Accurate workload design for web performance evaluation

RAÚL PEÑA ORTIZ

EDITORIAL
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENT D'INFORMÀTICA DE SISTEMES I COMPUTADORS

Accurate workload design for web performance evaluation

Thesis submitted in partial fulfillment of the requirements
for the degree of Ph.D in the subject of Computer Science

Presented by:

Raúl Peña-Ortiz

Supervised by:

Dr. Ana Pont Sanjuán
Dr. José Antonio Gil Salinas
Dr. Julio Sahuquillo Borrás

Valencia, Spain. January, 2013

This editorial is member of the UNE, which guarantees the diffusion and commercialization of its publications at national and international level.

ISBN: 978-84-9048-025-0 (printed version)
Re. editorial: 5613

"Education is the passport to the future, for tomorrow belongs to those who prepare for it today."

Malcolm X.

"If today were the last day of my life, would I want to do what I am about to do today?

The only way to do great work is to love what you do.

Don't settle.

You have to trust that the dots will somehow connect in your future."

Steve Jobs,
conference at Stanford University.

"But we're talking about an objective doctoral thesis."

Marathon Man, the film.

"Stay Hungry. Stay Foolish."

The Whole Earth Catalog,
back cover of final issue.

Acknowledgments

These pages are dedicated to those who have given me their unconditional support in tackling this challenge, especially to my advisors, colleagues, friends and dear family.

Abstract

The new web-based applications and services, which are becoming more and more popular every day, have completely changed the way users interact with the Web. In less than half a decade the role of users has changed from passive consumers of information to active and dynamic contributors to the contents offered. Moreover, this trend is expected to rise in the incoming Web.

This user's behavior is a major concern when defining web workloads in order to precisely estimate system performance for the current Web. However, the intrinsic difficulty to represent the user's dynamism in a workload model has led many research works to still use workloads non representative of the current web navigations.

This dissertation focuses on characterizing and reproducing more realistic workload for web performance by mimicking the behavior of the real web users.

The state-of-the-art in modeling and generating workloads for web performance studies presents several lacks in models and software that represent the different levels of user's dynamism. This fact motivates us to propose a more accurate workload model and to develop a new workload generator based on this model. Both of them have been validated against a traditional workload generation approach. To this end, a new testbed with the ability of reproducing traditional and dynamic workloads has been developed by integrating the proposed generator with a commonly used benchmark.

In this Ph.D dissertation we also analyze and measure for the first time, to the best of our knowledge, the impact of using representative dynamic user workloads on web performance metrics instead of traditional workloads. Experimental results demonstrate that the use of an accurate workload model that considers user's dynamism when navigating the Web clearly affects system performance metrics as well as the stress borderline of the server.

Finally, we explore the effect of considering the User-Browser Interaction as a part of user's dynamic behavior on web workload characterization. The study proves that representing user's dynamic interactions with the provided contents allows users to achieve their navigation goals sooner thus increasing the productivity of their navigations. In addition results demonstrate that this type of navigations also affects the stress borderline of the server and system resources utilization.

Resumen

Las nuevas aplicaciones y servicios web, cada vez más populares en nuestro día a día, han cambiado completamente la forma en la que los usuarios interactúan con la Web. En menos de media década, el papel que juegan los usuarios ha evolucionado de meros consumidores pasivos de información a activos colaboradores en la creación de contenidos dinámicos, típicos de la Web actual. Y, además, esta tendencia se espera que aumente y se consolide con el paso del tiempo.

Este comportamiento dinámico de los usuarios es una de las principales claves en la definición de cargas de trabajo adecuadas para estimar con precisión el rendimiento de los sistemas web. No obstante, la dificultad intrínseca a la caracterización del dinamismo del usuario y su aplicación en un modelo de carga, propicia que muchos trabajos de investigación sigan todavía empleando cargas no representativas de las navegaciones web actuales.

Esta tesis doctoral se centra en la caracterización y reproducción, para estudios de evaluación de prestaciones, de un tipo de carga web más realista, capaz de imitar el comportamiento de los usuarios de la Web actual.

El estado del arte en el modelado y generación de cargas para los estudios de prestaciones de la Web presenta varias carencias en relación a modelos y aplicaciones software que representen los diferentes niveles de dinamismo del usuario. Este hecho nos motiva a proponer un modelo más preciso y a desarrollar un nuevo generador de carga basado en este nuevo modelo. Ambas propuestas han sido validadas en relación a una aproximación tradicional de generación de carga web. Con este fin, se ha desarrollado un nuevo entorno de experimentación con la capacidad de reproducir cargas web tradicionales y dinámicas, mediante la integración del generador propuesto con un benchmark de uso común.

En esta tesis doctoral también se analiza y evalúa por primera vez, según nuestro saber y entender, el impacto que tiene el empleo de cargas de trabajo dinámicas en las métricas de rendimiento de los sistemas web, con respecto al uso de cargas tradicionales. Los resultados experimentales demuestran que usar modelos de carga más precisos, en los que se considera el comportamiento dinámico de los usuarios cuando navegan por la Web, afecta claramente a las métricas de rendimiento de los sistemas, así como a su frontera de estrés.

Finalmente en este trabajo se explora el efecto de considerar la interacción del usuario con el navegador web como parte de su comportamiento dinámico en la caracterización de la carga. El estudio muestra un aumento de la productividad de las navegaciones del usuario cuando se considera su interacción con los contenidos web a través de las facilidades que ofrecen los navegadores. Los resultados demuestran que los usuarios alcanzan antes sus objetivos generando nuevos patrones de navegación, que a su vez afectan al rendimiento de los sistemas web, tanto a sus fronteras de estrés como a la utilización de sus recursos.

Resum

Les noves aplicacions i serveis web, cada vegada més populars en el nostre dia a dia, han canviat completament la forma en què els usuaris interactuen amb la Web. En menys de mitja dècada, el paper que juguen els usuaris ha evolucionat de mers consumidors passius d'informació a actius col·laboradors en la creació de continguts dinàmics, típics de la Web actual. I, a més, aquesta tendència s'espera que augmente i es consolide amb el pas del temps.

Aquest comportament dinàmic dels usuaris és una de les principals claus en la definició de càrregues de treball adequades per a estimar amb precisió el rendiment dels sistemes web. No obstant això, la dificultat intrínseca a la caracterització del dinamisme de l'usuari i la seua aplicació en un model de càrrega, propícia que molts treballs d'investigació seguisquen encara emprant càrregues no representatives de les navegacions web actuals.

Aquesta tesi doctoral se centra en la caracterització i reproducció, per a estudis d'avaluació de prestacions, d'un tipus de càrrega web més realista, capaç d'imitar el comportament dels usuaris de la Web actual.

L'estat de l'art en el modelatge i generació de càrregues per als estudis de prestacions de la Web presenta diverses carències en relació a models i aplicacions programari que representen els diferents nivells de dinamisme de l'usuari. Aquest fet ens motiva a proposar un model més precís i a desenvolupar un nou generador de càrrega basat en aquest nou model. Ambdós propostes han sigut validades en relació a una aproximació tradicional de generació de càrrega web. Amb aquest fi, s'ha desenvolupat un nou entorn d'experimentació amb la capacitat de reproduir càrregues web tradicionals i dinàmiques, per mitjà de la integració del generador proposat amb un benchmark d'ús comú.

En aquesta tesi doctoral també s'analitza i avalua per primera vegada, segons el nostre saber i entendre, l'impacte que té l'ocupació de càrregues de treball dinàmiques en les mètriques de rendiment dels sistemes web, respecte a l'ús de càrregues tradicionals. Els resultats experimentals demostren que usar models de càrrega més precisos, en els que es considera el comportament dinàmic dels usuaris quan naveguen per la Web, afecta clarament les mètriques de rendiment dels sistemes, així com a la seua frontera d'estrés.

Finalment en aquest treball s'explora l'efecte de considerar la interacció de l'usuari amb el navegador web com a part del seu comportament dinàmic en la caracterització de la càrrega. L'estudi mostra un augment de la productivitat de les navegacions de l'usuari quan es considera la seua interacció amb els continguts web a través de les facilitats que oferixen els navegadors. Els resultats demostren que els usuaris aconseguixen abans els seus objectius generant nous patrons de navegació, que al seu torn afecten el rendiment dels sistemes web, tant a les seues fronteres d'estrés com a la utilització dels seus recursos.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

There are few technological success stories as dramatic as that of the Web. Originally designed to share static contents among a small group of researchers, the Web is being used today by many millions of people as a part of their daily routines and social lives. Our society is progressively becoming more densely connected, and the paradigm where users access the Web from a desktop computer is making way for a new paradigm dominated by pervasive mobile devices like smart phones and tablets.

This incessant evolution has been possible thanks to the continuous changes in technology that have introduced new features in the current and incoming Web, both in its applications, users, and infrastructure [Rod09]. For instance, e-commerce systems, on-line social networks, blogs, wikis or service oriented architectures are some examples that manifest how websites are evolving from typical hypermedia information repositories of the First Web Generation (Web 1.0) to hypermedia distributed applications and services representative of the Second Web Generation (Web 2.0). With the emergence of this kind of applications and services, users are no longer passive consumers, but they become participative contributors to the dynamic content accessible on the Web [CK08]. Nowadays, web contents and services are even more dynamic [Ore07], which consequently increases the number of changes over time in user's interactions with the Web [RSD+12]. Therefore, a new user's dynamic behavior can be distinguished. Moreover, this user's behavior is expected to be more relevant and meaningful in the incoming Web, also referred to as Web 3.0 [Hen09] or Future Internet [TGG+09].

As a system that is continuously changing, both in the offered applications and infrastructure, performance evaluation studies are necessary in order to provide sound proposals when designing new web-related systems [BC98], such as web services, web servers, proxies or content distribution policies. As in any performance evaluation process, accurate and representative workload models must be used in order to guarantee the validity of the results. Regarding web systems, the user's dynamic behavior makes difficult the design of accurate web workload representing realistic users' navigations.

In general, there are three main challenges that must be addressed when modeling the user's dynamic behavior on representative workloads:

Challenge I: The dynamism in user's behavior when surfing the Web must be taken into account [BC98]. That is, users' behaviors as they interact with web contents and services have to be characterized, modeling the different aspects that determine users' navigation decisions. For instance, personal preferences, navigation goals, visited resources or connectivity conditions.

Challenge II: The different user's roles when navigating a website must be identified and defined as user's behaviors [WOHM06]. For instance, searcher and surfer roles refer to users who start navigations with a query in a given searcher engine, or navigate the web by following direct hyperlinks, respectively [PP99].

Challenge III: Continuous changes in these user's roles during the same navigation session must be modeled and considered [GBGP10]. That is, changes in users' behaviors over the time have to be characterized.

This thesis focuses on modeling and analyzing representative workloads for performance studies with the aim of accurately estimating systems performance indexes in current and incoming Web. To this end, the three main challenges mentioned above are analyzed and addressed in a progressive way in order to provide a new and more representative web workload for performance evaluation.

1.1 Motivation and main goals

Although web evolution has introduced significant changes on user's behavior, many performance studies still check their approaches with traditional workloads, which are typical of the early Web 1.0 and do not represent current web trends. Three main shortcomings can be observed in the open literature about both commercial products and academic research results:

1. To date, web workload models do not consider user's dynamism in an appropriate and accurate way because they only take partially into account the mentioned challenges.

2. There is a lack of web workload generators that can reproduce representative traffic of Web 2.0 applications and services.

3. The effect of using dynamic workloads on web performance evaluation, instead of traditional workloads, has not been analyzed and measured yet to the best of our knowledge.

These shortcomings define the main objectives of this thesis and encouraged us to propose a new model to cover the main gaps found in current web workload characterization research. Based on this model, in this Ph.D dissertation a new web workload generator and testbed are proposed, validated and used for performance evaluation.

1.2 Contributions of the thesis

This thesis presents three main contributions:

- The first contribution is the proposal of a new workload model that permits to define accurate workloads for performance evaluation studies in current and incoming Web. This model takes into account the three mentioned challenges by introducing progressively different levels of dynamism in user's behavior when characterizing web workload.
- Based on this model, a web workload generator has been developed in order to reproduce user's dynamism in web performance studies, which is the second contribution of this thesis.
- Finally, the last main contribution is the analysis of using representative dynamic workload on the web performance metrics. To this end, we provide a new testbed for web performance evaluation by integrating our generator with a commonly used benchmark with the aim of contrasting performance metrics for traditional and dynamic workloads.

1.3 Research context

Part of this thesis has been developed in the context of the research project GENERICA, which was led by the *iSOCO S.L* company in collaboration with the *Web Architecture Research Group (Universitat Politècnica de València)* and the *Institute of Computer Technology*. This project was partially supported by the Spanish Government Grant (FIT 340000-2004-236) and the Regional Valencian Government and IMPIVA Grant (IMIDTD 2004/92, and IMIDTD 2005/15).

The global goal of GENERICA was the development of a methodology to evaluate performance and functionality of web applications typical in Web 2.0, by using a new workload generator with the ability to generate representative dynamic workload.

GENERICA was managed by the author of this Ph.D dissertation, who worked actively in the project results, both software development and several technical reports, and co-authored in some international and national publications.

1.4 Outline

The remainder of this thesis is organized as follows. First, Chapter 2 presents the current state of the art in characterizing and generating workloads for web performance evaluation. It reviews the most relevant perspectives to define web workloads, and classifies a representative subset of software tools proposed in the open literature according to their main features and their ability to generate workloads for the dynamic Web.

After that, Chapter 3 proposes a new web workload model with the aim of characterizing a more realistic workload to evaluate the performance of current web applications. The chapter describes the main concepts of this model and introduces some examples of user's dynamism representation.

Next, Chapter 4 propounds a new web workload generator based on the model. This chapter describes the main features, applications, and architecture of the generator. It also introduces an example of web performance evaluation using this software, that has been appropriately validated against a traditional workload generator. Then, with the aim of providing a more flexible tool to evaluate web performance, a new testbed with the ability of reproducing dynamic user workloads has been developed. Chapter 5 presents both testbed design and validation.

Chapters 6 and 7 analyze for the first time, to the best of our knowledge, the impact of considering user's dynamism on web workload characterization in performance studies. Chapter 6 proves that the web user's dynamic behavior is a crucial point that must be addressed in web performance in order to accurately estimate system performance indexes. On the other hand, Chapter 7 measures the effect of modeling the User-Browser Interaction as a part of user's dynamic behavior on web workload characterization.

Finally, Chapter 8 summarizes the main contributions of this thesis, presents some concluding remarks and the open research lines derived from this dissertation.

CHAPTER 2

Characterizing and generating workload for web performance evaluation

Web workload characterization studies that help us to model and reproduce users' behaviors grow in importance with the massively use of web applications and services [BC98]. Moreover, both types of applications are developed using new technologies that have a strong impact on the system performance. Some previous attempts have been published to reflect this fact. For instance, Cecchet et al. [CMZ02] investigate the effect of different J2EE application deployments on the performance scalability of application servers. Schneider et al. [SAAF08] point out that the use of AJAX and mashups generates more aggressive and bursty network usage compared to the overall HTTP traffic. Similar conclusions but considering server performance are presented in [ONUI09] by Ohara et al. Unfortunately, these studies only consider specific web paradigms, thus the workload used is not representative enough of current users' navigations.

This chapter presents the current state of the art in characterizing and generating workloads for web performance evaluation. First, Section 2.1 reviews a representative subset of the most relevant perspectives to define web workloads, and analyzes the main drawbacks that we have to tackle in order to obtain representative workloads for current web applications. Moreover, we go over different approaches to represent user's behavior on web workload characterization. After that, Section 2.2 evaluates and classifies the most commonly used software tools proposed in the open literature according to their main features and ability to generate workloads for the dynamic Web. Finally, Section 2.3 presents some concluding remarks about this work, which motivate the main contributions of this dissertation.

2.1 Workload models and the current Web

Web workload models are abstractions of the real workload that reproduce users' behaviors and ensure that a particular web application performs as it would do when

working with real users. To this end, the model represents a set of users of the web application and avoids those characteristics that are superfluous for a particular study. Workload models can be classified into two main groups: trace-based and analytical models.

Traces log the sequence of HTTP requests and commands received by a web application during a certain period of time under given conditions. Traces are obtained for a particular environment; that is, specific server speed, network bandwidth, browser cache capacity, etc. This means that if any system parameter varied, the obtained trace would be different. Therefore, the main challenge of trace-based models is to achieve a good representativeness, especially when requests received by different servers exhibit a large variability. Consequently, trace based models are not appropriate to model changes in the user's behavior.

The analytical approach uses mathematical models to simulate the user's behavior or the characteristics of specific workloads. These models allow us to consider different scenarios by setting some input parameters that specify the main characteristics of the workload to be characterized. Analytical models are a flexible approach for modeling changes in the user's behavior.

Some studies [FP01, SCK03] confirm how difficult is to model and generate representative web requests, especially when trying to model the characteristics of the dynamic Web and its users. In this dissertation, the main difficulties when modeling the user's behavior on realistic web workload have been summarized in the three challenges reported on page 1.

There have been few but interesting efforts to model user's behavior in order to obtain more representative users workloads for specific web applications. Menascé et al. [MA00] introduced the Customer Behavior Model Graph (CBMG) that describes patterns of user's behavior in the workloads of e-commerce websites by using an approach based on Finite State Machines (FSMs). The CBMG model consists of all pages of an on-line bookstore and the associated transition probability. For illustrative purposes, Figure 2.1 depicts an example of CBMG for a search process, showing that users may visit several pages and move among these pages according to the arcs weight. Numbers in the arcs indicate the probability of taking that transition. For example, the probability of going to the Product Detail page from the Search Results page is 60%. This value means that after a search, regardless of whether the search returns a list of books or a void list, the Product Detail page will be visited 60% of the times.

Duarte et al. [DMA+08] introduced the Visitor Behavior Model Graph (VBMG) for workload definition of the blogspace by extending CBMG. Blog visitors can be grouped into different categories according to their visiting patterns. These categories are characterized by different VBMGs in terms of the state transition probabilities. For example, Figure 2.2 shows the typical behavior of blurkers, who tend to read a lot of blogs but never post any comments. This behavior only considers that a blurker can *start reading a new blog* or can *continue reading the same blog*. Notice

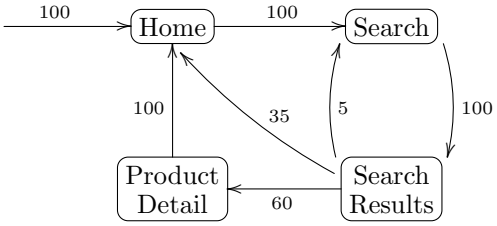


Figure 2.1: Example of a simplified CBMG model

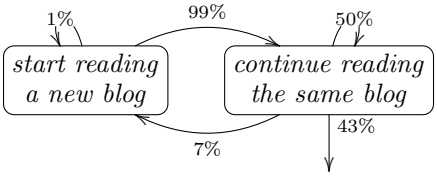


Figure 2.2: Example of a VBMG model for blurkers

that, if a blurker reads the same blog at least twice, he can also leave the blog with a probability of 43% (exit transition).

Shams et al. [SKF06] proposed an application modeling methodology to handle inter-request and data dependencies. The methodology relies on Extended Finite State Machines (EFSMs) that can model applications with higher-order request dependencies without encountering the state explosion problem [LY96], typical in FSM-based approaches. Consequently, EFSM is better suited for modeling web applications than CBMG and VBMG. Figure 2.3 depicts and example of EFSM for an e-commerce system, where nodes are states in the user's navigation and arcs are requests to the web application. Two request dependency state variables are used to enforce inter-request dependencies. The *items_in_cart* is an integer variable that indicates the number of items in the shopping cart and the *signed_on* is a boolean variable that states whether a user has signed on or not. For example, the *items_in_cart* variable is incremented by 1 when the user executes the *Add* request type (transition from S_2 to S_3), and it is decremented by 1 when the user executes the *Delete* request type (transition from S_3 to S_4). So that, the *Checkout* request type (transition from S_4 to S_5) is only allowed when the previous sequences of requests have resulted in at least one item in the shopping cart ($items\_in\_cart > 0$).

Benevenuto et al. [BRdMCA09] introduced the Clickstream Model to characterize user's behavior in On-line Social Network (OSN). This approach identifies and describes representative users' behaviors in OSNs by characterizing the type, frequency, and sequence of their activities. The modeling of the system implies two steps: i) to

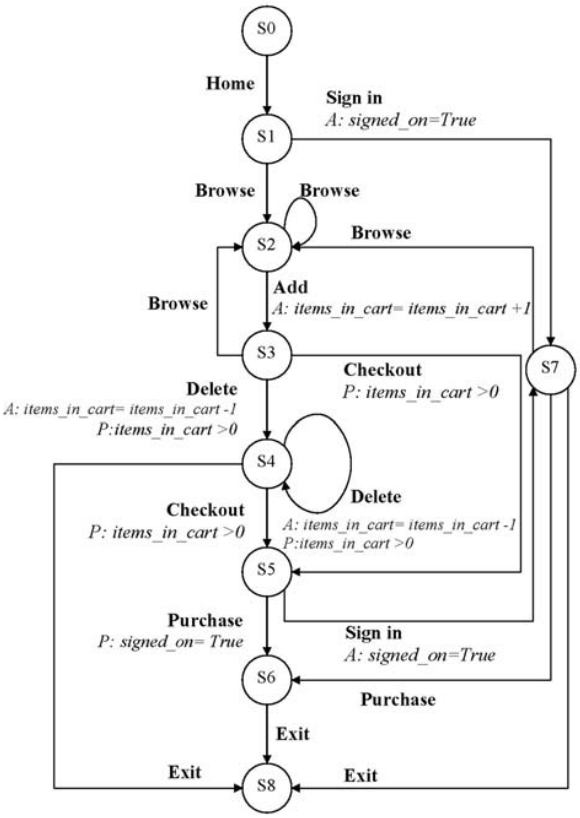


Figure 2.3: Example of a simplified EFSM model for an e-commerce system

Source [SKF06]

identify dominant user's activities in clickstreams, and ii) to compute the transition rates between activities. For illustrative purposes, Figure 2.4 shows the transition probability in the Clickstream Model for an OSN.

These four models only characterize web workload for specific paradigms or applications, but they either do not model user's dynamic behavior for a general context and in an appropriate and accurate way (*Challenge I*) or do not consider user's dynamic roles (*Challenges II and III*).

On the other hand, there is an evidence of an important change of user interaction with the Web. For instance, a recent study showed that 57.4% of web sessions involve parallel browsing behavior [HW10]. This behavior was originally found in the experienced users, who surf the Web by using multiple browsers tabs or windows to support backtracking or multitasking with the aim of enhancing their navigation productivity

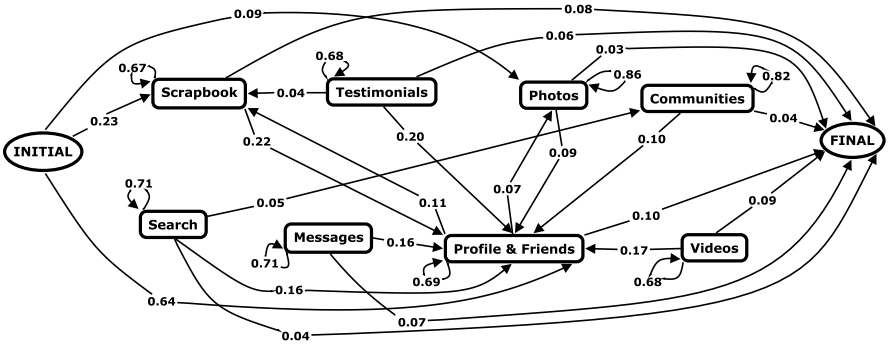


Figure 2.4: Transition probability in the Clickstream Model for an OSN

Source [BRdMCA09]

[AJK05, Tha08]. Moreover, the history-back button, included in any current web browser, is still one of the world's most heavily used user interface components in the web context, and accounts for up to 31% of all revisits [OWHM07]. This important change has been considered in several studies and tools to improve the website usability [ASW06], to test web applications [DLDP03] or learning user preferences [SZ00]. However, to the best of our knowledge, User-Browser Interaction (UBI) have not been taken into account when modeling user's dynamism on workload characterization in web performance studies yet.

2.2 Web workload generators overview

Workload generators are software products based on workload models to generate HTTP requests sequences similar to real requests. They are designed and implemented as versatile software tools for performing tuning or capacity planning studies.

Comparing web workload generators is a laborious and difficult task since they offer a large amount and diversity of features. In this section we contrast generators according to a wide set of features and capabilities, focusing on their ability to reproduce user's dynamism in performance studies for current Web.

To this end, Section 2.2.1 analyzes a representative subset of state-of-the-art workload generators as a first step, highlighting their main features as web performance evaluation software and their main disadvantages when reproducing accurate workload for current Web. After that, in Section 2.2.2 we evaluate and classify these generators, concentrating on those that consider user's dynamic behavior.

2.2.1 Software tools study

2.2.1.1 WebStone

WebStone [Min02b] was designed by Silicon Graphics in 1996 to measure the performance of web server software and hardware products. It was acquired by Mindcraft, Inc. that improved its reliability and portability as well as the reproducibility of performance tests. Moreover, new workloads for CGI, NSAPI and ISAPI tests were provided. Nowadays, both executable and source code for WebStone are available for free.

The benchmark generates a web server load by simulating multiple *web clients* navigating a website as shown in Figure 2.5. These clients can be considered as users, web browsers, or other software that makes requests to the website files, which can be classified in different categories according to their size. The simulation is carried out using multiple clients running on one or more computers to generate large loads on a web server. All the testings done by the benchmark are controlled by a *webmaster*, which is a program that can be run on one of the client computers or on a different one. The webmaster distributes the web client software and test configuration files to the client computers. After that, it starts the execution and waits for the clients to report the performance they measured. Finally, the webmaster combines the performance results from all the clients into a single report.

The performance measured by WebStone depends on the set of files used by the web clients. The set used by default is based on a model of the Silicon Graphics website in 1995, although it is possible to change the file set to one that better simulates the website of interest.

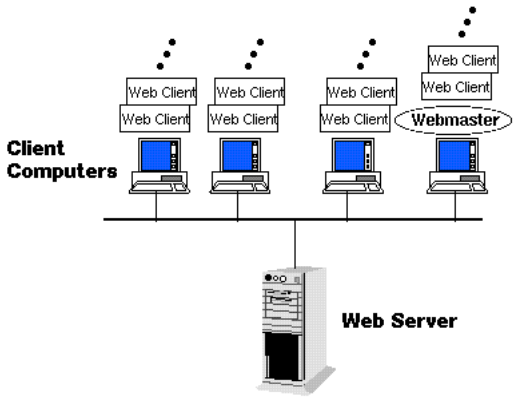


Figure 2.5: WebStone architecture

Source [Min02a]

To sum up, WebStone is one of the first software products proposed to measure the performance of web systems but it seems obsolete for the current Web. Table 2.1 summarizes its main features and disadvantages.

Main features	• Parameterized workload. • Distributed model for workload generation. • Open performance reports. • Open source solution.
Disadvantages	• Basic HTTP protocol only. • No users' navigations characterization. • No facilities to consider user's dynamism.

Table 2.1: Main features and disadvantages of WebStone

2.2.1.2 SPEC's Benchmarks for Web Servers

The Standard Performance Evaluation Corporation (SPEC) has commercialized *Benchmarks for Web Servers* [SPE09] from 1996 to the early 2012. This benchmarks' family is designed to measure the performance of systems offering services in the Web. The last member of the family, named *SPECweb2009*, includes many sophisticated and state-of-the-art enhancements to meet the modern demands of the current Web, such as requests to static and dynamic content (ASP, JSP, and PHP), simultaneous user sessions, parallel HTTP connections to request page images or simulates browser caching effects.

Figure 2.6 shows the logical components of SPECweb2009. The *prime client* initializes and manages the other *clients*, sets up the *web server* and the *back-end simulator*, and stores the results of the benchmark tests. The web server handles the requests issued by the clients by itself or by communicating with the back-end simulator in order to retrieve specific information needed to complete HTTP responses. This simulator emulates the communication between a web server and a back-end application server. Each benchmark client generates HTTP requests according to certain workloads that are defined by studying three representative types of web applications (banking, e-commerce, and support).

Table 2.2 presents the main features and disadvantages of SPECweb2009. As observed, SPEC software is a mature benchmark that has evolved with the Web, nonetheless it has not achieved to reproduce realistic workload in the performance studies for the current Web because it does not consider user's dynamism on workload characterization.

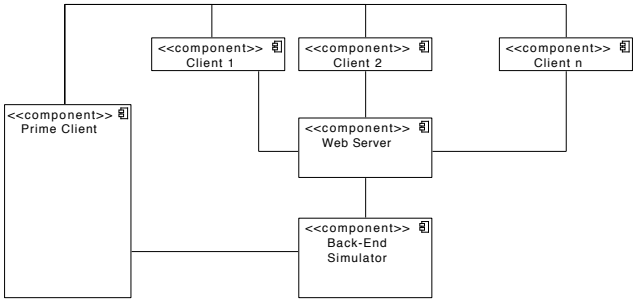


Figure 2.6: Logical components of SPECweb2009

Main features	• Parameterized workload. • Different types of workloads according to the kind of web application. • Distributed model for workload generation. • Full HTTP protocol (cookies, HTTPS, dynamic content, etc). • Performance reports. • Proprietary software.
Disadvantages	• No users' navigations characterization. • No facilities to consider user's dynamism.

Table 2.2: Main features and disadvantages of SPECweb2009

2.2.1.3 SURGE

The *Scalable URL Reference Generator (SURGE)* [Bar98] was developed by Barford in 1998 with the goal of measuring the server behavior while varying the user load. The need to develop SURGE appeared with the difficulty to generate representative traces for the Web because workloads generated by web users have a number of unusual features, such as the highly variable demands experienced by the web servers or the self-similarity shown by the network traffic [BC98]. To tackle these drawbacks, SURGE performs an analytical characterization of the user load and a set of mathematical models that generate the HTTP requests in the server [BBBC99]. These models characterize:

- The distribution of sizes of unique files requested from web servers.

- The distribution of sizes of all files transferred from web servers.
- The popularity of all requested files.
- The temporal locality of requested files.
- The active (ON) and inactive (OFF) periods of time for the emulated users.
- The number of documents transferred during an active period.

SURGE was designed as a scalable software framework where the previous models are combined according to the various components of the Web [BC97]. The software resides on a sets of clients that are connected to a web server as depicted in Figure 2.7. Each client executes a set of threads that request sets of documents which are then transferred by the server (ON time). After receiving a set, the thread sleeps for a some amount of time (OFF time) simulating the user's think time.

In summary, SURGE was a step forward on modeling accurate workload for evaluating the performance of Web 1.0. Specifically, it was able to produce self-similar network traffic under conditions of both high and low workload intensity. However, it also seems to be obsolete for the current Web because its generation process is based on analytical models that do not consider user's dynamism, and it cannot model 3-tier architectures for dynamic content generation. Table 2.3 summarizes the main features and disadvantages of SURGE.

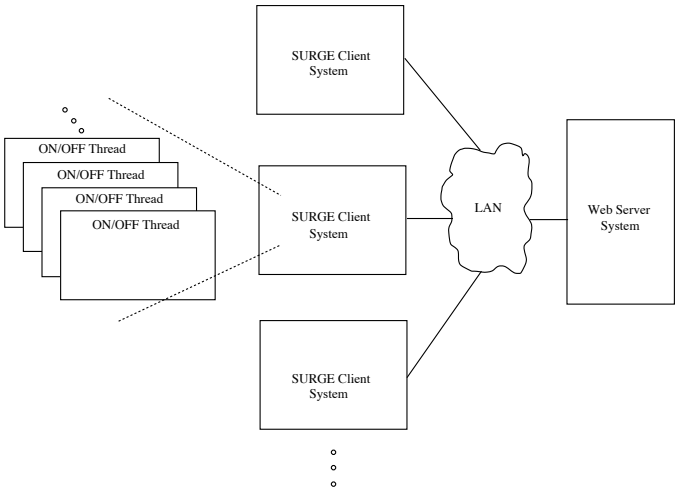


Figure 2.7: SURGE architecture

Source [BC97]

Main features	• Workload generation architecture based on analytical models. • Distributed model for workload generation. • Open source solution.
Disadvantages	• Basic HTTP protocol only. • No users' navigations characterization. • No facilities to consider user's dynamism.

Table 2.3: Main features and disadvantages of SURGE

2.2.1.4 S-Clients

Banga and Druschel proposed in their approach a new improved methodology for HTTP request generation [BD99]. In this context, *S-Clients* was designed with the aim of reproducing bursty traffic with peak loads exceeding the capacity of the server as well as the modeling delay and loss characteristics of Wide Area Networks (WANs).

Figure 2.8 shows the S-Clients design. It defines an architecture (Figure 2.8a) where a set of client machines are connected to the server machine being tested through a router, which has sufficient capacity to support the maximum client traffic specification. The purpose of the router is to simulate WAN effects by introducing an artificial delay and/or dropping packets at a controlled rate. Each client machine runs a number of *scalable client* processes. S-Clients splits the process of generating traced HTTP requests in two subprocesses: one for obtaining the connection and other for recovering the content (Figure 2.8b), so enabling a relative parallelism.

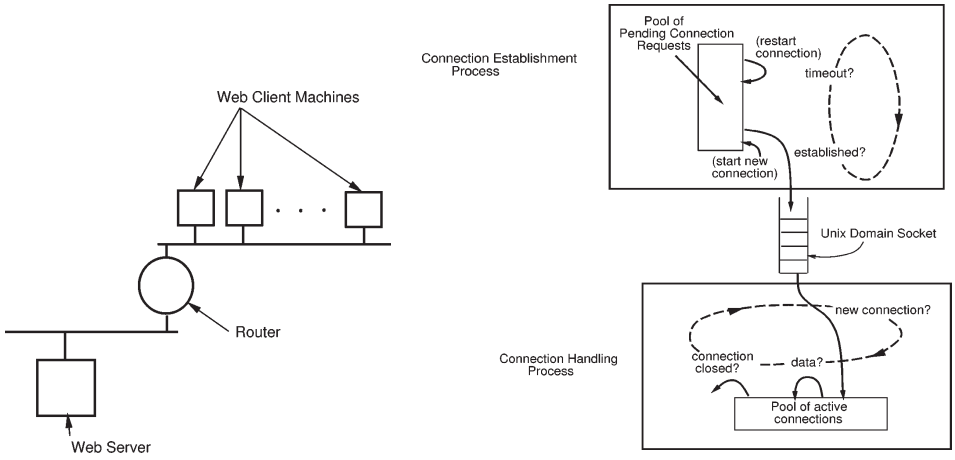
To sum up, S-Clients was an architecture devised to improve workload generators for Web 1.0 that is still interesting to be considered in Web 2.0. Table 2.4 presents the main features and disadvantages of S-Clients.

2.2.1.5 WebJamma

WebJamma was a library to generate HTTP traffic written by the Network Research Group at Virginia Tech [CAJ+99]. It is aimed at serving as baseline for developing a full web workload generator.

This library works in a simple way by taking a URL file that provides the source of the HTTP requests to be generated, so it cannot represent user's dynamism. It uses a multiprocessing architecture based on distributed generation nodes to test the performance of web caching subsystems.

In summary, *WebJamma* was an interesting open source library to generate HTTP requests in a easy way. Table 2.5 shows its main features and disadvantages.



(a) Basic architecture (b) A Scalable Client

Figure 2.8: S-Clients design

Source [BD99]

Main features

- Parameterized workload.
- A router to simulate WAN effects.
- A split generation process to avoid limits on HTTP requests.
- Open source solution.

Disadvantages

- Only an architecture for workload generation.
- Basic HTTP protocol only.
- No users' navigations characterization.
- No facilities to consider user's dynamism.

Table 2.4: Main features and disadvantages of S-Clients

2.2.1.6 TPC Benchmark™ W

TPC Benchmark™ W (TPC-W) is a transactional web benchmark defined by the Transaction Processing Performance Council [Tra02a]. It models a representative e-commerce system, specifically an on-line bookstore environment, with the aim of

Main features	• Easy to use as a baseline of other software generators. • Open source solution.
Disadvantages	• Basic stressing functionalities. • No users' navigations characterization. • No facilities to consider user's dynamism.

Table 2.5: Main features and disadvantages of WebJamma

evaluating the architecture performance on a generic profile. To this end, the benchmark provides both models of business-client and business-business and examines real features of e-commerce applications, such as: catalog, searcher, security, etc.

As shown in Figure 2.9, TPC-W presents a client-server architecture. The *remote browser emulators* are located in the client side and generate workload towards the e-commerce web application, which is located in the server side (*e-commerce server*). With the aim of reproducing a representative workload, the emulators simulate real users' behaviors when they surf the website by using the CBMG model, which is composed of all pages of the on-line bookstore and the associated transition probability. The server hosts the *system under test*, which consists of a web server and its storage of static contents, and an application server with a database system to generate dynamic content. The *payment gateway emulator* represents an entity to authorize users' payments. These three main architecture components are interconnected through a dedicated network.

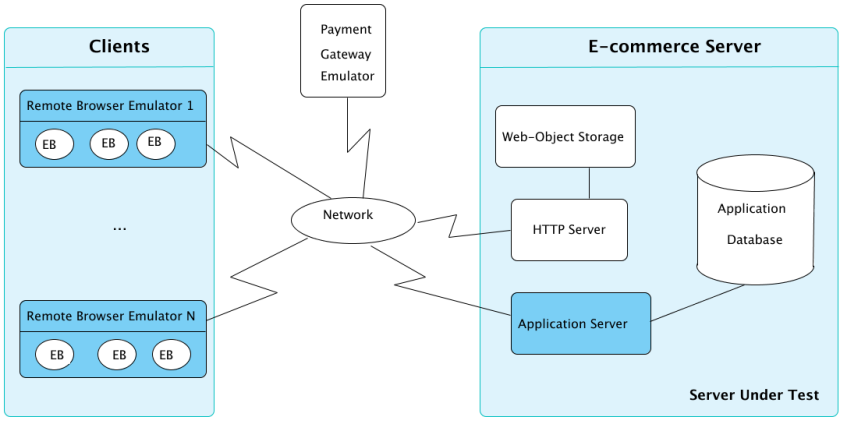


Figure 2.9: TPC-W architecture

To sum up, TPC-W was the first benchmark for e-commerce considering the users' behaviors on workload generation. To this end, TPC-W adopts CBMG model to define web workload in spite of this model only characterizes user's dynamic behavior partially, as introduced in Section 2.1. The benchmark, which has been commonly accepted by the scientific community in many research works [DMB01, ACC+02, GG03], presents the main features and disadvantages shown in Table 2.6.

Main features	• Parameterized workload. • Different types of workloads according to the type of scenario. • Distributed model for workload generation. • Full HTTP protocol (cookies, HTTPS, dynamic content, etc). • Basic facilities to consider user's behavior. • Performance reports. • Open source solution.
Disadvantages	• Basic users' navigations characterization. • No advanced facilities to consider user's dynamism.

Table 2.6: Main features and disadvantages of TPC-W

2.2.1.7 Web Polygraph

Web Polygraph is a performance testing tool for caching proxies, origin server accelerators, L4/7 switches, content filters, and other web intermediaries. It was originally developed at the University of California by Wessels and Rousskov in the context of the IRCache project [RWC99]. Nowadays, it is copyrighted by *The Measurement Factory* [MF12] that authorizes the use of Polygraph under the Apache License.

The benchmark consists of virtual clients and servers glued together with an experiment configuration file [RW03]. Clients, named *robots*, generate HTTP requests for the simulated objects. These requests may be sent directly to the servers (e.g. web servers), or through an intermediary (e.g. proxy cache or load balancer) using a configurable mix of HTTP/1.0 and HTTP/1.1 protocols, optionally encrypted with SSL or TLS. The benchmark can be configured to produce a variety of realistic and unrealistic workloads based on a synthetic workload characterization. As Polygraph runs, measurements and statistics are gathered for a detailed postmortem analysis.

In summary, Web Polygraph is a versatile tool for generating web traffic and measuring proxy performance that was chosen for several industry-wide benchmarking

events. Table 2.7 shows its main features and disadvantages focusing on workload generation.

Main features	• Synthetic workload characterization. • Distributed model for workload generation. • Full HTTP protocol. • Performance reports. • Successful industrial solution. • Apache License.
Disadvantages	• No users' navigations characterization. • No facilities to consider user's dynamism.

Table 2.7: Main features and disadvantages of Web Polygraph

2.2.1.8 LoadRunner

LoadRunner is one of the most popular industry-standard software products for functional and performance testing. It was originally developed by Mercury Interactive, but nowadays it is commercialized by Hewlett-Packard [HP12a].

Figure 2.10 shows how LoadRunner works [HP12b]. As observed, it tests a web application by emulating an environment where multiple users work concurrently. Moreover, it accurately measures, monitors, and analyzes performance and function-

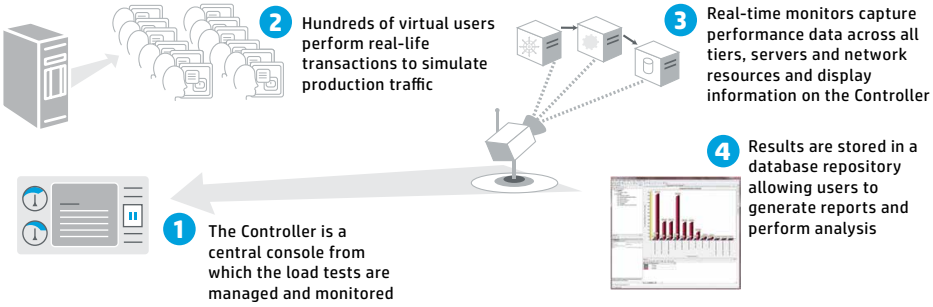


Figure 2.10: How LoadRunner works

Source [HP12b]

ality of the application while it is working under load. The testing process is controlled by a central console.

LoadRunner supports the definition of users' navigations, which are represented using a scripting language, to characterize users' families. Figure 2.11 depicts the sequential approach to scripting a Web 2.0 application using LoadRunner [HP12c]. First the basic steps are recorded, creating a *shell* script. Next, this script is then taken off-line, and undergoes further manual steps such as data parameterization and correlations. Finally, the desired performance scripts are obtained after adding transactions and any other required logic. LoadRunner scripting permits only to partially reproduce user's dynamism when generating web workload because it cannot define neither advanced interactions of users, such as parallel browsing behavior, nor continuous changes in user's behaviors.

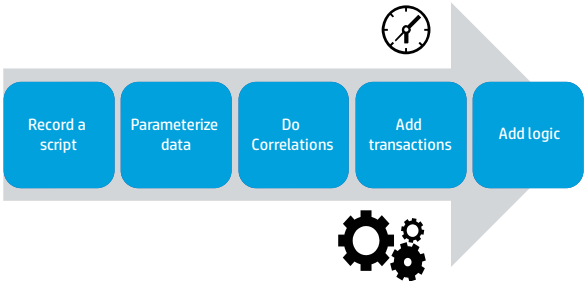


Figure 2.11: LoadRunner scripting for Web 2.0 applications

Source [HP12c]

To sum up, LoadRunner is one of the most important software products to test the functionality and performance of a web application. It presents some facilities to consider user's dynamism on the workload generation process, but only partially. Table 2.8 summarizes its main features and disadvantages.

2.2.1.9 WebLOAD

WebLOAD [Rad12] is a software for web performance commercialized by RadView since 1997. It is oriented to explore the performance of critical web applications by quantifying the utilization of the main server resources.

Figure 2.12 depicts the WebLOAD architecture. The *authoring environment* is a software tool to create scenarios that try to mimic the navigations of real users. To this end, it provides facilities to record, edit and debug *test scripts*, that are used to define the scenarios on workload characterization. The *execution environment* is a console to manage tests execution, whose results are analyzed in the *Analytics* application. Since WebLOAD is a distributed system, it is possible to deploy several *load generators* to reproduce the desired load. Load generators can also be used as

Main features	• Parameterized workload. • Different types of workloads according to the type users' families. • Distributed model for workload generation. • Full HTTP protocol (cookies, HTTPS, dynamic content, etc). • Basic facilities to consider user's behavior. • Advanced reports during performance evaluation studies. • Multi-platform.
Disadvantages	• No users' dynamic navigations characterization. • No advanced facilities to consider user's dynamism.

Table 2.8: Main features and disadvantages of LoadRunner

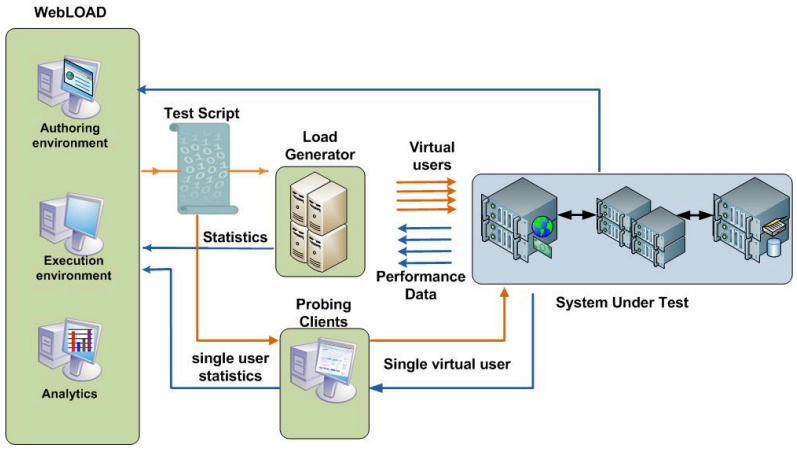


Figure 2.12: WebLOAD architecture

Source [Rad12]

probing clients where a single virtual user is simulated to evaluate specific statistics of a single user. These probing clients resemble the experience of a real user using the system while it is under load.

In summary, WebLOAD is a commercial software that presents some capability to generate user's dynamic behavior, but only in a partial way, when evaluating performance of a given web application. Table 2.9 shows the main features and disadvantages of WebLOAD.

Main features	• Parameterized workload. • Different types of workloads according to the type of scenario. • Distributed model for workload generation. • Full HTTP protocol (cookies, HTTPS, dynamic content, etc). • Basic facilities to consider user's behavior. • Advanced reports during performance evaluation studies. • Multi-platform.
Disadvantages	• No users' dynamic navigations characterization. • No advanced facilities to consider user's dynamism.

Table 2.9: Main features and disadvantages of WebLOAD

2.2.1.10 JMeter

JMeter [ASF12] is an open source solution presented by the Apache Software Foundation and designed to generate web workload with the aim of testing client/server software, such as web applications and services.

The generator is written entirely in Java and provides an easily configurable and visual API to define, execute and analyze web performance tests from the client side. It presents partial capability to generate dynamic user's workload by defining a *navigation test* based on patterns (e.g. regular expressions). Additionally, JMeter presents some facilities to check the functionality of a web application, such as test scripts which use assertions to validate that the application returns the expected results. Table 2.10 summarizes its main features and disadvantages.

2.2.1.11 Testing scripts and tools

With the increasing popularity of web applications, some software and testing factories or web developers have created several scripts and tools, which are usually basic and

Main features	• Parameterized workload. • Different types of workloads according to navigation tests. • Full HTTP protocol (cookies, HTTPS, dynamic content, etc). • Basic facilities to consider user's behavior. • Basic performance reports. • Open source solution. • Multi-platform.
Disadvantages	• No users' dynamic navigations characterization. • No advanced facilities to consider user's dynamism.

Table 2.10: Main features and disadvantages of JMeter

open source approaches. These tools capture HTTP requests and reproduce them for the purpose of stressing applications and testing their functionalities.

For instance, *HTTPERF* [MJ08] and *Deluge* [Bla05] were developed as tools for measuring web server performance in Hewlett-Packard and Thrown Clear Productions, respectively. HTTPERF is not focused on implementing one particular benchmark but on providing a robust high-performance tool that facilitates the construction of both micro- and macro-level benchmarks [MJ98]. In contrast, Deluge is a final stressing tool that includes three main components: i) `dlg_proxy` that records HTTP requests, ii) `dlg_attack`, which generates workload by reproducing recorded users' requests, and iii) `dlg_eval` that elaborates statistics from the generated results.

On the other hand, *HAMMERHEAD 2* [WDG11], *PTester* [Eri99], *Siege* [Ful12] and *Autobench* [Mid04] are examples of scripts and utilities deployed by the open source community to evaluate the quality of its developments.

Table 2.11 summarizes common features and disadvantages for these tools.

2.2.2 A survey on reproducing user's dynamism

In this section we classify the studied tools according to a wide set of features and capabilities. Below, the twelve features and capabilities used are defined to ease the understanding of the comparison study.

1. *Distributed architecture.* It refers to the ability to distribute the generation process among different nodes. The distribution of the workload generation significantly helps us to improve the workload accuracy.

Main features	• Easy to use and introduce in both development and testing processes. • Simple reports for functional and performance tests. • Open source solutions.
Disadvantages	• Basic stressing functionalities. • No users' navigations characterization. • No facilities to consider user's dynamism.

Table 2.11: Main features and disadvantages of testing scripts and tools

2. *Analytical-based architecture.* This feature represents the capability to use analytical and mathematical models to define the workload. These models allow to improve the workload quality by using them as workload parameters (e.g. user's behavior models or simulation architectures).

3. *Business-based architecture.* When defining a testing environment, the simulator architecture should implement the same features as the real environment (e.g. e-commerce architectures typically include a catalog, a product searcher or a payment gateway), so it is quite important to model the business logic deployed by the web application under test.

4. *Client parameterization.* This is the ability to parameterize generators nodes (e.g. number of users, allowed navigations set, or changes between navigations). In general, web dynamism highlights the need for a workload characterization based on parameters, and specially the related to user's behavior.

5. *Workload types.* Some generators organize the workload in categories or types, each one modeling a given user profile (e.g. searcher or buyer user profiles).

6. *Testing the web application functionality (functional testing).* This capability permits to define functional tests related to a real web application. These tests allow to guarantee the application correctness; that is, the application provides the defined functionality, which fulfills the quality and assurance requirements.

7. *Multi-platform* is referred to a software package that is implemented in multiple types of computer platforms inter-operating among them.

8. *Differences between LAN and WAN.* Simulations usually run in Local Area Network (LAN) environments. Most of the current simulators cannot model differences between LAN and WAN, where applications are usually located.

9. *Ease of use.* The generator should be a friendly application carrying out usability guidelines, mainly in commercial products.

10. *Performance reports.* The elaborated results by the generation process are usually presented by using both on-line and off-line graphical plots.

11. *Open source.* This feature allows the open source community to develop extensions or different generation alternatives over the generator architecture.

12. *User's dynamism.* This is the main feature we are interested in, because the dynamism in contents and users is the most relevant characteristic in the current Web that makes workload generation difficult.

Table 2.12 summarizes the studied software packages used to generate web workload as well as the grade (full or partial) in which they fulfill the features described above. These software packages can be classified in three groups according to their main application contexts:

- *Group I: Benchmarks that model the client and server paradigm in web context.* In this case, among the five studied benchmarks, only TPC-W provides a workload generation process that considers user's dynamism, but only partially. The others do not model user's dynamism because: i) they are simulation approaches that do not reproduce real workload (WebStone and Web Polygraph), or ii) they are based on analytical models that do not consider user's dynamism as a parameter (SPECweb and SURGE).

- *Group II: Software products to evaluate performance and functionality of a given web application*, such as LoadRunner, WebLOAD and JMeter. All of them provide abilities to generate web workload taking into account user's dynamism in a partial way.

- *Group III: Testing tools and other approaches for traffic generation*, that cannot reproduce user's dynamic behavior due to they are based on HTTP traces.

As observed, only four of the studied approaches (i.e. TPC-W, LoadRunner, WebLOAD and JMeter) present some capability to reproduce user's dynamism. Table 2.13 deals with the ability of considering user's dynamism in depth, and explores how each approach takes into account the three challenges. Notice that, the four generators provide some capability to partially reproduce the dynamism of users when they surf a website (*Challenge I*) but in a different way. For instance, TPC-W only considers a probabilistic approach to define users' navigations by using the CBMG model. On the other hand, LoadRunner, WebLOAD and JMeter provide scripting languages that permit to define users' navigations considering conditional transitions between their pages. Among these three generators, only the commercial products (LoadRunner and WebLOAD) offer software artifacts to represent the different behaviors of users (*Challenge II*), but they do not mind continuous changes in these behaviors.

FEATU./CAPAB. \ GENERATOR	WebStone	SPECweb	SURGE	Web Polygraph	TPC-W	LoadRunner	WebLOAD	JMeter	S-Clients	WebJamma	Deluge	HAMMERHEAD 2	PTester	Siege	HTTPERF	Autobench
	GROUP I					GROUP II			GROUP III							
Analytical-Based Architecture	◆	◆	◆	◆	◆	◆	◆	▲	◆	◆	◆	▲	◆	▲	◆	◆
Distributed Architecture	◆	◆	◆	◆		◆	◆		◆	◆	◆	◆	◆	◆	◆	◆
Business-Based Architecture		▲		▲	◆	◆	◆	◆								
Client Parameterization	◆	◆		▲	◆	◆	◆	◆		◆		▲			▲	
Workload Types		◆			◆	◆	◆	▲	◆		◆		◆	▲		◆
Functional Testing						◆	◆	▲				▲	▲	▲		▲
LAN and WAN									◆							
Multi-platform	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	▲	◆	▲	◆	◆
Ease of Use						◆	◆	▲								
Performance Reports	▲	◆		◆	◆	◆	◆	◆			▲	▲	▲	▲	▲	▲
Open Source	◆		◆	▲	◆			▲	◆	◆	◆	◆	◆	◆	◆	◆
User's Dynamism					▲	▲	▲	▲								

◆ Full support ▲ Partial support

Table 2.12: Web workload generators and grade in which main features are fulfilled

	TPC-W	**LoadRunner**	**WebLOAD**	**JMeter**
Challenge I	◎	⊛	⊛	⊛
Challenge II		Ⓢ	Ⓢ	
Challenge III				

◎ Analytical approach ⊛ Scripting Ⓢ Software artifact

Table 2.13: Web workload generators and how challenges of user's dynamism are fulfilled

2.3 Summary

This chapter has analyzed state-of-the-art workload models and generators for web performance evaluation, focusing on the capability to fulfill the three challenges that have been introduced in Chapter 1.

With the aim of improving workload models, few approaches (CBMG, VBMG, EFSM and Clickstream Model) provided some capabilities to represent user's behavior on web workload characterization, but they do not offer an accurate solution to model user's dynamism.

Furthermore, among the studied software tools, only one benchmark for e-commerce (TPC-W) and three software solutions to evaluate functionality and performance of a given web application (LoadRunner, WebLOAD and JMeter) provide mechanisms to reproduce users' navigations in current Web. However, these mechanisms do not consider all challenges when reproducing user's dynamic behavior, so they are not enough to mimic real patterns of HTTP requests.

These lacks in models and software motivate us to propose a more accurate workload model in order to develop a new workload generator with the aim of analyzing the effect of using dynamic workloads on web performance evaluation, instead of traditional workloads.

A first review of this state-of-the-art can be found in the context of the GENERICA project as a technical report [GENERICA'04a]. A summary of this work was published in [WOSP'05, IJEB'05]. In [WEBIST'11], an updated state-of-the-art was presented.

CHAPTER 3

DWEB: modeling user's dynamism on web workload characterization

This chapter proposes the Dynamic WEB workload model (DWEB) with the aim of characterizing a more realistic workload when evaluating the performance of current web applications.

DWEB tackles in a progressive way the three previously mentioned challenges when modeling the user's behavior on representative workloads. To this end, it defines a couple of new concepts: *user's navigations* and *user's roles*. These concepts characterize different levels of dynamism in the workload definition by means of modeling web users' behaviors.

The remainder of this chapter is organized as follows. Section 3.1 introduces the navigation concept in general terms, describes its notation and provides an example of modeling a first level of dynamism. Section 3.2 defines the concept of role and presents an example of representing a second level of dynamism. Finally, some concluding remarks are drawn in Section 3.3.

3.1 The user's navigation

The concept of *user's navigation* defines a first level of user's dynamism and satisfies the *Challenge I* by modeling user's dynamic behavior when interacting with the contents and services offered by the Web.

For instance, a typical navigation of a user searching for specific information, usually begins with a query on a web finder. Queries are frequently cancelled when the response time surpasses a certain value, which is characteristic for each user and his current navigation conditions. In the case of obtaining results, users usually visit the first site on the list or refine the search when receiving too much information. Analyzing this simple example, one can see that each user request depends not only on the response itself but also on other issues related to the quality of service (e.g. response time length or content amount), and the users' states. That is, users take

their navigation decisions according to their personal preferences, navigation goals, visited resources, network and connectivity conditions, etc.

The navigation concept is not just limited to reproduce human behavior, since it can be further applied to any web client, such as software automatons, that are easier to model than users because they follow a given navigation pattern. Nevertheless, the strong point of the concept lies in the flexibility to represent dynamism in user's behavior when interacting with the Web.

Formally, a user's navigation N is defined as a sequence of n URLs of HTTP requests where each visited URL depends on the previously visited ones, as defined in equation 3.1.

$$N = \{url_1, url_2, ..., url_n\} \ / \ \forall i = 2..n : url_i \text{ depends on } url_k \text{ for } k < i \text{ and } \textit{user's-state}_{i-1} \tag{3.1}$$

where url_i refers both to the content related to the resource i and its associated characteristics and, $\textit{user's-state}_{i-1}$ denotes the user's state resulting from the interaction with the previously visited resource.

A graph where nodes were pages of websites and arcs were transitions between pages is not enough as a visual representation of a user's navigation, because we need a visual modeling language that allows us to define user's state. Moreover, this language has to provide mechanisms to easily model the user's dynamism when navigating.

There are several successful extensions of Unified Modeling Language (UML) applied to web engineering. For instance, *User eXperience diagrams* [Con03] are introduced to model the storyboards and the dynamic information of pages in building model-driven web applications. Moreover, these diagrams were extended to rapidly develop and deploy public administration portals considering usability factors [FPZ07].

Due to these reasons, we decide to represent a navigation using a state machine view of UML. In general, a state machine is a graph of states and transitions that describes the response of an object to the events that it receives [RJB99]. We simplify this model by considering only a reduced set of graphical elements where states are web pages, as shown in Table 3.1. The result has been adopted for visual representation of the navigation concept.

For illustrative purposes, Figure 3.1 shows the visual navigation corresponding to a *Google search* where some dynamic issues in the user's behavior (e.g. dynamic think time or conditional and parallel requests) are introduced. Two main parts can be distinguished in this navigation:

1. The upper part of the diagram (before reaching branch *b1*) shows the two ways in which the search can be initiated:

   (1) On the left side, the user makes use of a search toolbar of a web browser (e.g. Google toolbar for Mozilla Firefox) to make the query directly.

   (2) On the right side of the figure, the user reaches branch *b1* after the *Google.HOME* node, where the user requests the main page to the web

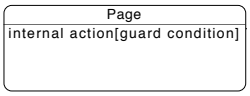
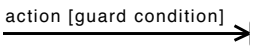
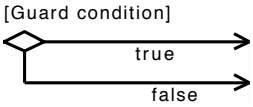
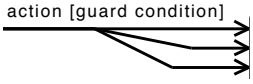
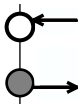
Notation	Name	Description
	*Beginning*	The initial state in a state machine represents the beginning of a navigation.
	*End*	The final state in a state machine defines the end of a navigation; that is, the user leaves the website.
	*Page*	Pages are states where a user can execute actions (e.g. user's think time) when their guard conditions are true. These conditions can also consider probabilities in the same way than CBMG.
	*HTTP request*	Transitions are HTTP requests to pages that are executed when their guard conditions are true. These conditions can also consider probabilities in the same way than CBMG.
	*If/else*	A simple condition is introduced to ease understanding of branches in user's way.
	*Parallel HTTP request*	A parallel request starts parallelism on navigating to execute n HTTP requests using n different threads.
	*End of parallelism*	It transforms the parallel navigation in a sequential navigation again by killing the threading.
	*Extension*	It introduces a new extension point in the navigation (e.g. call to external constraints or functions), that is used to highlight issues of dynamism.
	*Call other navigation*	It calls another navigation that is defined outside the model.
	*States set/get*	It allows users to store (set) and recover (get) some information at their states (e.g. cookies or visited contents).

Table 3.1: User's navigation notation

searcher engine (www.google.com). After that, he waits for a while (time referred to as the user's think time) and then the user makes the query.

2. In the bottom of the diagram, the user analyzes the results for a dynamic think time, which depends on the number of results, and takes a decision (conditional request):

   (1) If the web search engine provides results (path from *b1* to *b2*) the user analyzes them. After that, he can refine the results by making a new query, *refined query* in the figure (path from *b2* to *b1*), or access the top 10 sites provided by using multiple browsers tabs (one for each result). Finally, he finishes the navigation (path from *b2* to black dot through *X_TH_RESULTS.HOME* node).

   (2) On the contrary, when no results are provided (path from *b1* to *b3*), the user can make a new query, *other query* in the figure (path from *b3* to *b1*), or finish the process (path from *b3* to black dot).

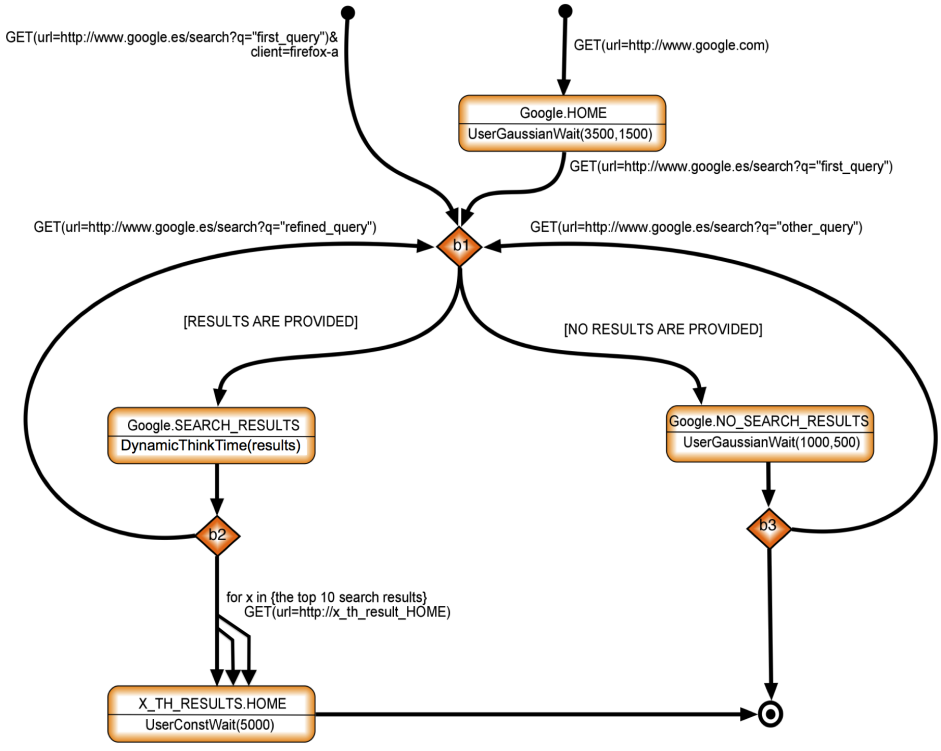


Figure 3.1: Google Search navigation pattern

3.2 The user's roles

The DWEB model proposes the concept of *user's roles* to fulfill the *Challenges II and III* by introducing a second level of dynamism in user's behavior. This level is related to the roles that users play when navigating the Web. The continuous changes of these roles defines users' behaviors.

For instance, assume people at the Import and Export Department in a typical multinational company who usually access the Web during their working time. Assume also that the company has got an intranet (web ERP) which allows web access. Most of the time, workers use Internet for professional purposes (e.g. intranet navigations, suppliers sites navigations, or professional web searches), but sometimes they use the web for leisure purposes (e.g. reading the news with Google Reader, performing personal searches, or checking the mail). So we can distinguish between two roles when a department member navigates the web: *working behavior* (professional navigations), and *leisure behavior* (personal navigations). Figure 3.2 defines the working and leisure behaviors of the example, and the likelihood to change between behaviors by using balanced arcs (the arc weight is the probability to change from the source behavior to the destination behavior). These behaviors are defined as automatons, where their nodes represent navigations, and their balanced arcs indicate the transitions between navigations (the arc weight indicates the probability to take that arc).

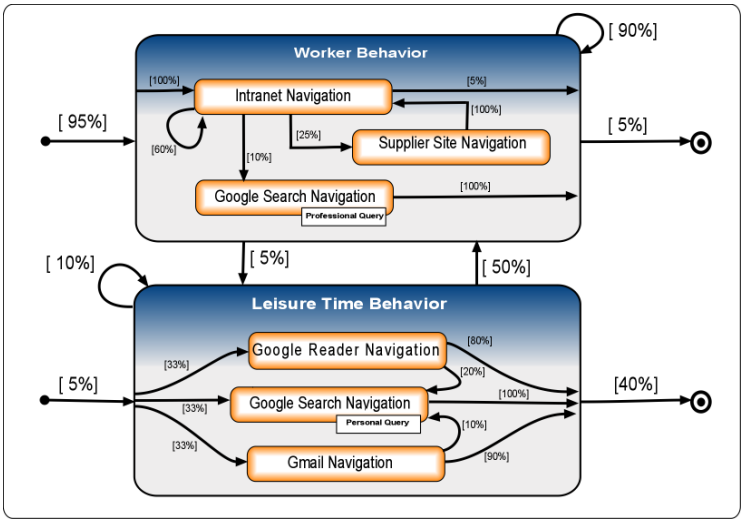


Figure 3.2: User's roles example: working and leisure behaviors

Formally, we define user's roles $R(C,\varphi:CxC\rightarrow N)$, where:

- C is the set of navigations; that is, $C = \{n_1, n_2 \ ... \ , n_k\}$ with $n_i \ \epsilon \ N$.
- $\varphi:CxC\rightarrow N$. φ is the function which provides the next navigation to be executed in terms of probabilities.

3.3 Summary

In this chapter we proposed the Dynamic WEB workload model (DWEB) with the aim of characterizing workload in a more accurate and appropriate way when evaluating the performance of current web applications. To this end, DWEB tackles in a progressive way the three challenges when modeling the user's behavior on representative workloads (introduced in Chapter 1) by using a couple of new concepts: *user's navigation* and *user's roles*.

The *user's navigation* defines a first level of user's dynamism by modeling user's dynamic behavior when interacting with the current Web, and consequently fulfills the *Challenge I*.

On the other hand, the concept of *user's roles* satisfies *Challenges II and III* by modeling a second level of dynamism related to the roles that users play when navigating the Web, and the continuous changes of these roles. Each role is explained in the terms defined by the navigation concept.

For illustrative purposes, some examples of using DWEB concepts have been presented, and the main visual notation has been described.

A summary of this contribution was published in [WOSP'05, COMCOMJ'09].

CHAPTER 4

GUERNICA: a workload generator for current Web

This chapter presents the Universal Generator of Dynamic Workload under WWW Platforms (GUERNICA), which is a web workload generator and testing tool to evaluate performance and functionality of web applications. GUERNICA was developed as a result of the cooperation among the *Web Architecture Research Group (Universitat Politècnica de València)*, *iSOCO S.L*, and the *Institute of Computer Technology*; thereby, bridging the gap between academia and industry.

The main aim of GUERNICA is its workload generation process, which is based on DWEB model that characterizes web workload modeling user's behavior as described in Chapter 3.

The remainder of this chapter is organized as follows. Section 4.1 introduces the main applications of GUERNICA, which permit to carry out performance studies based on the five phases detailed in Section 4.2. Next, the software architecture is described in Section 4.3, and the main features are recapitulated in Section 4.4. For illustrative purposes, Section 4.5 shows an example of web performance evaluation using the generator. Finally, we draw some concluding remarks in Section 4.6.

4.1 The application suite

GUERNICA is a software made up of three main applications (*workload generator*, *performance evaluator* and *performance tests planner*) as shown in Figure 4.1. Each application, described below, permits an autonomous distribution among different machines of the main activities in the evaluation of performance and functional specifications of a web application.

- The *performance tests planner* manages the testing process. Its main functionalities are: i) to define both performance and functional testing cases, ii) to plan their execution, iii) to monitor on-line results, which are generated by the

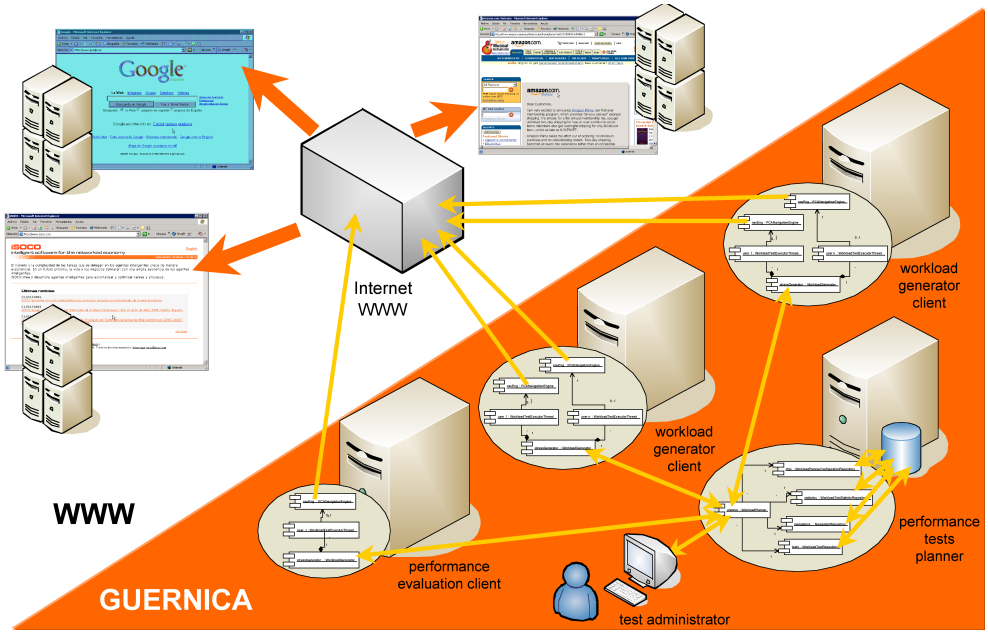


Figure 4.1: Main applications of GUERNICA

distributed clients, and iv) to elaborate final reports combining the obtained results.

- *Workload generators* reproduce web workload for stressing purposes. They are not required to be executed in the same machine as the planner, which configures generators and controls the executions. Each generator mimics user's behavior and notifies navigation statistics and results to the planner, which performs the corresponding graphical representation. Alternatively, a generator must stop its execution when the planner requires it.

- The *performance evaluator*, also known as *probe client*, is aimed at evaluating the major functionalities of a given web application while it is stressed by the workload generators, from the user's point of view. A probe client also can execute functional tests notifying their results to the planner.

In addition to these main applications, there are two software extensions that assist the suite:

- *CARENA* [NdlOG+05] is a `Mozilla` plugin that helps GUERNICA to define users' navigations. It captures and reproduces a real sequence of HTTP requests

that a user makes when surfing the Web. GUERNICA provides conversion from CARENA data format to its own data format.

- *SemViz*[BCCPB07] is an application to visualize knowledge based on ontologies by using 3D technology. GUERNICA transforms results of functional tests in an external format based on an ontology that SemViz can read and display graphically in a 3D representation.

4.2 Testing phases

GUERNICA permits to carry out a website evaluation of performance and functional specifications by following the next five phases (see Figure 4.2).

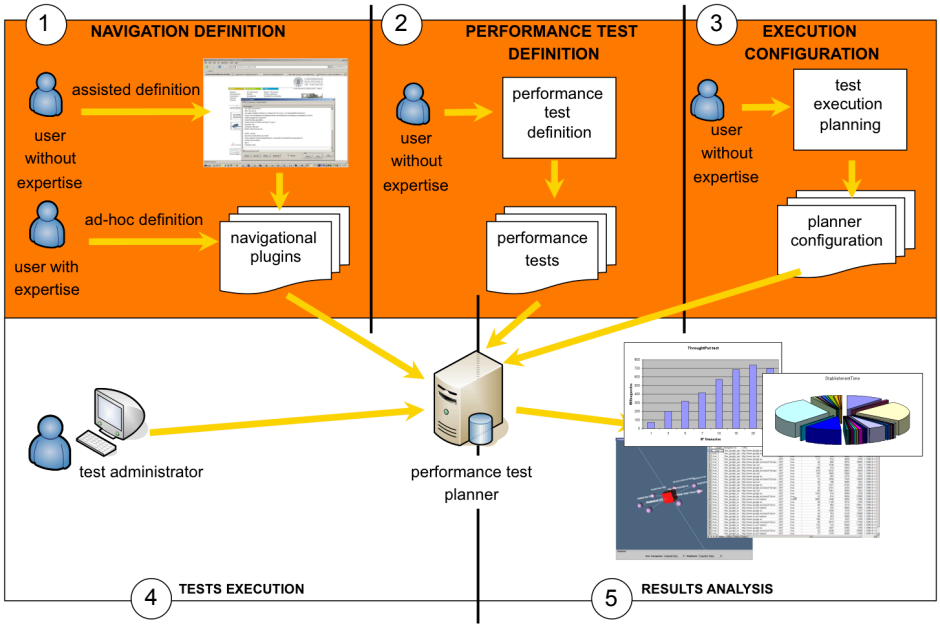


Figure 4.2: Testing phases in GUERNICA

1. The *navigation definition* phase characterizes the first level of user's dynamism by modeling his interactions with the dynamic web contents, typical of the current Web. To this end, GUERNICA adopts the user's navigation concept of DWEB, and implements it as a *navigational plugin*. As mentioned, CARENA

helps GUERNICA to define user's navigations by capturing real HTTP sequences that are converted, in a second step, to navigational plugins.

2. The *performance test definition* phase specifies the behaviors that users play in a website using *workload tests* that implement the user's roles concept of DWEB. A workload test specifies a set of navigational plugins that define user's behaviors by considering the capability of changing them with time.

3. The *execution configuration* phase sets others execution parameters, such as the number and type of users, the types of reports or the *workload distribution*. The distribution allows GUERNICA to improve the workload accuracy by dispersing its generation among different machines. Figure 4.3 shows the three main workload generation approaches that can be used by the *performance test planner* to coordinate the set of secondary workload generation processes.

   In the *ideal distributed model*, the planner, the generators, and the probe client are located in different machines. This is the best way to evaluate the performance of web applications because the probe is in an individual machine; therefore it is influenced neither by other generators nor by the planner.

   The *advanced distributed model* stipulates only three machines to host the planner, the generators and the probe. The main drawback of this approach is that all generators are in the same machine. In this model, the generated workload is not as real as in the ideal model because in a real environment different users are usually in different machines. Even so, this model can be used when there are not enough machines to perform the evaluation.

   In the *basic model*, the generators, the probe client, and the planner are located in the same machine. Thus, if the performance of this machine is not good enough, the approach will introduce *noise* in the tests. This model can be used as a first attempt in the workload generation to identify performance bottlenecks in web applications. Nevertheless, ideal or advanced approaches are required for a more accurate performance evaluation.

4. *Tests execution*. This phase executes workload tests gathering performance and functional statistics. Results collected by generators and probe clients are given to the planner, that groups, classifies and reaches a consensus among them in order to obtain a uniform set of results.

5. *Results analysis*. Finally, GUERNICA analyzes the performance of the system under test and represents performance indexes in different formats (e.g. graphical plots or tabular text). As introduced, SemViz visualizes results by using 3D technology.

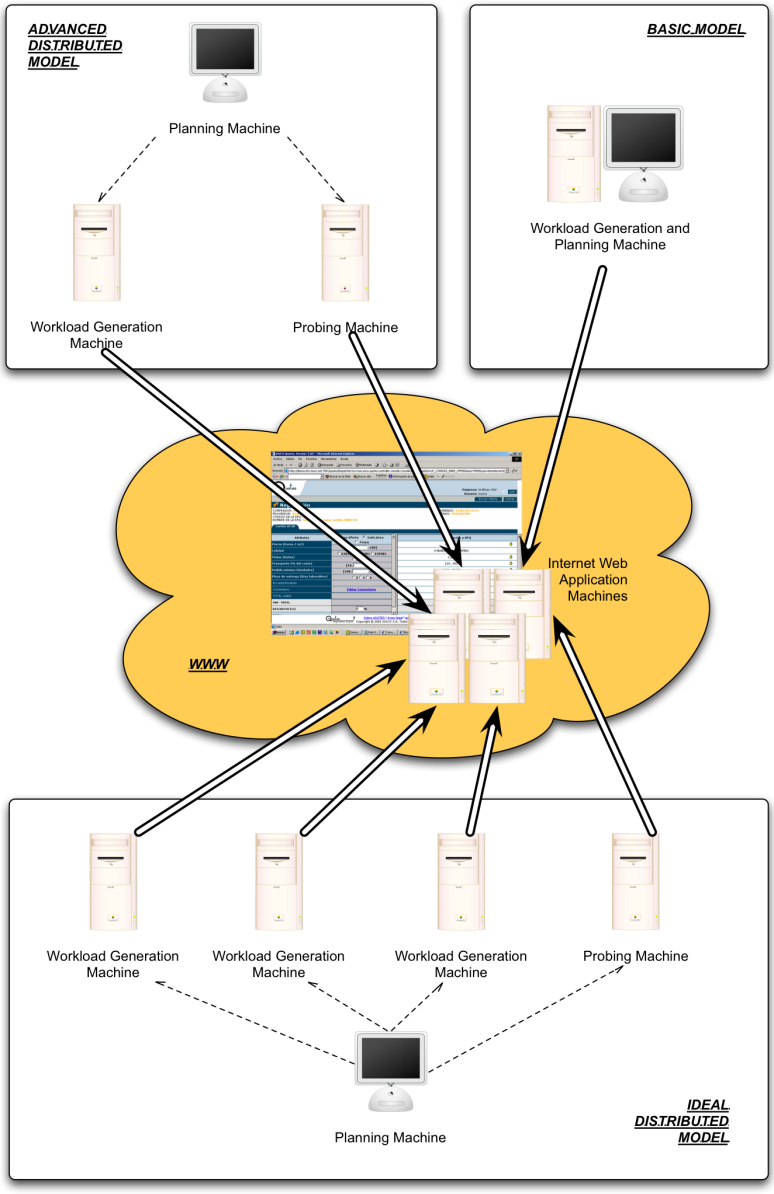


Figure 4.3: Distribution of workload generation

4.3 Architecture

The GUERNICA suite presents a distributed software architecture as depicted in the deployment diagram (UML 2.1) of Figure 4.4.

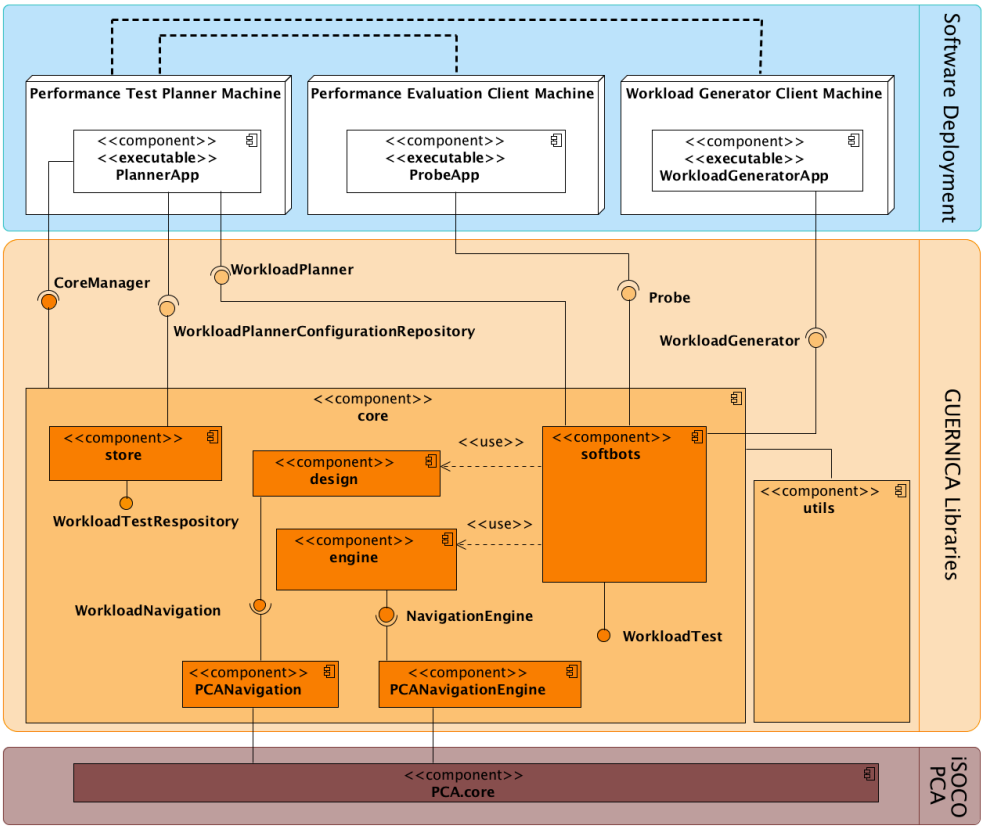


Figure 4.4: Architecture of GUERNICA

The three main applications (`WorkloadGeneratorApp`, `PlannerApp` and `ProbeApp`) have been programmed in `Java` using web services technology, so they run independently of the execution platform. Their business logics are provided by three interfaces of the `core` library: `WorkloadPlanner`, `Probe` and `WorkloadGenerator`.

The `core` is the main component of the architecture and carries out the workload generation process by using DWEB. Navigational plugins and workload tests are implemented by `WorkloadNavigation` and `WorkloadTest` interfaces, respectively. The `NavigationEngine` defines an API to reproduce the user's behavior; its configuration is described in terms of DWEB, and it is stored in a repository called `WorkloadTestRespository`. The engine can operate with any technology fulfilling

its API, but it currently supports Personal Content Aggregator (PCA) technology to implement the navigational plugin execution. To this end, the `PCANavigation` and the `PCANavigationEngine` classes have been provided.

The PCA technology, developed by iSOCO, offers: i) a scripting language that allows GUERNICA to easily define dynamic user's navigations for the current Web, and ii) an engine to carry out automatic executions for the navigational plugins.

Finally, the `util` package helps the main library, that can be accessed by using the `CoreManager` in a centralized way.

4.4 Main features

GUERNICA supports, totally or partially, the main features introduced in well-known workload generators proposed in the open literature (see Section 2.2.2), as well as the capability to represent the user's dynamism, as shown in Table 4.1.

FEATURE/CAPABILITY	GUERNICA
Analytical-Based Architecture	▲
Distributed Architecture	♦
Business-Based Architecture	▲
Client Parameterization	♦
Workload Types	♦
Functional Testing	▲
LAN and WAN	▲
Multi-platform	♦
Ease of Use	♦
Performance Reports	♦
Open Source	▲
User's Dynamism	♦

♦ Full support ▲ Partial support

Table 4.1: GUERNICA features

Regarding the totally supported features, the generator presents a distributed architecture based on web services. That is, GUERNICA allows to distribute the generation process among different nodes which emulate users working on different machines. Furthermore, the generator implements the DWEB model that offers important capabilities. For instance, it provides an analytical approach (user's roles) to characterize users' dynamic behaviors as well as their continuous changes (*Challenge II and III*). That is, GUERNICA can organize the workload in dynamic categories or types, each of them modeling a given user profile by using a workload test. The model also introduces some client variables to parameterize the user behavior (e.g.

the user's think time by means of a Gaussian distribution), which are provided by GUERNICA as a part of the scripting language. This language allows the generator to reproduce users' dynamic navigations (*Challenge I*). Table 4.2 summarizes how GUERNICA technology fulfills the challenges when representing user's dynamism.

	GUERNICA
Challenge I	Navigational plugins ⊛
Challenge II	Workload tests ⊙
Challenge III	

⊙ Analytical approach ⊛ Scripting

Table 4.2: Challenges of user's dynamism fulfilled by GUERNICA

Among other features supported by GUERNICA, one can observe its ease of use (e.g. CARENA to define user navigation), the ability to generate performance reports (e.g. 3D graphical representation), and the capability to model differences between LAN (where generators are usually run) and WAN (where applications are usually located). Moreover, the workload tests can be used as web application functional tests, and the generation process can be easily applied to different business architectures. Finally, it should be noted that a significant part of the code (the applications and the core packages that do not use PCA technology) has been written under an open source license.

4.5 Case study

The objective of this case study is not to present a detailed performance evaluation but to show how GUERNICA can be used in web performance studies.

The case study considers that users can behave as *searchers* or *surfers* [CPCP01] when navigating the Web. Searchers are users who start their navigations with a query in a search engine like Google. On the other hand, surfers prefer to navigate through the Web using its direct hyperlinks. With the aim of illustrating these two behaviors, different navigations looking for some information related with the *US president election in 2012* are chosen.

Figure 4.5 presents an automaton that defines the surfer and searcher users' behaviors, and the continuous changes of behavior when they are looking for some information about the candidates. The automaton shows how a typical user can proceed: i) to search the name of a candidate in the searcher engine, or ii) to navigate through one of the most important news sites (the CNN website). The automaton states the same probability (50%) to start a user's navigation searching in Google the democratic candidate name (Obama) or the republican (Romney). Then, the user can search the other candidate in Google (25%), can browse the CNN website to look for some information about the same candidate (50%), or can finish the navigation

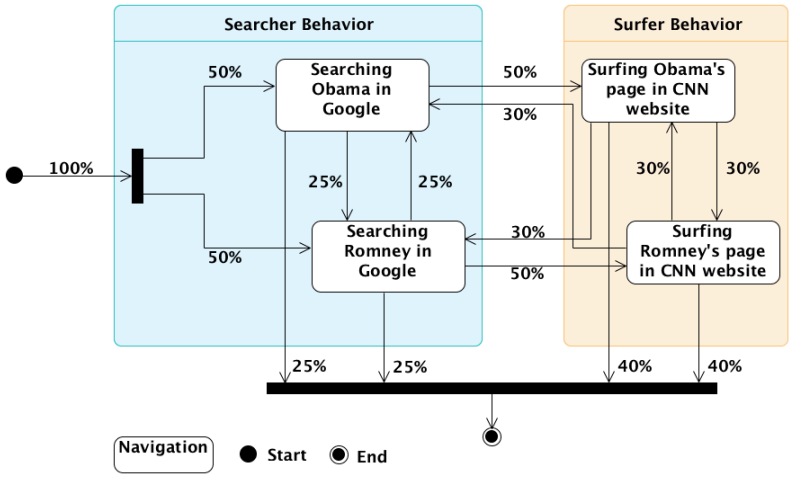


Figure 4.5: Web searcher and surfer user's behaviors

(25% of probability). When the user has found the information in the CNN website, he can navigate to the other candidate page in the site (30% of probability), or he can change his behavior by executing a new search looking for some information about the other candidate (30%). The third alternative is to finish the navigation (40%).

For illustrative purposes, Figure 4.6 specifies the searcher behavior by modeling a simple search in Google. As observed, once the main page of Google (`www.google.es`) is accessed, the user spends some time thinking (think time - 1000 ms) until he asks Google for some information (`www.google.es/search`). After that, Google returns the results page (`SEARCH_RESULTS`) and a new user's think time (given by a Gaussian distribution as 3500 ms of average with 1500 ms standard deviation) is provided. Then, the user has two options represented by two branches in the graph; if the searcher engine provides results for the *candidate name* (left branch) the user will access the candidate site (first site provided); otherwise, the user will finish the process (right branch). If the user accesses the candidate home page (`www.first_result.home`), he spends time thinking about the returned contents before finishing the navigation (black dot).

GUERNICA was implemented by using a model language based on XML labels to define *workload tests*, *navigational plugins* and *workload distribution*. Listing 4.1 shows the XML file that implements as a workload test the automaton of Figure 4.5. On the other hand, Listing 4.2 presents the navigational plugin for a simple search in Google (Figure 4.6). The `PCA-Plugin` is the XML file that defines, by using the PCA technology, the user's navigation.

Figure 4.6: A simple search in Google

Listing 4.1: Workload test

```
<?xml version="1.0" encoding="UTF-8"?>
<WorkloadTest id="test_searcher_vs_surfer">
  <UsersNumber>2</UsersNumber>
  <NavigationGraph>
    <InitialNavigations>
      <InitialNavigation id="google_obama" prob="0.50"/>
      <InitialNavigation id="google_romney" prob="0.50"/>
    </InitialNavigations>
    <NavigationTransitions>
      <NavigationTransition from="google_obama" to="cnn_obama" prob="0.50"/>
      <NavigationTransition from="google_obama" to="google_romney"  prob="0.25"/>
      <NavigationTransition from="google_romney" to="cnn_romney" prob="0.50"/>
      <NavigationTransition from="google_romney" to="google_obama" prob="0.25"/>
      <NavigationTransition from="cnn_obama" to="google_romney"  prob="0.30"/>
      <NavigationTransition from="cnn_obama" to="cnn_romney" prob="0.30"/>
      <NavigationTransition from="cnn_romney" to="google_obama" prob="0.30"/>
      <NavigationTransition from="cnn_romney" to="cnn_obama" prob="0.30"/>
    </NavigationTransitions>
  </NavigationGraph>
</WorkloadTest>
```

Listing 4.2: Searching Obama in Google for the US President Election 2012

```
<?xml version="1.0" encoding="UTF-8"?>

<Navigation id="google_obama">
  <InputData>
    <Param name="phrase" value="Obama"/>
  </InputData>
  <ExecutionCode>
      <PCA-Plugin name="google.pca.xml"/>
  </ExecutionCode>
  <StatisticsConfiguration>
    <StatisticAttribute name="NavigationTime"/>
    <StatisticAttribute name="ExecutionTime"/>
    <StatisticAttribute name="HttpRoute">
      <StatisticAttribute name="URL"/>
      <StatisticAttribute name="HttpMethod"/>
      <StatisticAttribute name="Stablished"/>
      <StatisticAttribute name="StablishmentTime"/>
      <StatisticAttribute name="TransferTime"/>
      <StatisticAttribute name="ThinkUserTime"/>
      <StatisticAttribute name="ContentSize"/>
    </StatisticAttribute>
  </StatisticsConfiguration>
</Navigation>
```

Finally, Listing 4.3 shows the XML file that distributes the workload generation process. This file configures GUERNICA in the basic generation approach with only a workload generator process (`generator-1`) in a single machine.

Listing 4.3: Workload distribution for the US President Election 2012

```
<?xml version="1.0" encoding="UTF-8"?>
<WorkloadPlannerConfiguration>
   <PlannerIdentifier>election-2004</PlannerIdentifier>
      <WorkloadGenerators>
         <WorkloadGenerator>
            <Id>generator-1</Id>
         </WorkloadGenerator>
      </WorkloadGenerators>
      <WorkloadTestAssignations>
         <WorkloadtestAssignation testId="test_searcher_vs_surfer"
                                  generatorId="generator-1"/>
      </WorkloadTestAssignations>
</WorkloadPlannerConfiguration>
```

Once the workload test has been run, we obtain the plan of conducted navigations and their HTTP requests, each one having different outcome. Listing 4.4 illustrates an example showing the results of an execution for two users. As observed, a set of global statistics is obtained, such as the total execution time of the experiment or the navigation time. For each HTTP request, the test execution produces different statistics, such as the `GET` or `POST` method, the time for establishing the connection, the transfer time, the user think time, the content size, accessed URL, or the successfulness when making the connection (`Established`).

Listing 4.4: Testing results for the US President Election 2012

```
<?xml version="1.0" encoding="UTF-8"?>

<WorkloadTestStatistics id="..." testId="test_searcher_vs_surfer"
  <UserNavigations>
    <NavigationStatistic navigationId="google_obama" date="...">
      <NavigationTime>17065</NavigationTime>
      <ExecutionTime>18453</ExecutionTime>
      <HttpRoute>
        <HttpRouteElement>
          <URL>http://www.google.es</URL>
          <HttpMethod>GET</HttpMethod>
          ...
        </HttpRouteElement>
        <HttpRouteElement>
          <URL>http://www.google.es/search?q=Obama</URL>
          <HttpMethod>GET</HttpMethod>
          <Established>true</Established>
          <EstablishmentTime>116</EstablishmentTime>
          <TransferTime>1084</TransferTime>
          <ThinkTime>5622</ThinkTime>
          <ContentSize>19896</ContentSize>
        </HttpRouteElement>
        <HttpRouteElement>
          ...
          <TransferTime>257</TransferTime>
          <ThinkTime>5000</ThinkTime>
          <ContentSize>15545</ContentSize>
        </HttpRouteElement>
      </HttpRoute>
    </NavigationStatistic>

    <NavigationStatistic navigationId="cnn_obama" date="...">
        ...
    </NavigationStatistic>

    <NavigationStatistic navigationId="google_romney" date="...">
        ...
    </NavigationStatistic>

    <NavigationStatistic navigationId="cnn_romney" date="...">
        ...
    </NavigationStatistic>
  </UserNavigations>
  <UserNavigations>
    <NavigationStatistic navigationId="google_obama" date="...">
        ...
    </NavigationStatistic>
  </UserNavigations>
</WorkloadTestStatistics>
```

4.6 Summary

This chapter has presented GUERNICA, a new web workload generator based on DWEB model with the aim of reproducing dynamic users workload in a more accurate and appropriate way than traditional approaches to workload generation.

GUERNICA implements the main DWEB concepts with the aim of adopting the model: i) the user's navigation is incorporated with a scripting approach named *navigational plugin*, and the *workload test* carries out the concept of user's roles from an analytical perspective. Furthermore, the generator represents the physical distribu-

tion of users in the Web by providing a distributed architecture, which permits to set up different approaches (basic, advanced, and ideal) when generating web workload.

Additionally, GUERNICA supports, totally or partially, the main features introduced in well-known workload generators proposed in the open literature. These features work together under a five-phases methodology allowing GUERNICA to carry out performance and functional evaluation of web application in the current Web.

The main results of this chapter were published in [WOSP'05, COMCOMJ'09]. Further details about GUERNICA design and features can be found in the context of the GENERICA project as a technical reports [GENERICA'04b, GENERICA'04c]. Furthermore, more information about CARENA and SemViz can be obtained in [GENERICA'04d, GENERICA'04e].

CHAPTER 5

GUERNICA validation: a new testbed for web performance evaluation

Before widely using GUERNICA in web performance studies, we need to validate it against a traditional approach to workload generation. To this end, we devise a new testbed with the ability of reproducing different types of workloads. After the validation process, the testbed will be used to analyze the effect of applying dynamic workloads on the web performance metrics, instead of traditional workloads. This chapter introduces the testbed design and describes the validation process.

The new testbed has to accomplish three main goals. First, it must define and reproduce traditional web workloads by using a parameterized and extensible architecture that allows us to integrate the workload generation process of GUERNICA. Second, it must be able to provide client and server metrics with the aim of being used for web performance evaluation studies. Finally, it should be representative of web transactional systems that have been established in recent years, such as e-commerce websites, blogs or OSNs.

Among the evaluated benchmarks in Section 2.2, TPC-W is the best candidate to provide an appropriate testbed for our purposes, because it satisfies the previous goals and also considers user's behavior on workload generation although in a partial way. Consequently, with the aim of validating GUERNICA against the traditional approach to workload generation of TPC-W, we deploy a new testbed for web performance evaluation by integrating our generator into the benchmark.

The remainder of this chapter is organized as follows. Section 5.1 presents the main features of the TPC-W implementation that is adopted as a framework of the testbed. After that, in Section 5.2, we show the devised architecture of integrating GUERNICA into TPC-W. Section 5.3 and Section 5.4 describe the experimental setup and the main measured performance metrics in the validation process, respectively. This process is discussed in Section 5.5. Finally, we draw some concluding remarks in Section 5.6.

5.1 The TPC-W framework

As mentioned in Section 2.2.1.6, the TPC-W is a transactional web benchmark that models an on-line bookstore environment. The benchmark specification [Tra02b] defines a full website map for the on-line bookstore that consists of 14 unique pages and their navigation transitions. Figure 5.1 depicts a reduced TPC-W website map, where pages with related functionality are included in the same group: *ordering, shopping, browsing, admin* and *search*. Navigation hyperlinks among them are also indicated.

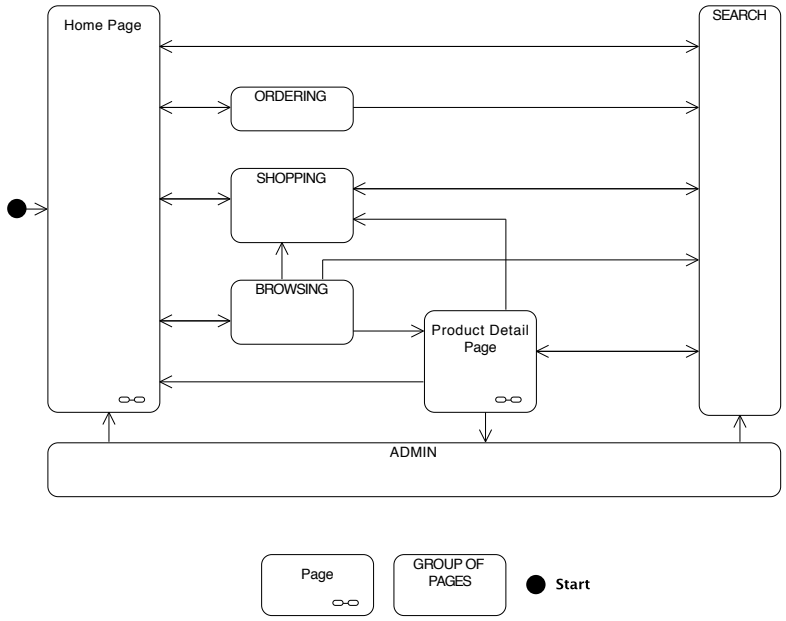


Figure 5.1: TPC-W reduced website map

The *search* group provides a book searcher by using the Search page to request the query and the Search Results page to show a list of results. The *browsing* group embraces the Best-sellers and the New Products pages, which arrange the bookstore catalog according to the sales and the publication date, respectively. The *shopping* group is the largest set of pages and provides i) sale functionality by managing the shopping cart (ShoppingCart page), ii) the buy request and its confirmation (Buy Request page and Buy Confirm page, respectively), and iii) the pay through a secured navigation (Customer Registration page). The *ordering* group includes a set of pages that allows checking the order status (Order Inquiry page and Order Display page). The *admin* group manages the catalog of books (using Admin Request and Product Updated pages). Finally, the most referred pages (Home page and Product Detail page) are also included. *Search* and *shopping* groups implement the most interactive

and personalized functionality in the website, so they are potentially interesting as dynamic user workload.

The benchmark provides a standard environment that is independent of the underlying technology, designed architecture and deployed infrastructure. A TPC-W `Java` implementation developed by the UW-Madison Computer Architecture Group [CRML01] was selected as framework of our testbed. As shown in Figure 5.2, the architecture client side is a `Java` console application that provides two interfaces for workload generation; an `Emulated Browser (EB)` and a factory (`EBFactory`) to create, configure and manage it. These interfaces allow us to define new processes for workload generation. The server side was developed as a `Java` web application made of a set of `Servlets`. Each `Servlet` resolves client requests by looking in the database information.

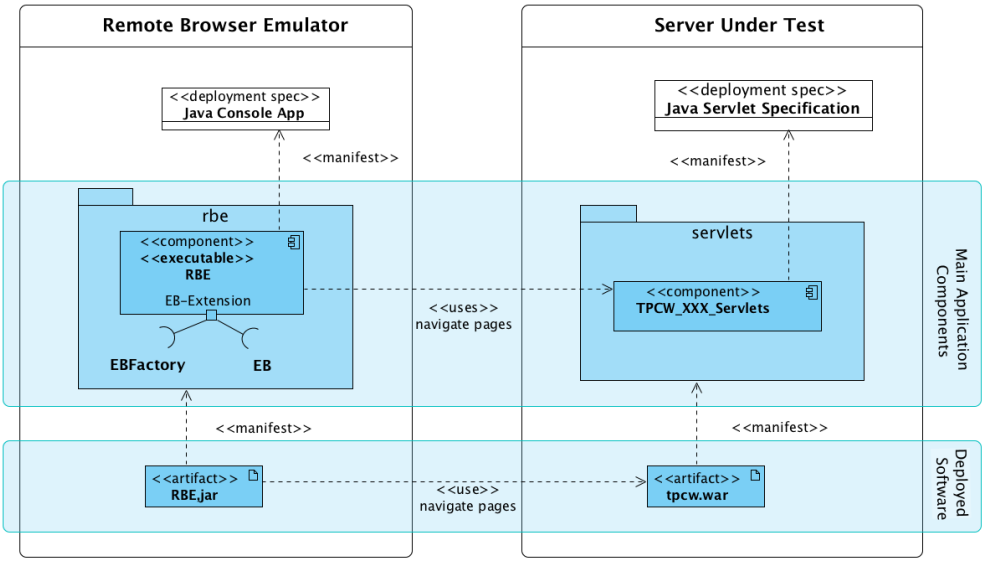


Figure 5.2: Main software components of TPC-W `Java` implementation

5.2 Testbed architecture

The architecture of integrating GUERNICA into TPC-W is organized in three main layers as depicted in Figure 5.3 and detailed below.

- The top layer is defined at the client side of TPC-W and supplies the two interfaces related to the workload generation process (`EB`, `EBFactory`), as introduced in Section 5.1.

- The bottom layer is related to the process of workload generation in GUERNICA, detailed in Chapter 4.

- Finally, the intermediate layer defines the integration between GUERNICA and TPC-W. This integration is provided by an independent `Java` library named `TGI`. This library implements a new type of EB (`DwebEB`) that uses the `GUERNICA core` to reproduce user's dynamic behavior in the workload generation process. In order to simplify the new EB, a workload generation engine (`DwebExecutorEngine`) is implemented to carry out the generation process. A browser factory (`DwebEBFactory`) is also developed to manage the creation and configuration of the new EB.

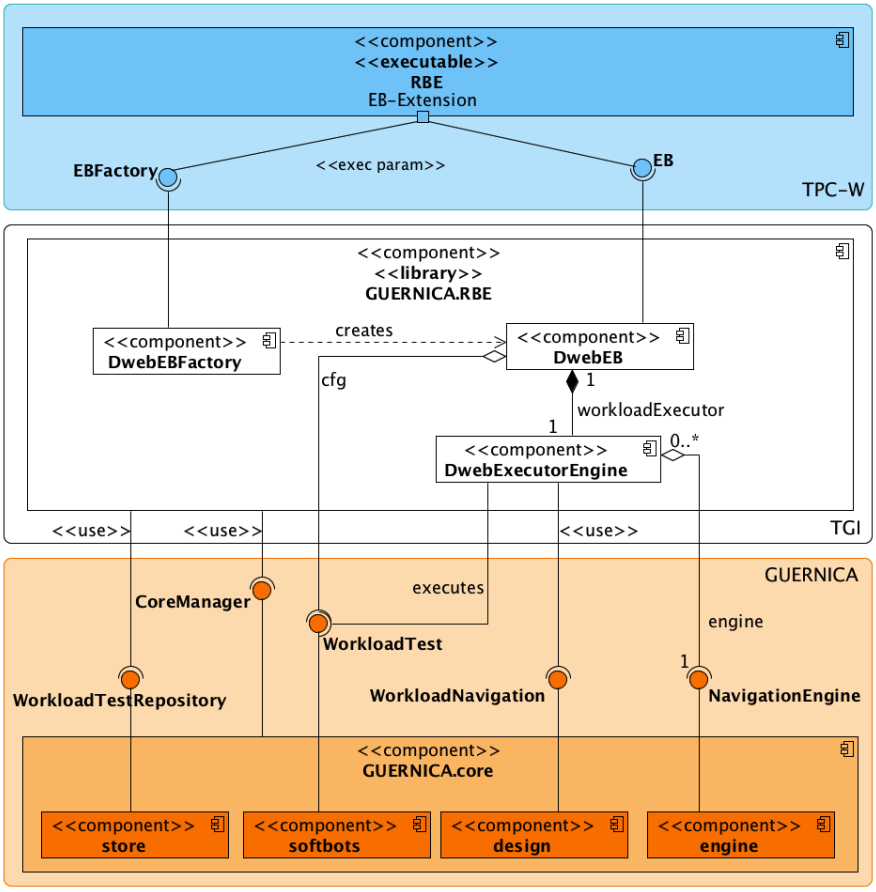


Figure 5.3: Testbed architecture

5.3 Experimental setup

The experimental setup used in this dissertation is a typical two-tier configuration consisting of an Ubuntu Linux Server back-end tier and an Ubuntu Linux client front-end tier. The back-end runs the on-line bookstore, which core is a `Java` web application (`TPC-W web app`) deployed on the Tomcat web application server. Requests to static content, such as images, are served by the Apache web server, which redirects requests for dynamic content to Tomcat. `TPC-W web app` generates the dynamic content by fetching data from the MySQL database. On the other hand, the front-end tier is able to generate the workload either using conventional or dynamic models. Both web application and workload generators are run on the SUN Java Runtime Environment 5.0 (`JRE 5.0`). Figure 5.4 illustrates the hardware/software platform of the experimental setup used in this dissertation.

Given the multi-tier configuration of this environment, system parameters (both in the server and in the workload generators) have been properly tuned to avoid that middleware and infrastructure bottlenecks interfere in the results. The on-line bookstore has been configured with 300 EBs and a large number of items (100,000 books) that forced us to balance accesses in the database (e.g. the pool connection size), static content service by Apache (e.g. the number of processes to attend HTTP requests), or dynamic content service by Tomcat (e.g. the number of threads providing dynamic contents). For each experiment, the measurements were performed for several runs with a 20-minute collecting-data phase after a 15-minute warm-up phase.

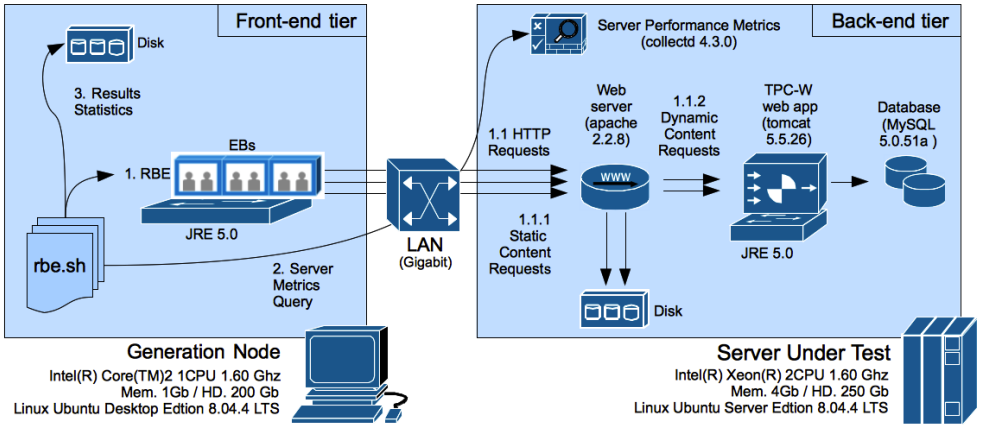


Figure 5.4: Experimental setup

5.4 Performance metrics

Table 5.1 summarizes the performance metrics available in the experimental setup. The main metrics measured on the client side are the *total number of requests per page* and the *response time*, which is expressed as *Web Interaction Response Time (WIRT)*. On the server side, the platform collects the server performance statistics required by the TPC-W specification (*CPU and memory utilization, database I/O activity, system I/O activity, and web server statistics*) as well as other optional statistics. These metrics allow a better understanding of the system behavior under test and permit to check the techniques used to improve performance when applying a dynamic workload. The collected metrics can be classified in two main groups: metrics related with the usage of main hardware resources, and performance metrics for the software components of the back-end. For evaluation purposes, we used a middleware named `collectd` [For12] that collects system performance statistics periodically.

5.5 GUERNICA validation

This section validates GUERNICA by using the devised testbed to compare our workload generation approach against the TPC-W approach. According to the TPC-W specification three scenarios are defined when characterizing the web workload: shopping, browsing, and ordering. The shopping scenario presents intensive browsing and ordering activities while the browsing and ordering scenarios reduce ordering and browsing activities, respectively. TPC-W describes these scenarios as three different full CBMGs.

Regarding the validation test, we contrast both workload characterization approximations (i.e. CBMG and DWEB) for each scenario. Figure 5.5 depicts the shopping scenario workload as an illustrative example. Note that we are able to model the same workload by using only the navigation concept of DWEB, and disabling all the parameters used to include user dynamism in the workload characterization. The validation test considers 50 EBs because the `Java` implementation of the TPC-W generator presents some limitations in the workload generation process. The measurements were performed for 50 runs and obtaining confidence intervals with a 99% confidence level.

For illustrative purposes, this section presents results of a subset of the most significative metrics when running TPC-W for the three scenarios defined by CBMG and DWEB.

Figure 5.6 and Figure 5.7 depicts client and server performance metrics for the shopping scenario, respectively.

Resource			Metric	Description/Formula
Client Side			Response Time (WIRT)	WIRT is defined by TPC-W as $t2 - t1$, where $t1$ is the time measured at the Emulated Browsers when the first byte of the first HTTP request of the web interaction is sent by the browser to the server, and $t2$ is the time when the last byte of the last HTTP response that completes the web interaction is received.
			Average Response Time ($\overline{WIRT}$)	$\overline{WIRT} = \frac{\sum_{i \in Pages} WIRT_i * Req_i}{\sum_{i \in Pages} Req_i}$
			Req_{page}	Requests per Page (Req_{page}) are the total number of connections for a page requested by Emulated Browsers and accepted by the server.
Server Side	Hardware	CPU	U_{CPU}	Metrics for hardware resources include utilization for all of them, and throughput for the disk and the network.
		Memory	U_{memory}	
		Disk	U_{disk}, X_{disk}	
		Network	U_{net}, X_{net}	
	Software	Apache	$X_{apache}, CPU_{apache}, MEM_{apache}$	Performance metrics for software components of server include: their throughput, the CPU and memory consumption, the number of processes or threads, etc
		Tomcat	$X_{tomcat}, CPU_{tomcat}, MEM_{tomcat}$	
		MySQL	$X_{mysql}, CPU_{mysql}, MEM_{mysql}$	

Table 5.1: Performance metrics classification according to the evaluated resource

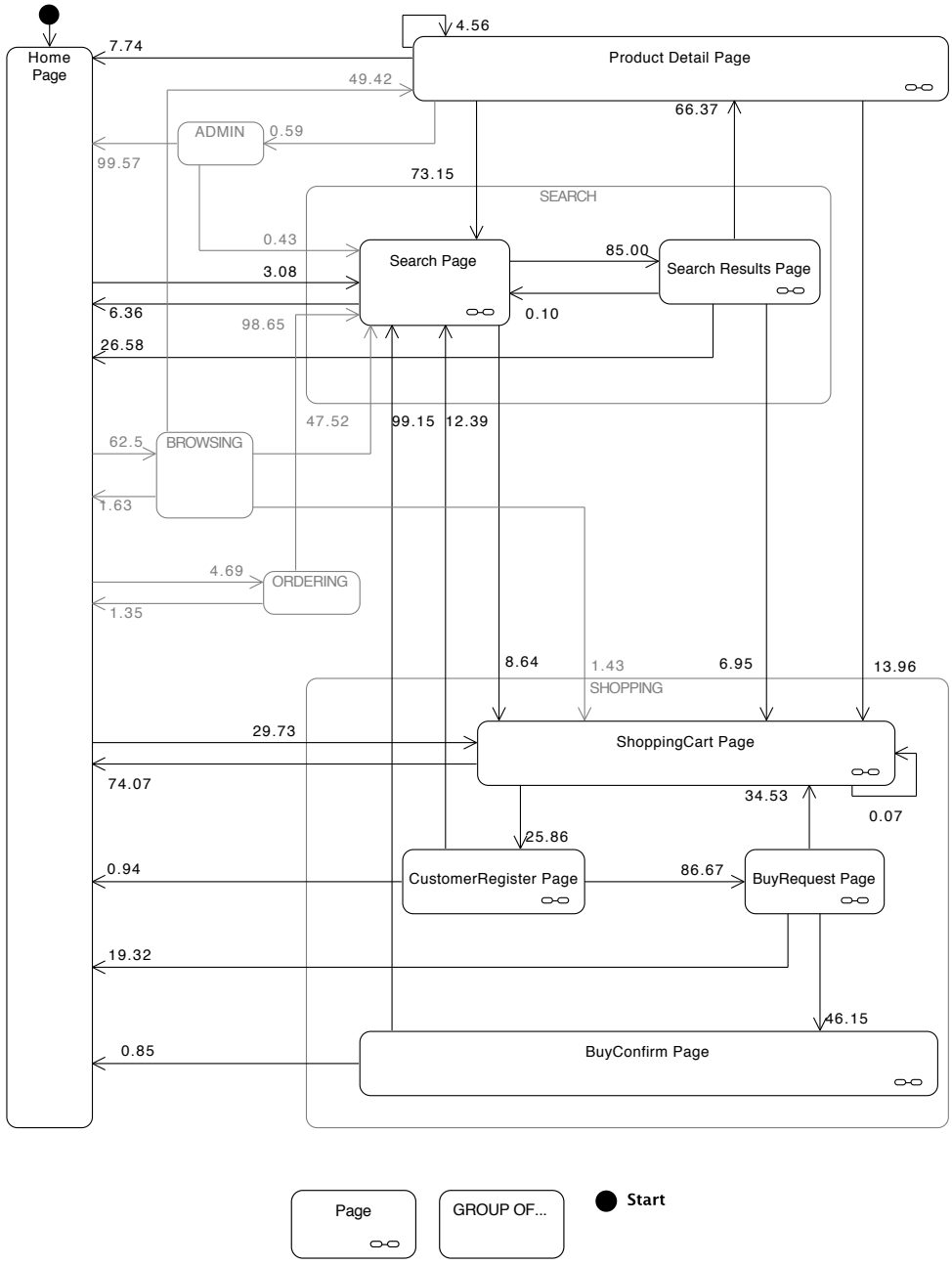
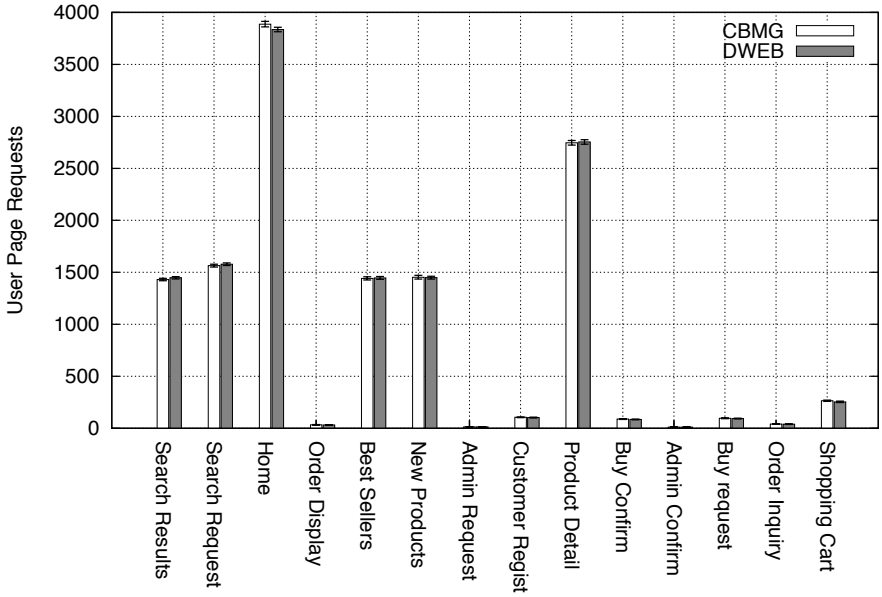
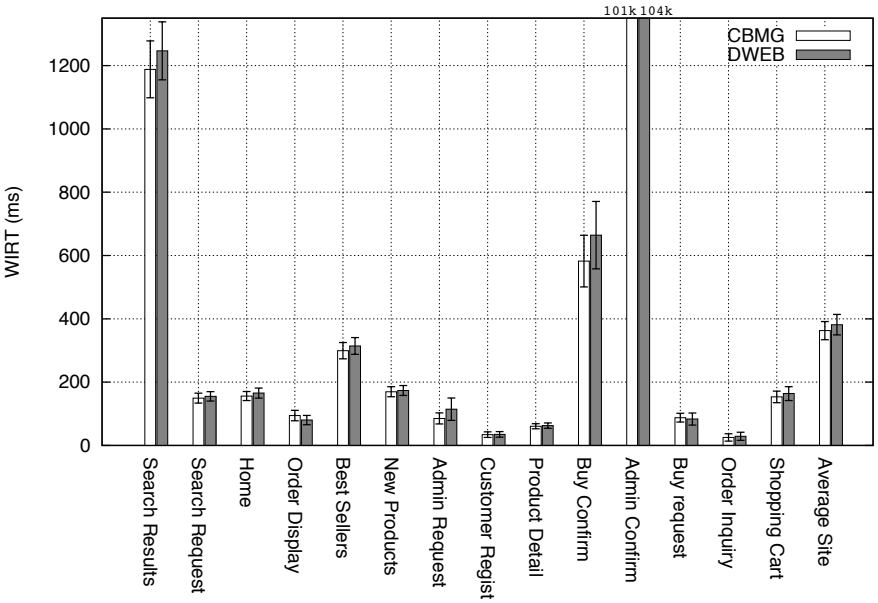


Figure 5.5: CBMG model for shopping scenario in GUERNICA validation

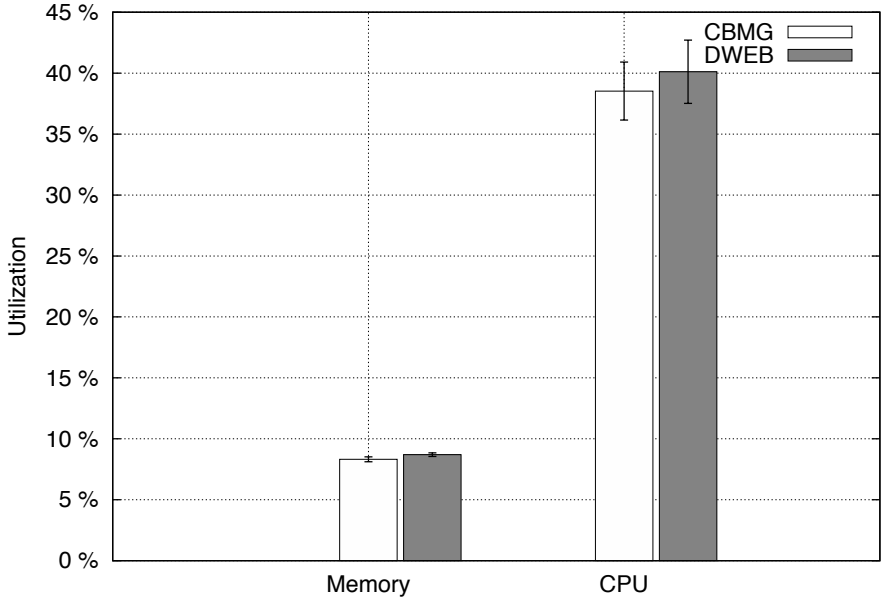


(a) User page requests

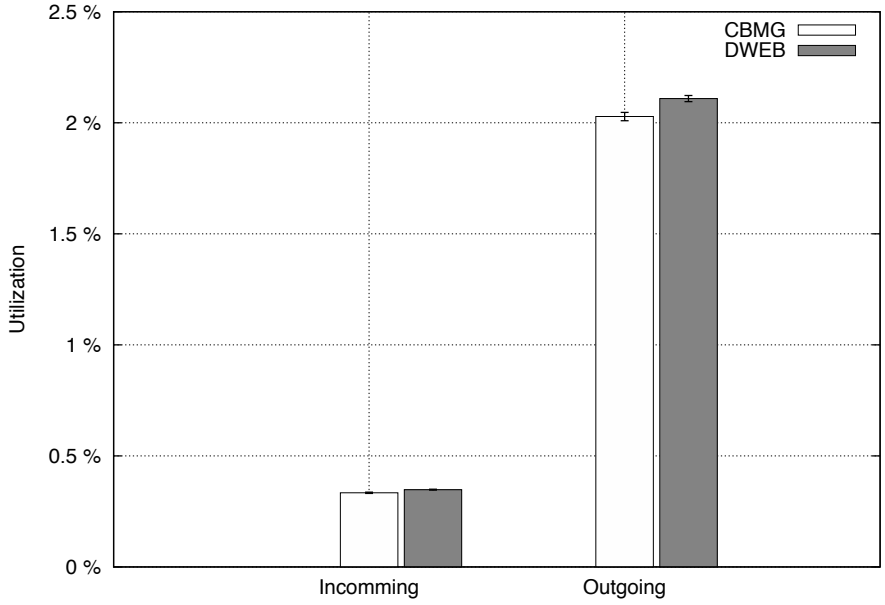


(b) WIRT

Figure 5.6: Client metrics obtained for the shopping scenario in GUERNICA validation

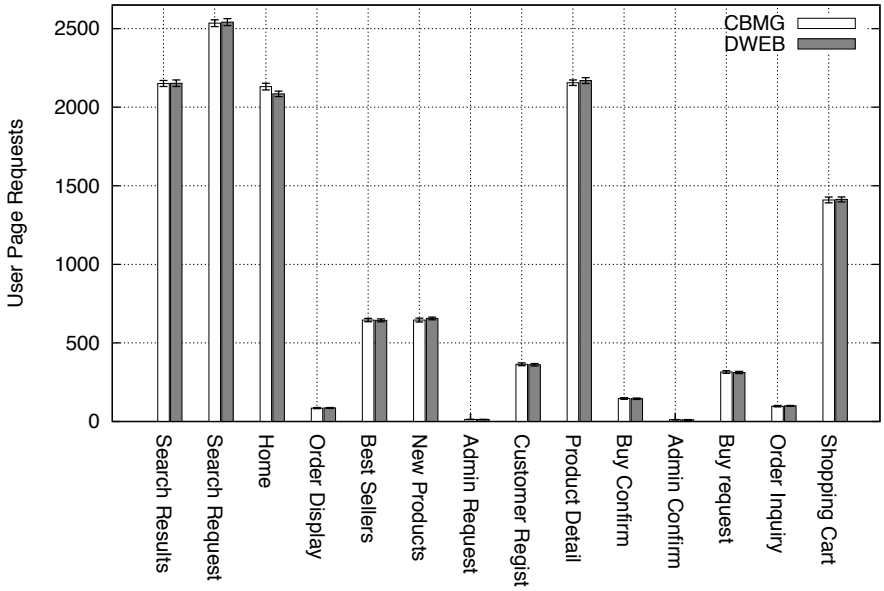


(a) Server memory and CPU utilization

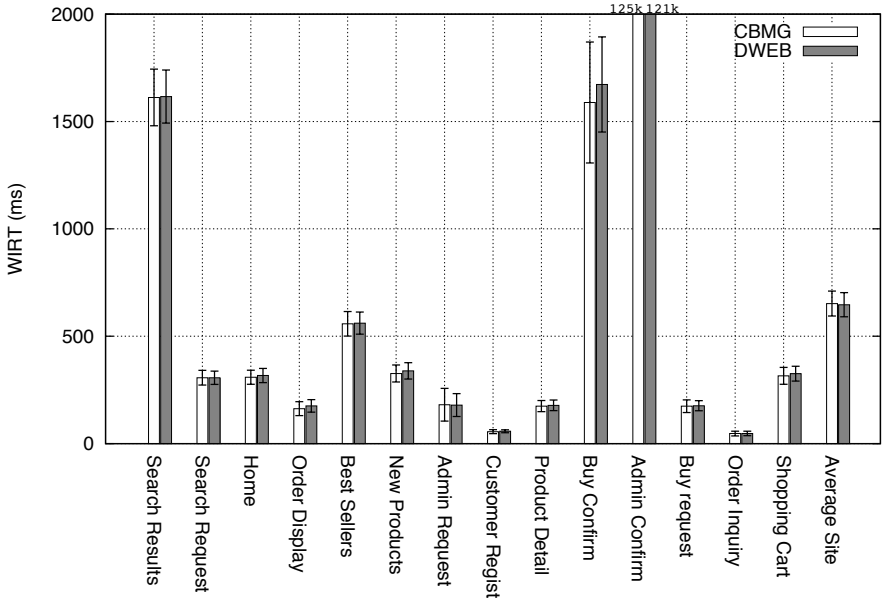


(b) Server network utilization

Figure 5.7: Server metrics obtained for the shopping scenario in GUERNICA validation

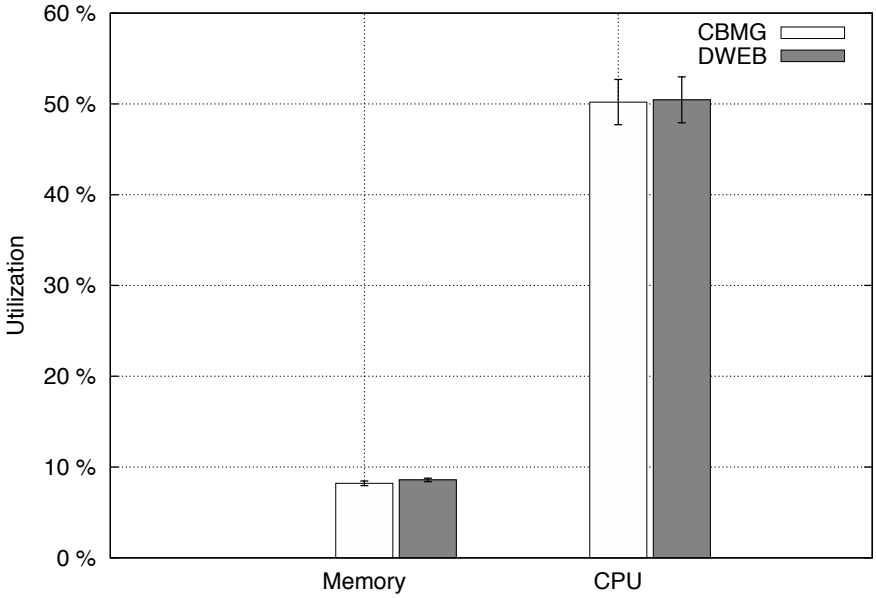


(a) User page requests

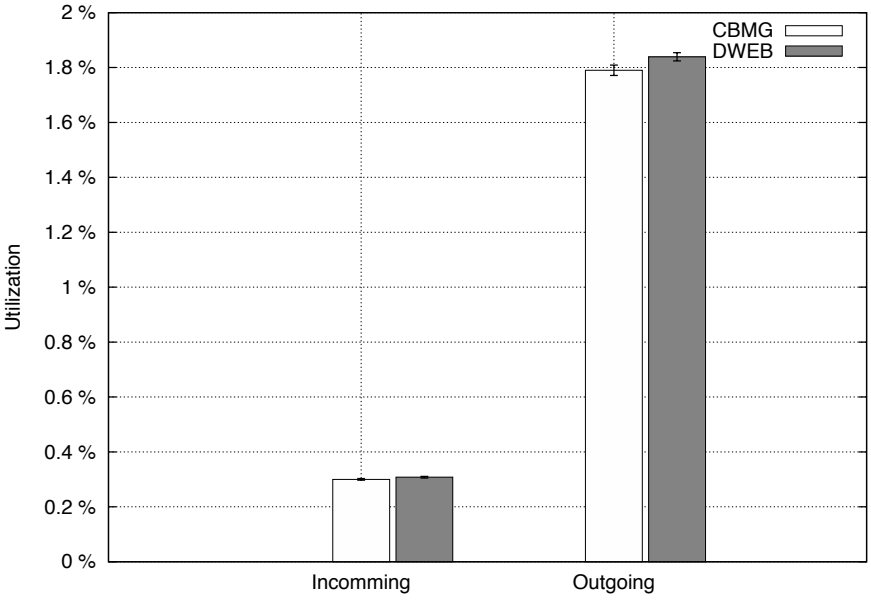


(b) WIRT

Figure 5.8: Client metrics obtained for the browsing scenario in GUERNICA validation

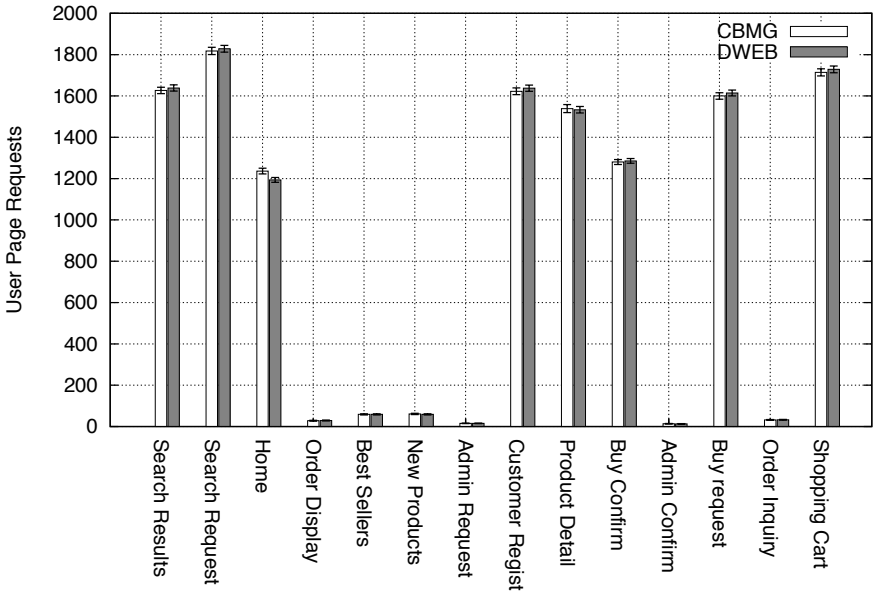


(a) Server memory and CPU utilization

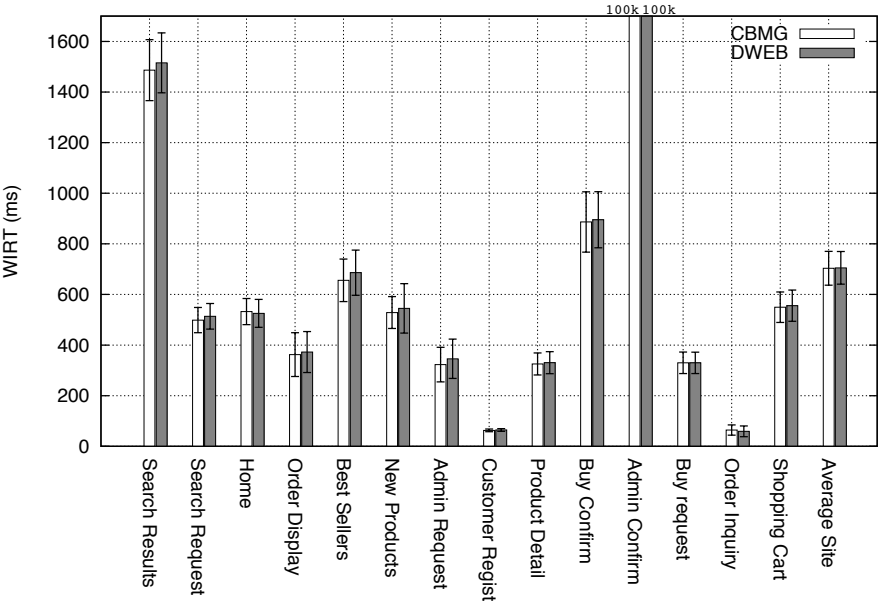


(b) Server network utilization

Figure 5.9: Server metrics obtained for the browsing scenario in GUERNICA validation

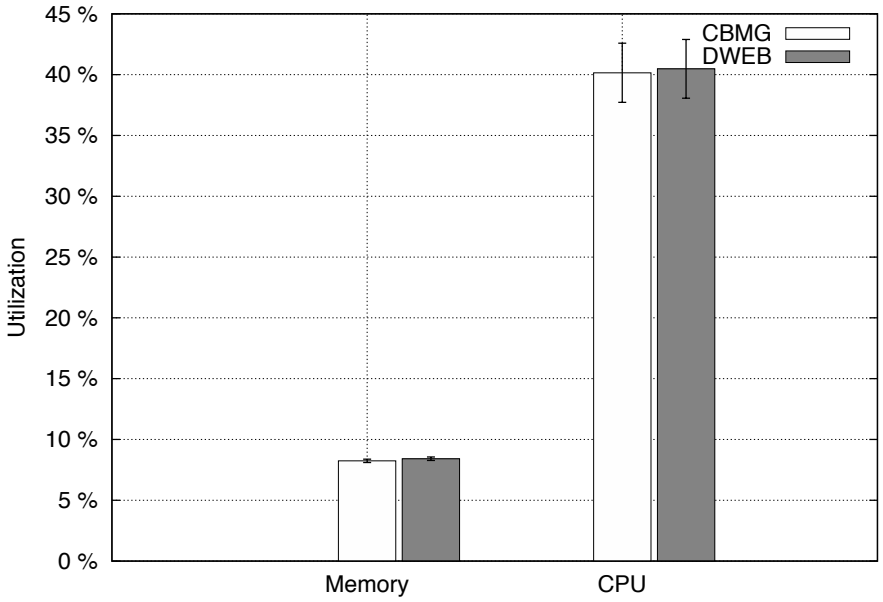


(a) User page requests

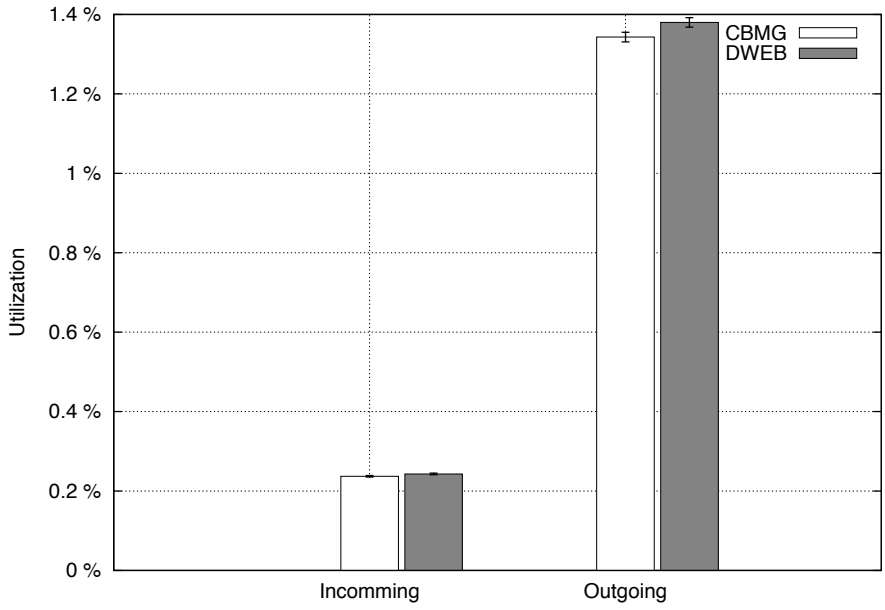


(b) WIRT

Figure 5.10: Client metrics obtained for the ordering scenario in GUERNICA validation



(a) Server memory and CPU utilization



(b) Server network utilization

Figure 5.11: Server metrics obtained for the ordering scenario in GUERNICA validation

As shown in Figure 5.6a, both approximations generate a similar number of page requests. Figure 5.6b shows that the DWEB response time is, on average, 5% higher than that of CBMG, because some pages (e.g. Search Results or Buy Confirm) present very wide confidence intervals in this scenario. However, this difference does not affect the server performance metrics since, as observed in Figure 5.7, the highest utilization is below 40% in both cases. The utilization for CPU and memory is rather low and similar in both cases (see Figure 5.7a). Incoming and outgoing traffics do not increase network utilization more than 2% in the studied workloads (see Figure 5.7b). Finally, the disk utilization is lower than 0.2% in both workloads (not shown in the figures).

Browsing scenario results are illustrated in Figure 5.8 and Figure 5.9. Both workloads generate a similar number of page requests and response time as shown in Figure 5.8a and Figure 5.8b, respectively. On the other hand, the server is characterized by a middle level of stress in both cases. CPU utilization is by 50%, while the memory, network and disk utilizations are low, as observed in Figures 5.9a and Figure 5.9b, respectively.

Figure 5.10 and Figure 5.11 depict the results for the ordering scenario. The former shows that both workloads present similar levels of client metrics. The latter presents how the highest server's utilization is lower than 40% in both cases.

Finally, we can conclude that the DWEB model and GUERNICA can generate accurate traditional workloads for web performance studies based on TPC-W. Moreover, due to their designs, our new testbed can be used to generate web workloads with user's dynamic behavior.

5.6 Summary

This chapter has discussed the validation of GUERNICA against a traditional approach to workload generation. With this aim, we have developed a new testbed for performance evaluation with the ability of generating dynamic user workload by integrating GUERNICA into TPC-W.

We have validated our approach by contrasting the new testbed main functionalities and behavior against TPC-W, and found that both implementations present similar behavior in traditional web workloads. Moreover, our approach represents a more valuable alternative because DWEB and GUERNICA are able to model and reproduce user's dynamism on workload characterization in an accurate and appropriate way, respectively. The devised testbed will help us to prove it.

In [CLEIej'12], a summary of both testbed and validation introduced in this chapter was presented.

CHAPTER 6

The impact of dynamic user workloads on web performance

This chapter proves that the web user's dynamic behavior is a crucial issue that must be addressed in web performance studies in order to accurately estimate system performance indexes. To this end, using the testbed presented in Chapter 5, we analyze and measure for the first time, to the best of our knowledge, the effect of considering different levels of dynamic workload on web performance evaluation, instead of traditional workloads.

Results show that CPU utilization can increase as large as 30% with dynamic user workloads. These more realistic workloads show that processor utilization is not uniformly balanced along time, but overloaded peaks rise when considering user's dynamic behavior. As a consequence, the probability of a long response time is higher, and the number of user's abandonments can increase (up to 40%).

The remainder of this chapter is organized as follows. Section 6.1 presents the dynamic workloads proposals and how they can be modeled by the approaches under study. Then, Section 6.2 shows the effect of the different workloads on the web system performance. Finally, we draw some concluding remarks in Section 6.3.

6.1 Workload design

This section presents the experimental workloads used in the study. We would like to emphasize that the aim of this work is not to present a detailed dynamic workload based on current user's behavior, but to explore how typical web performance metrics are affected by introducing different degrees of dynamism.

This study assumes CBMG model to define traditional web workload, and DWEB model to introduce different levels of user's dynamism on workload characterization. First, Section 6.1.1 focuses only on user's navigation in order to represent dynamism on workload characterization by using the *user's navigation concept*. Second, Section

6.1.2 introduces a more realistic dynamic workload that also models changes on the user's roles by using the *user's roles concept*.

6.1.1 Considering dynamism on user's navigations

In a first step, a dynamic workload (DWEB workload I - DW1) is defined with the aim of introducing user's dynamic navigations on workload characterization. For this purpose, we assume a common scenario of e-commerce where the main objective is to avoid the defection of customers. A large percentage of new customers - more than 60% in some sectors - defect before their third anniversary with an e-commerce website [RS00]. Consequently, these websites care deeply about customer retention and consider loyalty vital to the success of their on-line operations.

For the studied scenario, regarding customer retention in the on-line bookstore, we define a loyalty promotion consisting of a general discount only for those customers who buy at least once a month. The promotion introduces a new behavior with four cases of dynamism as summarized in Table 6.1.

Case	**Description**
1	If customers do not remember their last order status, they will check them by navigating into the *ordering* group of pages.
2	Because the customer has to buy at least once a month to keep the discount, a buying session must finish with a payment when he has not bought anything during that month.
3	An experienced customer only buys a book when its cost is 25% cheaper than in other markets.
4	The higher the number of provided search results, the longer the time that a user takes to read and think about them.

Table 6.1: Cases of dynamism in the loyalty promotion behavior

DWEB allows the modeling of these cases of dynamism, which cannot be represented with this level of accuracy using traditional approaches such as CBMG.

Figure 5.5, which was introduced in Chapter 5 as the shopping scenario for the on-line bookstore, depicts the workload described using the traditional approach (CBMG workload). On the other hand, Figure 6.1 shows the workload characterization using DWEB (DW1 workload). Both graphs focus on *searching* and *shopping* groups of the TPC-W website, and highlight their main pages and transitions.

CBMG workload considers the *think time* based on the TPC-W specification [Tra02b]. This benchmark defines the user think time (TT) as $TT = T2 - T1$, where $T1$ is the time measured at the emulated browser when the last byte of the last web interaction is received from the server, and $T2$ is the time measured when

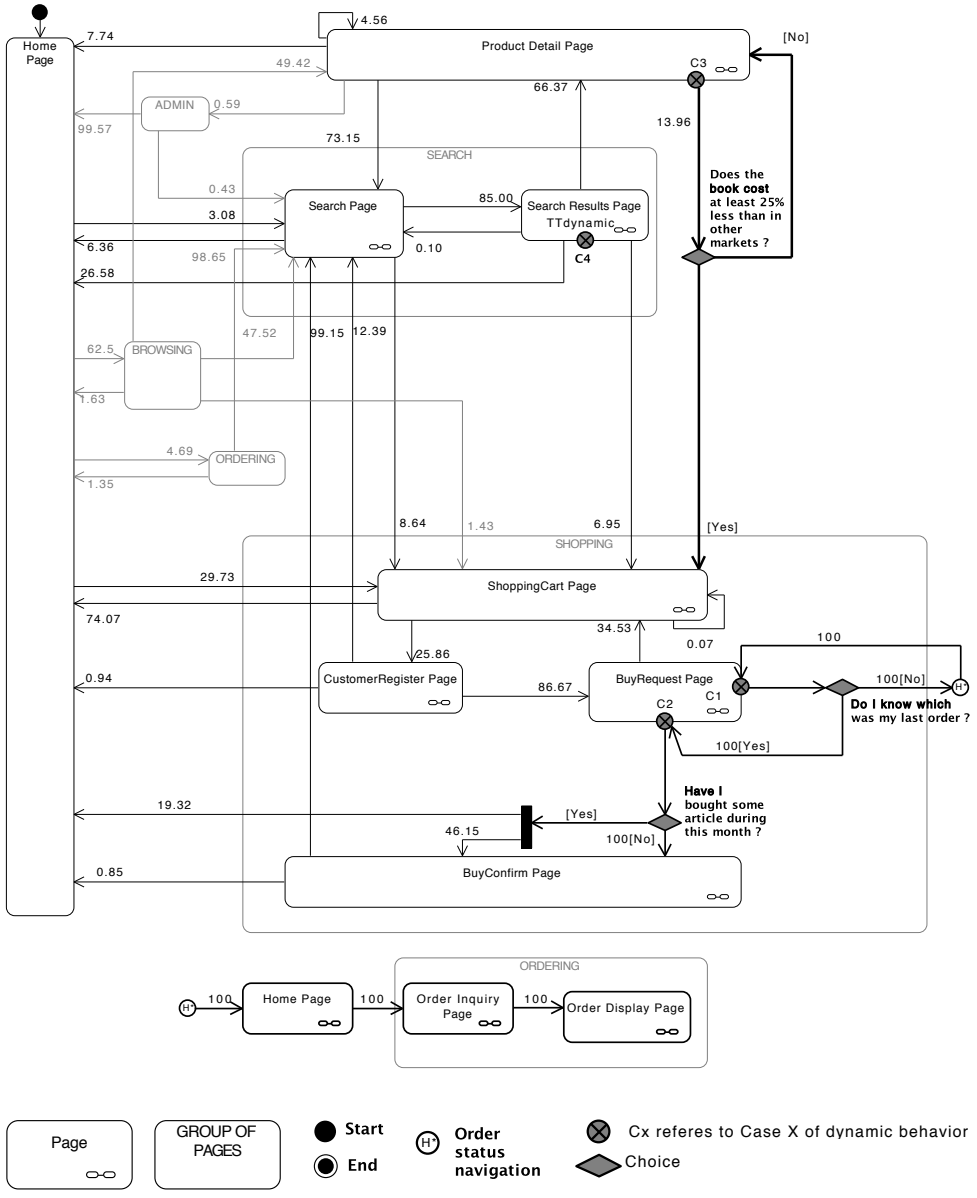


Figure 6.1: DWEB workload I - DW1: navigation for loyalty promotion behavior

the first byte of the first HTTP request of the next web interaction is sent from the emulated browser to the server. TPC-W considers that each think time must be taken independently from a negative exponential distribution, with the restriction that the average value must be greater than 7 seconds and lower than 8 seconds, as shown in equation (6.1). An important drawback of this approach is that it does not consider the results of the current search (web contents) on the actual user think time as occurs in real navigations.

$$TT_{TPC-W} = -ln(r) * p \ , \ where \ (0 < r < 1) \ \wedge \ (7 \leq p \leq 8) \tag{6.1}$$

In contrast, DWEB allows us to define a *dynamic think time* ($TT_{dynamic}$) according to the number of items returned by the search as shown in equation 6.2, which is closer to real web activities like the one defined in *case 4*. DW1 workload is still assuming TT for the less dynamic pages in the website (e.g. the Home page or the Product Detail page), but it uses $TT_{dynamic}$ in the Search Results page.

$$\begin{aligned} TT_{dynamic} = -ln(r) * p \ , \ where \ (0 < r < 1) \wedge \\ (p = 7 + \frac{Number \ of \ Search \ Results}{Max. \ Search \ Results}) \end{aligned} \tag{6.2}$$

The remaining cases of dynamism have been characterized using conditional transitions with DWEB. The transition from the Buy Request to the Buy Confirm pages depends on the last customer's purchase. If the customer did not buy any book during a given month, he has to commit the buying process, otherwise, he may finish the purchase or navigate to the Home page according to estimated probabilities of arcs as defined in *case 2*. Notice that, when a customer does not remember the date of his last purchase, he must visit the *ordering* group in order to find out it, as defined in *case 1*. Finally, *case 3* has been implemented in DW1 workload with a conditional transition between the Product Detail and the Shopping Cart pages. This transition only allows users to add a book to the shopping cart in case that its cost is 25% cheaper than in other markets.

6.1.2 One step ahead: evolving user's profile using dynamic roles

A common behavior characteristic of the web users community is the dynamism in the user's roles. In other words, the different roles that users adopt and the changes among them. Chang et al. [CACB97] reported three phases of marketing in an e-commerce website that can induce the mentioned dynamic evolution of web users. These phases are: pre-sales, on-line, and after sales. The pre-sales phase includes company efforts to attract customers by advertising, public relations, new products or service announcements, and other related activities such as discounts in some products or freebies (e.g. Apple promotions: Back to school and 12 Days of Christmas). Customers' electronic purchasing activities take place in the on-line sales where orders

and charges are done through web facilities. The after-sales phase includes customer service, problem resolution, etc.

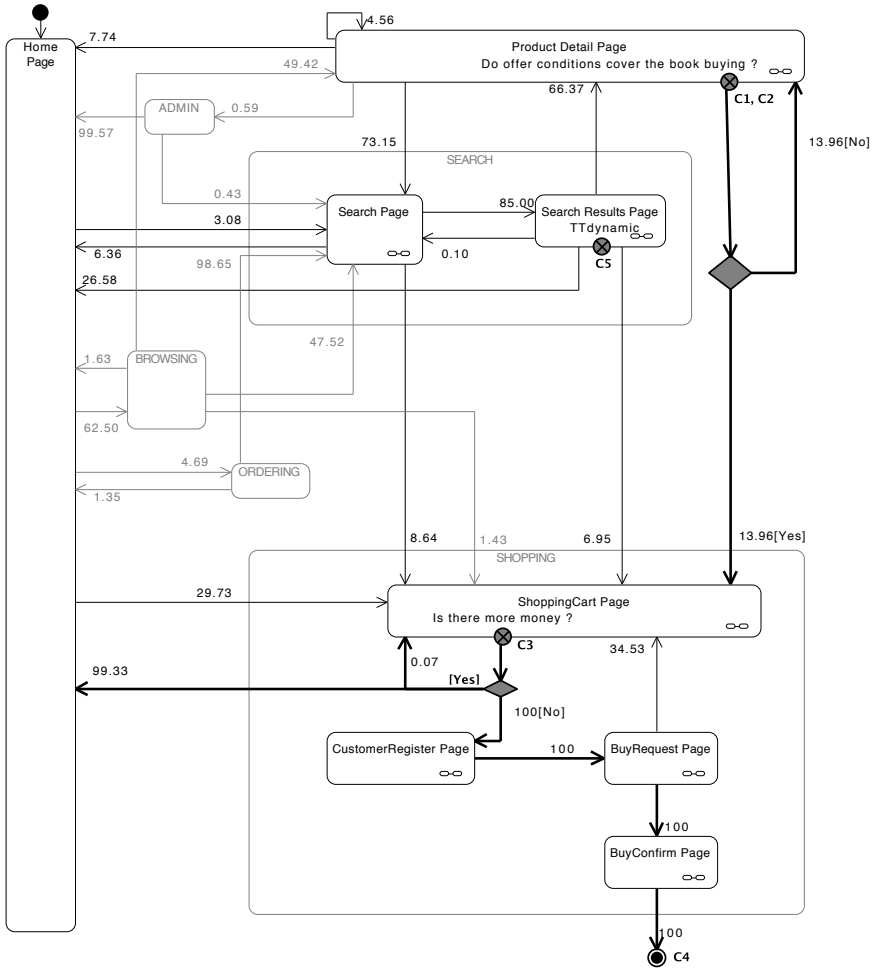
In the second scenario, we define a new DWEB workload (DWEB workload II - DW2) that reproduces how user's behavior evolves from pre-sales to on-line user profiles. We adopt the loyalty promotion behavior presented above as an example of on-line user profile, and define a new behavior based on a pre-sales promotion on the studied on-line bookstore. The pre-sales promotion consists of 1000 bonus to acquire common books in a buying session. This promotion should present a different user behavior with five cases of dynamism as summarized in Table 6.2.

Case	**Description**
1	Books that are best-sellers or new products cannot be added to the shopping cart because the bonus is only valid for common books.
2	The customer can buy a book only if the cost of the resulting shopping cart is lower than the bonus value.
3	A buying session (navigation session) finishes with a payment when the shopping cart cost is at least 75% of the bonus value.
4	A customer leaves the website when the buying session finishes.
5	The higher the number of provided search results, the longer the time that a user takes to read and think about them.

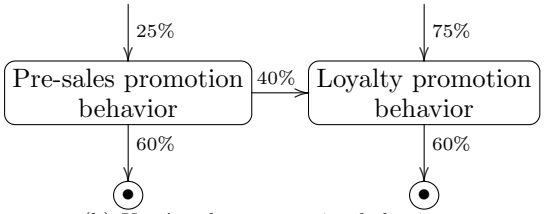
Table 6.2: Cases of dynamism in the new pre-sales promotion behavior

Figure 6.2 depicts the DW2 workload. This characterization defines the pre-sales promotion behavior using the DWEB navigation concept (Figure 6.2a). *Case 1* and *case 2* have been implemented with a conditional transition between the Product Detail and the Shopping Cart pages. This transition depends on the user's state (the available user's bonus) and the user's navigation path. That is, it only allows users to add a book to the shopping cart in case that they arrive at the Product Detail page from other pages than the Best-sellers or New Products ones. The transition from the Shopping Cart to the Customer Register page depends on the content of the first page (*case 3*), and it implies the end of the user navigation when the buying process is committed (*case 4*). We also use the dynamic think time based on the number of items returned by the search (*case 5*).

Finally, we combine both promotion behaviors (behaviors for loyalty and pre-sales promotions) as dynamic roles by using the user's roles concept of DWEB (Figure 6.2b). The user's roles automaton considers that the average response of the pre-sales promotion is 25% of users, based on the suggestions made in [SAP02]. The transition between the pre-sales promotion and the loyalty promotion behaviors models the evolution from a new user to a loyal customer, according to the average percentage of



(a) Navigation for pre-sales promotion behavior



(b) User's roles: promotion behaviors

Figure 6.2: DWEB workload II - DW2: characterization based on user's dynamic roles

customer retention (40%) reported in [RS00], while the arcs arriving to a final state represent the users' defection (60%).

6.2 Impact of the dynamic workloads on web system performance

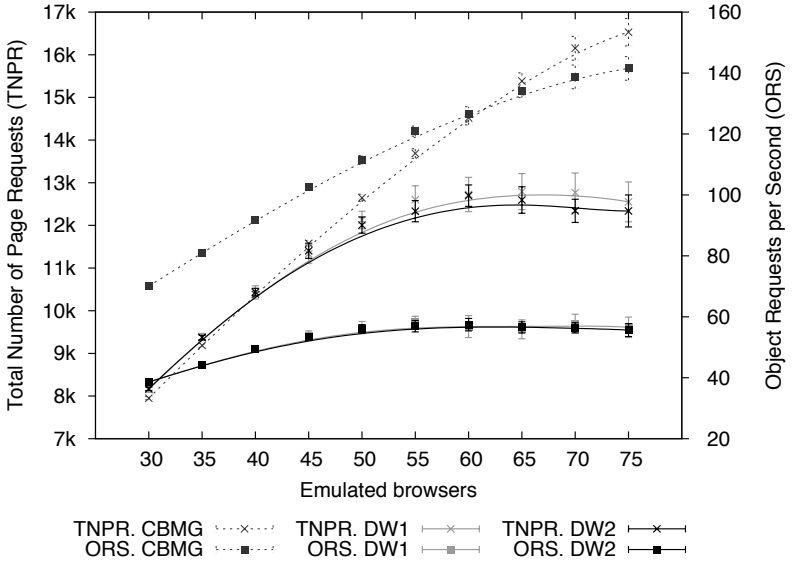
Experimental tests have been devised to compare performance metrics obtained with CBMG workload versus those obtained with the studied DWEB workloads (DW1 and DW2). With the aim of finding out the stress borderline of the server for each workload, we varied the number of EBs ranging from 30 to 75 in 5-user steps. All the experiments were done repeating 50 runs to obtain a margin of error with 99% confidence level.

Although experiments measured all the performance metrics listed in Table 5.1, only those present significant differences between traditional and dynamic workloads (e.g. number of requests, response time, CPU utilization and MySQL throughput) are shown in Figure 6.3 and Figure 6.4. Note that, in general, considering user's dynamism degrades the service conditions more than using traditional workloads, even though dynamic workloads generate a lower number of requests.

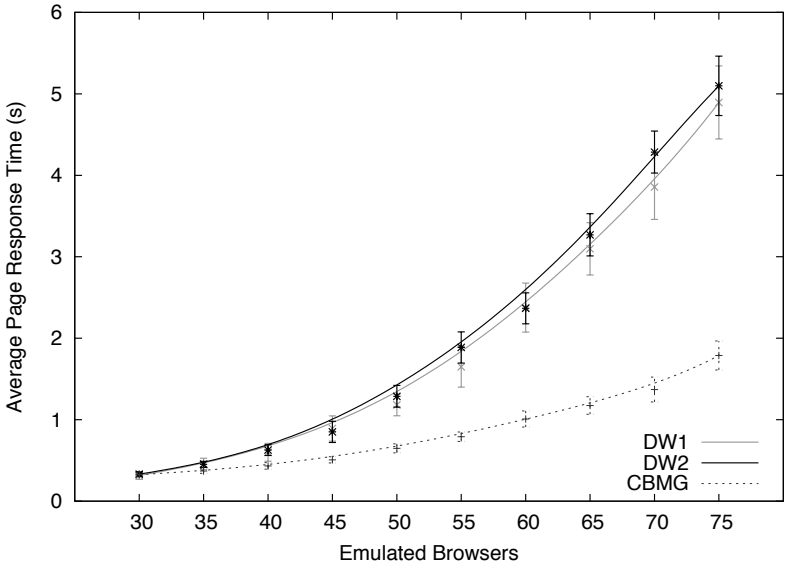
The *object requests per second* generated by CBMG workload is 45% to 60% higher than the generated by dynamic workloads (see Figure 6.3a). However, the response time for DWEB workloads presents exponential curves with more pronounced slopes than CBMG curve (see Figure 6.3b), for instance, DW2 workload increases the difference with respect to CBMG workload by 10%. On the other hand, the *total number of page requests* shows different values depending on the stressing server conditions (see Figure 6.3a). When the server is characterized by a poor stress level (e.g. less than 40 browsers), the number of page requests generated by the dynamic workloads is by 3% higher than the generated by CBMG, because $TT_{dynamic}$ reduces idle times when there are simultaneous requests on a search process. However, CBMG workload requests a higher number of pages than DWEB workloads for a significative level of stress (e.g. more than 40 browsers), because the dynamism produces, in general, more complex requests that require extra service time and consequently, reducing service rate.

Regarding the utilization of the main hardware resources (CPU, memory, network and disk), only the processor presents significant differences. As expected, the CPU utilization increases with the number of emulated browsers as shown in Figure 6.4a. However, although the workloads present similar CPU utilization for a low number of browsers, differences between dynamic and traditional workloads can be as large as 30% when considering dynamism. Notice that a high CPU utilization means that the processor acts as the main performance bottleneck.

To better understand why the CPU utilization is so high, we studied how the main software components use the processor (Apache, Tomcat and MySQL). As observed in Table 6.3, MySQL almost monopolizes the processor since its execution time is more

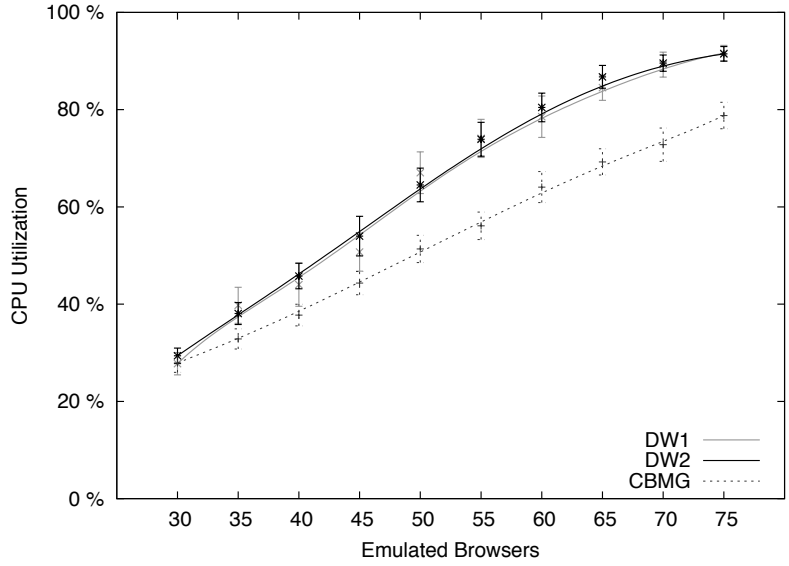


(a) Requests

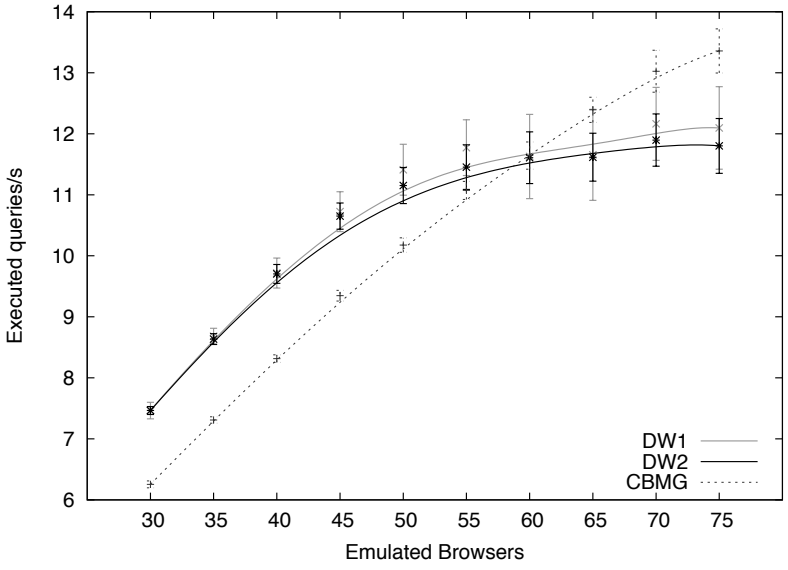


(b) Page Response Time

Figure 6.3: Main performance client metrics values



(a) CPU utilization



(b) MySQL throughput

Figure 6.4: Main performance server metrics values

EBs	Workload	Apache	Tomcat	MySQL
30	CBMG	7.71	4.55	886.48
	DW1	2.64	4.65	972.87
	DW2	2.61	4.93	1044.66
35	CBMG	9.36	5.41	1103.24
	DW1	3.44	6.47	1390.73
	DW2	3.25	6.34	1462.59
40	CBMG	11.01	6.45	1301.23
	DW1	3.87	7.51	1612.87
	DW2	3.86	7.92	1693.67
45	CBMG	13.16	8.03	1573.35
	DW1	4.52	9.05	1862.63
	DW2	4.67	9.59	2003.51
50	CBMG	14.95	9.64	1760.76
	DW1	5.45	11.52	2486.49
	DW2	5.32	11.45	2398.61
55	CBMG	17.13	11.59	1979.87
	DW1	6.00	12.95	2703.26
	DW2	6.04	13.03	2732.06
60	CBMG	19.04	12.98	2171.47
	DW1	6.45	13.82	2868.51
	DW2	6.55	14.11	2948.50
65	CBMG	21.97	15.42	2489.40
	DW1	6.81	14.97	3043.73
	DW2	7.00	15.19	3152.04
70	CBMG	23.51	16.81	2575.40
	DW1	6.97	14.99	3190.34
	DW2	7.28	15.72	3222.43
75	CBMG	25.68	18.08	2748.36
	DW1	7.23	15.33	3246.03
	DW2	7.32	15.80	3264.57

Table 6.3: CPU consumption (in jiffies) foreach application

than two orders of magnitude higher than the time devoted to Tomcat, specially with dynamic workloads. Figure 6.4b shows the executed queries rate by MySQL for each workload. As observed, for a relatively low number of emulated browsers (e.g. 45), this rate is by 15% higher when considering dynamism. But, more than 60 browsers cause that the number of executed queries becomes almost constant with dynamic workloads. That is due to a higher CPU utilization (greater than 80%) as depicted in Figure 6.4a. Consequently, MySQL database is the major candidate to be a software bottleneck.

With the aim of evaluating the impact of dynamism on server stress peaks, we analyze the database usage done by dynamic workloads. Before this study, we need to understand how MySQL database works. This database includes `qcache` as a cache of

executed queries, where queries result in *hit* or *miss*. We also distinguish *not cached* queries as a part of misses that cannot be cached for their dynamic nature or their complexity [Ora12]. Listing 6.1 shows a *not cached* query example at TPC-W website.

```
SELECT ol_i_id FROM orders, order_line
WHERE
        orders.o_id = order_line.ol_o_id
        AND NOT (order_line.ol_i_id = 93234)
        AND orders.o_c_id IN (
                SELECT o_c_id FROM orders, order_line
                WHERE orders.o_id = order_line.ol_o_id AND
                        orders.o_id > (SELECT MAX(o_id) ...)
        ) ...
```

Listing 6.1: *Not cached* queries example at TPC-W website

For a deeper study, we compared the cache status of the executed queries versus CPU utilization for several stress levels. Figure 6.5 shows the values for 35, 55 and 75 emulated browsers as examples of poor stress, significative stress, and overloaded situation at the server side, respectively. DW2 workload (the right column) presents a higher number of misses than DW1 and CBMG workloads (center and left columns, respectively). CPU overload peaks become larger and larger with the increase of *not cached* queries, which is caused by the dynamic query nature, specially for the DW2 workload that presents higher increase than the DW1 workload. Note that hits are not represented because their execution time is around one millisecond while the time taken by misses might be more than eight seconds.

Finally, we present an example illustrating how the system level of stressing caused by user's dynamism might induce higher probability of user abandonment. According to a Jupiter Research report [Jup06] commissioned by Akamai, web page rendering should be not longer than four seconds to avoid user abandonment. Figure 6.6 depicts how dynamic workloads increase the probability that a response takes over four seconds, specially when considering changes of user's behaviors in a overloaded system. Consequently, if the server is not properly tuned, the extra workload induced by the user's dynamic behavior can increase the probability of user abandonment (up to 40%).

In summary, results show that considering user's dynamism when characterizing web workload affects system performance. Dynamism on workloads characterization introduces new patterns of HTTP requests. These patterns, in general, reduce the number of requests to objects but increases the number of requests to dynamic web content so incurring differences in the system performance metrics, specially on the processor usage and the database throughput. As a result, the server performance degradation affects service conditions, increasing the average response time that induces a higher number of user abandonments. Results have also proved that considering user's dynamic navigations on workload characterization has a higher impact

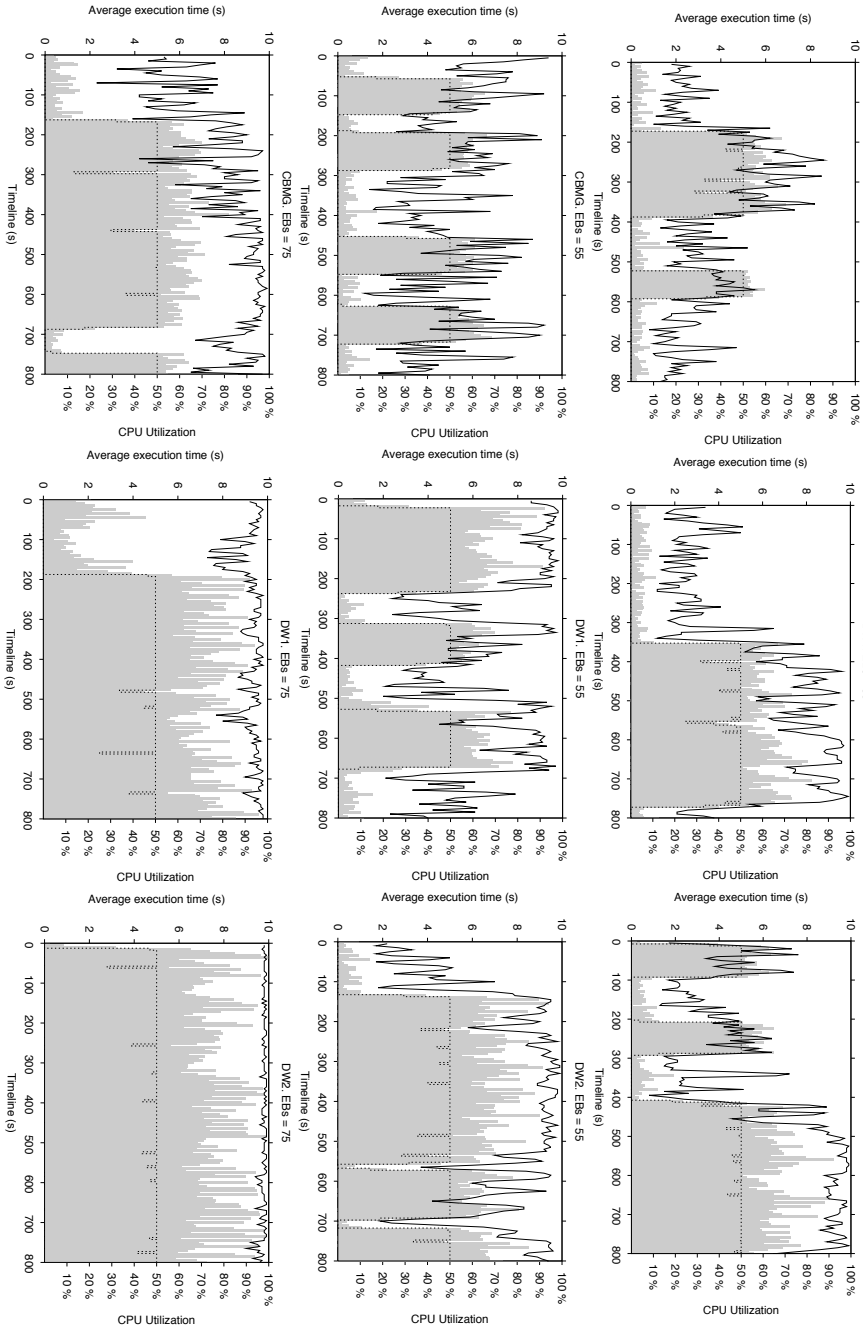


Figure 6.5: CPU utilization by query cache status

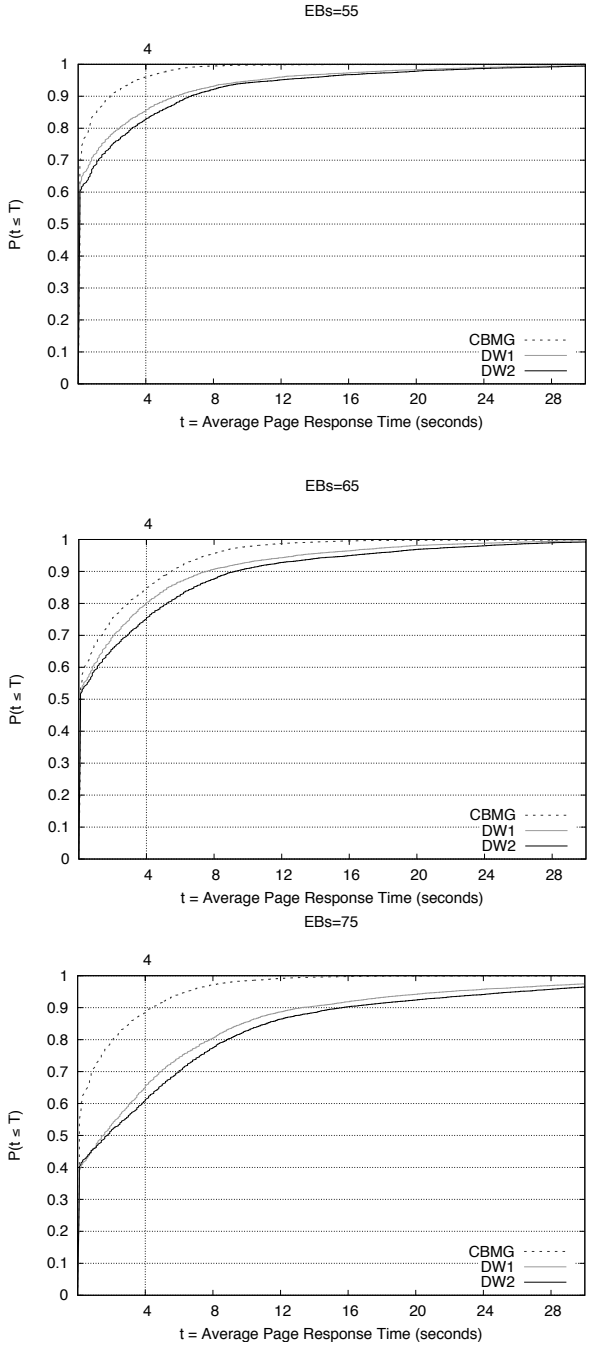


Figure 6.6: Cummulative distribution for page response time

on the system performance metrics than modeling changes in user's roles because the dynamism is more present in navigations than in user's role changes.

6.3 Summary

This chapter has explored the effects of using dynamic user workload on web performance evaluation studies. The obtained results have been compared against those generated with traditional workloads. To this end, a scenario based on a typical e-commerce website has been recreated and different user's dynamic behaviors have been reproduced. We have used DWEB model to define the dynamic workloads because it permits easily to model these behaviors, which cannot be represented with this level of accuracy with traditional approaches, such as the CBMG model. Furthermore, DWEB allowed us to define the workloads considering not only the dynamism in user's interaction but also the dynamic changes in the role when navigating the website. With the aim of evaluating the effect of these levels of dynamism on the system performance, a large and representative set of the most commonly used performance metrics has been measured for each experiment, such as client metrics (total number of page requests or response time), server hardware metrics (CPU utilization), or metrics for the main software components at the server (executed queries rate).

Experimental results have shown that dynamic workloads affect the stress borderline on the server, which is more degraded than when using traditional workloads in the studied cases. The object requests rate has decreased between 45% and 60%, but dynamic workloads have also changed the request nature, which has become more dynamic. This change has implied an important growth by 15% in the number of executed queries by the database with a significant increase in their dynamic nature that led the database to be an overloaded application. Consequently, the CPU utilization increased by 30% and consolidates the processor as the main performance bottleneck. The server performance degradation affects the service conditions increasing the average response time exponentially, which yield to higher probability of user abandonment (up to 40%). Furthermore, we observed that considering user's dynamic navigations on workload characterization has a stronger impact on the system performance metrics than modeling changes in the role, because dynamism is more present in navigations than in user's roles.

In [IEEE-NCA'12, COMCOMJ'13], a summary of the study introduced in this chapter was presented.

CHAPTER 7

The impact of User-Browser Interaction on web performance

This chapter analyzes and measures the effect of considering the User-Browser Interaction (UBI) as a part of user's dynamic behavior on web workload characterization in performance studies. To this end, we evaluate a typical e-commerce scenario by using the testbed presented in Chapter 5, and compare the obtained results for different behaviors that take user's interaction with browser interface facilities into account, such as the use of the back button and parallel browsing originated by browser tabs or opening new windows when surfing a website.

Experimental results show that these interaction patterns allow users to achieve their navigation objectives sooner than when browsing in a sequential way, so increasing their productivity up to 200% when surfing the website. In addition, results prove that when this type of behaviors is taken into account, performance indexes can widely differ and relax the stress borderline of the server. For instance, the server utilization drops as much as 45% due to parallel browsing behavior permitting the system either to devote more resources to other applications or to serve more users.

The remainder of this chapter is organized as follows. Section 7.1 presents the workloads considering UBI when modeling user's behavior. Section 7.2 shows the effect of the proposed workloads on the web system performance. Finally, we draw some concluding remarks in Section 7.3.

7.1 Workload design

This work uses DWEB to introduce different levels of user's dynamism on workload characterization, especifically dynamism related to the interaction with the web browser. Section 7.1.1 focuses on user's goals when surfing a website and defines user's navigations according to this end. After that, a more realistic dynamic workload is defined considering rapid return to recently visited pages by using the history-back

button on web browsers. Section 7.1.2 introduces parallel tab browsing behavior on workload characterization.

7.1.1 The back button: rapid return to recently visited pages

Web browsers includes a stack-based model for web page navigation [CMJ02]. In this model, there are two traditional ways of displaying pages in the browser: *load* and *reload*. Pages are loaded when the user clicks on a link, types a URL or selects a favorite page. The effect of load is to add the page to the top of the visited pages stack. Pages are reloaded with the back and forward buttons, and the effect is to alter the position within the stack. Each back click shows the next page down the stack until the stack bottom is reached. Forward clicks shows pages up the stack until the stack top is reached.

An important factor of the back button success is the chance of rapid return to recently visited pages [CG00], which can avoid new HTTP requests and consequently changes the causes of the stressing conditions of a given website. However, only certain visited pages can be cached by a web browser with the aim of going back using the back button. That is, some web contents such as audio or video streaming, dynamic contents or web forms are not cached according to their HTTP headers. For these types of contents the back button does not have any effect because pages are completely reloaded.

With the purpose of measuring the effect of UBI in web performance evaluation we compare different dynamic workloads. In a first step, a workload (LOY) conducted by user's goals is defined. To this end, we extend the loyalty promotion behavior presented in Section 6.1.1 by adding a new case of dynamism to establish that a user leaves the website when his goals are satisfied. The new behavior is characterized by five cases of dynamism as summarized in Table 7.1.

Secondly, a new DWEB workload (LOYB) is defined by extending the LOY workload with an extra case of dynamism that characterizes the use of the back button on web browser (see Table 7.1).

Figure 7.1 and Figure 7.2 show the workloads generated using DWEB for the loyalty promotion behavior conducted by goals (LOY) and its extended version (LOYB) considering the back button, respectively. Both workloads assume TT (see equation 6.1) as *think time* for the less dynamic pages in the website (e.g. the Home page or the Product Detail page), and $TT_{dynamic}$ (see equation 6.2) in the Search Result page, that is closer to real web activities like that defined in *case 4* listed in Table 7.1.

The remaining cases of dynamism have been characterized using conditional transitions with DWEB. The transition from the Buy Request page to the Buy Confirm page depends on the last customer's purchase. If the customer does not buy any book during a given month, he has to commit the buying process, which means the end of the navigation session (*case 5*). Otherwise, he may finish the purchase or navigate to the Home page according to estimated probabilities of arcs as defined in *case 2* and

Case	Description
1	If customers do not remember their last order status, they will check them by navigating into the *ordering* group of pages.
2	Because the customer has to buy at least once a month to keep the discount, a buying session must finish with a payment when he has not bought anything during that month.
3	A experienced customer only buys a book when its cost is 25% cheaper than in other markets.
4	The higher the number of provided search results, the longer the time that a user takes to read and think about them.
5	A customer leaves the website when the buying session finishes because his goals have been satisfied.

Extra case in the extended behavior

6	A customer can return to recently visited listings of books (browsing listings or search results) without repeating a request to the web server through the back button.

Table 7.1: Cases of dynamism in the loyalty promotion behaviors conducted by goals

shown in both figures. Notice that when a customer does not remember the date of his last purchase, he must visit the *ordering* group in order to find out it, as defined in *case 1*. *Case 3* has been implemented with a conditional transition between the Product Detail and the Shopping Cart pages. This transition represents users adding a book to the shopping cart because its cost is 25% cheaper than in other markets. Finally, the back button has been characterized in LOYB workload as a cache that only stores the immediately previous listing page (*case 6*).

7.1.2 Optimizing user productivity: the parallel tab browsing behavior

Parallel browsing describes a behavior where users visit web pages in multiple concurrent threads by using web browsers tabs or windows. To help the understanding of how parallel browsing works, Figure 7.3 illustrates a parallel tab browsing session applied to the on-line bookstore. It shows how a web user uses parallel browsing based on three web browser tabs to improve his navigation time avoiding searches of books. The user begins a navigation session in a window with the aim of buying a book that fulfills several requirements as soon as possible. After he visits some pages, a search result is provided. At this point of time, he starts a parallel tab browsing

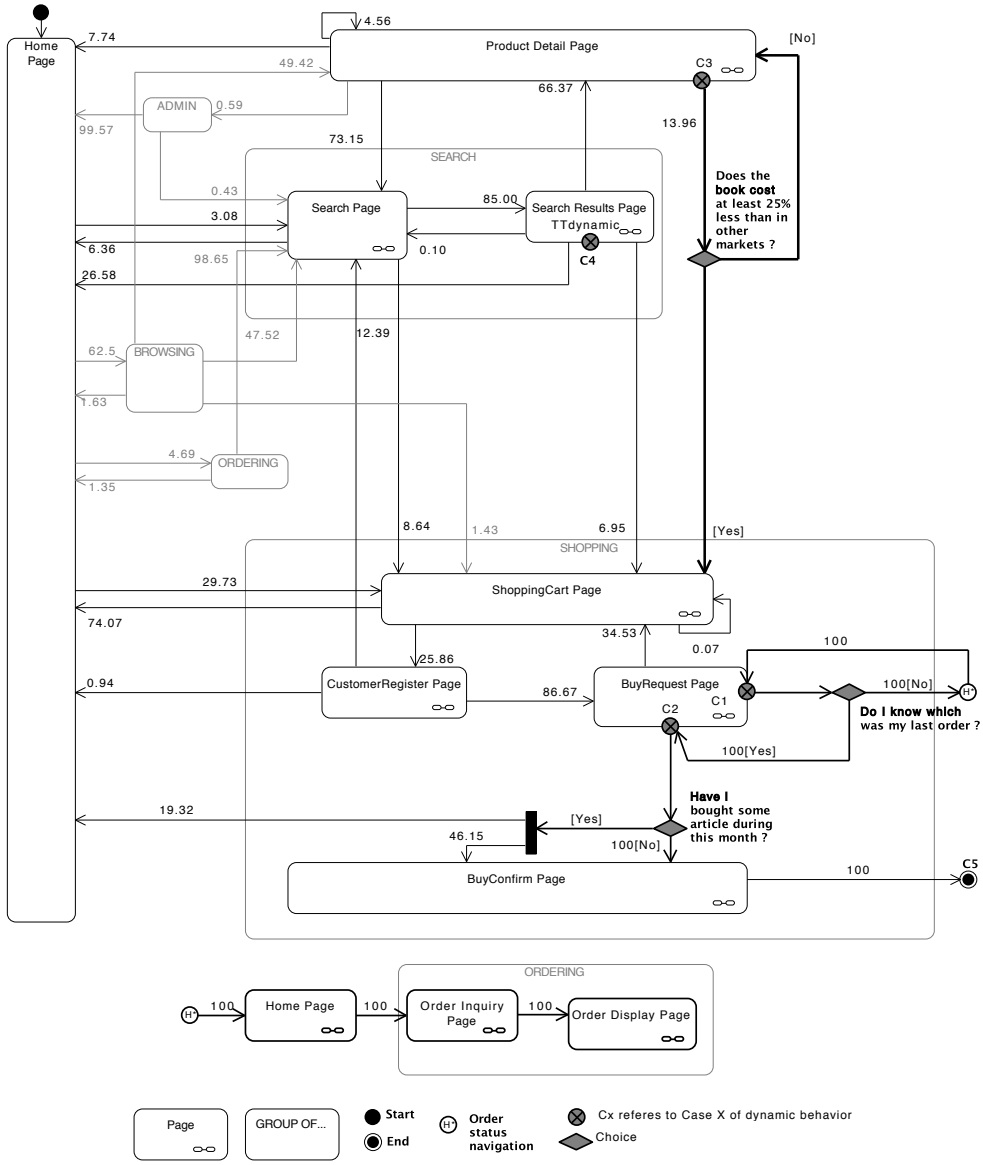


Figure 7.1: LOY workload: loyalty promotion behaviors conducted by goals

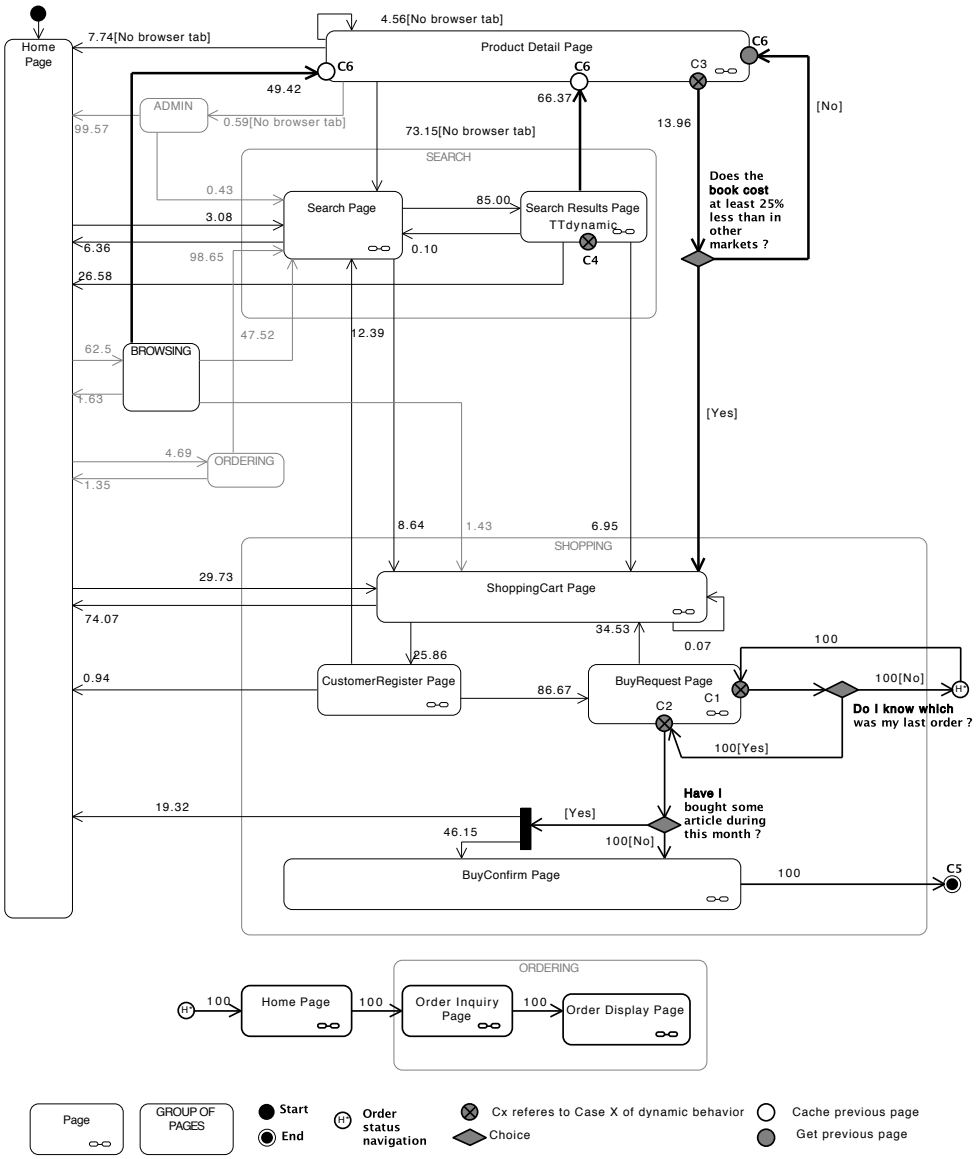


Figure 7.2: LOYB workload: LOY workload considering the back button

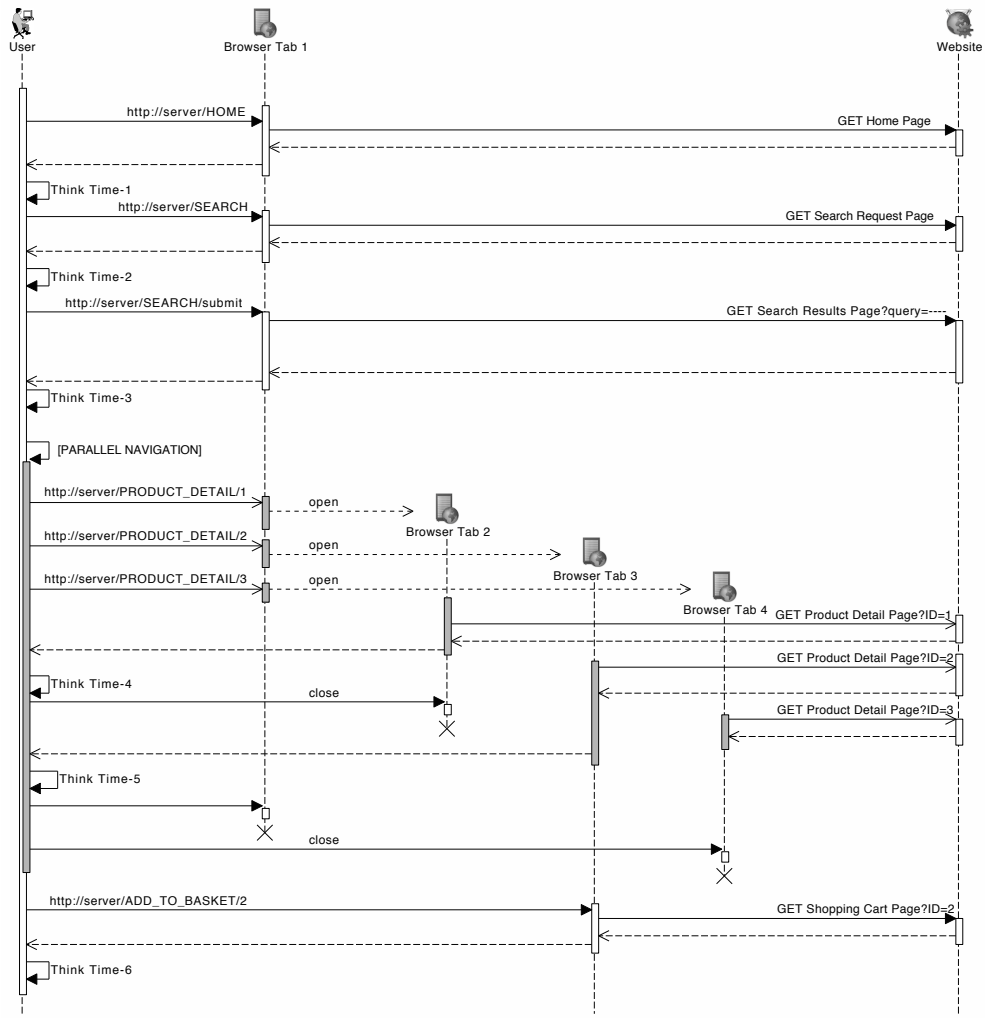


Figure 7.3: Example of parallel tab browsing session

behavior by opening three new tabs with the detail of three different books selected from the results. Then, he takes some time calculated according to equation 6.1, such as *Think Time 4 or Think Time 5* (see Figure 7.3), to evaluate the content of each tab until he finds a book satisfying his requirements. When a book does not fulfill the requirements, the user closes its associated tab and switches to the next one. Otherwise he discards the rest of tabs and adds the book to the shopping cart,

finishing the parallel browsing session. Notice that the user does not take any *think time* on a book detail page when he discards its tab.

In the third scenario, we define a new DWEB workload (LOYT) that reproduces parallel tab browsing behavior in the loyalty promotion. With this aim, we extended the loyalty promotion behavior presented above by characterizing the navigation on web browser tabs as summarized in Table 7.2.

Case	**Description**
6	When a user has to review a listing of books, such as the result of a search or a browsing request, he begins a parallel tab browsing session with three tabs.
7	A user closes a tab when its book does not fulfill his buying requirements or when he has found the the wished book in other tab.

Table 7.2: Extra cases of dynamism in the loyalty promotion behavior conducted by goals to represent parallel tab browsing

Figure 7.4 depicts the LOYT workload. This characterization defines the same behavior as LOY workload (Figure 7.1) except the navigations related to parallel tab browsing (*cases 6 and 7*). *Case 6* has been implemented with a pool of three parallel navigation threads, one for each tab. A navigation thread is killed when its book does not fulfill the user's requirements, or when the user finds the required book in other thread, as defined in *case 7*. The navigation becomes sequential again when only a thread is kept alive. Notice that even though LOYT workload does not consider the possibility of opening new tabs from another existing tab in this study, DWEB and GUERNICA provide mechanisms to model and execute multi-level in tab branching, respectively, if required.

7.2 Impact of UBI on web performance

Experimental tests have been carried out to compare the performance achieved with the loyalty promotion workload (LOY) against those of the extended versions (LOYB and LOYT) which consider UBI when modeling the user's behavior. With the aim of finding out the stress borderline of the server for each workload, we varied the number of users ranging from 50 to 250 in 50-user steps.

Additional metrics measured at the client side have been defined in order to quantify the user's productivity, such as *number of finished sessions*, *session length* and *number of visited pages per session*. Notice that a navigation session finishes when a user leaves the website after his goals are achieved; that is, after he buys some books in the case study. Thus, the higher the number of finished sessions, the higher the user's productivity.

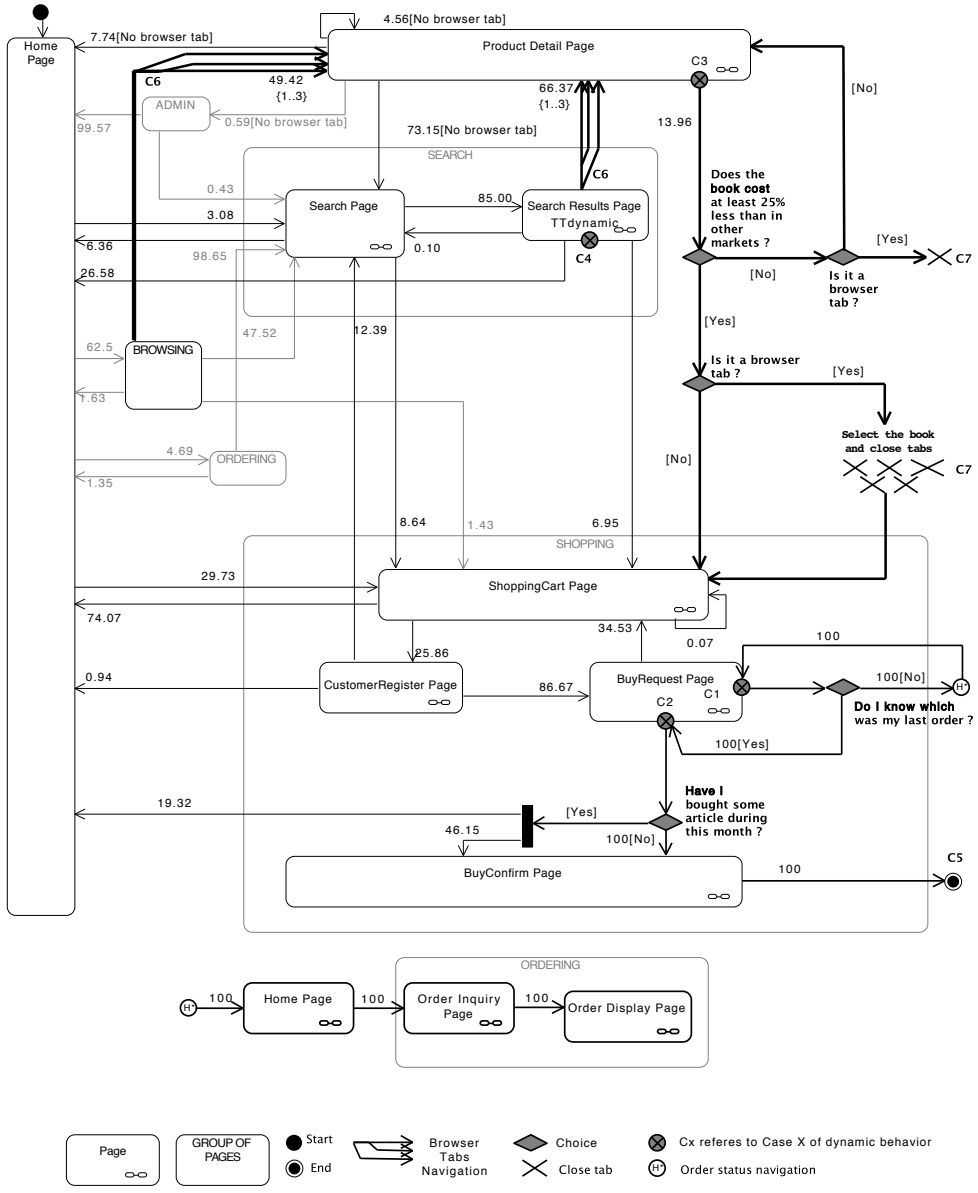


Figure 7.4: LOYT workload: parallel tab browsing behavior in LOY workload

Although we measured all the performance metrics listed in Table 5.1, only those showing significant differences in the studied workloads are discussed below.

Figure 7.5 shows how the user's productivity (measured in number of finished sessions) increases when considering UBI on workload characterization, specially when a parallel browsing behavior is introduced, because both the *session length (in seconds)* and the *number of visited pages per session* are lower in these workloads (see Table 7.3 considering 100 simultaneous users as example). Therefore, user's productivity is improved when the user changes his navigation patterns as a result of parallel browsing behavior (up to 200%) or using the back button (up to 100%).

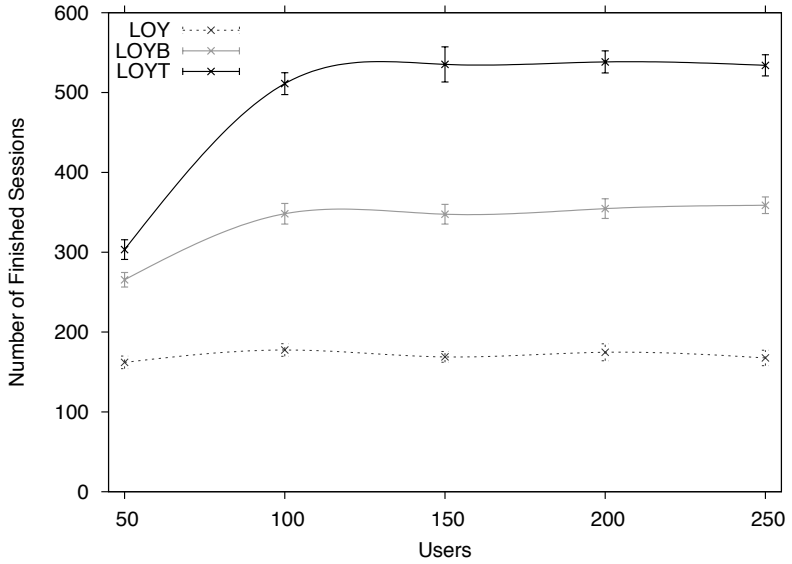


Figure 7.5: User's productivity evolution

Metric	LOY	LOYB	LOYT
Number of finished sessions	177.530	348.180	526.380
Session length (sec)	598.024	230.032	132.839
Number of visited pages per session	78.909	26.146	18.814

Table 7.3: Mean user productivity considering 100 simultaneous users in the system

In general, the extended workloads also increase the server throughput, despite of the LOY workload degrades the service conditions more than considering the interactive behaviors as depicted in Figure 7.6. Considering a significative number of simultaneous users (e.g. from 100 users on), the *total served pages* is on average by 25% and 90% higher for the extended workloads (Figure 7.6). To identify the causes

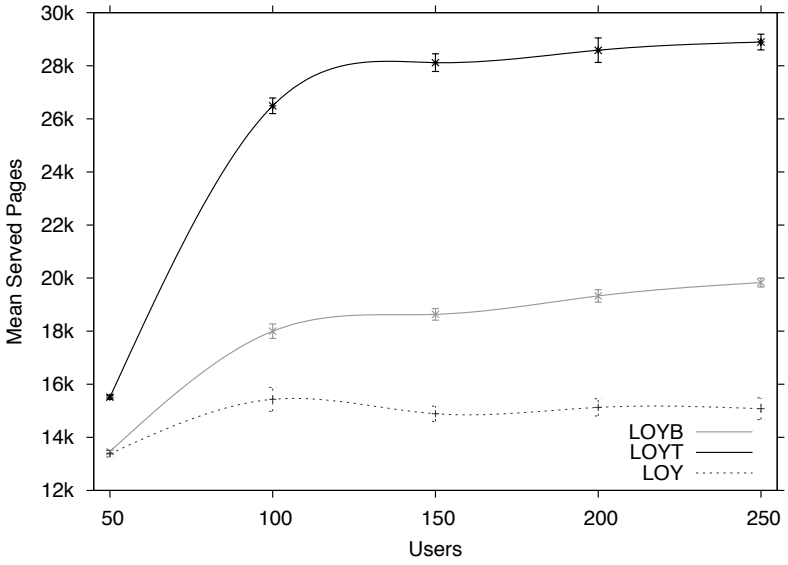
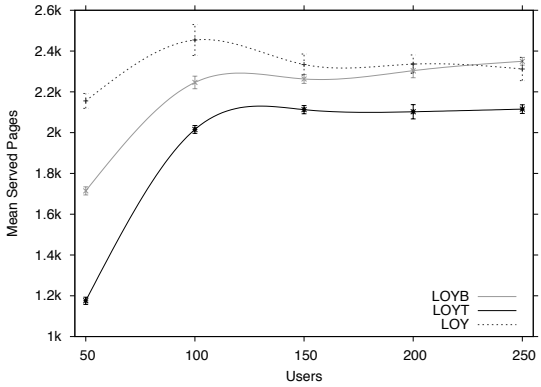


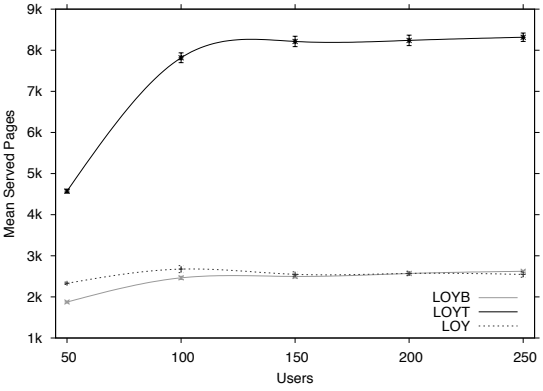
Figure 7.6: Total served pages

of this high increase, served pages are classified in three main types: *search results page*, *product detail page* and *others*. As observed in Figure 7.7, the interactive behaviors generate less requests to the search engine (Figure 7.7a) and increase the number of requests to the product detail page (Figure 7.7b) and to the others (Figure 7.7c), especially for the LOYT workload. This new pattern of HTTP requests reduces the complexity of database queries when decreasing searches, so the throughput of the web server increases as shown in Figure 7.8. Specifically, the *Apache HTTP requests per second* generated by LOY workload is by 25% and 75% lower than those generated by the LOYB and the LOYT workloads, respectively.

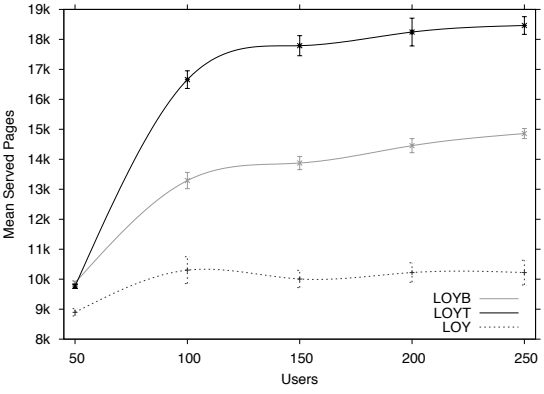
Figure 7.9 shows the *CPU utilization*, which is the only element that presents significant differences among the main hardware resources. Stress conditions have been classified in three levels according to the CPU utilization values: low stress ($U_{CPU} < 50\%$), significant stress ($50\% \leq U_{CPU} \leq 80\%$), and high stress - here by overload at server - ($U_{CPU} > 80\%$). As can be seen, the processor utilization for the extended workloads is always lower than for the LOY workload. For instance, considering 100 users, the utilization decreases by 15% and 45% for the LOYB and the LOYT workloads, respectively. Notice that the high CPU utilization value denotes that the processor acts as the main performance bottleneck but the stress borderline moves from 100 users for the LOY workload to 150 and 200 users for the LOYB and the LOYT workloads, respectively. This means that the web server allows the system to serve about 50% and 100% more users for the LOYB and the LOYT workloads.



(a) Search results page



(b) Product detail page



(c) Others

Figure 7.7: Mean served pages by type

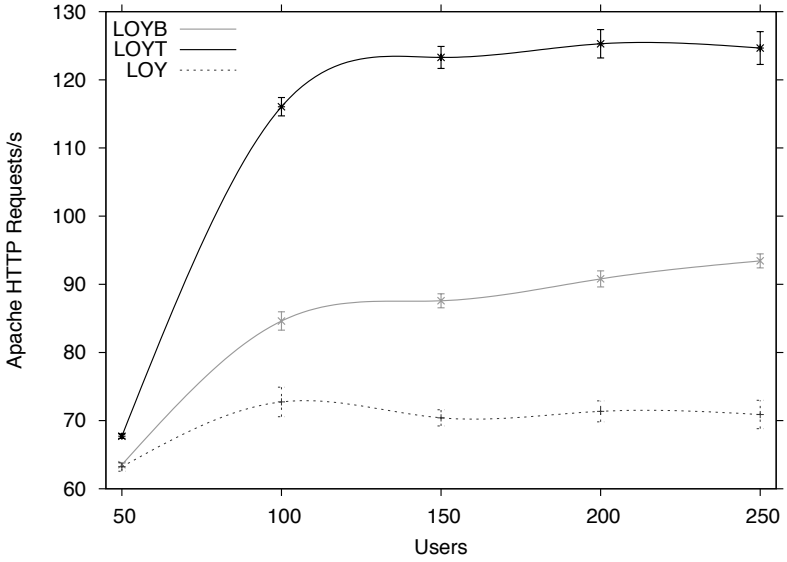


Figure 7.8: Apache throughput

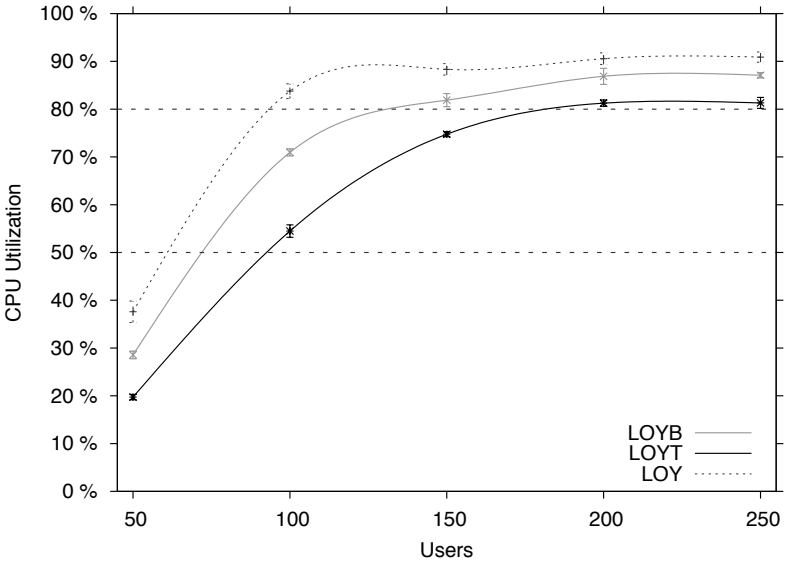


Figure 7.9: CPU utilization

To better understand why the CPU utilization decreases, we studied how the main software components (Apache, Tomcat and MySQL) make use of the processor. As observed in Table 7.4, MySQL almost monopolizes the processor for the different workloads since its execution time is more than two orders of magnitude higher than the time devoted to Tomcat and Apache. Consequently, MySQL database is the major candidate to be a software bottleneck. However, despite of the executed queries rate for the extended workloads can be as large as 30% and 93% higher than for the LOY workload (Figure 7.10), the CPU time consumed by MySQL for the extended workloads is by 30% and 55% lower than for the LOY workload. Moreover, the CPU consumption of Tomcat and Apache increases for the extended workloads. That is, there is a change in the patterns of HTTP requests and consequently in the type of executed queries at MySQL.

(a) LOY

Soft. \ Users	50	100	150	200	250
MySQL	1315	2883	2807	2768	2897
Tomcat	10	22	22	23	23
Apache	4	9	10	10	10

(b) LOYB

Soft. \ Users	50	100	150	200	250
MySQL	931	2439	2606	2540	2624
Tomcat	9	24	26	27	28
Apache	3	9	11	12	13

(c) LOYT

Soft. \ Users	50	100	150	200	250
MySQL	567	1808	2381	2434	2424
Tomcat	8	27	33	34	34
Apache	3	10	15	17	18

Table 7.4: CPU consumption (in jiffies) for each application

We performed a deeper study to provide a sound understanding about how the database is used by these workloads. As introduced in Chapter 6, MySQL database includes `qcache`[Ora12] as a cache of executed queries, where a query results in a *hit* or *miss*. Figure 7.11 shows the mean execution time for hits, misses and total queries considering 50, 100 and 250 simultaneous users in the system as examples of the different stress conditions: low stress (Figure 7.11a), significant stress (Figure 7.11b) and overload at server (Figure 7.11c). As observed, the mean execution time for a query when considering the extended workloads is always lower than for LOY work-

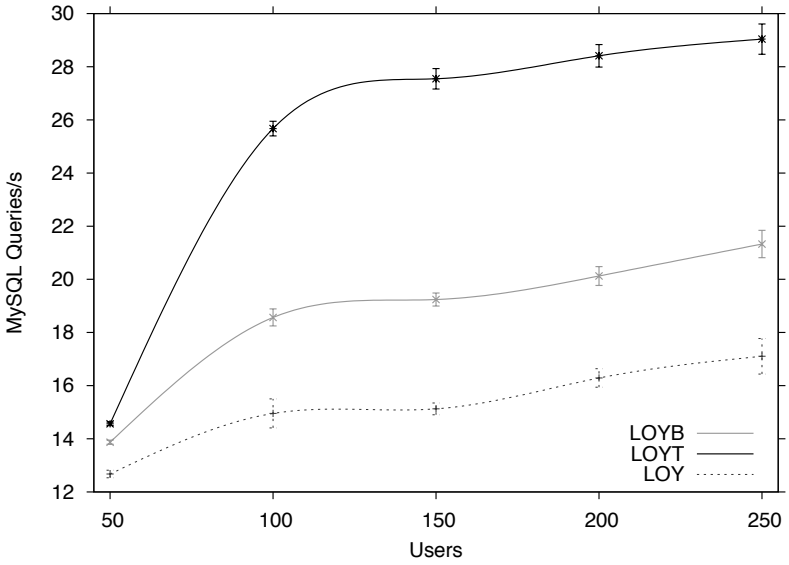


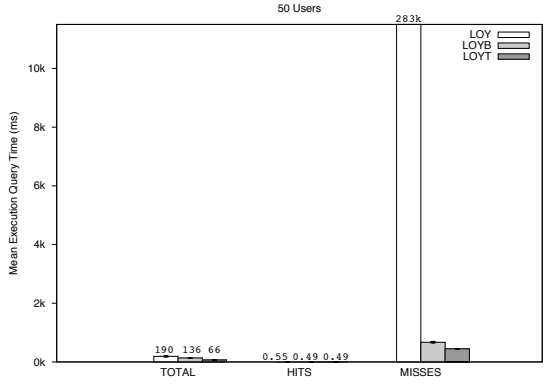
Figure 7.10: MySQL Throughput

load because decreasing the number of searches reduces the complexity, on average, of misses (the execution time decreases for the extended workloads) and increases the number of hits.

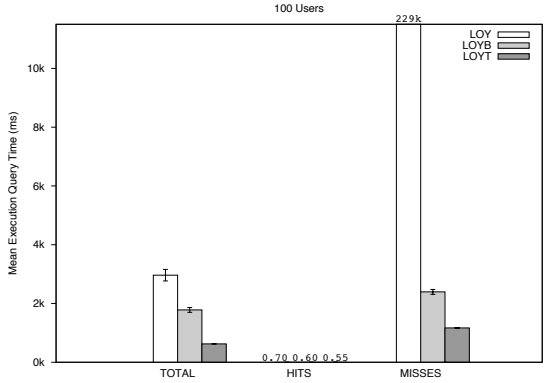
In summary, results show that considering user interaction with web browser facilities positively affects the system performance. This is because of considering UBI when modeling user's behavior introduces new patterns of HTTP requests. These patterns, in general, increase the number of requests to objects but reduce requests to the search engine allowing users to achieve a higher productivity. This fact causes noticeable differences in the performance metrics, specially in the processor utilization and the server throughput. As a result, the stress borderline of the server is affected.

7.3 Summary

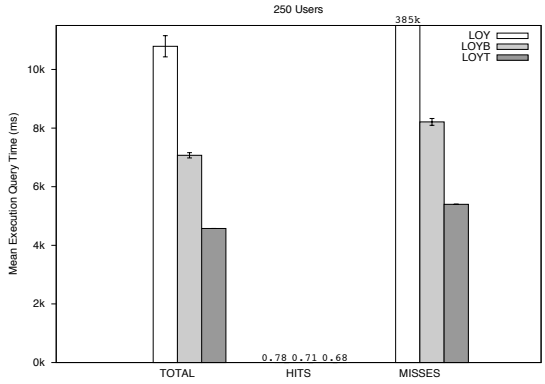
This chapter has explored the effects of considering User-Browser Interaction in web performance evaluation studies. To this end, the previously presented scenario based on a typical e-commerce website has been extended and a dynamic workload conducted by user's goals has been reproduced. Additionally, with the aim of improving the user's productivity, we included in the navigation patterns the use of the back button and the parallel tab browsing, as examples of user interactions with web browsers facilities. DWEB model has been used to take these behaviors into account in an easy and flexible way when modeling web browsing patterns.



(a) Low stress conditions



(b) Significative stress conditions



(c) Overload at server

Figure 7.11: Execution time per query type

The study has considered a wide set of traditional performance metrics measured both at client side and server side. In addition, new metrics have been defined in order to quantify the productivity in the web navigations from the user point of view, such as number of finished session, number of visited pages per session or session length.

This study proved that navigations using the back button or opening new tabs, which result from considering a user dynamic interaction with the provided contents, clearly allow users to achieve their goals in less time, so increasing their navigation productivity. These browsing patterns also affect the utilization and the throughput of the main system resources, and consequently the stress borderline on the server. Experimental results have shown that parallel tab browsing behavior noticeably increases the user's productivity (measured in number of finished sessions) up to 200% with respect to browsing the website in a sequential way. The new navigation patterns also increase the number of served pages (up to 90%). Nevertheless, they generate less requests to the search engine so reducing the complexity of the executed database queries and decreasing their execution time. This means an important drop in the processor utilization (up to 45%) and a noticeable rise in the web server throughput (up to 75%). As a consequence, the stress borderline is relaxed permitting the system either to support more applications or serve more users.

In [SAC'13], a summary of the study introduced in this chapter will be presented.

CHAPTER 8

Conclusions and open research lines

This chapter presents the main conclusions of this dissertation, summarizes the contributions and lists the publications in international and national journals and conferences related to this work. Possible research lines that are opened by this study are also drawn.

8.1 Conclusions

The current Web introduces a new paradigm where users become contributors to the dynamic contents and services offered. As a result, the behavior of users and their interaction patterns have evolved and are currently more active and dynamic. Consequently, web performance studies should take this new user behavior into account in the workload models to guarantee the validity of the results. According to the open literature three different problems must be addressed when modeling the user's behavior on web workload characterization: i) the dynamic behaviors of users [BC98], ii) the different roles that they can play when surfing the Web [WOHM06], and iii) the continuous changes among these roles [GBGP10].

This thesis has analyzed the current state of the art in modeling and generating workloads for web performance evaluation, focusing on the capability to fulfill the three open problems before mentioned. The lack of appropriate models and software applications motivated us to propose a more accurate workload model (first thesis contribution) and to develop a new workload generator (second thesis contribution), based on this model, with the aim of analyzing the effect of using dynamic workloads on web performance evaluation, instead of traditional workloads (third thesis contribution).

The proposed model, named *DWEB*, characterizes web workload in a more accurate and appropriate way by using a couple of concepts: *user's navigation* and *user's roles*. Both concepts tackle in a progressive way the implicit dynamism in user's interaction and user's roles, respectively, when navigating the Web.

Once this model was defined, the *GUERNICA* web workload generator was developed in order to reproduce more realistic workload mimicking the behavior of the real web users community. GUERNICA implements the two concepts of DWEB with the aim of adopting the model, and represents the physical distribution of users in the Web, which improves the workload accuracy by providing a distributed architecture.

Finally, we analyzed and measured the effect of using representative dynamic workloads on the web performance metrics. To this end, we previously validated GUERNICA in a new *testbed* for web performance studies. This testbed integrates our generator with the TPC-W benchmark in order to contrast performance metrics for traditional and dynamic workloads.

As a first step, we evaluated the impact of representing different levels of user's dynamism in web workload characterization. The obtained results were compared against those obtained with traditional workloads proving that dynamic workloads negatively affect the stress borderline on the server in the studied cases. Furthermore, we observed that considering user's dynamic navigations on workload characterization has a stronger impact on the system performance metrics than modeling changes in the role, because dynamism is more present in the navigations than in modeling changes.

In a second study, we explored the effects of considering User-Browser Interaction in web performance evaluation. This study proved that navigations using the back button or opening new tabs, which result from considering a user dynamic interaction with the provided contents, clearly allow users to achieve their goals in less time, so increasing productivity. These browsing patterns also affect the utilization and the throughput of the main system resources, and consequently the stress borderline on the server, which is relaxed permitting the system either to support more applications or serve more users.

The work presented in this dissertation permits us to state that workload models have to consider the different levels of user's dynamism in order to precisely estimate system performance for the dynamic Web.

8.2 Open research lines

The work done through this dissertation opens several research lines and new research fields based on its contributions. Some of them are briefly described below:

- DWEB could be formalized with the aim of: i) developing a graphical designer software, and ii) generating automatically web workloads from real traffic logs. To this end, approaches as the model driven development or the semantic knowledge would be considered.
- GUERNICA can be improved as a software product, or its generation process can be integrated in one of the most important commercial or open source community solutions with the aim of transferring the research knowledge into the industry.

- Other contexts of the current Web could be modeled, analyzed and evaluated using our approach, such as On-line Social Network or Mobile Apps.
- Traffic generation using DWEB and GUERNICA could be applied to other research contexts. After a research stay at Coventry University, the possibility of extending our approach to the web security context was suggested by Dr. Siraj Ahmed Shaikh. To this end, we can study how to generate traffic for:
  - Distributed Denial of Service (DDoS) attacks. These attacks are explicit attempts to disrupt the services of a provider, which are intended for its legitimate clients, by consuming computing and networking resources.
  - Flash Events (FEs) scenarios, which represent a period of time when a web-server experiences a dramatic increase in incoming traffic.
  - Flash-Crowd attacks, that are a popular form of DDoS targeted at application-level floods, where attackers mimic FEs by deploying a large, distributed bot network and generating legitimate application requests that overwhelm the victim.

8.3 Publications related to the thesis

Table 8.1 classifies the main publications related to this dissertation, which are detailed below, according to the publication type and provides the next information:

- *REFERENCE*: The publication identifier.
- *RANK*: The publication class according to the next rankings:
  - CORE: Conference class in the Computing Research Education (CORE) classification [Aus06] of 2006.
  - ERA: Conference class in the Excellence in Research for Australia (ERA) classification [Aus10] of 2010.
  - JCR: Journal class in the Journal Citation Reports (JCR) classification [Tho11] of 2009 and 2011.
  - SJR: Journal class in the SCImago Journal Rank (SJR) [SCI11] of 2009 and 2011.
- *CITATIONS*: Paper citations (source Google Scholar [Goo12]).

REFERENCE	RANK	CITATIONS	TYPE	
[IJEB'05]	-	5	Journals	
[COMCOMJ'09]	JCR-Q2 SJR-Q1	12		
[CLEIej'12]	-	1		
[COMCOMJ'13]	JCR-Q2 SJR-Q1	-		
[INFOSCI'13][a]	-	-		
[HET-NET'04]	-	2	International	Conferences
[WOSP'05]	CORE-B	5		
[CLEI'06]	-	-		
[WEBIST'11]	ERA-C	-		
[CLEI'11]	-	-		
[IEEE-NCA'12]	ERA-A	-		
[SAC'13]	ERA-B	-		
[JSWEB'05]	-	-	National	
[JJPP'10]	-	-		
[JJPP'11]	-	-		
[GENERICA'04a]	-	-	Reports	
[GENERICA'04b]	-	-		
[GENERICA'04c]	-	-		
[GENERICA'04d]	-	-		
[GENERICA'04e]	-	-		

Table 8.1: List of main publications

[a]Submitted

[HET-NET'04] Raúl Peña-Ortiz, Julio Sahuquillo, Ana Pont, and José Antonio Gil. Modeling users' dynamic behavior in web aplication environments. In *2nd International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NET'04)*, ISBN 0-99540151-6-9, pages P50/1–P50/10. Ilkley, West Yorkshire, UK, July, 2004.

[GENERICA'04a] GENERICA. Estudio del estado actual de los generadores de carga. Technical report. iSOCO S.L, December 2004.

[GENERICA'04b] GENERICA. Arquitectura. Technical report. iSOCO S.L, December 2004.

[GENERICA'04c] GENERICA. Arquitectura del Motor de navegaciones. Technical report. iSOCO S.L, December 2004.

[GENERICA'04d] GENERICA. Arquitectura de CARENA. Technical report. iSOCO S.L, December 2004.

[GENERICA'04e] GENERICA. Componente de visualización de navegaciones. Technical report. iSOCO S.L, December 2004.

[IJEB'05] Raúl Peña-Ortiz, Julio Sahuquillo, Ana Pont, and José Antonio Gil. Modeling users' dynamic behavior in e-business environments using navigations. *International Journal of Electronic Business (IJEB)*, ISSN 1741-5063, 3(3):225–242, May-June 2005. doi: 10.1504/IJEB.2005.007268.

[WOSP'05] Raúl Peña-Ortiz, Julio Sahuquillo, Ana Pont, and José Antonio Gil. Modeling continuos changes of the user's web dynamic behavior in the WWW. In *Fifth International Workshop on Software and Performance (WOSP'05)*, ISBN 1-59593-087-6, pages 175–180. Palma de Mallorca, Spain, July 2005.

[JSWEB'05] Raúl Peña-Ortiz, Julio Sahuquillo, Ana Pont, and José Antonio Gil. Modelado de patrones de comportamiento de usuarios WWW de segunda generación. In *I Jornadas Científico-Técnicas en Servicios Web (JSWEB'05)*, ISBN 84-9732-455-2, pages 251–254. Granada, Spain, September, 2005.

[CLEI'06] Raúl Peña-Ortiz, Julio Sahuquillo, Ana Pont, and José Antonio Gil. Incorporación de modelado dinámico a un generador de carga para la Web 2.0. In *XXXII Conferencia Latinoamericana de Informática (CLEI'06)*, pages 158–168. Santiago de Chile, Chile, August, 2006.

[COMCOMJ'09] Raúl Peña-Ortiz, Julio Sahuquillo, Ana Pont, and José Antonio Gil. DWEB model: Representing Web 2.0 dynamism. *Computer Communications Journal (COMCOMJ)*, ISSN 0140-3664, 32(6):1118–1128, 2009. doi: 10.1016/j.comcom.2009.01.002.

[JJPP'10] Raúl Peña-Ortiz, Julio Sahuquillo, Ana Pont, and José Antonio Gil. Generating dynamic workload for web performance evaluation. In *XXI Jornadas de Paralelismo (JJPP'10)*, ISBN 978-84-92812-49-3, pages 711–718. Valencia, Spain, September, 2010.

[WEBIST'11] Raúl Peña-Ortiz, Julio Sahuquillo, Ana Pont, and José Antonio Gil. WEB WORKLOAD GENERATORS: A survey focusing on user dynamism representation. In *7th International Conference on Web Information Systems and Technologies (WEBIST'11)* , ISBN 978-989-8425-51-5, pages 119–126. Noordwijkerhout, The Netherlands, May, 2011.

[JJPP'11] Raúl Peña-Ortiz, José Antonio Gil, Julio Sahuquillo, and Ana Pont. Incorporación del dinamismo del usuario en un benchmark de comercio electrónico. In *XXII Jornadas de Paralelismo (JJPP'11)*, ISBN 978-84-694-1791-1, pages 459–465. La Laguna, Tenerife, Spain, September, 2011.

[CLEI'11] Raúl Peña-Ortiz, José Antonio Gil, Julio Sahuquillo, and Ana Pont. Integración del comportamiento dinámico del usuario en TPC-W. In *XXXVII Conferencia Latinoamericana de Informática (CLEI'11)*, pages 34–48. Quito, Ecuador, October, 2011.

[CLEIej'12] Raúl Peña-Ortiz, José Antonio Gil, Julio Sahuquillo, and Ana Pont. Providing TPC-W with web user dynamic behavior. *CLEI electronic journal (CLEIej)*, ISSN 0717-5000, 15(2):1. August, 2012.

[IEEE-NCA'12] Raúl Peña-Ortiz, José Antonio Gil, Julio Sahuquillo, and Ana Pont. The impact of user's dynamic behavior on web performance. In *11th IEEE International Symposium on Network Computing and Applications (IEEE-NCA'12)*, ISBN 978-0-7695-4773-2/12, pages 143–150. Cambridge, Massachusetts, USA, August, 2012. doi: 10.1109/NCA.2012.9.

[COMCOMJ'13] Raúl Peña-Ortiz, José Antonio Gil, Julio Sahuquillo, and Ana Pont. Analyzing web server performance under dynamic user workloads. *Computer Communications Journal (COMCOMJ)*, ISSN 0140-3664. Accepted but not yet published. doi: 10.1016/j.comcom.2012.11.005.

[SAC'13] Raúl Peña-Ortiz, José Antonio Gil, Julio Sahuquillo, and Ana Pont. The impact of User-Browser Interaction on web performance. In *28th Symposium On Applied Computing (SAC)*. Accepted but not yet published.

[INFOSCI'13] Raúl Peña-Ortiz, José Antonio Gil, Julio Sahuquillo, and Ana Pont. Surfing the web using browser interface facilities: a performance evaluation approach. *Information Sciences (INFOSCI)*, ISSN 0020-0255. Submitted.

APPENDIX A

Acronyms

Web 1.0 First Web Generation 1

Web 2.0 Second Web Generation 1

CBMG Customer Behavior Model Graph 6

FSM Finite State Machine 6

VBMG Visitor Behavior Model Graph 6

EFSM Extended Finite State Machine 7

OSN On-line Social Network 7

SPEC Standard Performance Evaluation Corporation 11

SURGE Scalable URL Reference Generator 12

TPC-W TPC Benchmark^{TM} W 15

EB Emulated Browser 49

WAN Wide Area Network 14

LAN Local Area Network 23

DWEB Dynamic WEB workload model 27

UML Unified Modeling Language 28

GUERNICA Universal Generator of Dynamic Workload under WWW Platforms 33

PCA Personal Content Aggregator 39

WIRT Web Interaction Response Time 52

UBI User-Browser Interaction 9

CORE Computing Research Education 95

ERA Excellence in Research for Australia 95

JCR Journal Citation Reports 95

SJR SCImago Journal Rank 95

DDoS Distributed Denial of Service 95

FE Flash Event 95

APPENDIX B

Glossary

3-tier architecture 3-tier architecture is a client–server architecture that typically comprises a presentation tier, a business or data access tier, and a data tier. 13

AJAX AJAX (Asynchronous JavaScript and XML) is a group of interrelated web development techniques used on the client-side to create asynchronous web applications. 5

ASP Active Server Pages, also known as Classic ASP, was Microsoft's first server-side script engine for dynamically generated web pages. 11

blurker A blog lurker (blurker) is someone who reads a lot of blogs but never posts any comments. 6

CGI The Common Gateway Interface (CGI) is a standard method for web server software to delegate the generation of dynamic web content to executable files, which are known as CGI scripts. 10

ERP Enterprise Resource Planning (ERP) is an application used by large organizations to manage inventory, resources, and business processes across departments. 31

ISAPI Internet Server Application Programming Interface (ISAPI) is an N-tier application programming interface of Internet Information Services (IIS). The most prominent application of IIS and ISAPI is Microsoft's web server. 10

J2EE Java Platform Enterprise Edition (J2EE) is an enterprise Java computing platform that provides an API and runtime environment for developing and running enterprise software, including network and web services, and other large-scale, multi-tiered, scalable, reliable, and secure network applications. 5

JSP JavaServer Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. 11

L4/7 switches Layer 4/7 switching refers to content or application aware intelligent switching and deep packet inspection of IP application traffic. By increasing performance, security, availability, and scalability through content-aware intelligence, Layer 4/7 switching has become an indispensable component of new data center and network infrastructures. 17

NSAPI Netscape Server Application Programming Interface (NSAPI) is an application programming interface for extending web server software. 10

PHP General-purpose server-side scripting language originally designed for web development to produce dynamic web pages. 11

APPENDIX C

Bibliography

[ACC+02] Cristiana Amza, Anupam Chanda, Alan L. Cox, Sameh Elnikety, Romer Gil, Karthick Rajamani, Emmanuel Cecchet, Julie Marguerite, and Willy Zwaenepoel. Specification and implementation of dynamic web site benchmarks. In *Workshop on Workload Characterization*, pages 3–13, November 2002.

[AJK05] Anne Aula, Natalie Jhaveri, and Mika Käki. Information Search and Re-access Strategies of Experienced Web Users. In *Conference on World Wide Web*, pages 583–592, May 2005.

[ASF12] ASF (Apache Software Foundation). Apache JMeter™ [online]. 2012. Available from: `http://jmeter.apache.org/` [cited July 2012].

[ASW06] Ernesto Arroyo, Ted Selker, and Willy Wei. Usability tool for analysis of web designs using mouse tracks. In *Conference on Human factors in computing systems*, pages 484–489. ACM, April 2006.

[Aus06] Australian Research Council. Australian Ranking of ICT conferences [online]. 2006. Available from: `http://core.edu.au/rankings/ConferenceRankingMain.html` [cited April 2006].

[Aus10] Australian Research Council. Archived material from ERA 2010 [online]. 2010. Available from: `http://www.arc.gov.au/era/era_2010/archive/era_journal_list.htm` [cited October 2012].

[Bar98] Paul Barford. The SURGE Web Workloaad Generator [online]. 1998. Available from: `http://pages.cs.wisc.edu/~pb/software_data.html` [cited 2005].

[BBBC99] Paul Barford, Azer Bestavros, Adam Bradley, and Mark E. Crovella. Changes in Web client access patterns: Characteristics and caching implications. *World Wide Web Internet and Web Information Systems*, 2(1):15–28, 1999.

[BC97] Paul Barford and Mark Crovella. An architecture for a WWW workload generator. In *World Wide Web Consortium Workshop on Workload Characterization*, October 1997.

[BC98] Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. In *SIGMETRICS '98/PERFORMANCE '98, Joint International Conference on Measurement and Modeling of Computer Systems*, pages 151–160, June 1998.

[BCCPB07] Mercedes Blázquez-Cívico, Jesús Contreras-Cino, Raúl Peña-Ortiz, and V. Richard Benjamins. *Trends on Legal Knowledge, the Semantic Web and the Regulation of Electronic Social Systems*, chapter Visualization of Semantic Content. European Press Academic Publishing, 2007.

[BD99] Gaurav Banga and Peter Druschel. Measuring the capacity of a Web server under realistic loads. *World Wide Web Internet and Web Information Systems*, 2(1/2):69–83, 1999.

[Bla05] Michael Blakeley. Deluge: website stress test tool [online]. 2005. Available from: `http://deluge.sourceforge.net/` [cited 2010].

[BRdMCA09] Fabrício Benevenuto, Tiago Rodrigues de Magalhães, Meeyoung Cha, and Virgilio A.F Almeida. Characterizing user behavior in online social networks. In *Internet Measurement Conference*, pages 49–62, November 2009.

[CACB97] Liu Chang, Kirk P. Arnett, Louis M. Capella, and Robert C. Beatty. Web sites of the Fortune 500 companies: Facing customers through home pages. *Information & Management Journal*, 31(6):335–345, January 1997.

[CAJ+99] Hua Chen, Marc Abrams, Tommy Johnson, Anup Mathur, Ibraz Anwar, and John Stevenson. Wormhole caching with HTTP PUSH method for a satellite-based web content multicast and replication system. In *Web Caching Workshop*, 1999.

[CG00] Andy Cockburn and Saul Greenberg. Issues of Page Representation and Organisation in Web Browser's Revisitation Tools. *Australasian Journal of Information Systems*, 7(2), May 2000.

[CK08] Graham Cormode and Balachander Krishnamurthy. Key differences between Web 1.0 and Web 2.0. *First Monday Journal*, 13(6), 2008.

[CMJ02] A. Cockburn, B. McKenzie, and M. JasonSmith. Pushing back: evaluating a new behaviour for the back and forward buttons in

web browsers. *International Journal of Human-Computer Studies*, 57(5):397–414, November 2002.

[CMZ02] Emmanuel Cecchet, Julie Marguerite, and Willy Zwaenepoel. Performance and scalability of EJB applications. In *Conference on Object-oriented programming, systems, languages, and applications*, pages 246–261. ACM Request Permissions, November 2002.

[Con03] Jim Conallen. *Building Web Applications With UML*. Addison-Wesley Professional, 2003.

[CPCP01] Ed H. Chi, Peter Pirolli, Kim Chen, and James Pitkow. Using information scent to model user information needs and actions and the Web. In *Conference on Human factors in computing systems*, pages 490–497, 2001.

[CRML01] Harold W. Cain, Ravi Rajwar, Morris Marden, and Mikko H. Lipasti. An Architectural Evaluation of Java TPC-W. In *International Symposium on High-Performance Computer Architecture*, page 229, January 2001.

[DLDP03] Giuseppe A. Di Lucca and Massimiliano Di Penta. Considering browser interaction in web application testing. In *International Workshop on Web Site Evolution*, pages 74–81. IEEE, September 2003.

[DMA+08] Fernando Duarte, Bernardo Mattos, Jussara Almeida, Virgilio A.F Almeida, Mariela Curiel, and Azer Bestavros. Hierarchical characterization and generation of blogosphere workloads. Technical report, October 2008.

[DMB01] Ronald C. Jr Dodge, Daniel A. Menascé, and Daniel Barbará. Testing e-commerce site scalability with TPC-W. In *Computer Measurement Group Conference*, pages 457–466, December 2001.

[Eri99] Peter Eriksson. PTester [online]. 1999. Available from: `http://www.lysator.liu.se/~pen/ptester/` [cited 2005].

[For12] Florian Forster. Collectd – The system statistics collection daemon [online]. 2012. Available from: `http://collectd.org/` [cited June 2012].

[FP01] Sally Floyd and Vern Paxson. Difficulties in simulating the Internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, August 2001.

[FPZ07] Pierfrancesco Foglia, Cosimo Antonio Prete, and Michele Zanda. Modelling Public Administration Portals. In *Encyclopedia of Portal Technologies and Applications*, pages 606–614. 2007.

[Ful12] Jeffrey Fulmer. SIEGE [online]. 2012. Available from: `http://www.joedog.org/siege-home/` [cited July 2012].

[GBGP10] Sharad Goel, Andrei Broder, Evgeniy Gabrilovich, and Bo Pang. Anatomy of the long tail: ordinary people with extraordinary tastes. In *ACM International Conference on Web Search and Data Mining*, pages 201–210. Yahoo Research, February 2010.

[GG03] Daniel F. García and Javier García. TPC-W e-commerce benchmark evaluation. *Computer*, 36(2):42–48, February 2003.

[Goo12] Google. Google Scholar [online]. 2012. Available from: `http://scholar.google.com/` [cited October 2012].

[Hen09] Jim Hendler. Web 3.0 Emerging. *Computer*, 42(1):111–113, 2009.

[HP12a] Hewlett-Packard. HP LoadRunner [online]. 2012. Available from: `http://www8.hp.com/us/en/software-solutions/software.html?compURI=1175451` [cited July 2012].

[HP12b] Hewlett-Packard. Load factor: performance testing for web applications business white paper. Business white paper, June 2012.

[HP12c] Hewlett-Packard. Setting the pace for testing modern applications. Business white paper, May 2012.

[HW10] Jeff Huang and Ryen W. White. Parallel browsing behavior on the web. In *Conference on Hypertext and hypermedia*, pages 13–18. ACM, June 2010.

[Jup06] Jupiter Research. Retail Web Site Performance: Consumer Reaction to a Poor Online Shopping Experience. Technical report, Akamai, June 2006.

[LY96] David Lee and Mihalis Yannakakis. Principles and methods of testing finite state machines-a survey. *Proceedings of the IEEE*, 84(8):1090–1123, 1996.

[MA00] Daniel A. Menascé and Virgilio A.F Almeida. *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning*. Prentice Hall, 2000.

[MF12] The Measurement Factory. Web Polygraph [online]. 2012. Available from: `http://www.web-polygraph.org/` [cited July 2012].

[Mid04] Julian T.J Midgley. AUTOBENCH [online]. 2004. Available from: `http://www.xenoclast.org/autobench` [cited 2005].

[Min02a] Mindcraft. WebStone 2.x Benchmark Description [online]. 2002. Available from: `http://www.mindcraft.com/webstone/ws201-descr.html` [cited 2005].

[Min02b] Mindcraft. WebStone: The Benchmark for Web Servers [online]. 2002. Available from: `http://www.mindcraft.com/webstone/` [cited 2005].

[MJ98] David Mosberger and Tai Jin. httperf—a tool for measuring web server performance. *Performance Evaluation Review*, 26(3), December 1998.

[MJ08] David Mosberger and Tai Jin. HTTPERF [online]. 2008. Available from: `http://www.hpl.hp.com/research/linux/httperf/` [cited 2010].

[NdlOG+05] Ingrid Juliana Niño, Bernardo de la Ossa, José Antonio Gil, Julio Sahuquillo, and Ana Pont. CARENA: a tool to capture and replay web navigation sessions. In *End-to-End Monitoring Techniques and Services*, pages 127–141, May 2005.

[ONUI09] Moriyoshi Ohara, Priya Nagpurkar, Yohei Ueda, and Kazuaki Ishizaki. The Data-centricity of Web 2.0 Workloads and its Impact on Server Performance. In *IEEE International Symposium on Workload Characterization*, pages 133–142, October 2009.

[Ora12] Oracle. MySQL 5.1 Reference Manual: How the Query Cache Operates [online]. 2012. Available from: `http://dev.mysql.com/doc/refman/5.1/en/query-cache-operation.html` [cited April 2012].

[Ore07] Tim Oreilly. What is Web 2.0: Design patterns and business models for the next generation of software. *Communications & Strategies*, 1:17, January 2007.

[OWHM07] Hartmut Obendorf, Harald Weinreich, Eelco Herder, and Matthias Mayer. Web page revisitation revisited: implications of a long-term click-stream study of browser usage. In *Conference on Human factors in computing systems*, pages 597–606, April-May 2007.

[PP99] Peter L.T. Pirolli and James E. Pitkow. Distributions of surfers' paths through the World Wide Web: Empirical characterizations. *World Wide Web*, 2(1/2):29–45, 1999.

[Rad12] RadView Software. WebLOAD [online]. 2012. Available from: `http://www.radview.com/product/Product.aspx` [cited July 2012].

[RJB99] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. 1999.

[Rod09] Pablo Rodriguez. Web Infrastructure for the 21st Century. In *Conference on World Wide Web*. Telefónica I+D, January 2009.

[RS00] Frederick F. Reichheld and Phil Schefter. E-Loyalty: Your Secret Weapon on the Web. *Harvard Business Review Magazine*, 78:105, December 2000.

[RSD+12] Kira Radinsky, Krysta Svore, Susan Dumais, Jaime Teevan, Alex Bocharov, and Eric Horvitz. Modeling and predicting behavioral dynamics on the web. In *Conference on World Wide Web*, pages 599–608. ACM, April 2012.

[RW03] Alex Rousskov and Duane Wessels. High performance benchmarking with Web Polygraph. *Software: Practice and Experience*, 1:1–10, 2003.

[RWC99] Alex Rousskov, Duane Wessels, and Glenn Chisholm. The First IRCache Web Cache Bake-off. In *Web Caching Workshop*, 1999.

[SAAF08] Fabian Schneider, Sachin Agarwal, Tansu Alpcan, and Anja Feldmann. The new web: Characterizing AJAX traffic. In *International Conference on Passive and Active Network Measurement*, pages 31–40, April 2008.

[SAP02] Srini S. Srinivasan, Rolph Anderson, and Kishore Ponnavolu. Customer loyalty in e-commerce: an exploration of its antecedents and consequences. *Journal of Retailing*, 78(1):41–50, 2002.

[SCI11] SCImago Lab. SCImago Journal and Country Rank [online]. 2011. Available from: `http://www.scimagojr.com` [cited December 2012].

[SCK03] Weisong Song Shi, Elizabeth Collins, and Vijay Karamcheti. Modeling object characteristics of dynamic web content. *Journal of Parallel and Distributed Computing*, 63(10):963–980, October 2003.

[SKF06] Mahnaz Shams, Diwakar Krishnamurthy, and Behrouz Far. A model-based approach for testing the performance of web applications. In *International Workshop on Software Quality Assurance*, pages 54–61. ACM, November 2006.

[SPE09] SPEC. SPEC's Benchmarks for Web Servers [online]. 2009. Available from: `http://www.spec.org/benchmarks.html#web` [cited 2012].

[SZ00] Young-Woo Seo and Byoung-Tak Zhang. Learning user's preferences by analyzing Web-browsing behaviors. In *International conference on autonomous agents*, pages 381–387. ACM, June 2000.

[TGG+09] Georgios Tselentis, Alex Galis, Anastasius Gavras, Srdjan Krco, Volkmar Lotz, Elena Simperl, Burkhard Stiller, and Theodore Zahariadis. *Towards the Future Internet: A European Research Perspective*. IOS Press, May 2009.

[Tha08] Andrew Thatcher. Web search strategies: The influence of Web experience and task type. *Information Processing & Management*, 44(3), May 2008.

[Tho11] Thomson Reuters. Journal Citation Reports [online]. 2011. Available from: `http://thomsonreuters.com/products_services/science/science_products/a-z/journal_citation_reports/` [cited December 2012].

[Tra02a] Transaction Processing Performance Council. TPC Benchmark™ W [online]. 2002. Available from: `http://www.tpc.org/tpcw/` [cited 2005].

[Tra02b] Transaction Processing Performance Council. TPC Benchmark™ W Specification. Version 1.8. Technical report, February 2002.

[WDG11] Geoff Wong, Mick Dwyer, and Jon Gifford. Hammerhead 2.0 [online]. 2011. Available from: `http://sourceforge.net/projects/hammerhead/` [cited July 2012].

[WOHM06] Harald Weinreich, Hartmut Obendorf, Eelco Herder, and Matthias Mayer. Off the beaten tracks: exploring three aspects of web navigation. In *Conference on World Wide Web*, pages 133–142, June 2006.