 性能测试需求分析方法
- 性能测试计划的编写要素
- 测试环境的搭建策略及容量生成手段
- 性能调优的原理及前后端性能分析理论
- 如何使用 LoadRunner 完成性能测试分析
- 性能测试报告的类型及特点
- 性能测试团队的组成和工作职责