Institute of Software Technology

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Fachstudie Nr. 193

Evaluation of Load Testing Tools

Gustav Murawski, Philipp Keck, Sven Schnaible

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. Lars Grunske
Supervisor:	André van Hoorn
Commenced:	2014/04/02
Completed:	2014/10/02
CR-Classification:	D.4.8

Abstract

This study evaluates common load testing tools with focus on AJAX based web applications, including JSF ViewState and WebSockets as special cases. Load tests are created and executed using an internal application provided by the collaborating company NovaTec GmbH. The evaluation focuses on the efficient creation and maintainance of workload specifications, which is of great importance when employing load tests in larger projects. Eight tools have been tested and recommendations are given for different use cases at the end of the paper.

Zusammenfassung

Diese Studie bewertet gebräuchliche Lasttest-Werkzeuge, wobei das Augenmerk auf AJAX-basierte Web-Anwendungen, den JSF ViewState und WebSockets als spezieller Fall gerichtet ist. Lasttests werden durch eine interne Anwendung erstellt und ausgeführt. Diese Anwendung wird von der zusammenarbeitenden Firma NovaTec GmbH bereit gestellt. Die Auswertung konzentriert sich auf die effiziente Erstellung und Wartung von Lasttest-Spezifikationen, was von großer Bedeutung ist, falls solche Lasttests in umfangreicheren Projekten verwendet werden. Es wurden acht Werkzeuge getestet und die dazugehörigen Empfehlungen auf Basis verschiedener Anwendungsfällen am Ende der Ausarbeitung ausgesprochen.

Contents

Chapter 1

Introduction

1.1 Motivation

For modern web applications it is critical to remain available and guarantee short response times—even if accessed by large numbers of concurrent users. Studies show that response times above one second cause dissatisfaction, which makes users switch to a competitor sooner or later [Cheung and Lee, 2005].

Building applications that meet these performance requirements starts with planning and testing these requirements in the early stages of the development process. Load tests are carried out in order to ensure that the required number of customers can be served simultaneously without exceeding a certain response time and performance requirements such as throughput and utilization.

In practice, load tests are rarely an integral part of the development process, see P. Haberl [2012]. The main reasons (other than the lack of time or money) may be the common preconception that meaningful load tests are too difficult to create and maintain as well as short-sighted calculations showing that the benefits of load tests don't outweigh their costs.

However, there are lots of tools available—both open-source and commercial—that support the creation and execution of load tests. Because of the high variety of these tools, it may be difficult to find a suitable solution for the individual use case.

1.2 Goals

This study attempts to serve as a guidance for choosing the right load testing tool—particularly for testing web-based applications that make use of AJAX and WebSockets. The results will be reported in form of a market overview, a criteria catalog and transparent experiments that allow for quick comparison of the tools. We also make a recommendation on which tool to use for several different use cases.

1.3 Collaboration with NovaTec Consulting GmbH

This case study is conducted in collaboration with NovaTec Consulting GmbH in Leinfelden-Echterdingen. NovaTec is an owner-operated, independent German IT consulting firm that amongst other things specializes in performance management and load testing. In the context of load testing, NovaTec offers the freely available product LoadIT ([NovaTec Solutions GmbH, 2014]), which is an extension for the popular JMeter tool ([Apache Foundation, 2014]).

1.4 Document Structure

Chapter 2 presents our organization. Chapter 3 presents relevant foundations. Chapter 4 gives a market overview. Chapter 5 presents the evaluation criteria. The evaluation procedure is presented in Chapter 6. Chapter 7 presents the evaluation results.

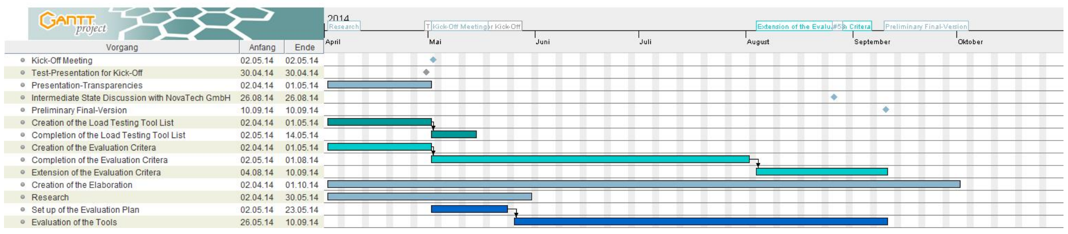
Chapter 2

Organization

This chapter will describe the milestones and phases of the study, which was conducted over six months in the period between 2nd May 2014 and 1st October 2014.

2.1 Timeline

The following Gantt chart shows the milestones and phases of the study.



Vorgang	Anfang	Ende
Kick-Off Meeting	02.05.14	02.05.14
Test-Presentation for Kick-Off	30.04.14	30.04.14
Presentation-Transparencies	02.04.14	01.05.14
Intermediate State Discussion with NovaTech GmbH	26.08.14	26.08.14
Preliminary Final-Version	10.09.14	10.09.14
Creation of the Load Testing Tool List	02.04.14	01.05.14
Completion of the Load Testing Tool List	02.05.14	14.05.14
Creation of the Evaluation Critera	02.04.14	01.05.14
Completion of the Evaluation Critera	02.05.14	01.08.14
Extension of the Evaluation Critera	04.08.14	10.09.14
Creation of the Elaboration	02.04.14	01.10.14
Research	02.04.14	30.05.14
Set up of the Evaluation Plan	02.05.14	23.05.14
Evaluation of the Tools	26.05.14	10.09.14

2.2 Phases

Phase 1: Begin of the Study The study began with the kickoff meeting on 2nd May 2014 in the department of NovaTec Consulting GmbH in Leinfelden-Echterdingen. After presenting our first ideas to NovaTec, we were introduced to the basics of Load Testing Tools and the requirements of the study. Our presentation also contained slides about our general course, goals, a preliminary criteria catalog, and a first list of load testing tool.

Phase 2: Research In this phase we familiarized us with the topic of load testing and also searched for load testing tool, which we listed in the market overview (see Chapter 4).

Phase 3: Evaluation Criteria Several criteria for our evaluation were developed. They are based on the kickoff meeting notes and further research. Our criteria catalog was planned to complete over time in case we find further important criteria points while testing our selected tools. The rating and structure of the specific criteria points are defined by us.

Phase 4: Creation of the Load Testing Tool List While being in this phase, we searched for five to eight load testing tools to test within our study. We chose JMeter as the most popular and few load testing tools were also recommended by NovaTec to be considered within our elaboration.

Phase 5: Set up of the Evaluation Plan Before we were able to start the evaluation of the tools we selected, we had to determine the evaluation procedure first. After we set it up we used this procedure for every load testing tool we chose.

Phase 6: Evaluation of the Tools In this phase we used our evaluation plan on all selected load testing tools and took down all information we gathered in notes and tables.

Phase 7: Creation of the Elaboration Creating the elaboration was continuous process since the beginning of our project. We categorized our gathered information in a compact introduction to explain the first steps of load testing followed by the market overview. Afterwards we show our specifically set up criteria catalog with the evaluation procedure. At last we represent all our evaluation results in both text and tabular form.

2.3 Milestones

▷ **1st Milestone 2nd May 2014**: Kick-Off Meeting with NovaTec. We presented our goals and a first load testings tool list and criteria catalog. Subsequently there was a discussion about additional points that needed to be taken into our elaboration and which topics needed improvement and more attention.

▷ **2nd Milestone 26th August 2014**: Intermediate State Discussion with NovaTec. On this meeting we presented all our results till then. Our progress came to 80 percent completion rate at this time.

▷ **3rd Milestone 2nd October 2014**: Handing in of the final version. At this point we were done with all the testings, summarized all our evaluation results and did several feedback rounds.

▷ **4th Milestone Mid October 2014**: Presentation of our study with focus on testing with WebSocket protocol.

Chapter 3

Basics of Load Testing

This chapter provides an overview about load testing. In Section 3.1 we point out the meaning of load testing. Section 3.2 presents the basics of performance testing, while Section 3.4 and Section 3.5 introduce different workload specification and generation approaches, respectively.

3.1 Meaning of Load Testing

Note that the term 'load test' is ambivalent. It may either refer to performance tests in general or to the most basic form of performance testing as explained in Section 3.3. For *performance testing*, Abbors et al. [2012] give the following definition:

> "The idea behind performance testing is to validate the system under test in terms of its responsiveness, stability, and resource utilization when the system is put under certain synthetic workload in a controlled environment"

3.2 Performance Testing Basics

Performance tests examine the performance of a system under a particular workload. Performance can be defined by many different metrics in this context, e.g., response time, throughput, etc.

A typical test setup is depicted in Figure 3.1. It requires several workload agents to send requests to the system under test. The overall amount and time distribution of these requests is determined by a workload specification and distributed by the controller. During the processing of the requests, the system under test as well as the workload agents measure certain metrics to gather information about the reaction of the system. The data is collected and used to generate a report.

3.3 Variants of Performance Tests

There are several variations of performance tests that can be characterized by their goals. We follow the classification of Subraya and Subrahmanya [2000].

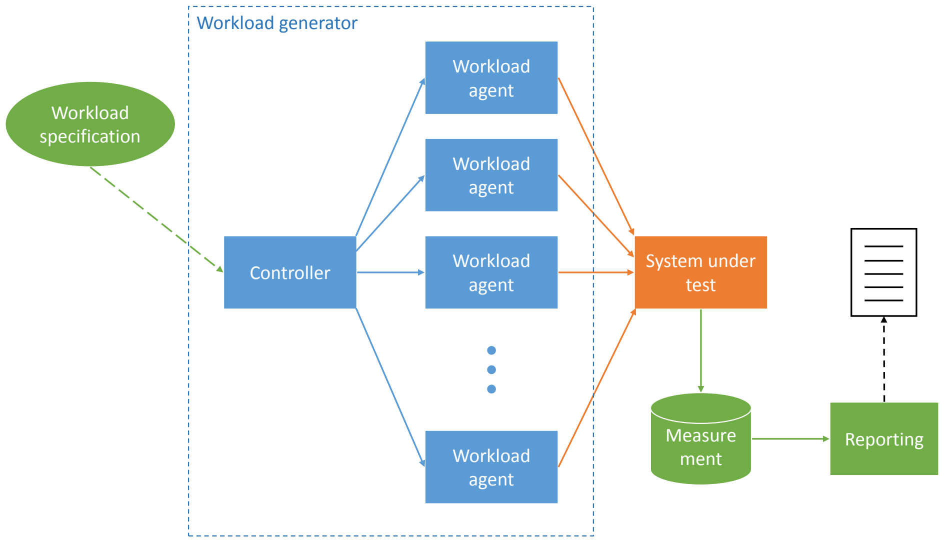


Figure 3.1. General load test architecture

- ▻ **Load tests**: The goal of a load test is to simulate realistic, day-to-day conditions and to observe how the application server can handle it. In particular, think times and arrival rates are used to reflect the observed load patterns in production use.
- ▻ **Stress tests**: A stress test analyzes the behaviour of an overloaded system. It serves to test the stability and robustness of the system under extreme circumstances.
- ▻ **Endurance/Durability/Soak tests**: Endurance tests put load on the system for a longer period of time. The goal is to find memory leaks and other causes of performance degradation.

3.4 Workload Specification Approaches

In this section we will define and explain the different possibilities to generate certain workload specification successfully. The record-and-play approach will be introduced in Section 3.4.1 and scripting in Section 3.4.2.

3.4.1 Record-and-Play

One of the quickest ways to get a workload specification is to once execute a sequence of actions and have the computer record it. Since reproducing the actions by simulating mouse clicks and other UI interactions would be too slow, performance tests are based on sending requests. Therefore, it suffices to record all the request that have been made during the action sequence. There are multiple possibilities to intercept these requests:

▻ **Built-in browser**: The load testing tool contains a mini-browser, in which the user executes the actions.

▻ **Browser plug-in**: The load testing tool provides a plug-in for popular browsers so that the user can perform the actions in her normal browser environment.

▻ **Proxy**: The load testing tool sets up a proxy server that is registered in a normal browser and records the requests.

The main advantage of this approach is that it is fast and easy. In particular, AJAX calls are supported by design.

The record-and-play approach can be improved by breaking down the requests into single steps, allowing the user to recombine the steps to create a different specification. This approach can also be combined with the scripting approach (see next section), if the recording output is provided as a script instead of a list or tree of recordings in the UI.

The possibility to modify and rearrange the recording is a major improvement when it comes to maintainability and flexibility of the tests. However, special parameters (like JSF View State IDs, see Chapter 6) may require some modification, which can be rather easy or difficult, depending on the support of the respective load testing tool.

3.4.2 Scripting

By providing a suitable script language such as JavaScript, a load testing tool can allow the user to specify action sequences in code. In combination with the recording approach introduced above, scripts don't have to be written entirely.

The most significant advantages of this approach originate from the fact that the scripts are in fact code, just like the application they intend to test. Therefore, the load test specification can be versioned and organized by the same configuration management tools and even alongside the code itself, which increases maintainbability. Also, scripts provide a deeper control of the test execution. On the other hand, using the scripting interface may require some training, especially if the tester is not already familiar with the scripting language used by the load testing tool.

3.5 Workload Generation Approaches

Executing a load test usually involves executing one or more workload specifications a couple of times. The number, the temporal distribution and the combination of different specification executions can be handled quite differently.

3.5.1 Open and Closed Workloads

First, we distinguish between open and closed workloads. Schroeder et al. [2006] give the following definition for open and closed workloads:

▷ **Closed workloads**: A new request is only triggered by the completion of a previous request (after a certain *think time*).

▷ **Open workloads**: A new request is only triggered by a new user arrival.

In the context of web applications, particularly those using AJAX, most requests depend on preceding requests. Thus, a sequence of requests always belongs together and—if considered separately—behaves like a closed workload. Therefore, we do not apply the definitions of open and closed workloads on single requests, but on entire executions of workload specifications.

Most tools do not make a difference between open and closed workloads. Instead, it is possible to specify how many parallel "virtual" users should be simulated and how often each of these users performs a workload specification. While the users themselves are executed concurrently, the repetition of a specification that a single user executes is always sequential. Therefore, the generated workloads are generally closed workloads, because a user starts a new cycle only when his previous cycle is completed. But by setting the number of repetitions to a single execution of the specification and by in turn increasing the total number of concurrent users, one can generate open workloads. The typical characteristics of open workloads like arrival rates or distributions can also be achieved by delaying some of the virtual users.

Schroeder et al. [2006] also consider partly-open systems, which combine external arrival rates (open workload) with repetitions (closed workload). In order not to overflow the system, a repetition only occurs with a certain probability, so that the workload is not strictly closed.

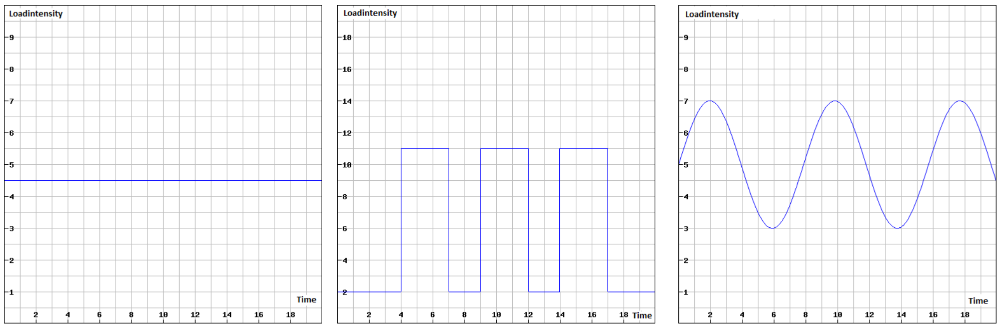
3.5.2 Load Intensity

The second characteristic of a workload is its intensity distribution over time, which is determined by the arrival rate. We only list the most basic patterns and refer to v. Kistowski et al. [2014] for more details.

Note that in production use, users would usually not start coming in all at once, but increasingly over a (short) period of time. This is usually modeled as a **ramp-up** time at the beginning of the test.

Constant

A rather simple way of modeling load distribution is to create a constant number of concurrent users (see Figure 3.2a), which operate independently and send one request after



(a) Constant **(b)** Peaks **(c)** Periodic

Figure 3.2. Different load intensities

another, optionally including a ”think time”. This approach ensures that there is always a constant number of users active in the system.

Peaks

For some application types, a typical production workload is not constant, but rather heavy at a few times only. For example, the registration system used for the tutorial groups at the University of Stuttgart is only used at the beginning of the semester. In the first few weeks, there are a few times with very high workload, while during the rest of the time, the server is rarely used.

This can be modeled as a constant (base) workload combined with workload peaks, that are distributed evenly over the entire time period. A graphical representation of such a load intensity can be seen in Figure 3.2b.

Seasonal/Periodic/Cyclic

Another way of modeling workload is to have a different amount of concurrent users that never drops below a predefined lower limit and never rises above a predefined upper limit, like seen in Figure 3.2c. The concurrent user numbers will alternate between these limits and thus simulate a daily or weekly routine.

Advanced Model-based Approaches

In order to model user behaviour more realistically (especially for large numbers of users), workload generation models can be employed. The execution still consists of basic steps that need to be recorded or scripted (see above). These steps are then combined by a workload generation model to form a more complex workload specification. In particular, the workload model defines the number, frequency and order of the basic steps.

For more details about workload generation models, we refer to Hoorn et al. [2008], Roy et al. [2013], Feitelson [2002], Schroeder et al. [2006] and v. Kistowski et al. [2014].

Chapter 4

Market Overview and Shortlist

In the following, we provide the full list of all load testing tools that we have found on the Internet during our research. At the end of the chapter we list the tools that we selected for the evaluation.

4.1 Market Overview

In this section, we shortly describe the load testing tools we found during our market research.

JMeter (Apache)

Apache JMeter is the de-facto standard open-source tool. While the base functionality is already quite good, it comes with plenty of plug-ins for almost every use case.

▷ License: Apache License 2.0

▷ Website: `http://jmeter.apache.org/`

Load Test (AppPerfect)

The tool developed by AppPerfect provides all important features and also supports scheduling and monitoring. WebSockets are not supported.

▷ License: Proprietary

▷ Website: `http://www.appperfect.com/products/load-test.html`

Load Tester (webperformance)

Webperformance offers a tool focused on browser-recording and replay based on the Selenium framework. Monitoring is supported and a cloud service can be used for load generation. WebSockets are not supported.

- ▻ License: Proprietary
- ▻ Website: `http://www.webperformance.com`

loadIT (NovaTec)

NovaTec loadIT is technically an extension of Apache JMeter. As such, it supports plug-ins written for JMeter and has a similar user interface. There are various improvements and additions.

- ▻ License: Proprietary (free of charge)
- ▻ Website: `http://www.loadit.de/`

LoadRunner (HP)

HP's LoadRunner is based on scripts, has a large number of features and supports a wide range of technologies, including WebSockets.

- ▻ License: Proprietary
- ▻ Website: `http://www8.hp.com/us/en/software-solutions/loadrunner-load-testing/index.html`

Loadtest (alexfernandez)

Loadtest is based on Node.js and provides a command-line as well as a JavaScript interface, and provides only the most basic features.

- ▻ License: MIT license
- ▻ Website: `https://github.com/alexfernandez/loadtest`

LoadUIWeb (SmartBear)

LoadUIWeb Pro is based on browser simulation (in contrast to simply sending predefined requests) and execution in the cloud. WebSockets are not supported directly.

▻ License: Proprietary

▻ Website: `http://smartbear.com/products/qa-tools/load-testing/`

NeoLoad (NeoTys)

NeoTys NeoLoad features a great user guidance and many automated tasks.

▻ License: Proprietary

▻ Website: `http://www.neotys.com/product/overview-neoload.html`

PhantomJS

PhantomJS is a JavaScript framework for website testing that can also be used for performance testing.

▻ License: BSD

▻ Website: `http://phantomjs.org/`

RPT (IBM)

IBM's Rational Performance Tester is a large software package closely integrated with the Java language and the Eclipse framework.

▻ License: Proprietary

▻ Website: `http://www-03.ibm.com/software/products/de/performance/`

Silk Performer (Borland)

Borland's Silk Performer is a feature-rich tool that integrates with virtually any other technology.

- ▻ License: Proprietary
- ▻ Website: `http://www.borland.com/products/silkperformer/`

SOAtest/LoadTest (Parasoft)

Parasoft's SOAtest is designed for API testing of web services, so there is no browser recording and other features common to load testing tools for web applications. However, sending HTTP messages is supported and Parasoft's LoadTest of managing and evaluating load tests, so that load testing web applications is possible. WebSockets are not supported.

- ▻ License: Proprietary
- ▻ Website: `http://www.parasoft.com/soatest`

The Grinder

The Grinder is another popular open-source tool.

- ▻ License: Custom open source license
- ▻ Website: `http://grinder.sourceforge.net/`

Thor

Thor is a utility tool for testing WebSocket connections, based on Node.js.

- ▻ License: MIT license
- ▻ Website: `https://www.npmjs.org/package/thor`

Tsung

Tsung is a load testing tool based on the Erlang programming langauge that supports both HTTP and WebSockets (among other protocols), although the support for the latter is only experimental.

- ▻ License: GPLv2
- ▻ Website: `http://tsung.erlang-projects.org/`

WAPT (Softlogica)

Softlogica's WAPT provides a selection of the most useful and important features on a clean and intuitive user interface that guides the user.

- ▻ License: Proprietary
- ▻ Website: `http://www.loadtestingtool.com`

WebLOAD (RadView)

The most distinctive features of RadView's WebLOAD are its modern Office-like interface, recording to JavaScript and full WebSocket support.

- ▻ License: Proprietary
- ▻ Website: `http://radview.com/`

Webserver Stress Tool (Paessler)

Paessler's Webserver Stress Tool is script-based and only includes rather basic features, but it is advertised to be fast. WebSockets are not supported.

- ▻ License: Proprietary
- ▻ Website: `http://www.de.paessler.com/webstress`

4.2 Shortlist

From the tools that are listed above, we first eliminated the commandline tools like LoadTest, because they lack functionality and are unsuited for the usage in bigger projects. Then we eliminated tools whose main purpose is not load testing, like PhantomJS and SOAtest. Among the remaining tools we selected the following eight representative tools: JMeter, loadIT, NeoLoad, WAPT, RPT, LoadUIWeb Pro, LoadRunner, and Silk Performer.

Chapter 5

Criteria Catalog

In this chapter we list and explain our evaluation criteria. These are grouped into meta criteria (Section 5.2), which are not part of the product itself, non-functional (Section 5.3) and functional (Section 5.4) criteria.

5.1 Scales

For each criterion, we define a scale that it is evaluated on. Some are simply lists of those items that are supported by the tool, or only rated on a binary scale where

″y″ means the item is present/supported and

″n″ means it is not present/supported.

Most criteria are rated on a three-valued ordinal scale where

″-″ means the item does not fulfill the expectations we have in a load testing tool,

″0″ means the item at least fulfills our expectations and

″+″ means the item exceeds our expectations.

5.2 Meta Criteria

In the following subsections we represent analysis methods of meta data within separated subject areas. These results are acquired by statistical methods and valuations. Here we segmented our meta criteria in the following subsections: At first we amplify which licence type our tested product has. In the next subsections we rate the quality and quantity of the documentation and the support service and alternative provided support channels. In the following there are several criteria points summarized in the chapter usability. We also consider the stability rate of the product as well as the extensibility.

5.2.1 License

Even though the license does not influence the rating, it is specified for reference. Using an open source tool may have several advantages that include the ability to debug and customize the tool using its source code. On the other hand, commercial solutions often come with included support. Since support, documentation and extensibility are treated as individual criteria, the license itself is not included in the overall rating. But it will be remarked, if a load testing tool can be used with a commercial or free licence and if the source code is available.

5.2.2 Documentation

The term documentation covers the entirety of information on the use and design of a product. The existence of the following kinds of documentation is evaluated (no matter if it is being provided by the tool publisher or by third parties):

- User manual
- Wiki
- Tutorial (text)
- Tutorial (video)
- Integrated documentation (help-function)
- For open source tools: Detailed JavaDoc or similar source code documentation

Incomplete or outdated documentation is not considered.

5.2.3 Support

Support is a set of communication channels that allows users to get help with problems that could not be solved using the documentation. The existence of the following kinds of support channels is evaluated:

- E-Mail / Contact form
- Telephone
- (Highly available) hotline
- User/Community forum
- Mailing list
- Moderated forum with expert support

Closed or outdated support channels are not considered, but listed for reference.

5.2.4 System Platforms

All supported platforms, i.e., operating systems that the controller and/or the agents can be executed on, are listed. This is not to be confused with the platforms for potential under-test applications, which a tool may have special support for.

5.3 Non-Functional Criteria

5.3.1 Usability

We split the usability into three sub-criteria: simplicity, user guidance, and an undo function.
The simplicity is shown on an ordinal scale ranging from ″-″ to ″+″:

″-″ The user interface is too complex and unstructured or the structure is too confusing. Functions are labeled in an unintuitive way.

″0″ The user interface is easily comprehensible with a little help from the manual. Functions are organized in a coherent structure and labeled appropriately.

″+″ The user interface is structured similarly to other well-known applications, places functions where the average expert user would expect them to be and supports the user with directed hints and expressive icons, so that consulting the manual is not necessary.

The user guidance is shown on an ordinal scale ranging from ″-″ to ″+″:

″-″ The user is not guided in any way and has to find his way through the tool on his own.

″0″ The tool uses wizards and tells the user what he can adjust.

″+″ The tool guides the user through the process of a load test and highlights settings that need to be configuired.

The presence of an undo function is evaluated on a binary scale.

5.3.2 Stability

The stability of the tool is evaluated by the absence of the following events during a one-hour testing:

- Lacking UI responsiveness
- UI crash
- Load generator crash

5.3.3 Extensibility

The existence of the following interfaces is evaluated:

- Protocols
- Reporting formats

5.4 Functional Criteria

Load testing tools offer a broad range of different and sometimes very specific features. This evaluation concentrates on features that are of general interest as well as features that are necessary to test JSF applications. Note that not only features directly included in the product are evaluated, but also features that are available through a plug-in. Here we made a point on separating this topic into the subsections supported protocols and workload specification.

5.4.1 Supported Protocols (K.O. Criterion)

It is evaluated if the tool supports HTTP, HTTPS and WebSockets. Tools that do not support HTTP are excluded completely from the evaluation.

5.4.2 Workload Specification

Some possible ways to generate workload specifications have been described in Section 3.4. For the evaluation of the tools, the extent to which they support these workload generation approaches is evaluated. The workload generation approaches are subdivided into two main methods, namely record and play and also scripting.

Record-and-Play

First, the setup procedure of the record-and-play environment is evaluated. This includes the supported methods (Proxy and/or Tunnel), web browsers (Firefox, Chrome, Internet Explorer, etc.) and the ability of the tool to automatically open and configure the respective browsers. In particular, the tool's internal proxy server needs to be able to handle HTTPS traffic. These single items are documented using a binary scale each.

Second, the further processing of the recording is evaluated. A tool can support the user in various ways to allow her to create a workload specification from her recording quickly:

- Automatic grouping of requests (detecting requests that belong together)
- Manual grouping of requests (by selecting groups or by clicking a button when the current group is complete)
- Custom naming of individual requests by the user (possibly at recording time)
- Outputting the recording as an editable script (which inherently allows copy&paste, editing and commenting). If so:
  - Is the script commented to help the user find individual requests?
  - Is the script well structured and easily readable?
  - Does the script structure allow to deactivate and/or exchange requests quickly?

- Outputting the recording as a list on the UI, which may allow to:
  - Reorder requests
  - Dynamically deactivate requests (for testing purposes)
  - Copy&Paste requests
  - Edit request parameters

The support for each of these is documented on a binary scale and the overall user experience using the record-and-play function is rated on an ordinal scale from ″-″ to ″+″.

In addition, the ability to handle the JSF View State ID, which is required for our specific test application (see Chapter 6), is evaluated on an ordinal scale from ″-″ to ″+″:

″-″ The JSF View State ID must be retrieved manually and inserted into subsequent requests. The procedure to do so is not documented and/or time-consuming.

″0″ The JSF View State ID must be retrieved manually and inserted into subsequent requests. The procedure to do so is well documented and the tool allows to apply the new View State ID to multiple subsequent requests at once.

″+″ The JSF View State ID is detected and handled automatically.

Note that testing tools that cannot handle JSF (JavaServer Faces) View State at all are excluded from our evaluation.

Scripting

If the tool offers some kind of scripting language to create or enhance workload specifications, the following criteria are evaluated:

▻ Is the language a common language that an average web developer might already know?

▻ Are the features of the language documented with respect to the use in the load testing tool?

▻ Does the load testing tool offer syntax highlighting?

▻ Does the load testing tool offer auto completion?

5.4.3 Execution

In the following subsections we define which aspects of a tool have to be taken into account to provide an evaluation basis for the whole functionality of the load testing tool. The different aspects are clearly sophisticated. Some of them have their own matching rate with ideal load testing tool features which are predefined in every subsection.

Combination of Test Parts

A useful feature to quickly assemble different workload specifications is the combination of tests from smaller (previously recorded or scripted) parts. In addition to the mere possibility to combine and rearrange test parts, control structures like branching (`if`) and looping (`while`, `for`) have been evaluated.

Parametrisation

In order to run workload specifications, which follow the same sequence of requests, but are executed with different parameters, a feature is required to assign certain request parameters to values retrieved from a list or file. This feature—if present—has been evaluated by using different login data for otherwise identical workload specifications, and rated on an ordinal scale from ″-′ to ″+″:

″-″ Although there is no particular support for parametrisation, it is possible to duplicate workload specifications and change individual parameters manually.

″0″ The tool supports inserting different values for specific parameters. Values are entered in a table.

″+″ The tool supports inserting different values for specific parameters and can also evaluate value expressions. Values can be entered in a table or imported from a file.

Pre-test Execution

After creating a workload specification, the user wants to verify if it runs correctly against her application server, mostly without running an entire load test to do so. A feature that ignores all settings about load generation models, distributed load generation agents, etc. and only executes a workload specification once is useful to determine whether the workload specification and application server are correctly configured for the main test execution. Only the existence of such a feature is evaluated on a binary scale.

Workload Models

It is evaluated if the tool supports the following workload models:

- Constant distribution
- Peaks
- Periodic, Cyclic, Seasonal
- User-defined function

It is also evaluated if the tool supports Ramp-Up.

Distributed Execution

An important part of load test executions is to ensure that the load generator itself is not the bottleneck. Therefore, multiple machines are used as load generators and they need to be coordinated. The extent to which the load testing tool facilitates the configuration, execution, and data collection on multiple machines is evaluated on an ordinal scale from ″-″ to ″+″:

″-″ Distributed test execution is not supported at all by the tool and could only be performed manually.

″0″ Distributed test execution is possible, but a lot of manual configuration is required. The tool supports at least starting and stopping remote machines simultaneously.

″+″ Distributed test execution is supported by the tool. It helps setting up remote machines and transferring configurations and test data to those machines. After executing the test, the tool is able to collect the recorded results automatically.

Test Abortion

It is evaluated on a binary scale, whether the tool supports canceling a running test within a reasonably short time, also on all remote machines, if those can be controlled from within the tool.

5.4.4 Reporting

This chapter only deals with the report function of the load testing tool. The different aspects we take into account are clearly distinguished in report contents, -format, -generation and report management. The subsections contain an ideal feature that we expect load testing tools to have.

Report Contents

To evaluate the reports generated by the load testing tools, mostly the information included in the reports is taken into account:

- Input data and test environment
- Request times
- Throughput
- Utilization

Report Format

The available formats for reports should on the one hand provide a clearly represented overview of the test results and on the other hand provide structured and detailed information. Furthermore, a machine-readable format that allows for further use of the data in other applications (e.g., Microsoft Excel©) can be useful.

The available reporting formats as a whole are rated using an ordinal scale from ″-″ to ″+″:

″-″ The reports can be displayed within the tool’s interface and options for export are very limited or unsatisfying, so that a good visualization cannot be achieved even after exporting the raw data to another tool.

″0″ The reports can be displayed and there are several useful export options, at least one of which is a structured data format. Results are put into graphs to provide an overview.

″+″ Reports are displayed both as graphs and as tables. The offered export formats are well structured and offer detailed information.

Report Generation

It is investigated whether the testing tool allows customization of its reports. This involves the possibility to select the parts of the report (rated on a boolean scale), as well as the possibility to provide additional data sources such as the results of profiling instruments, etc.

Moreover, it is an advantage if a tool provides partial results during the test or when a test fails, both documented on a binary scale.

Report Management

When conducting a lot of load tests, it is important to have some kind of report management. Features like local report management, centralized report management, report sharing and report comparison can be provided by a load testing tool to facilitate the report management. Additionally, a tool might be able to compare the results of multiple test executions and generate comparative reports. These individual features are documented on a binary scale.

5.4.5 Maintainability of Workload Specifications

Note that this is not a strictly separate criterion, since it is mainly influenced by other features. But because of its importance in production use, we evaluate it separately.

In practice, an important part of load testing is to keep up with the changes of the system under test. Therefore, a good load testing tool should support the maintenance of load tests. We identified three major areas that affect maintainability.

▷ **Response Validation**: Due to misconfiguration, network problems, or server overload, a request can terminate with an error. Such errors can occur in different forms: The server may be unreachable, it may send HTTP Status codes like 500 or 404, it may send (small, quick) responses that indicate an internal subsystem failure (e.g., `<error>This server is overloaded, please return later.</error>`) and are usually handled by the UI layer in the browser, or it may simply send false responses (such as empty result sets).

To ensure that the server was actually able to handle the load and did not only reply with error messages, which requires a lot less time per client, load testing tools should provide simple and efficient means to verify the responses. Possible verification methods include:

For the evaluation we list the supported verification methods.

- Automatically detect and report network failures and negative HTTP Response Code
- Regex evaluation of responses or similar textual evaluation
- XPath queries on the response
- Automatic detection of common error message types
- Compare actual responses to recorded responses

▷ **Modifying outdated tests**: When the tester needs to modify a large workload specification, it is important that the tool allows reordering, deleting and modifying the requests quickly. In addition, a group-edit or batch-change or global search-replace feature can save a lot of time.

▷ **Modular test design**: By re-using parts of workload specifications and assembling longer tests from smaller modules, the amount of specification data can be reduced, which means that there is less to maintain. Load testing tools should allow the user to extract parts of a test, create a module and reference it from other tests.

▷ **Manageable file format**: If the load testing tool stores its data in a format (like XML) that is recognized by source code management tools, the tests can be stored and managed besides the software itself. It also makes merging and branching possible for the workload specifications.

The overall ability of the tools to support load test maintenance is rated on an ordinal scale from ″-″ to ″+″.

″-″ The tool does not have specific features to increase maintainability.

″0″ The tool only offers some of the features listed above or some of them do not work reliably.

″+″ The tool offers most of the features listed above and they are easy to use and work reliably.

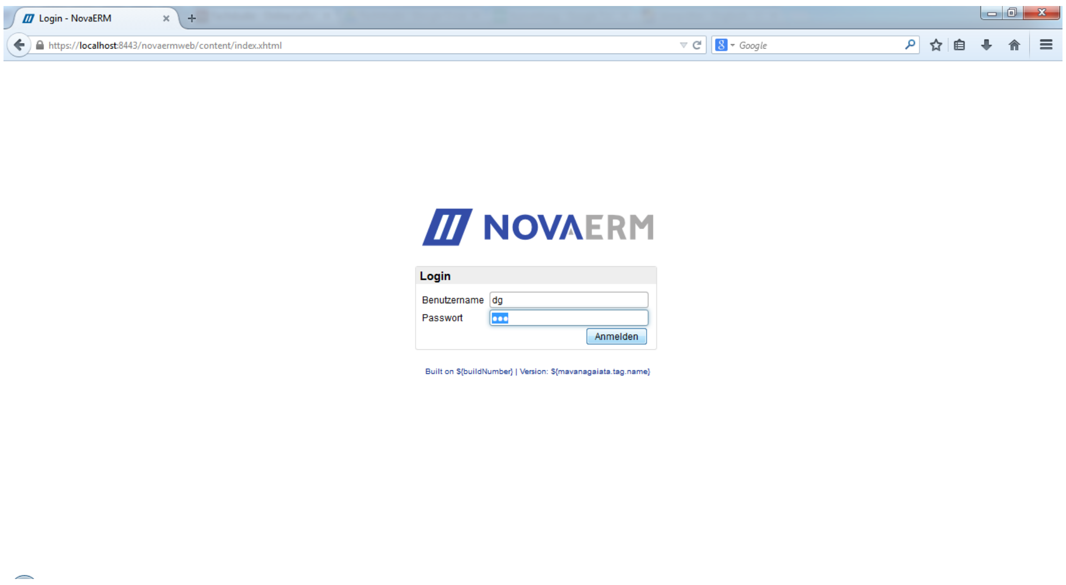
Chapter 6

Evaluation Procedure

This chapter presents the system under test (see Section 6.1), the test environment (see Section 6.2), and the test scenario (see Section 6.3) used for the evaluation.

6.1 The System Under Test

The cooperation partner NovaTec (see Section 1.3) has provided an internally used and developed application called *NovaERM* (see Figure 6.1) for this study. NovaERM is based on Java Enterprise technologies like Enterprise Java Beans, Java Persistence API, Java Server Faces 2.2, PrimeFaces and also JavaScript and therefore makes heavy use of AJAX. In order to test the WebSockets capabilities of the load testing tools, NovaERM has been extended by a feature based on WebSockets.


Login - NovaERM
https://localhost:8443/novaermweb/content/index.xhtml

Login

Benutzername dg
Passwort ●●●

Built on ${buildNumber} | Version: ${mavanagaiata.tag.name}

Figure 6.1. Login of the tested application

6.2 Test Environment

All tests have been executed on regular laptops running Windows 7. In order to avoid negative performance impacts, the test systems were connected to AC power during the tests.

As application server, WildFly 8.1 was used to deploy the application. Also the latest versions of the browsers supported by the tools (Chrome, Firefox Internet Explorer) were used. Please note that the actually measured performance of the application is not of interest in this study. In particular, the correctness and stability of the measurement results, which the tools produced, have not been verified.

6.3 Test Scenario

In order to assess the tools' capability to record browser requests and generate workload specifications, we created a suitable test scenario. The test scenario includes a login, two JSF views (each of which has its own View State ID) and several AJAX requests. Since the test scenario is recorded from user interactions, we illustrate the scenario using screenshots:

The first action is the login (see Figure 6.1). After filling in the login form, the user clicks the login button ("Anmelden"), which triggers a POST request to the server.

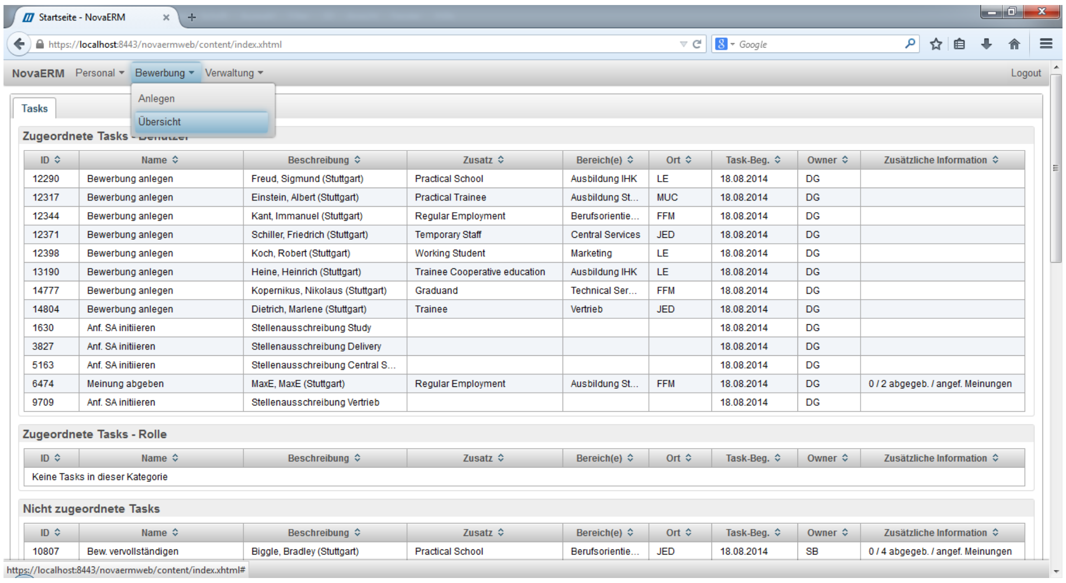
In response to the login request, the server redirects the browser (using a 302 response code) to the homepage, which is the first JSF view. When loading the homepage, the browser loads a number of additional JS and CSS files from the server.

The homepage shows a list of tasks (see Figure 6.2), which is not used in the test scenario. Instead, the tester opens a search form for applicants from the menu ("Bewerbung", "Bewerberübersicht"). The search form is displayed in a new PrimeFaces tab using an AJAX post-back request to the JSF view. The tester then switches to the search mode for applicants (instead of tasks) using the radio-box ("Bewerbung") at the top, which triggers another AJAX request to refresh the entire user interface below.

The third step is to produce search results. In order to be able to perform the recording quickly, the tester does not enter a search query, but simply clicks the search button ("Suchen") to retrieve all records from the database, which is done in another AJAX request. The search usually takes less than a second.

After the results have been retrieved (see Figure 6.3), the user clicks the first entry. This opens a new tab (see Figure 6.4) containing information about the selected applicant. The tab is opened using a GET request to a specific URL, i.e., this step does not include another AJAX request, but it creates a new JSF view, which is assigned a different view state ID (!).

To ensure that the load testing tools are capable of detecting the new JSF view state ID, the tester triggers two AJAX requests in the new view by clicking the checkbox labelled



Zugeordnete Tasks - Benutzer

ID	Name	Beschreibung	Zusatz	Bereich(e)	Ort	Task-Beg.	Owner	Zusätzliche Information
12290	Bewerbung anlegen	Freud, Sigmund (Stuttgart)	Practical School	Ausbildung IHK	LE	18.08.2014	DG	
12317	Bewerbung anlegen	Einstein, Albert (Stuttgart)	Practical Trainee	Ausbildung St...	MUC	18.08.2014	DG	
12344	Bewerbung anlegen	Kant, Immanuel (Stuttgart)	Regular Employment	Berufsorientie...	FFM	18.08.2014	DG	
12371	Bewerbung anlegen	Schiller, Friedrich (Stuttgart)	Temporary Staff	Central Services	JED	18.08.2014	DG	
12398	Bewerbung anlegen	Koch, Robert (Stuttgart)	Working Student	Marketing	LE	18.08.2014	DG	
13190	Bewerbung anlegen	Heine, Heinrich (Stuttgart)	Trainee Cooperative education	Ausbildung IHK	LE	18.08.2014	DG	
14777	Bewerbung anlegen	Kopernikus, Nikolaus (Stuttgart)	Graduand	Technical Ser...	FFM	18.08.2014	DG	
14804	Bewerbung anlegen	Dietrich, Marlene (Stuttgart)	Trainee	Vertrieb	JED	18.08.2014	DG	
1630	Anf. SA initiieren	Stellenausschreibung Study				18.08.2014	DG	
3827	Anf. SA initiieren	Stellenausschreibung Delivery				18.08.2014	DG	
5163	Anf. SA initiieren	Stellenausschreibung Central S...				18.08.2014	DG	
6474	Meinung abgeben	MaxE, MaxE (Stuttgart)	Regular Employment	Ausbildung St...	FFM	18.08.2014	DG	0 / 2 abgegeb. / angef. Meinungen
9709	Anf. SA initiieren	Stellenausschreibung Vertrieb				18.08.2014	DG	

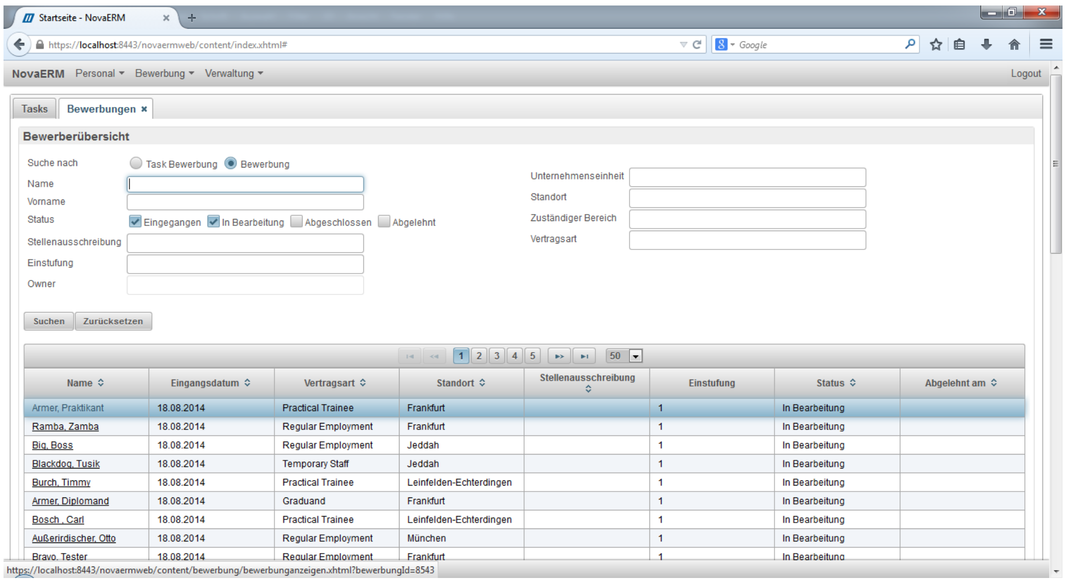
Zugeordnete Tasks - Rolle

ID	Name	Beschreibung	Zusatz	Bereich(e)	Ort	Task-Beg.	Owner	Zusätzliche Information
Keine Tasks in dieser Kategorie								

Nicht zugeordnete Tasks

ID	Name	Beschreibung	Zusatz	Bereich(e)	Ort	Task-Beg.	Owner	Zusätzliche Information
10807	Bew. vervollständigen	Biggle, Bradley (Stuttgart)	Practical School	Berufsorientie...	JED	18.08.2014	SB	0 / 4 abgegeb. / angef. Meinungen

Figure 6.2. The tasklist in NovaERM



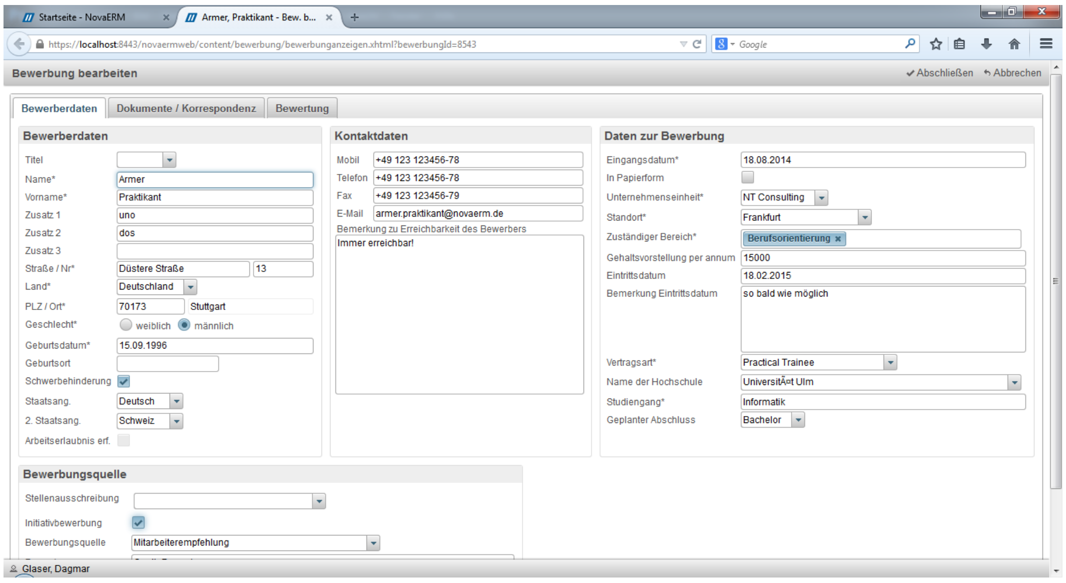
Bewerberübersicht

Suche nach	Task Bewerbung [✓ Bewerbung]
Name	
Vorname	
Status	[✓ Eingegangen] [✓ In Bearbeitung] Abgeschlossen Abgelehnt
Stellenausschreibung	
Einstufung	
Owner	
Unternehmenseinheit	
Standort	
Zuständiger Bereich	
Vertragsart	

Name	Eingangsdatum	Vertragsart	Standort	Stellenausschreibung	Einstufung	Status	Abgelehnt am
Armer, Praktikant	18.08.2014	Practical Trainee	Frankfurt		1	In Bearbeitung	
Ramba, Zamba	18.08.2014	Regular Employment	Frankfurt		1	In Bearbeitung	
Big, Boss	18.08.2014	Regular Employment	Jeddah		1	In Bearbeitung	
Blackdog, Tusik	18.08.2014	Temporary Staff	Jeddah		1	In Bearbeitung	
Burch, Timmy	18.08.2014	Practical Trainee	Leinfelden-Echterdingen		1	In Bearbeitung	
Armer, Diplomand	18.08.2014	Graduand	Frankfurt		1	In Bearbeitung	
Bosch , Carl	18.08.2014	Practical Trainee	Leinfelden-Echterdingen		1	In Bearbeitung	
Außerirdischer, Otto	18.08.2014	Regular Employment	München		1	In Bearbeitung	
Bravo, Tester	18.08.2014	Regular Employment	Frankfurt		1	In Bearbeitung	

https://localhost:8443/novaermweb/content/bewerbung/bewerbunganzeigen.xhtml?bewerbungId=8543

Figure 6.3. Search screen for applicants



Startseite - NovaERM
Armer, Praktikant - Bew. b...

https://localhost:8443/novaermweb/content/bewerbung/bewerbunganzeigen.xhtml?bewerbungId=8543

Bewerbung bearbeiten

✓ Abschließen ↰ Abbrechen

Bewerberdaten | Dokumente / Korrespondenz | Bewertung

Bewerberdaten

Titel	
Name*	Armer
Vorname*	Praktikant
Zusatz 1	uno
Zusatz 2	dos
Zusatz 3	
Straße / Nr*	Düstere Straße 13
Land*	Deutschland
PLZ / Ort*	70173 Stuttgart
Geschlecht*	weiblich [✓ männlich]
Geburtsdatum*	15.09.1996
Geburtsort	
Schwerbehinderung	✓
Staatsang.	Deutsch
2. Staatsang.	Schweiz
Arbeitserlaubnis erf.	

Kontaktdaten

Mobil	+49 123 123456-78
Telefon	+49 123 123456-78
Fax	+49 123 123456-79
E-Mail	armer.praktikant@novaerm.de
Bemerkung zu Erreichbarkeit des Bewerbers	Immer erreichbar!

Daten zur Bewerbung

Eingangsdatum*	18.08.2014
In Papierform	
Unternehmenseinheit*	NT Consulting
Standort*	Frankfurt
Zuständiger Bereich*	Berufsorientierung
Gehaltsvorstellung per annum	15000
Eintrittsdatum	18.02.2015
Bemerkung Eintrittsdatum	so bald wie möglich
Vertragsart*	Practical Trainee
Name der Hochschule	UniversitÃ¤t Ulm
Studiengang*	Informatik
Geplanter Abschluss	Bachelor

Bewerbungsquelle

Stellenausschreibung	
Initiativbewerbung	✓
Bewerbungsquelle	Mitarbeiterempfehlung

Glaser, Dagmar

Figure 6.4. Page with detailed information about the application

”Initiativbewerbung” twice. Since the UI underneath the checkbox depends on its state, each click triggers an AJAX request.

After all four steps have been completed, the tester first stops the recording and then logs out of the application and closes the two tabs to ensure that the next test will start in a new session.

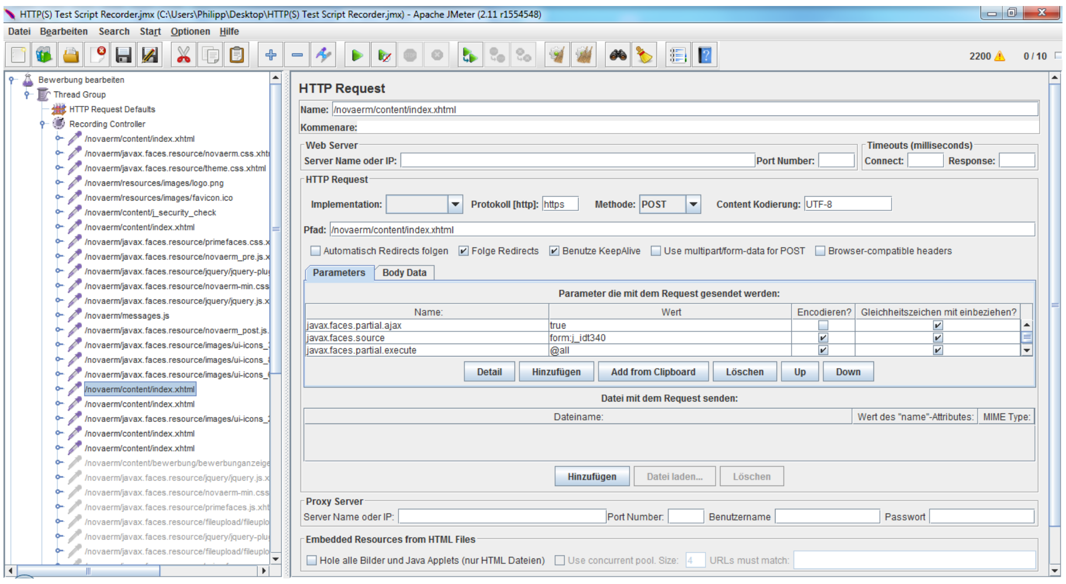
Chapter 7

Evaluation Results

This chapter provides the documented evaluation results of the eight selected load testing tools (see Section 4.2).

7.1 JMeter

The manufacturing company of JMeter is Apache and the license of this load testing tool is Apache 2.0. The version tested in this elaboration is JMeter 2.11, which is free to use for companies and private persons. JMeter 2.11 can be ran on every operating system as long as it has a working Java Virtual Machine (JVM).



HTTP(S) Test Script Recorder.jmx (C:\Users\Philipp\Desktop\HTTP(S) Test Script Recorder.jmx) - Apache JMeter (2.11 r1554548)

```
Bewerbung bearbeiten
  Thread Group
    HTTP Request Defaults
    Recording Controller
      /novaerm/content/index.xhtml
      /novaerm/javax.faces.resource/novaerm.css.xht
      /novaerm/javax.faces.resource/theme.css.xhtml
      /novaerm/resources/images/logo.png
      /novaerm/resources/images/favicon.ico
      /novaerm/content/j_security_check
      /novaerm/content/index.xhtml
      /novaerm/javax.faces.resource/primefaces.css.x
      /novaerm/javax.faces.resource/novaerm_pre.js.x
      /novaerm/javax.faces.resource/jquery/jquery-plu
      /novaerm/javax.faces.resource/novaerm-min.css
      /novaerm/javax.faces.resource/jquery/jquery.js.x
      /novaerm/messages.js
      /novaerm/javax.faces.resource/novaerm_post.js.
      /novaerm/javax.faces.resource/images/ui-icons_
      /novaerm/javax.faces.resource/images/ui-icons_
      /novaerm/javax.faces.resource/images/ui-icons_
      /novaerm/content/index.xhtml
      /novaerm/content/index.xhtml
      /novaerm/javax.faces.resource/images/ui-icons_
      /novaerm/content/index.xhtml
      /novaerm/content/index.xhtml
      /novaerm/content/bewerbung/bewerbunganzeige
      /novaerm/javax.faces.resource/jquery/jquery.js.x
      /novaerm/javax.faces.resource/novaerm-min.css
      /novaerm/javax.faces.resource/primefaces.js.xht
      /novaerm/javax.faces.resource/fileupload/fileuplo
      /novaerm/javax.faces.resource/jquery/jquery-plu
      /novaerm/javax.faces.resource/fileupload/fileuplo
```

HTTP Request

Name: /novaerm/content/index.xhtml
Kommenare:

Web Server
Server Name oder IP: Port Number:

Timeouts (milliseconds)
Connect: Response:

HTTP Request
Implementation: Protokoll [http]: https Methode: POST Content Kodierung: UTF-8
Pfad: /novaerm/content/index.xhtml

☐ Automatisch Redirects folgen ☑ Folge Redirects ☑ Benutze KeepAlive ☐ Use multipart/form-data for POST ☐ Browser-compatible headers

Parameter die mit dem Request gesendet werden:

Name:	Wert	Encodieren?	Gleichheitszeichen mit einbeziehen?
javax.faces.partial.ajax	true	☐	✓
javax.faces.source	form:j_idt340	✓	✓
javax.faces.partial.execute	@all	✓	✓

Datei mit dem Request senden:

Dateiname:	Wert des "name"-Attributes:	MIME Type:

Proxy Server
Server Name oder IP: Port Number: Benutzername Passwort

Embedded Resources from HTML Files
☐ Hole alle Bilder und Java Applets (nur HTML Dateien) ☐ Use concurrent pool. Size: 4 URLs must match:

Figure 7.1. JMeter's UI

7.1.1 Download and Installation

JMeter comes packaged as a zip archive. There is no setup required other than extracting the archive and providing a JVM. In particular, there is no selective installation and no helper for plug-in installation. To simplify the search for and installation of suitable plug-ins, the website `jmeter-plug-ins.org` provides predefined sets of common plug-ins. The source code of JMeter is freely available.

7.1.2 User Interface

The user interface is rather complex and has only partly been translated to languages other than English. The main concept is to build a tree of elements that form the test. All kinds of different elements like tools for recording, the recordings themselves, controllers for the execution, results are organized in that main tree. Adding new elements requires experience, a good tutorial or lucky guessing, since items are not labeled intuitively. There are no wizards or beginner tips that introduce the interface and help with the first steps.

For the experienced user, the JMeter user interface serves its purpose and allows for quick interaction with the software. JMeter does not provide an undo/redo feature.

7.1.3 Workload Specification

JMeter allows recording requests through a proxy server. While this proxy server works with all major browsers, JMeter does not configure or open them automatically. The HTTPS certificate also needs to be installed manually.

For the JSF View State ID, there is no dedicated feature in JMeter to automatically handle or even detect it. However, JMeter is able to deal with the View State, since it supports variables. Detailed instructions to manually configure the View State ID handling are available on the internet.

WebSocket support is provided through a plug-in. Though the capability is very simple. One can send and receive messages and specify the connection to be used. Also it is not possible to record the ongoing WebSocket connection, hence the communication has to be created by hand.

Recorded requests are not grouped automatically, but they can be grouped manually after the recording is completed. This is a little cumbersome, though. Renaming, reordering and deleting recorded requests as well as creating entirely new ones is easily possible.

JMeter uses a sophisticated GUI for viewing and editing request details. Request parameters can be edited quickly in a table.

JMeter provides integration with various scripting languages. However, none of them can be used to drive a load test from code entirely. Instead, JMeter offers several points of integration with the scripting languages: Timers, pre- and post-processors, samplers, assertions and listeners can be implemented as a script instead of defining them on the UI. The older ones of the supported languages are BeanShell and BSF. While their interpreters still work, they have been deprecated in favor of a JSR-223 interpreter, which—among others—also supports BeanShell, JavaScript, and many others.

Although JMeter is able to include the scripts from external files, so that an external editor could be used, it also provides an integrated editor with syntax highlighting. Other than that, the editor has only few features, which include bracket highlighting and block folding—but there is no auto completion or integrated documentation.

There is not much documentation on how to use the scripting languages in JMeter. The most important information (that is the available variables) is given directly in the editor window. And of course, there is a general documentation of the scripting languages' syntax available on the internet.

In addition, JMeter has a (small) expression language to evaluate variable values for request parameters, etc.

7.1.4 Execution

JMeter does not offer a pre-test execution. It is possible to combine different test parts, also by using loops and branches. While JMeter does not offer a specific feature for running a set of requests with varying parameters, a limited kind of parametrization can be achieved by duplicating an entire sub-tree.

Natively, JMeter only offers constant load generation with ramp-up. More complicated workload specifications can be achieved with plug-ins or manually by using timers and loops.

Setting up a distributed load generation environment is simple and JMeter both starts and stops/aborts distributed execution quickly.

JMeter shows most of its results already during the test execution. It cannot import external data or use profiling tools to collect additional data. Distributed execution is also possible and very easy to set up.

7.1.5 Monitoring

JMeter shows most of its results already during the test execution. But it is not able to import external data or use profiling tools to collect additional data.

7.1.6 Organization, Presentation and Export of Test Results

JMeter does not have a result management; results are stored in special nodes in the tree. Running a subsequent test either overwrites the existing results, or mixes them with the new results.

Results are presented using graphs and tables. There is a variety of ”listeners” available to collect and display different kinds of data. The design is quite simple and graphs sometimes do not scale correctly so that the actual information is too small and/or the graph does not fit the containing window.

Most of the graphs can be exported as images and the raw data can be exported in CSV format. When used with the UI, all exports need to be triggered individually. In headless mode the export is done automatically. The final results contain request and response times, appropriate utilization concepts within the same graph and performance values represented by the PerfMon-Profiler.

7.1.7 Maintainability

To detect outdated load test specifications, the various response assertions in JMeter can be used. However, these are mostly designed to detect server failures. In particular, there is no feature to compare responses for similarity to a recorded response. With some manual work, it is possible to create reasonable assertions, and JMeter offers several ways to do so. There are assertions for duration, length and title of the response, simple (string based) matchers, and for more complicated cases there is the scripting language integration. Altogether, a detection for changed applications can be hand-crafted in JMeter.

When the application has changed, it is rather difficult to adjust the tests in JMeter. While reordering and editing of single requests is easily possible, there is no feature to modify multiple requests at once. Therefore, the most efficient solution to, e.g., react to a changed JSF ID is to open the file in a text editor and use the search-and-replace function there.

The fact that JMeter uses an XML based file format makes the latter quite easy. Aside from being easily editable from other tools, the XML format is also well suited for versioning.

Modularization is possible in JMeter by moving requests to new modules in the ″WorkBench″ and then using the ″Module controller″ to include them into workload specifications. While the interface and implementation of this feature is plain and simple, it is still very effective in combination with the drag-drop functionality in JMeter′s main tree.

7.1.8 Extensibility

JMeter has a wide interface for plug-ins that gives users the opportunity to extend nearly every aspect of the tool including the support of other protocols and the introduction of new graphs.

7.1.9 Support

JMeter has an integrated HTML documentation that is also available online. While the offline viewer is quite difficult to handle, the contents are comprehensive and helpful. For additional help, there is both a forum and a mailing list. Many hints, tips and tutorials in video and text format can be found on the web. The wiki was not accessible during our evaluation.

7.1.10 Summary

JMeter is the only open source tool we evaluated. While the GUI is very complex and sometimes cumbersome to use, there is enough documentation and a large community to help with problems. Besides that, JMeter shines through it's extensibility which led to many useful plug-ins. With the "Module controller" and the versionable file format JMeter also allows for sustainable load testing.

7.2 loadIT

loadIT is based on JMeter. Its manufacturer NovaTec offers the proprietary tool free of charge, unless it is being used by other consulting companies.

7.2.1 Download and Installation

Up to the tested version 1.3, the tool required a license that could be obtained by registration. In the newer version 1.4, loadIT can be installed by simply downloading and extracting a zip archive. On Windows, the BAT file to start the tool did not work (though this may be due to misconfiguration), but the SH file intended for Linux did.

7.2.2 User Interface

Although loadIT uses a more modern UI design, the interface structure is basically the same as JMeter's. NovaTec offers additional tree elements annotated with "(NT)" that provide more features and are slightly more usable. In particular, loadIT has good default settings like the "Prepopulate default test tree" feature.

While experienced users and also JMeter users will easily find their way in loadIT, the user interface might still be unneccesarily complex for beginners.

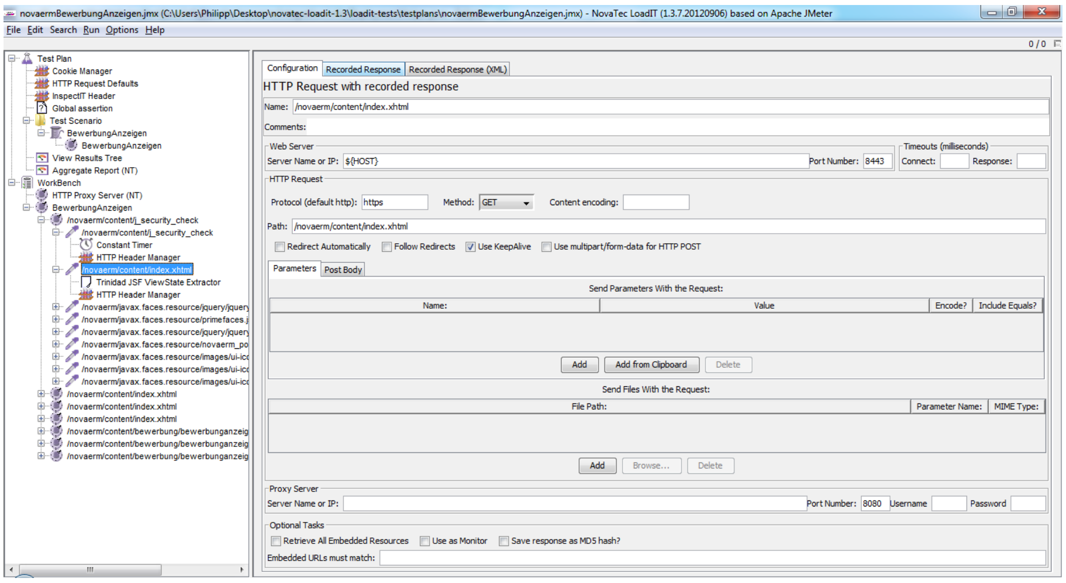
7.2.3 Record and Play and Further Editing

loadIT provides a couple of improvements that simplify the creation of workload specifications. It has a modified proxy server and recorder, which is capable of opening the default browser automatically and is easier to set up. However, it cannot configure the browsers, either.

The JSF View State ID can be handled by a dedicated "extractor", which needs to be activated manually and works fine.

The JMeter WebSocket plug-in seems not to be compatible with loadIT.

Recorded requests are grouped automatically in a reasonable way and (re-)grouping them manually is easier than in JMeter. Renaming, reordering and deleting recorded requests as well as creating entirely new ones is easily possible.



novaermBewerbungAnzeigen.jmx (C:\Users\Philipp\Desktop\novatec-loadit-1.3\loadit-tests\testplans\novaermBewerbungAnzeigen.jmx) - NovaTec LoadIT (1.3.7.20120906) based on Apache JMeter

File Edit Search Run Options Help

0 / 0

```
Test Plan
    Cookie Manager
    HTTP Request Defaults
    InspectIT Header
    Global assertion
    Test Scenario
        BewerbungAnzeigen
            BewerbungAnzeigen
    View Results Tree
    Aggregate Report (NT)
WorkBench
    HTTP Proxy Server (NT)
    BewerbungAnzeigen
        /novaerm/content/j_security_check
            /novaerm/content/j_security_check
                Constant Timer
                HTTP Header Manager
            /novaerm/content/index.xhtml
                Trinidad JSF ViewState Extractor
                HTTP Header Manager
            /novaerm/javax.faces.resource/jquery/jquery
            /novaerm/javax.faces.resource/primefaces.j
            /novaerm/javax.faces.resource/jquery/jquery
            /novaerm/javax.faces.resource/novaerm_po
            /novaerm/javax.faces.resource/images/ui-icc
            /novaerm/javax.faces.resource/images/ui-icc
            /novaerm/javax.faces.resource/images/ui-icc
        /novaerm/content/index.xhtml
        /novaerm/content/index.xhtml
        /novaerm/content/index.xhtml
        /novaerm/content/bewerbung/bewerbunganzeig
        /novaerm/content/bewerbung/bewerbunganzeig
        /novaerm/content/bewerbung/bewerbunganzeig
```

Configuration | Recorded Response | Recorded Response (XML)

HTTP Request with recorded response

Name: /novaerm/content/index.xhtml

Comments:

Web Server — Server Name or IP: ${HOST} Port Number: 8443

Timeouts (milliseconds) — Connect: Response:

HTTP Request — Protocol (default http): https Method: GET Content encoding:

Path: /novaerm/content/index.xhtml

☐ Redirect Automatically ☐ Follow Redirects ☑ Use KeepAlive ☐ Use multipart/form-data for HTTP POST

Parameters | Post Body

Send Parameters With the Request:

Name:	Value	Encode?	Include Equals?

Add | Add from Clipboard | Delete

Send Files With the Request:

File Path:	Parameter Name:	MIME Type:

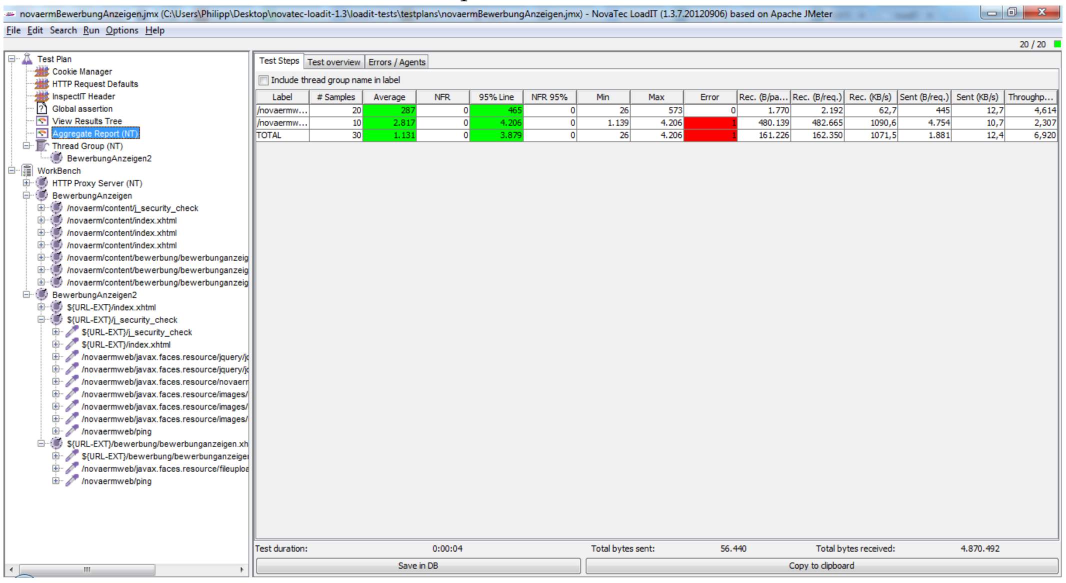
Add | Browse... | Delete

Proxy Server — Server Name or IP: Port Number: 8080 Username Password

Optional Tasks — ☐ Retrieve All Embedded Resources ☐ Use as Monitor ☐ Save response as MD5 hash?

Embedded URLs must match:

(a) Request view in loadIT



novaermBewerbungAnzeigen.jmx (C:\Users\Philipp\Desktop\novatec-loadit-1.3\loadit-tests\testplans\novaermBewerbungAnzeigen.jmx) - NovaTec LoadIT (1.3.7.20120906) based on Apache JMeter

File Edit Search Run Options Help

20 / 20

```
Test Plan
    Cookie Manager
    HTTP Request Defaults
    InspectIT Header
    Global assertion
    View Results Tree
    Aggregate Report (NT)
    Thread Group (NT)
        BewerbungAnzeigen2
WorkBench
    HTTP Proxy Server (NT)
    BewerbungAnzeigen
        /novaerm/content/j_security_check
        /novaerm/content/index.xhtml
        /novaerm/content/index.xhtml
        /novaerm/content/index.xhtml
        /novaerm/content/bewerbung/bewerbunganzeig
        /novaerm/content/bewerbung/bewerbunganzeig
        /novaerm/content/bewerbung/bewerbunganzeig
    BewerbungAnzeigen2
        ${URL-EXT}/index.xhtml
        ${URL-EXT}/j_security_check
            ${URL-EXT}/j_security_check
            ${URL-EXT}/index.xhtml
            /novaermweb/javax.faces.resource/jquery/jc
            /novaermweb/javax.faces.resource/jquery/jc
            /novaermweb/javax.faces.resource/novaerm
            /novaermweb/javax.faces.resource/images/
            /novaermweb/javax.faces.resource/images/
            /novaermweb/javax.faces.resource/images/
            /novaermweb/ping
        ${URL-EXT}/bewerbung/bewerbunganzeigen.xh
            ${URL-EXT}/bewerbung/bewerbunganzeigen
            /novaermweb/javax.faces.resource/fileuploa
            /novaermweb/ping
```

Test Steps | Test overview | Errors / Agents

☐ Include thread group name in label

Label	# Samples	Average	NFR	95% Line	NFR 95%	Min	Max	Error	Rec. (B/pa...	Rec. (B/req.)	Rec. (KB/s)	Sent (B/req.)	Sent (KB/s)	Throughp...
/novaermw...	20	287	0	465	0	26	573	0	1.770	2.192	62,7	445	12,7	4,614
/novaermw...	10	2.817	0	4.206	0	1.139	4.206	1	480.139	482.665	1090,6	4.754	10,7	2,307
TOTAL	30	1.131	0	3.879	0	26	4.206	1	161.226	162.350	1071,5	1.881	12,4	6,920

Test duration: 0:00:04 Total bytes sent: 56.440 Total bytes received: 4.870.492

Save in DB | Copy to clipboard

(b) Results table in loadIT

Figure 7.2. loadIT's UI

7.2.4 Scripting

In contrast to JMeter, loadIT only provides the BeanShell scripting languages—and none of the others. Also, there is no syntax highlighting in loadIT.

7.2.5 Execution

In addition to the features that are already included in JMeter (see Section 7.1), loadIT is able to retrieve dynamic request parameters from a central database, which distributes them to the load agents. Also, there are some advanced features for the deployment and synchronization of load agents that are not part of our evaluation criteria. Distributed execution is also possible and very easy to set up.

7.2.6 Monitoring

loadIT shows most of its results already during the test execution. It is integrated with DynaTrace and NovaTec's inspectIT to get profiling data.

7.2.7 Organization, Presentation and Export of Test Results

Like JMeter, loadIT does not have an integrated report management, but it is able to store results to a database. Also, it provides more detailed and structured tables. The final results contain request and response times, appropriate utilization concepts within the same graph and performance values represented by the PerfMon-Profiler.

7.2.8 Maintainability

All of JMeter's response assertion features are available in loadIT, too, except for the missing scripting languages. Additionally, there is a feature to automatically compare the actual responses during test execution with recorded responses. This feature is not automatically active during test execution and it will not cause the tests to fail. Instead, it can be called manually after a test execution to compare the results. The comparison works quite well in the default configuration and can be adjusted manually using a configuration file, so that values that are known to change (like session IDs) will not be recognized as errors. A small drawback is the integrated diff viewer, which fails to display large results (longer than 300 lines) and truncates them, so that the conflict is sometimes not visible. Nevertheless, the feature can be used in a reliable way once it has been adjusted to avoid false positives.

Mass-edit is not present in loadIT, either. The file format is similar to JMeter and allows for manual search-and-replace as well as versioning. Modularization is possible in the same way, but loadIT additionally supports the modularization by grouping recorded requests automatically.

7.2.9 Extensibility

loadIt inherits the plug-in interface of JMeter (see Section 7.1.8).

7.2.10 Support

loadIT has a wiki which cant be joined by users. For help, there is a mailing list (of JMeter) and you are able to contact the manufacturers directly via e-mail or telephone.

7.2.11 Summary

By extending JMeter, loadIT inherits all of JMeters strong points, but is not always able to eliminate its drawbacks. The major improvements of JMeter include the handling of the JSF View State ID, dynamic request parameters, the itegration with DynaTrace and inspectIT, and the improved modularization.

7.3 NeoLoad

The manufacturing company of NeoLoad is Neotys and the license of this load testing tool is proprietary. The test version is 4.2.2. This version runs on Windows, Linux, Solaris, IBM AIX, HP-UX, and is also executable by VMware.

7.3.1 Download and Installation

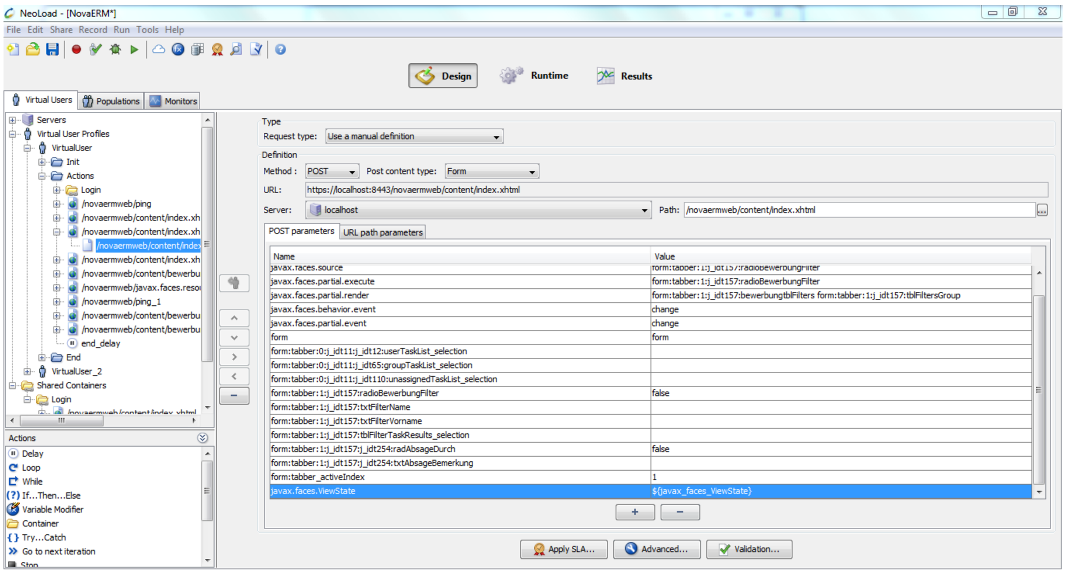
NeLoad requires a license that can be obtained by registration. Users also have the opportunity to download a 30 days trial-version and NeoLoad has a build-in update function. NeoLoad is not open source.

7.3.2 User Interface

The user interface of NeoLoad has a clear and compact structure and the tool functions are all very user-friendly and comprehensible. NeoLoad guides the user step-by-step through the test process and automatically opens tabs where configuration is needed. Another point to mention is the undo-function.

7.3.3 Record and Play and Further Editing

NeoLoad allows recording requests through either a proxy server or a tunnel. The proxy server works with all major browsers and NeoLoad automatically configures and opens them. When using HTTPS, a certificate has to be installed manually. NeoLoad also provides automatic JSF support, dynamic parameters, and support of the WebSocket protocol. All WebSocket connections are automatically recorded along the HTTP requests and listed in



Type

Request type: Use a manual definition

Definition

Method : POST Post content type: Form

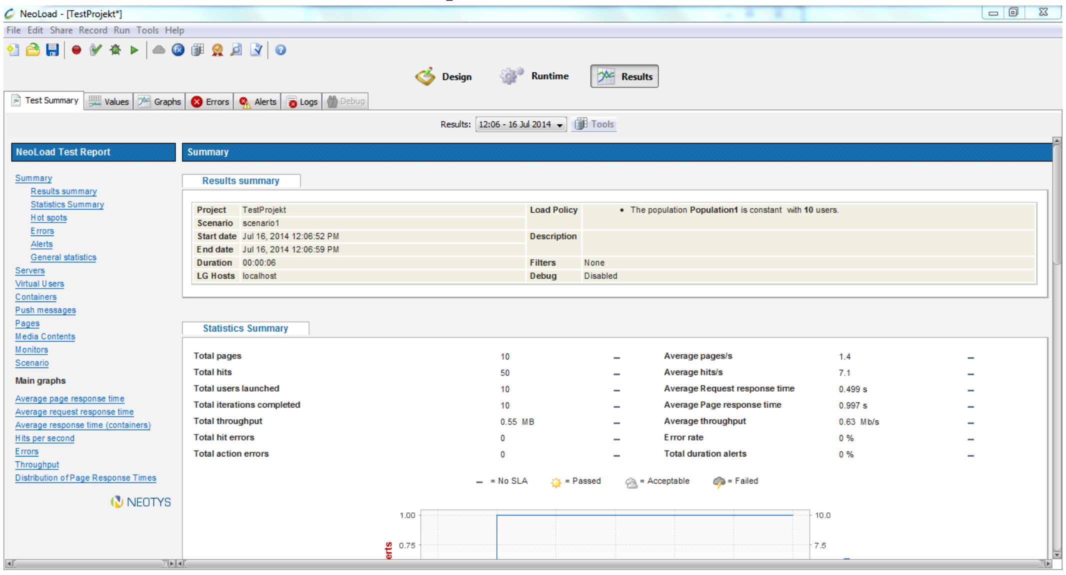
URL: https://localhost:8443/novaermweb/content/index.xhtml

Server: localhost Path: /novaermweb/content/index.xhtml

POST parameters | URL path parameters

Name	Value
javax.faces.source	form:tabber:1:j_idt157:radioBewerbungFilter
javax.faces.partial.execute	form:tabber:1:j_idt157:radioBewerbungFilter
javax.faces.partial.render	form:tabber:1:j_idt157:bewerbungtblFilters form:tabber:1:j_idt157:tblFiltersGroup
javax.faces.behavior.event	change
javax.faces.partial.event	change
form	form
form:tabber:0:j_idt11:j_idt12:userTaskList_selection	
form:tabber:0:j_idt11:j_idt65:groupTaskList_selection	
form:tabber:0:j_idt11:j_idt110:unassignedTaskList_selection	
form:tabber:1:j_idt157:radioBewerbungFilter	false
form:tabber:1:j_idt157:txtFilterName	
form:tabber:1:j_idt157:txtFilterVorname	
form:tabber:1:j_idt157:tblFilterTaskResults_selection	
form:tabber:1:j_idt157:j_idt254:radAbsageDurch	false
form:tabber:1:j_idt157:j_idt254:txtAbsageBemerkung	
form:tabber_activeIndex	1
javax.faces.ViewState	${javax_faces_ViewState}

(a) Request view in NeoLoad



Results: 12:06 - 16 Jul 2014 Tools

Summary

Results summary

Project	TestProjekt	**Load Policy**	• The population **Population1** is constant with **10** users.
Scenario	scenario1		
Start date	Jul 16, 2014 12:06:52 PM	**Description**	
End date	Jul 16, 2014 12:06:59 PM		
Duration	00:00:06	**Filters**	None
LG Hosts	localhost	**Debug**	Disabled

Statistics Summary

Total pages	10	–	**Average pages/s**	1.4	–
Total hits	50	–	**Average hits/s**	7.1	–
Total users launched	10	–	**Average Request response time**	0.499 s	–
Total iterations completed	10	–	**Average Page response time**	0.997 s	–
Total throughput	0.55 MB	–	**Average throughput**	0.63 Mb/s	–
Total hit errors	0	–	**Error rate**	0 %	–
Total action errors	0	–	**Total duration alerts**	0 %	–

– = No SLA ☀ = Passed ☁ = Acceptable ⛈ = Failed

(b) Results summary in NeoLoad

Figure 7.3. NeoLoad's UI

the UI. Sent messages are listed as single entries while recieved messages are listed in the view of the connection request.

Recorded requests are not grouped automatically, but they can be grouped manually after the recording is completed. Reordering and deleting recorded requests as well as creating entirely new ones is easily possible. But unfortunately, it is not able to export the configured action groups as a script.

During the recording NeoLoad takes screenshots of the pages and shows them with the requests which makes it easier to orient oneself in the list of requests.

7.3.4 Scripting

NeoLoad allows the user to insert JavaScript sections and also provides documentation. The editor has syntax-highlighting but unfortunately no auto completion.

7.3.5 Execution

NeoLoad offers a pre-test execution to validate the workload specification and the test environment. Various elements, like loops, branches, and delays, make it possible to create advanced workload specifications.

When configuring the load intensity, the user can specify a constant load intensity, a ramp-up, peaks, a custom function, and combinations of these.

Setting up a distributed load generation environment is very simple and convenient. It is customizable and has an implemented abort function.

7.3.6 Monitoring

NeoLoad shows most of its results already during the test execution. It can also import external data or use profiling tools to collect additional data itself.

7.3.7 Organization, Presentation and Export of Test Results

The data of each test execution is stored in a separate data set. NeoLoad also has a feature to compare the result of different executions.

Results are presented using graphs and tables. The design is very clear and graphs always scale correctly. Also, the extent of represented information is satisfying.

Most of the graphs can be exported as images: RTF, PDF, HTML, and XML. The raw data can be exported in CSV format. All exports can be triggered individually or in groups. The final results contain request and response times, appropriate utilization concepts, and performance values.

7.3.8 Maintainability

To detect outdated load test specifications, the various response assertions in NeoLoad can be used. However, these are mostly designed to detect server failures. In particular, there is no feature to compare responses for similarity to a recorded response. With some manual work, it is possible to create reasonable assertions, and NeoLoad offers several ways to do so. There are assertions for duration, length and title of the response, simple (string-based) matchers.

When the application has changed, it is rather difficult to adjust the tests in NeoLoad. While reordering and editing of single requests is easily possible, there is no feature to edit multiple requests at once.

The modularization is provided with so-called ”Shader Containers” that allow for such modules (e.g., Login) to be used in multiple workload specifications.

7.3.9 Extensibility

NeoLoad is in no way extensible.

7.3.10 Support

NeoLoad has a manual which is accessible online and another one integrated in the load testing tool itself. Many hints, tips, and tutorials in video and text format can be found on the web or in the official forum. For additional help, Neotys is providing support contact by mail and a phone hotline.

7.3.11 Summary

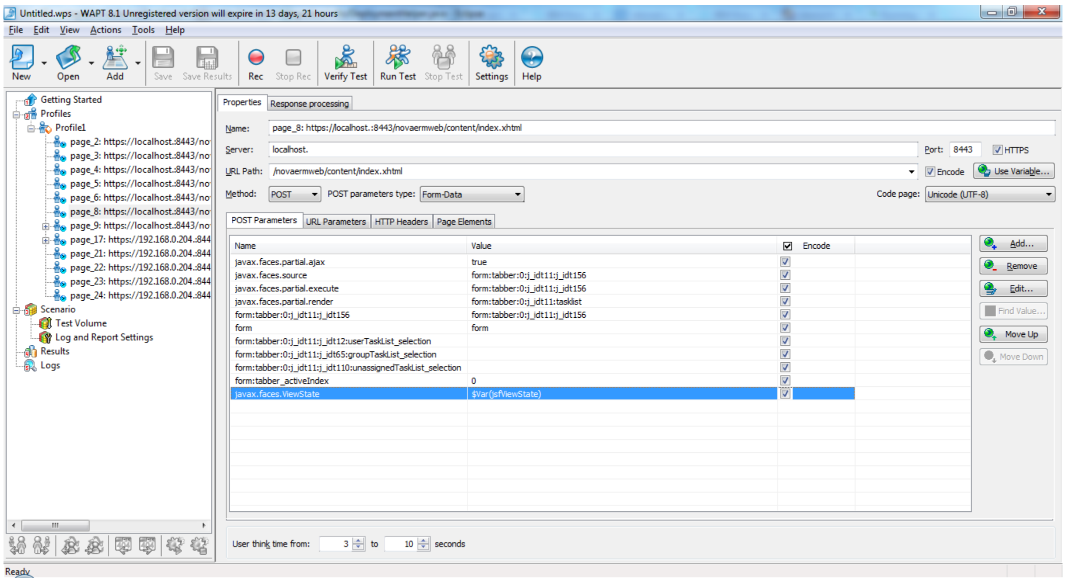
NeoLoad is a very easy-to-use tool that has nearly every feature we wished a good load testing tool to have, including automatic JSF ViewState handling. The features we missed were automatic grouping of request, automatic comparision of respones, and mass-edit of requests.

7.4 WAPT

The manufacturing company of WAPT is Softlogica and the license of this load testing tool is proprietary. WAPT is available in two editions: the regular for around 700 USD and the Pro Edition with additional functionality for around 1,200 USD. We tested the regular as well as the Pro edition and indicate the features only available to Pro users.

7.4.1 Download and Installation

The tested version of WAPT requires no registration. The manufacturing company of WAPT also provides a demo version of their load testing tool and users have the opportunity to choose a selective tool installation. WAPT is not open source.

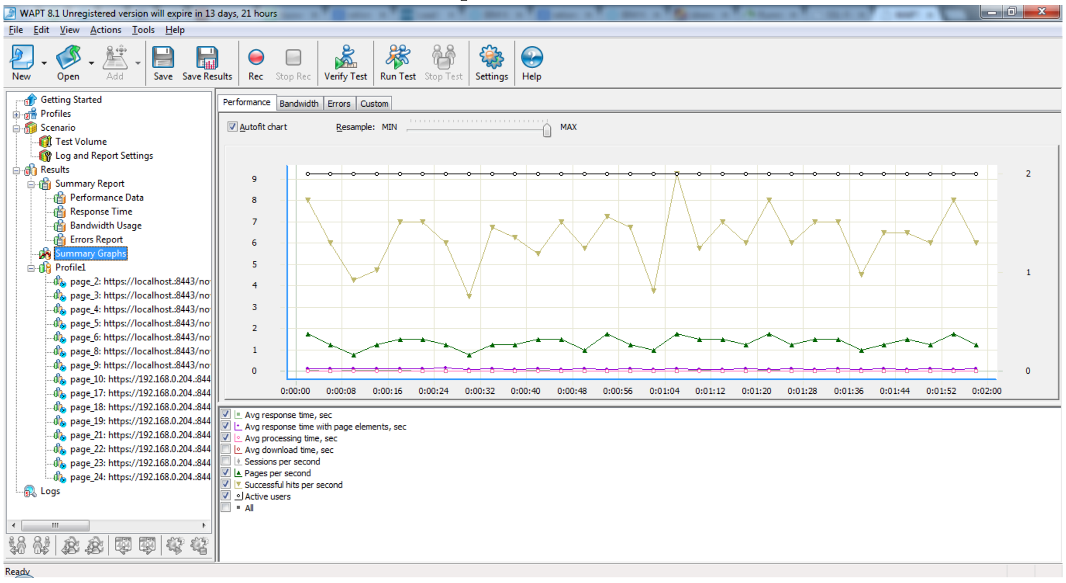

Untitled.wps - WAPT 8.1 Unregistered version will expire in 13 days, 21 hours
File Edit View Actions Tools Help
New
Open
Add
Save
Save Results
Rec
Stop Rec
Verify Test
Run Test
Stop Test
Settings
Help
Getting Started
Profiles
Profile1
page_2: https://localhost.:8443/no
page_3: https://localhost.:8443/no
page_4: https://localhost.:8443/no
page_5: https://localhost.:8443/no
page_6: https://localhost.:8443/no
page_8: https://localhost.:8443/no
page_9: https://localhost.:8443/no
page_17: https://192.168.0.204.:844
page_21: https://192.168.0.204.:844
page_22: https://192.168.0.204.:844
page_23: https://192.168.0.204.:844
page_24: https://192.168.0.204.:844
Scenario
Test Volume
Log and Report Settings
Results
Logs
Properties
Response processing
POST Parameters
URL Parameters
HTTP Headers
Page Elements
Add...
Remove
Edit...
Find Value...
Move Up
Move Down
Ready

Name: page_8: https://localhost.:8443/novaermweb/content/index.xhtml
Server: localhost. Port: 8443 [✓] HTTPS
URL Path: /novaermweb/content/index.xhtml [✓] Encode
Method: POST POST parameters type: Form-Data Code page: Unicode (UTF-8)

Name	Value	Encode
javax.faces.partial.ajax	true	✓
javax.faces.source	form:tabber:0:j_idt11:j_idt156	✓
javax.faces.partial.execute	form:tabber:0:j_idt11:j_idt156	✓
javax.faces.partial.render	form:tabber:0:j_idt11:tasklist	✓
form:tabber:0:j_idt11:j_idt156	form:tabber:0:j_idt11:j_idt156	✓
form	form	✓
form:tabber:0:j_idt11:j_idt12:userTaskList_selection		✓
form:tabber:0:j_idt11:j_idt65:groupTaskList_selection		✓
form:tabber:0:j_idt11:j_idt110:unassignedTaskList_selection		✓
form:tabber_activeIndex	0	✓
javax.faces.ViewState	$Var(jsfViewState)	✓

User think time from: 3 to 10 seconds

(a) Request view in WAPT


WAPT 8.1 Unregistered version will expire in 13 days, 21 hours
File Edit View Actions Tools Help
New
Open
Add
Save
Save Results
Rec
Stop Rec
Verify Test
Run Test
Stop Test
Settings
Help
Getting Started
Profiles
Scenario
Test Volume
Log and Report Settings
Results
Summary Report
Performance Data
Response Time
Bandwidth Usage
Errors Report
Summary Graphs
Profile1
page_2: https://localhost.:8443/no
page_3: https://localhost.:8443/no
page_4: https://localhost.:8443/no
page_5: https://localhost.:8443/no
page_6: https://localhost.:8443/no
page_8: https://localhost.:8443/no
page_9: https://localhost.:8443/no
page_10: https://192.168.0.204.:844
page_17: https://192.168.0.204.:844
page_18: https://192.168.0.204.:844
page_19: https://192.168.0.204.:844
page_21: https://192.168.0.204.:844
page_22: https://192.168.0.204.:844
page_23: https://192.168.0.204.:844
page_24: https://192.168.0.204.:844
Logs
Performance
Bandwidth
Errors
Custom
Resample: MIN MAX
9
8
7
6
5
4
3
2
1
0
2
1
0
0:00:00
0:00:08
0:00:16
0:00:24
0:00:32
0:00:40
0:00:48
0:00:56
0:01:04
0:01:12
0:01:20
0:01:28
0:01:36
0:01:44
0:01:52
0:02:00
Ready

[✓] Autofit chart

- [✓] Avg response time, sec
- [✓] Avg response time with page elements, sec
- [✓] Avg processing time, sec
- [ ] Avg download time, sec
- [ ] Sessions per second
- [✓] Pages per second
- [✓] Successful hits per second
- [✓] Active users
- [ ] All

(b) Summary graph in WAPT

Figure 7.4. The UI of WAPT

7.4.2 User Interface

The user interface of WAPT has a clear and compact structure and the tool's functions are all very user-friendly and comprehensible. But unfortunately, it has no undo-function.

7.4.3 Record and Play and Further Editing

WAPT allows recording requests through a proxy server. Configuring it for SSL/HTTPS is complicated. While the proxy server works with all major browsers, WAPT configures and opens them automatically. WAPT even has an integrated browser, which is very cumbersome to use though. Also, the HTTPS certificate needs to be installed manually, without any help of instructions, because there are no official ones. WAPT supports JSF, which has to be activated manually, but does not provide any introductions again. Configurable WebSockets are also not provided in WAPT.

Recorded requests can be grouped automatically or manually after the recording is completed. But the manual method is very laborious and the automatic one is too obscure to understand it immediately. Renaming, reordering, and deleting recorded requests as well as creating entirely new ones is possible.

7.4.4 Scripting

There is only a scripting language for the Pro version and also a very scarce documentation for the user. The editor features syntax-highlighting but no auto completion.

7.4.5 Execution

WAPT offers a pre-test execution to validate the workload specification and the test environment. It is possible to combine different test parts, also by using loops and branches.

WAPT also offers a specific feature for running a set of requests with varying parameters (better in PRO) and constant load generation only with ramp-up, peaks, and periodic variants (only rectangular). It is not possible to specify custom load intensity functions.

Setting up a distributed load generation environment is simple, but only possible with the Pro version. WAPT has an abort function and distributed execution is also possible and very easy to set up.

WAPT offers (basic) response assertion. It is able to detect failed requests by checking the response text for (the absence of) certain matches (”contains”). There is no feature to compare the responses during execution with those recorded when the workload specification was created. Furthermore, it is possible to detect responses that exceed a time limit.

7.4.6 Monitoring

WAPT shows most of its results already during the test execution and is able to use profiling tools to collect additional data; only with the Pro version though. External data cannot be imported.

7.4.7 Organization, Presentation and Export of Test Results

WAPT does not have a dedicated result management interface; results are stored a special files, which can be organized in the file system only.

Results are presented using graphs and tables. The design is very clear and graphs always scale correctly. But unfortunately the extent of the represented information is limited.

The overall report can be exported in HTML format; individual graphs can be exported as images (PNG). The raw data can be exported in CSV format, but only in the Pro version. The final results contain request and response times and appropriate performance values.

7.4.8 Maintainability

To detect outdated load test specifications, the various response assertions in WAPT can be used. Unfortunately the response assertions in WAPT have to be configured manually and that is a very strict and inflexible way to do so. Only the response time and the presence and absence of a string can be validated. However, these are mostly designed to detect server failure. In particular, there is no feature to compare responses for similarity to a recorded response. The editing of several actions is not possible: the search functions works successfully in all areas, but not the replace function. Modular testing is also not executable and in fact there does not exist a versionable file format.

7.4.9 Extensibility

WAPT is in no way extensible.

7.4.10 Support

WAPT has a manual which is accessible online as well as one integrated in the load testing tool itself. Many hints, tips, and tutorials in video and text format can be found on the web. For additional support, there is both a community forum and customer support via e-mail.

7.4.11 Summary

The functionality of WAPT is limited in nearly every category making it useless for load test with very special requirements. Also the lack of features that support maintainability hinder the use for sustainable load testing in large projects.

7.5 RPT (Rational Performance Tester)

The manufacturing company of RPT is IBM and the license of this load testing tool is proprietary. The price for the tool itself is around 2,253.86 EUR (as per July 2014). The test version of RPT used in this elaboration can be executed on Windows, Linux, and AIX systems.

7.5.1 Download and Installation

The tested version of RPT requires a license that can be obtained by registration. The manufacturing company of RPT provides a demo version of their load testing tool. Another beneficial point is that this load testing tool has an integrated update function. IBM does not provide source code for the tool.

7.5.2 User Interface

As usual with IBM products, the user interface is based on Eclipse RCP. Therefore, the structure of the RPT user interface is quite intuitive for regular Eclipse users, but may be confusing for other load testers, since it does not exactly follow the process of load testing. As part of the Eclipse integration, RPT offers the typical search and undo functions and organizes its data into projects.

7.5.3 Record and Play and Further Editing

RPT allows recording requests through a proxy server. This proxy server works with all major browsers and RPT configures and opens them automatically. The HTTPS certificate does not need to be installed manually.

Recorded requests can be grouped automatically or manually after the recording is completed. Renaming is only possible for action groups; reordering, and deleting recorded requests as well as creating entirely new ones is easily possible.

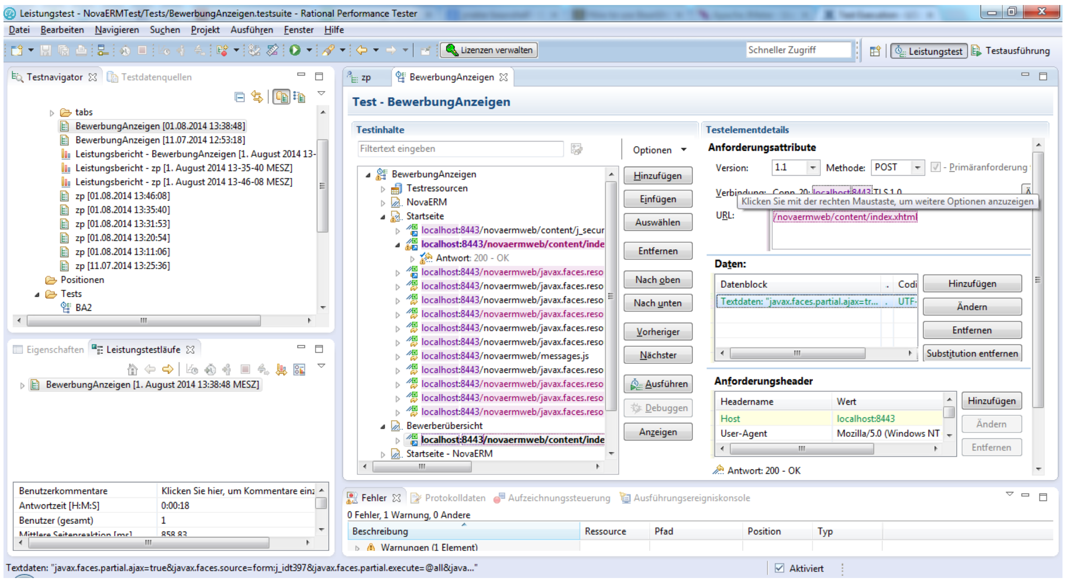
RPT automatically correlates response and request data, including the JSF ViewState.

7.5.4 Scripting

RPT uses Java as its scripting language. There are good syntax-highlighting and auto completion functions available (default Eclipse functionality).

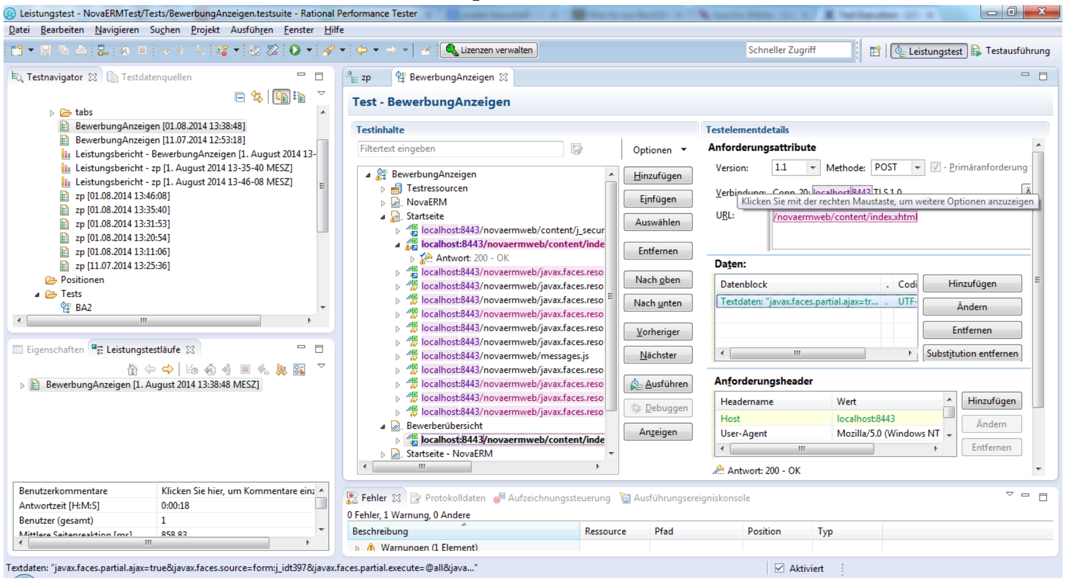
7.5.5 Execution

RPT offers a pre-test execution to validate the workload specification and the test environment. It is possible to combine different test parts, also by using loops and branches.



```
Leistungstest - NovaERMTest/Tests/BewerbungAnzeigen.testsuite - Rational Performance Tester
Datei  Bearbeiten  Navigieren  Suchen  Projekt  Ausführen  Fenster  Hilfe
Lizenzen verwalten          Schneller Zugriff          Leistungstest  Testausführung

Testnavigator  Testdatenquellen
  tabs
  BewerbungAnzeigen [01.08.2014 13:38:48]
  BewerbungAnzeigen [11.07.2014 12:53:18]
  Leistungsbericht - BewerbungAnzeigen [1. August 2014 13-
  Leistungsbericht - zp [1. August 2014 13-35-40 MESZ]
  Leistungsbericht - zp [1. August 2014 13-46-08 MESZ]
  zp [01.08.2014 13:46:08]
  zp [01.08.2014 13:35:40]
  zp [01.08.2014 13:31:53]
  zp [01.08.2014 13:20:54]
  zp [01.08.2014 13:11:06]
  zp [11.07.2014 13:25:36]
 Positionen
 Tests
  BA2

Eigenschaften  Leistungstestläufe
 BewerbungAnzeigen [1. August 2014 13:38:48 MESZ]

Benutzerkommentare              Klicken Sie hier, um Kommentare einz
Antwortzeit [H:M:S]             0:00:18
Benutzer (gesamt)               1
Mittlere Seitenreaktion [ms]    858.83

zp  BewerbungAnzeigen
Test - BewerbungAnzeigen
Testinhalte
Filtertext eingeben                                     Optionen
 BewerbungAnzeigen                                      Hinzufügen
  Testressourcen                                        Einfügen
  NovaERM                                               Auswählen
  Startseite
   localhost:8443/novaermweb/content/j_secur            Entfernen
   localhost:8443/novaermweb/content/inde
    Antwort: 200 - OK                                   Nach oben
   localhost:8443/novaermweb/javax.faces.reso           Nach unten
   localhost:8443/novaermweb/javax.faces.reso
   localhost:8443/novaermweb/javax.faces.reso           Vorheriger
   localhost:8443/novaermweb/javax.faces.reso           Nächster
   localhost:8443/novaermweb/javax.faces.reso
   localhost:8443/novaermweb/javax.faces.reso           Ausführen
   localhost:8443/novaermweb/messages.js                Debuggen
   localhost:8443/novaermweb/javax.faces.reso
   localhost:8443/novaermweb/javax.faces.reso           Anzeigen
   localhost:8443/novaermweb/javax.faces.reso
   localhost:8443/novaermweb/javax.faces.reso
  Bewerberübersicht
   localhost:8443/novaermweb/content/inde
  Startseite - NovaERM

Testelementdetails
Anforderungsattribute
Version: 1.1   Methode: POST   - Primäranforderung
Verbindung: Conn. 20: localhost:8443 TLS 1.0
Klicken Sie mit der rechten Maustaste, um weitere Optionen anzuzeigen
URL: /novaermweb/content/index.xhtml
Daten:
Datenblock                                  Codi      Hinzufügen
Textdaten: "javax.faces.partial.ajax=tr...  UTF-      Ändern
                                                      Entfernen
                                                      Substitution entfernen
Anforderungsheader
Headername     Wert                                   Hinzufügen
Host           localhost:8443                         Ändern
User-Agent     Mozilla/5.0 (Windows NT                Entfernen
Antwort: 200 - OK

Fehler  Protokolldaten  Aufzeichnungssteuerung  Ausführungsereigniskonsole
0 Fehler, 1 Warnung, 0 Andere
Beschreibung              Ressource   Pfad   Position   Typ
 Warnungen (1 Element)

Textdaten: "javax.faces.partial.ajax=true&javax.faces.source=form:j_idt397&javax.faces.partial.execute=@all&java..."    Aktiviert
```

(a) Request view in RPT



```
Leistungstest - NovaERMTest/Tests/BewerbungAnzeigen.testsuite - Rational Performance Tester
Datei  Bearbeiten  Navigieren  Suchen  Projekt  Ausführen  Fenster  Hilfe
Lizenzen verwalten          Schneller Zugriff          Leistungstest  Testausführung

Testnavigator  Testdatenquellen
  tabs
  BewerbungAnzeigen [01.08.2014 13:38:48]
  BewerbungAnzeigen [11.07.2014 12:53:18]
  Leistungsbericht - BewerbungAnzeigen [1. August 2014 13-
  Leistungsbericht - zp [1. August 2014 13-35-40 MESZ]
  Leistungsbericht - zp [1. August 2014 13-46-08 MESZ]
  zp [01.08.2014 13:46:08]
  zp [01.08.2014 13:35:40]
  zp [01.08.2014 13:31:53]
  zp [01.08.2014 13:20:54]
  zp [01.08.2014 13:11:06]
  zp [11.07.2014 13:25:36]
 Positionen
 Tests
  BA2

Eigenschaften  Leistungstestläufe
 BewerbungAnzeigen [1. August 2014 13:38:48 MESZ]

Benutzerkommentare              Klicken Sie hier, um Kommentare einz
Antwortzeit [H:M:S]             0:00:18
Benutzer (gesamt)               1
Mittlere Seitenreaktion [ms]    858.83

zp  BewerbungAnzeigen
Test - BewerbungAnzeigen
Testinhalte
Filtertext eingeben                                     Optionen
 BewerbungAnzeigen                                      Hinzufügen
  Testressourcen                                        Einfügen
  NovaERM                                               Auswählen
  Startseite
   localhost:8443/novaermweb/content/j_secur            Entfernen
   localhost:8443/novaermweb/content/inde
    Antwort: 200 - OK                                   Nach oben
   localhost:8443/novaermweb/javax.faces.reso           Nach unten
   localhost:8443/novaermweb/javax.faces.reso
   localhost:8443/novaermweb/javax.faces.reso           Vorheriger
   localhost:8443/novaermweb/javax.faces.reso           Nächster
   localhost:8443/novaermweb/javax.faces.reso
   localhost:8443/novaermweb/javax.faces.reso           Ausführen
   localhost:8443/novaermweb/messages.js                Debuggen
   localhost:8443/novaermweb/javax.faces.reso
   localhost:8443/novaermweb/javax.faces.reso           Anzeigen
   localhost:8443/novaermweb/javax.faces.reso
   localhost:8443/novaermweb/javax.faces.reso
  Bewerberübersicht
   localhost:8443/novaermweb/content/inde
  Startseite - NovaERM

Testelementdetails
Anforderungsattribute
Version: 1.1   Methode: POST   - Primäranforderung
Verbindung: Conn. 20: localhost:8443 TLS 1.0
Klicken Sie mit der rechten Maustaste, um weitere Optionen anzuzeigen
URL: /novaermweb/content/index.xhtml
Daten:
Datenblock                                  Codi      Hinzufügen
Textdaten: "javax.faces.partial.ajax=tr...  UTF-      Ändern
                                                      Entfernen
                                                      Substitution entfernen
Anforderungsheader
Headername     Wert                                   Hinzufügen
Host           localhost:8443                         Ändern
User-Agent     Mozilla/5.0 (Windows NT                Entfernen
Antwort: 200 - OK

Fehler  Protokolldaten  Aufzeichnungssteuerung  Ausführungsereigniskonsole
0 Fehler, 1 Warnung, 0 Andere
Beschreibung              Ressource   Pfad   Position   Typ
 Warnungen (1 Element)

Textdaten: "javax.faces.partial.ajax=true&javax.faces.source=form:j_idt397&javax.faces.partial.execute=@all&java..."    Aktiviert
```

(b) RPT's UI during execution

Figure 7.5. The UI of RPT

RPT also offers an excellent feature for running a set of requests with varying parameters.

It provides constant load generation and peaks with ramp-up, but there is no periodic load intensity available.

The load generation of RPT is customizable and has an abort function. Distributed execution is also possible and very easy to set up.

RPT offers response assertion. In particular, it is able to detect failed requests by checking the response text for (the absence of) certain matches. There is also a reliable feature to compare the responses during execution with those recorded when the workload specification was created.

7.5.6 Monitoring

RPT does not show the results during the test execution. But it is able to import external data, by IMB Tivoli, or use profiling tools to collect additional data.

7.5.7 Organization, Presentation and Export of Test Results

RPT has a result management. Results are presented using graphs and tables. The design is not one of the beneficial points of this load testing tool, but the graphs always scale correctly. Also, the extent of the represented information is satisfying.

Reports can be exported in HTML or RTF format. The raw data can be exported in CSV format. All exports can be triggered individually or in groups. The final results contain request and response times and appropriate performance values.

7.5.8 Maintainability

Load test specifications can be executed once locally to check if they still work. RPT automatically checks for typical error codes/messages and also lists ”potential correlation errors”, which can be used to detect relevant application changes. There are, however, a few false positives and they cannot be set to ignore manually (unlike with other errors in Eclipse).

Inserting manual response assertions is also possible using checks for response time, size, title, or content matching. The latter can be done very conveniently by highlighting the respective passage in the recorded response and adding it to the response check.

Editing request data individually is cumbersome because RPT does not split up the HTTP GET parameters, but shows all of them as a single-line text block (in GET query format). Mass-editing is still possible and can be done very easily, because RPT offers a good global search-and-replace function that can be applied to very specific parts of the test.

Modular test design is part of RPTs design concept. Using compound tests, the user can assemble complex load test specifications from individual recordings using loops and branches if required. The clearly structured tree view makes it easy to do so. All parts of a load test design are stored as individual files and organized in folders within the Eclipse project. This causes many files to be created. Unfortunately, the RPT specific file formats including the most important file format (`.testsuite`) are not based on plain text files, so that they cannot be versioned. As a solution, IBM provides CVS integration from the IDE, which has not been evaluated in this study.

7.5.9 Extensibility

Besides HTTP other protocols can be included through plug-ins. RPT can also be extended though Java that can be used to perform various tasks like setting up the test environment and gathering additional data during the execution.

7.5.10 Support

RPT has a manual which is accessible online and another one integrated in the load testing tool itself. Many hints, tips, and tutorials in video and integrated text format can be found on the web. RPT also has a moderated forum in which users are able to interchange. For additional support, there is a customer support via e-mail and telephone available. Although there is a wiki for RPT hosted on the IBM website, it is mostly empty.

7.5.11 Summary

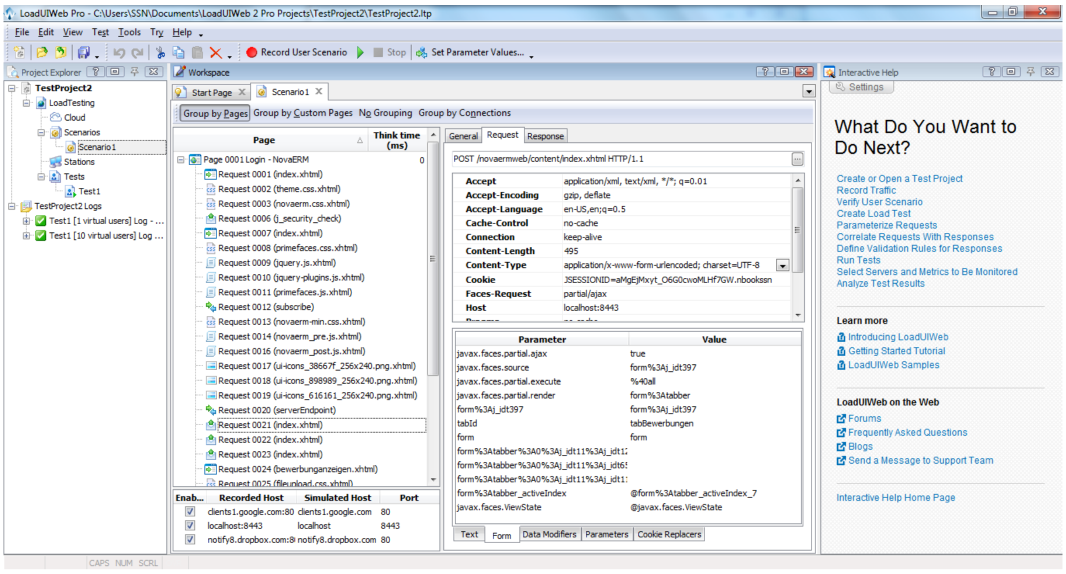
RPT has all features needed for common load testing. A plus is the automatic correlation of data that can be a big time saver when creating the workload specification. The fact that RPT is based on Eclipse RCP makes it a good choice for users that are already familiar with Eclipse or other IBM products.

7.6 LoadUIWeb Pro

The manufacturing company of LoadUIWeb Pro is SmartBear and the license of this load testing tool is proprietary. The price for the tool itself is around 6,439 EUR. The test version of LoadUIWeb Pro used in this elaboration can be ran on Windows/Linux/AIX.

7.6.1 Download and Installation

LoadUIWeb Pro requires a license that can be obtained by registration. Users also have the opportunity to download a 14-days trial version and LoadUIWeb Pro has a build-in update function. Since LoadUIWeb Pro is not open-source, there is no source code delivered with the executable.



Page	Think time (ms)
Page 0001 Login - NovaERM	0
Request 0001 (index.xhtml)	
Request 0002 (theme.css.xhtml)	
Request 0003 (novaerm.css.xhtml)	
Request 0006 (j_security_check)	
Request 0007 (index.xhtml)	
Request 0008 (primefaces.css.xhtml)	
Request 0009 (jquery.js.xhtml)	
Request 0010 (jquery-plugins.js.xhtml)	
Request 0011 (primefaces.js.xhtml)	
Request 0012 (subscribe)	
Request 0013 (novaerm-min.css.xhtml)	
Request 0014 (novaerm_pre.js.xhtml)	
Request 0016 (novaerm_post.js.xhtml)	
Request 0017 (ui-icons_38667f_256x240.png.xhtml)	
Request 0018 (ui-icons_898989_256x240.png.xhtml)	
Request 0019 (ui-icons_616161_256x240.png.xhtml)	
Request 0020 (serverEndpoint)	
Request 0021 (index.xhtml)	
Request 0022 (index.xhtml)	
Request 0023 (index.xhtml)	
Request 0024 (bewerbunganzeigen.xhtml)	
Request 0025 (fileupload.css.xhtml)	

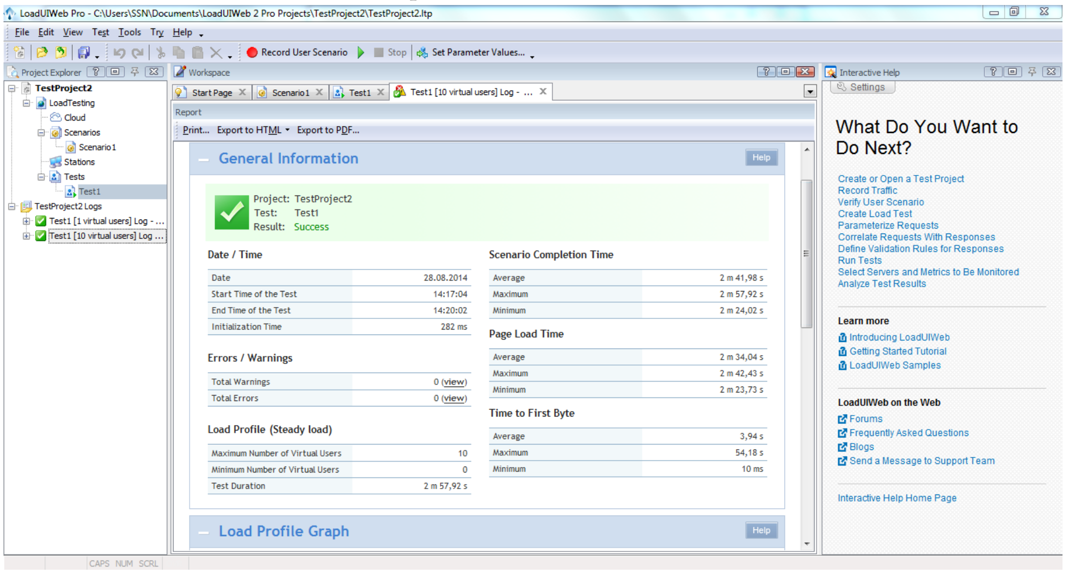
Enab...	Recorded Host	Simulated Host	Port
☑	clients1.google.com:80	clients1.google.com	80
☑	localhost:8443	localhost	8443
☑	notify8.dropbox.com:8	notify8.dropbox.com	80

POST /novaermweb/content/index.xhtml HTTP/1.1

Accept	application/xml, text/xml, */*; q=0.01
Accept-Encoding	gzip, deflate
Accept-Language	en-US,en;q=0.5
Cache-Control	no-cache
Connection	keep-alive
Content-Length	495
Content-Type	application/x-www-form-urlencoded; charset=UTF-8
Cookie	JSESSIONID=aMgEjMxyt_O6G0cwoMLHf7GW.nbookssn
Faces-Request	partial/ajax
Host	localhost:8443

Parameter	Value
javax.faces.partial.ajax	true
javax.faces.source	form%3Aj_idt397
javax.faces.partial.execute	%40all
javax.faces.partial.render	form%3Atabber
form%3Aj_idt397	form%3Aj_idt397
tabId	tabBewerbungen
form	form
form%3Atabber%3A0%3Aj_idt11%3Aj_idt1	
form%3Atabber%3A0%3Aj_idt11%3Aj_idt6	
form%3Atabber%3A0%3Aj_idt11%3Aj_idt1	
form%3Atabber_activeIndex	@form%3Atabber_activeIndex_7
javax.faces.ViewState	@javax.faces.ViewState

(a) Request view in LoadUIWeb Pro



LoadUIWeb Pro - C:\Users\SSN\Documents\LoadUIWeb 2 Pro Projects\TestProject2\TestProject2.ltp
File Edit View Test Tools Try Help
Record User Scenario
Stop
Set Parameter Values...
Project Explorer
TestProject2
LoadTesting
Cloud
Scenarios
Scenario1
Stations
Tests
Test1
TestProject2 Logs
Test1 [1 virtual users] Log - ...
Test1 [10 virtual users] Log ...
Workspace
Start Page
Scenario1
Test1
Test1 [10 virtual users] Log - ...
Report
Print...
Export to HTML
Export to PDF...
Help
Interactive Help
Settings
What Do You Want to Do Next?
Create or Open a Test Project
Record Traffic
Verify User Scenario
Create Load Test
Parameterize Requests
Correlate Requests With Responses
Define Validation Rules for Responses
Run Tests
Select Servers and Metrics to Be Monitored
Analyze Test Results
Learn more
Introducing LoadUIWeb
Getting Started Tutorial
LoadUIWeb Samples
LoadUIWeb on the Web
Forums
Frequently Asked Questions
Blogs
Send a Message to Support Team
Interactive Help Home Page
CAPS NUM SCRL

General Information

Project: TestProject2
Test: Test1
Result: Success

Date / Time

Date	28.08.2014
Start Time of the Test	14:17:04
End Time of the Test	14:20:02
Initialization Time	282 ms

Errors / Warnings

Total Warnings	0 (view)
Total Errors	0 (view)

Load Profile (Steady load)

Maximum Number of Virtual Users	10
Minimum Number of Virtual Users	0
Test Duration	2 m 57,92 s

Scenario Completion Time

Average	2 m 41,98 s
Maximum	2 m 57,92 s
Minimum	2 m 24,02 s

Page Load Time

Average	2 m 34,04 s
Maximum	2 m 42,43 s
Minimum	2 m 23,73 s

Time to First Byte

Average	3,94 s
Maximum	54,18 s
Minimum	10 ms

Load Profile Graph

(b) Result summary in LoadUIWeb Pro

Figure 7.6. The UI of LoadUIWeb Pro

7.6.2 User Interface

The user interface of LoadUIWeb Pro has a clear and compact structure and the tool functions are all very user-friendly and comprehensible. The menu lists Undo/Redo actions, but they do not work at all.

7.6.3 Record and Play and Further Editing

LoadUIWeb Pro allows recording requests through a proxy server. This proxy server works with all major browsers and LoadUIWeb Pro configures and opens them automatically. The HTTPS certificate does not need to be installed manually. JSF is also supported but has to be activated manually. LoadUIWeb Pro does not have an integrated browser and does not provide any configurable WebSockets.

Recorded requests can be grouped automatically or manually after the recording is completed. But the automatic method has a very bad detection. Renaming, reordering, and deleting recorded requests as well as creating entirely new ones is possible. Unfortunately, it is not possible to copy requests or deactivate a request for testing purposes.

7.6.4 Scripting

LoadUIWeb Pro has no scripting elements.

7.6.5 Execution

LoadUIWeb Pro offers a pre-test execution to validate the workload specification and the test environment. It is possible to combine different test parts with so-called complex scenarios. These complex scenarios support static loops but no branching.

LoadUIWeb Pro also offers a feature for running a set of requests with varying parameters. It provides constant load generation and ramp-ups, but there is no periodic load intensity available. There is also no way to run response-checks for error messages.

The load generation of LoadUIWeb Pro is customizable and has an abort function. Distributed execution is also possible and very easy to set up.

7.6.6 Monitoring

LoadUIWeb Pro shows most of its results already during the test execution. But it is not able to import external data or use profiling tools to collect additional data.

7.6.7 Organization, Presentation and Export of Test Results

LoadUIWeb Pro has a result management. Results are presented by using graphs. The design gives a favourable impression and the graphs always scale correctly. Also, the extent of the represented information is satisfying.

Reports can be exported in HTML or PDF format. All exports can be triggered individually or in groups. The final results contain request and response times and appropriate utilization concepts. Unfortunately, the raw data cannot be exported in a structured format like CSV.

7.6.8 Maintainability

To detect outdated load test specifications, the various response assertions in LoadUIWeb Pro can be used. However, these are mostly designed to detect server failures. In particular, there is no feature to compare responses for similarity to a recorded response. The validation of responses is possible, but offers only string-matching.

When the application has changed, it is rather difficult to adjust the tests in LoadUIWeb Pro. While reordering and editing of single requests is easily possible, there is no feature to modify multiple requests at once. Therefore, the most efficient solution to, e.g., react to a changed JSF ID is to open the file in a text editor and use the search-and-replace function there.

The fact that LoadUIWeb Pro uses an XML based file format makes the latter quite easy. Aside from being easily editable from other tools, the XML format is also well suited for versioning.

Modularization is possible in LoadUIWeb Pro with so-called ”Complex scenarios” which is a list of other scenarios to execute.

7.6.9 Extensibility

LoadUIWeb Pro is in no way extensible.

7.6.10 Support

LoadUIWeb Pro has a manual which is accessible online and another one integrated in the load testing tool itself. Many hints, tips, and tutorials in video and text format can be found on the web. LoadUIWeb Pro also has a moderated forum in which users are able to interchange. For additional support, there is a customer support via e-mail and telephone available. During the evaluation we found no wiki or mailing list.

7.6.11 Summary

LoadUIWeb Pro has all basic features needed by a load testing tool, but lacks functionality to customize the workload specification and results in an extensive manner.

7.7 LoadRunner

The manufacturing company of LoadRunner is HP and the license of this load testing tool is proprietary. The price for the tool itself is around 0.56-2.00 USD per virtual user and day. The test version of LoadRunner used in this elaboration can be ran on Windows XP and all other types of Windows Servers.

7.7.1 Download and Installation

The tested version of LoadRunner requires a license that could be obtained by registration. The manufacturing company of LoadRunner is also providing a demo version of their load testing tool and users have the opportunity to choose for a selective tool installation. But unfortunately there is no source for this tool available.

7.7.2 User Interface

LoadRunner is split up into multiple applications for designing, running and analysing load tests (besides some more applications). These applications have three different designs and user interface principles, which is confusing. The user guidance is quite satisfying but shows a deficit at some points. A beneficial criterion is that LoadRunner has an undo-function.

7.7.3 Record and Play and Further Editing

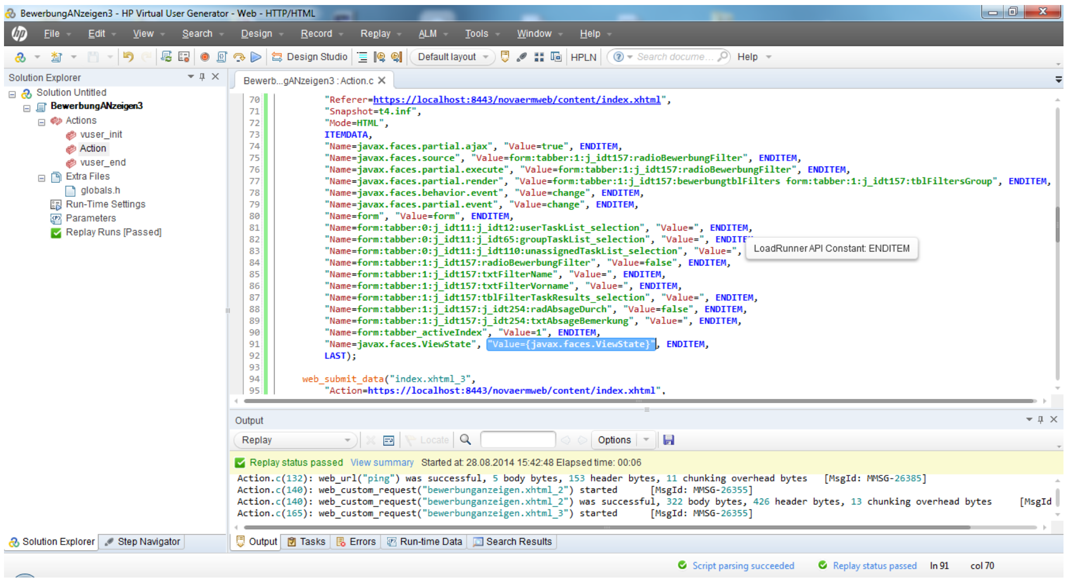
LoadRunner allows recording requests through a proxy server. This proxy server works with all major browsers and LoadRunner configures and opens them automatically. Also the HTTPS certificate does not need to be installed manually. LoadRunner supports JSF but is only able to operate with one View State ID automatically.

The recorded requests are organized as code and therefore it is inherently possible to reorder and delete requests as well as creating new ones. Unfortunately, the generated code is barely commented and the variable naming is not always clear.

LoadRunner supports the WebSockets protocol quite remarkably. All connections and sent messages are recorded and included in the script. Additional callback functions for the OnOpen, OnMessage, OnError and OnClose allow the implementation of client logic and/or logging.

7.7.4 Scripting

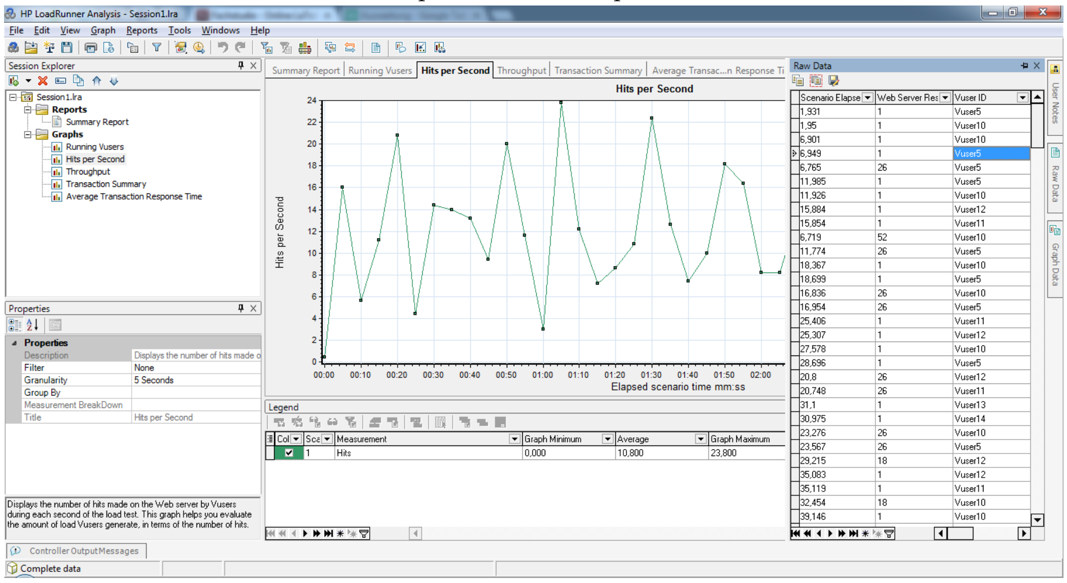
The languages available are C, Visual Basic, and Java. The corresponding documentation is very comprehensible. Beside that the editor also features syntax-highlighting and an auto-complete function.



```
        "Referer=https://localhost:8443/novaermweb/content/index.xhtml",
        "Snapshot=t4.inf",
        "Mode=HTML",
        ITEMDATA,
        "Name=javax.faces.partial.ajax", "Value=true", ENDITEM,
        "Name=javax.faces.source", "Value=form:tabber:1:j_idt157:radioBewerbungFilter", ENDITEM,
        "Name=javax.faces.partial.execute", "Value=form:tabber:1:j_idt157:radioBewerbungFilter", ENDITEM,
        "Name=javax.faces.partial.render", "Value=form:tabber:1:j_idt157:bewerbungtblFilters form:tabber:1:j_idt157:tblFiltersGroup", ENDITEM,
        "Name=javax.faces.behavior.event", "Value=change", ENDITEM,
        "Name=javax.faces.partial.event", "Value=change", ENDITEM,
        "Name=form", "Value=form", ENDITEM,
        "Name=form:tabber:0:j_idt11:j_idt12:userTaskList_selection", "Value=", ENDITEM,
        "Name=form:tabber:0:j_idt11:j_idt65:groupTaskList_selection", "Value=", ENDITE
        "Name=form:tabber:0:j_idt11:j_idt110:unassignedTaskList_selection", "Value=",
        "Name=form:tabber:1:j_idt157:radioBewerbungFilter", "Value=false", ENDITEM,
        "Name=form:tabber:1:j_idt157:txtFilterName", "Value=", ENDITEM,
        "Name=form:tabber:1:j_idt157:txtFilterVorname", "Value=", ENDITEM,
        "Name=form:tabber:1:j_idt157:tblFilterTaskResults_selection", "Value=", ENDITEM,
        "Name=form:tabber:1:j_idt157:j_idt254:radAbsageDurch", "Value=false", ENDITEM,
        "Name=form:tabber:1:j_idt157:j_idt254:txtAbsageBemerkung", "Value=", ENDITEM,
        "Name=form:tabber_activeIndex", "Value=1", ENDITEM,
        "Name=javax.faces.ViewState", "Value={javax.faces.ViewState}", ENDITEM,
        LAST);

    web_submit_data("index.xhtml_3",
        "Action=https://localhost:8443/novaermweb/content/index.xhtml",
```

```
Replay status passed   View summary   Started at: 28.08.2014 15:42:48 Elapsed time: 00:06
Action.c(132): web_url("ping") was successful, 5 body bytes, 153 header bytes, 11 chunking overhead bytes   [MsgId: MMSG-26385]
Action.c(140): web_custom_request("bewerbunganzeigen.xhtml_2") started      [MsgId: MMSG-26355]
Action.c(140): web_custom_request("bewerbunganzeigen.xhtml_2") was successful, 322 body bytes, 426 header bytes, 13 chunking overhead bytes     [MsgId
Action.c(165): web_custom_request("bewerbunganzeigen.xhtml_3") started      [MsgId: MMSG-26355]
```

(a) Workload specification as script in LoadRunner



Properties	
Description	Displays the number of hits made o
Filter	None
Granularity	5 Seconds
Group By	
Measurement BreakDown	
Title	Hits per Second

Displays the number of hits made on the Web server by Vusers during each second of the load test. This graph helps you evaluate the amount of load Vusers generate, in terms of the number of hits.

Col	Sca	Measurement	Graph Minimum	Average	Graph Maximum
✓	1	Hits	0,000	10,800	23,800

Scenario Elapse	Web Server Res	Vuser ID
1,931	1	Vuser5
1,95	1	Vuser10
6,901	1	Vuser10
6,949	1	Vuser5
6,765	26	Vuser5
11,985	1	Vuser5
11,926	1	Vuser10
15,884	1	Vuser12
15,854	1	Vuser11
6,719	52	Vuser10
11,774	26	Vuser5
18,367	1	Vuser10
18,699	1	Vuser5
16,836	26	Vuser10
16,954	26	Vuser5
25,406	1	Vuser11
25,307	1	Vuser12
27,578	1	Vuser10
28,696	1	Vuser5
20,8	26	Vuser12
20,748	26	Vuser11
31,1	1	Vuser13
30,975	1	Vuser14
23,276	26	Vuser10
23,567	26	Vuser5
29,215	18	Vuser12
35,083	1	Vuser12
35,119	1	Vuser11
32,454	18	Vuser10
39,146	1	Vuser10

(b) Results view in LoadRunner

Figure 7.7. The UI of LoadRunner

7.7.5 Execution

LoadRunner offers a pre-test execution to validate the workload specification and the test environment. Because the requests are organized in code, it is very easy to implement loop, branches, and parametrization.

Concerning workload intensity, LoadRunner features constant load with ramp-up and custom functions.

Distributed execution is also possible and very easy to set up.

7.7.6 Monitoring

LoadRunner shows most of its results already during the test execution and uses profiling tools to collect additional data. But unfortunately it is not able to import external data.

7.7.7 Organization, Presentation and Export of Test Results

Results are presented by using graphs and tables. While the user interface looks very crowded, the graphs always scale correctly and a wide range of information is available.

LoadRunner can export to a wide variety of file format, including HTML, PDF, some image formats and also CSV and XLS as structured formats.

7.7.8 Maintainability

The fact that the requests are organized as code, makes the workload specifications easy to maintain. Modularization is achieved by simply extracting the requests into a new function. Also code is very easy-to-use with versioning software. The editing of multiple requests at once can be done with the text-replace function.

To detect outdated workload specifications LoadRunner allows for response validation and the comparison of responses to the recorded ones only during the pre-test execution, which is adequate for the purpose of maintainability.

7.7.9 Extensibility

LoadRunner is designed on top of an extensibility framework called Addin Tree. This allows everyone to fully customize the tool with plug-ins.

7.7.10 Support

LoadRunner has a manual which is accessible online and another one integrated in the load testing tool itself. Many hints, tips, and tutorials in video and text format can be found on the web. LoadRunner also has a moderated forum in which users are able to interchange. For additional support, there is a customer support via e-mail and telephone available and an exclusive hotline for clients. During the evaluation we found no wiki or mailing list.

7.7.11 Summary

LoadRunner organizes the workload specification as code, which inherently adds many features like response validation and modularization. Apart from that, LoadRunner has a wide functionality that allows it to be used for nearly every use case.

7.8 Silk Performer

The manufacturing company of Silk Performer is Borland and the license of this load testing tool is proprietary. The test version of Silk Performer used in this elaboration can only be ran on Windows XP.

Unfortunately we were not able to fully evaluate Silk Performer, because the load generator was not able to communicate with our test application. Hence we only show our results as far as we have come in the evaluation procedure.

7.8.1 Download and Installation

The tested version of Silk Performer requires a license that could be obtained by registration. Borland is also providing a demo version of their load testing tool and users have the opportunity to choose for a selective tool installation. Another beneficial point is that this load testing tool has an integrated update function. But unfortunately there is no source for this tool available.

7.8.2 User Interface

The user interface is quite extensive but still ease-to-use. The user is guided with wizards and Silk Performer also features an undo-function.

7.8.3 Record and Play and Further Editing

Silk Perfomer allows recording requests through a proxy server. This proxy server works with all major browsers and Silk Performer configures and opens them automatically. Silk Performer also features the automatic handling of session ID's, including the JSF ViewState.

The recorded requests are organized as code and therefore it is inherently possible to reorder and delete requests as well as creating new ones. Unfortunately, the generated code is barely commented and the variable naming is not always clear.

7.8.4 Support

Silk Performer has a manual which is accessible online and another one integrated in the load testing tool itself. Many hints, tips, and tutorials in video and text format can be found on the web, plus it has a wiki, too. Silk Performer also has a moderated forum in which users are able to interchange. For additional support, there is a customer support via e-mail. During the evaluation we found no mailing list.

7.9 Summary

This section gives a short summary of the evaluation results in the form of tables. The entries evaluation result either on a binary scale (y/n) or on an ordinal scale (-/0/+) defined in the criteria catalog (see Chapter 5).

Documentation and Support

	JMeter	loadIT	NeoLoad	WAPT	RPT	LoadUIWeb	LoadRunner	Silk Performer
User Guide	y	y	y	y	y	y	y	y
Wiki	y	y	n	n	y	n	n	y
Tutorial: Text	y	n	y	y	y	y	y	y
Tutorial: Video	y	n	y	y	y	y	y	y
Integrated Manual	y	n	y	y	y	y	y	y
Code Documentation	n	omit.	omit.	omit.	omit.	omit.	omit.	omit.
E-Mail / Contact Form	n	y	y	y	y	y	y	y
Telephone	n	y	y	n	y	y	y	n
Customer-Hotline	n	n	n	n	n	n	y	n
Community Forum	y	n	y	y	y	y	y	y
Mailing List	y	y	n	n	n	n	n	n
Moderated Forum	n	n	y	n	y	y	y	y

Usability and Extensibility

	JMeter	loadIT	NeoLoad	WAPT	RPT	LoadUIWeb	LoadRunner	Silk Performer
Simplicity	-	0	+	+	0	0	-	0
User Guidance	-	-	+	+	0	0	0	+
Undo-Function	n	n	y	n	y	n	y	y
Protocols	y	y	n	n	y	n	y	
Reporting formats	y	y	n	n	n	n	y	

Record and Play - Supported Technologies

	JMeter	loadIT	NeoLoad	WAPT	RPT	LoadUIWeb Pro	LoadRunner	Silkperformer
HTTPS	y	y	y	y	y	y	y	y
WebSockets	y	y	y	n	n	n	y	
Proxy-Method	y	y	y	y	y	y	y	y
Automatic Proxy-Configuration	n	n	y	y	y	y	y	y
Firefox-Support	y	y	y	y	y	y	y	y
Chrome-Support	y	y	y	y	y	y	y	y
IE-Support	y	y	y	y	y	y	y	y
Integrated Browser	n	n	n	y	n	n	n	n
Automatic Opening	n	y	y	y	y	y	y	y
JSF-Support	-	+	+	-	+	+	0	+

Record and Play - Further Editing

	JMeter	loadIT	NeoLoad	WAPT	RPT	LoadUIWeb Pro	LoadRunner	Silkperformer
Grouping automatic	n	y	n	y	y	y	n	n
Grouping manual	y	y	y	y	y	y	y	y
Renaming	y	y	y	y	n	y	y	y
Output as Script	n	n	n	n	n	n	y	y
Script-Commentation	omit.	omit.	omit.	omit.	omit.	omit.	-	-
Script Readability	omit.	omit.	omit.	omit.	omit.	omit.	0	-
Output in the UI	y	y	y	y	y	y	n	n
Reorder Requests	y	y	y	y	y	y	y	y
Deactivation of Requests	y	y	y	n	y	n	y	y
Copy/Paste	y	y	y	y	y	y	y	y
Parameter-Modification	y	y	y	y	y	y	y	y

Scripting

	JMeter	loadIT	NeoLoad	WAPT	RPT	LoadUIWeb Pro	LoadRunner	Silk Performer
Documentation	-	-	+	0	+	omit.	+	0
Autocomplete	n	n	n	n	y	omit.	y	n
Syntax-Highlighting	y	n	y	y	y	omit.	y	y

Execution

	JMeter	loadIT	NeoLoad	WAPT	RPT	LoadUIWeb	LoadRunner	Silk Performer
Composition	y	y	y	y	y	y	y	
Loops and Branching	y	y	y	y	y	n	y	
Parametrisation	-	0	+	+	+	+	+	
Pre-Test Execution	n	n	y	y	y	y	y	
Constant Distribution	y	y	y	y	y	y	y	
Peaks	n	n	y	y	y	n	n	
Periodic	n	n	y	y	n	n	n	
User-Defined	y	y	y	n	y	y	y	
Ramp-Up	y	y	y	y	y	y	y	
Distributed Execution	+	+	+	+	+	+	+	
Test Abortion	y	y	y	y	y	y	y	

Maintainability

	JMeter	loadIT	NeoLoad	WAPT	RPT	LoadUIWeb	LoadRunner	Silk Performer
Response-Validation	y	y	y	y	y	n	y	
Automatic Response-Comparison	n	y	n	n	y	n	y	
Group-Editing	n	n	n	n	y	n	n	
Modular Test Design	y	y	y	n	y	y	y	
Manageable file format	y	y	y	n	n	y	y	
Overall	0	+	0	-	0	0	+	

Results

	JMeter	loadIT	NeoLoad	WAPT	RPT	LoadUIWeb	LoadRunner	Silk Performer
Input Data and Test Environment	n	n	n	n	n	n	n	
Content: Request/Response Times	y	y	y	y	y	y	y	
Content: Throughput	y	y	y	y	y	n	y	
Content: Utilization	y	y	y	n	n	y	n	
Report Format	0	0	+	+	+	-	+	
Selective Export	y	y	y	y	y	n	y	
Profiling	n	y	y	y	y	n	n	
Import of extern Measurement Data	n	y	y	n	y	n	y	
Results during the Execution	y	y	y	y	n	y	y	
Organization of serveral Data Sets	n	n	y	n	y	y	n	
Comparison of Test Results	n	n	y	n	y	n	y	

7.10 Conclusion

In this study, we selected eight load testing tools and evaluated them using the criteria catalog we created beforehand. We now give our opinion on which tools to consider for a given use case.

At first sight, all of the load testing tools evaluated in this study are quite similar and have clearly influenced each other: They all offer a recording-based approach to generate workload specifications in the form of a list of requests, let the user correct, refine and parametrise the generated requests. But beyond that, the tools differ significantly in the number and quality of their features, as well as the ideas and concepts they are based on.

Some users may need to use a particular tool, because it is the only one that offers integration with their system under test or the rest of their testing environment. For other users, who are not restricted to a single tool, we provide a guideline to facilitate their decision.

For users with little or no experience in load testing, who just want to try and find out the costs and benefits of load testing, we do *not* recommend to use one of the open-source tools, only because they are free and small in download size. Instead, the trial versions of the commercial products should also be considered, which are also free. The decision for a suitable load testing tool should be made beforehand, possibly with the hints below. NeoLoad and WAPT provide a user-friendly interface and help getting started.

Probably one of the first decisions to be made is the decision between a script-based and a UI-form-based tool, because it has a great impact on how the user works with the tool and how changes, parametrisation, versioning, etc. need to be done. LoadRunner and Silk Performer are the only two tools in this evaluation with code-based workload specifications.

If the load tests are to be employed in a production environment on a day-to-day basis, we recommend looking at loadIT and LoadRunner, because they offer the best features to help maintain workload specifications.

Furthermore, the offered features of the tool should be considered. While no user will need all of the features, we recommend to make sure that the desired features are available in the preferred tool using the tables above, especially for WAPT and LoadUIWeb Pro, since they are far from being feature-complete. Choosing a tool with many features that are not needed for the particular use case may on the other hand result in confusion and slower working speed. This might be the case with IBM's Rational Performance Tester, which offers most features.

Last but not least, one has to decide between an open-source and a commercial tool. This is especially important for more complicated or unusual kinds of load tests, since they might require changes to or internal knowledge of the tool. While the commercial vendors offer support for their products (which is often included) and can therefore help with special cases, the free tools have many special cases already built-in or available through plug-ins, but they may be hard to find and understand. Being the only truly open-source tool in the evaluation, JMeter can also be adjusted by the tester to fit the requirements. This is probably how loadIT was founded, but unfortunately loadIT is not open-source, so that it cannot be adjusted as easily.

For good WebSocket support, we recommend LoadRunner, because of its ability to simulate client-side logic, which is needed to test complex WebSocket communication.

The above thoughts can be useful to decide for or against a particular load testing tool, but we explicitly point out that only a small selection of the available load testing tools have been evaluated, so that we recommend looking at other similar tools before making a decision.

The purpose of this study was to find a good load testing tool for the test application NovaERM. In our opinion, NeoLoad would be best suited. It comes with all required features (especially excellent JSF support) and is not overfilled with functionality, which is not needed. NeoLoad's simple and clear user interface is also an important point, because NovaERM is developed mostly by working students. Hence, the staff is changing on a regular basis. Also, if NovaERM will use WebSockets in the future, NeoLoad already has basic support.

Bibliography

[Abbors et al. 2012] F. Abbors, T. Ahmad, D. Truscan, and I. Porres. Mbpet – a model-based performance testing tool. In A. Alimohammad and P. Dini, editors, *4th International Conference on Advances in System Testing and Validation Lifecycle*, page 1–8. IARIA, 2012.

[Apache Foundation 2014] Apache Foundation. Apache JMeter, 2014. URL `http://jakarta.apache.org/jmeter/`.

[Cheung and Lee 2005] C. Cheung and M. Lee. The asymmetric effect of website attribute performance on satisfaction: An empirical study. In *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 175c–175c, Jan 2005.

[Feitelson 2002] D. G. Feitelson. Workload modeling for performance evaluation. In *Performance Evaluation of Complex Systems: Techniques and Tools*, pages 114–141. Springer Verlag, 2002.

[Hoorn et al. 2008] A. V. Hoorn, M. Rohr, and W. Hasselbring. Generating probabilistic and intensity-varying workload for web-based software systems. In *Performance Evaluation – Metrics, Models and Benchmarks: Proceedings of the SPEC International Performance Evaluation Workshop (SIPEW '08), volume 5119 of Lecture Notes in Computer Science (LNCS*, pages 124–143. SPEC, Springer, 2008.

[NovaTec Solutions GmbH 2014] NovaTec Solutions GmbH. loadIT, 2014. URL `http://www.loadit.de/home/`.

[P. Haberl 2012] K. V. M. W. P. Haberl, A. Spillner. Survey 2011: Software test in practice, 2012. Translation of Umfrage 2011: Softwaretest in der Praxis, dpunkt.verlag.

[Roy et al. 2013] S. Roy, T. Begin, and P. Gonçalves. A Complete Framework for Modelling and Generating Workload Volatility of a VoD System. In *IEEE Int. Wireless Communications and Mobile Computing Conference*, pages 1168 – 1174, Calgari, Italy, 2013.

[Schroeder et al. 2006] B. Schroeder, A. Wierman, and M. Harchol-Balter. Open versus closed: A cautionary tale. In *Proceedings of the 3rd Conference on Networked Systems Design & Implementation - Volume 3*, NSDI'06, pages 18–18, Berkeley, CA, USA, 2006. USENIX Association.

[Subraya and Subrahmanya 2000] B. M. Subraya and S. V. Subrahmanya. Object driven performance testing in web applications. In *APAQS*, pages 17–28, 2000.

[v. Kistowski et al. 2014] J. v. Kistowski, N. R. Herbst, and S. Kounev. Modeling variations in load intensity over time. In *Proceedings of the Third International Workshop on Large Scale Testing*, LT '14, pages 1–4, New York, NY, USA, 2014. ACM.

Declaration

We declare that this thesis is the solely effort of the authors. We did not use any other sources and references than the listed ones. We have marked all contained direct or indirect statements from other sources as such. Neither this work nor significant parts of it were part of another review process. We did not publish this work partially or completely yet. The electronic copy is consistent with all submitted copies.

Stuttgart, 2014/10/02