Dweb model: Representing Web 2.0 dynamism

Raúl Peña-Ortiz a,*, Julio Sahuquillo b,1, Ana Pont b,1, José A. Gil b,1

a Instituto Tecnológico de Informática (ITI), Ciudad Politécnica de la Innovación, Edif. BG – Acc. B. Office D3, 2nd Floor, Camí de Vera s/n, 46022 Valencia, Spain
b Polytechnic University of Valencia, Computer Engineering Department (DISCA), Camí de Vera s/n, 46022 Valencia, Spain

Article history:
Received 20 October 2008
Accepted 23 January 2009
Available online xxxxxx

Abstract

Due to the increase in the number and popularity of applications such as e-commerce or on-line booking systems, typical of the Web 2.0, dynamic contents are becoming more and more frequent. This trend suggests the review of widely accepted paradigms and models for the World Wide Web. As a system that is continuously changing, both in the offered applications and in its infrastructure, performance evaluation studies are a main concern to provide sound proposals when designing new web-related systems. Although the dynamism in the workload characterization has also been tackled in previous research, it has not been modeled in a precise way yet because of its complex nature.

In this paper we propose the Dweb model which represents the dynamism of current web applications in the workload characterization. Dweb is based on three main concepts that allow to model dynamic web workloads: (i) how to model the users' dynamism, (ii) how to represent the different roles that users play in the web, and (iii) how to model the continuous changes in the users' behavior.

In a previous work [7,8] we introduced a first approach to model the dynamism of the web workload based on users' behavior models. This paper extends the previous model to characterize the web workload generation and distribution, taking into account the dynamism in web users' behavior, the load balance among different hosts, and also the definition of different generation patterns. Furthermore, a dynamic workload generator has been implemented to show a practical application of this model, which has been illustrated through a case study.

The remainder of this paper is organized as follows. Section 2 discusses the reasons that motivated us to perform this work. Section 3 presents the proposed workload model in a formal way. Section 4 describes its implementation in the proposed workload designing new web-related systems [4], such as web services, web servers, or proxy policies. As in any performance evaluation process, the use of an accurate and representative workload model is a main concern in order to guarantee the representativeness and validity of the results. In the case of the WWW, the implicit dynamism of the users' behavior makes it difficult to design accurate workload models representing users' navigation. For example, a high number of users' navigation sessions begin searching a dynamic resource in a specialized site [5] and continues visiting one or more sites depending on the results of the search. In general, the main challenges when modeling current web workloads are: (i) how to model the users' dynamism, (ii) how to represent the different roles that users play in the web, and (iii) how to model the continuous changes in the users' behavior [6].

In this paper we propose the Dweb model which represents the dynamism of current web applications in the workload characterization. Dweb is based on three main concepts that allow to model dynamic web workloads: (i) how to model the users' dynamism, (ii) how to represent the different roles that users play in the web, and (iii) how to model the continuous changes in the users' behavior.

In a previous work [7,8] we introduced a first approach to model the dynamism of the web workload based on users' behavior models. This paper extends the previous model to characterize the web workload generation and distribution, taking into account the dynamism in web users' behavior, the load balance among different hosts, and also the definition of different generation patterns. Furthermore, a dynamic workload generator has been implemented to show a practical application of this model, which has been illustrated through a case study.

The remainder of this paper is organized as follows. Section 2 discusses the reasons that motivated us to perform this work. Section 3 presents the proposed workload model in a formal way. Section 4 describes its implementation in the proposed workload designing new web-related systems [4], such as web services, web servers, or proxy policies. As in any performance evaluation process, the use of an accurate and representative workload model is a main concern in order to guarantee the representativeness and validity of the results. In the case of the WWW, the implicit dynamism of the users' behavior makes it difficult to design accurate workload models representing users' navigation. For example, a high number of users' navigation sessions begin searching a dynamic resource in a specialized site [5] and continues visiting one or more sites depending on the results of the search. In general, the main challenges when modeling current web workloads are: (i) how to model the users' dynamism, (ii) how to represent the different roles that users play in the web, and (iii) how to model the continuous changes in the users' behavior [6].

In a previous work [7,8] we introduced a first approach to model the dynamism of the web workload based on users' behavior models. This paper extends the previous model to characterize the web workload generation and distribution, taking into account the dynamism in web users' behavior, the load balance among different hosts, and also the definition of different generation patterns. Furthermore, a dynamic workload generator has been implemented to show a practical application of this model, which has been illustrated through a case study.

The remainder of this paper is organized as follows. Section 2 discusses the reasons that motivated us to perform this work. Section 3 presents the proposed workload model in a formal way. Section 4 describes its implementation in the proposed workload designing new web-related systems [4], such as web services, web servers, or proxy policies. As in any performance evaluation process, the use of an accurate and representative workload model is a main concern in order to guarantee the representativeness and validity of the results. In the case of the WWW, the implicit dynamism of the users' behavior makes it difficult to design accurate workload models representing users' navigation. For example, a high number of users' navigation sessions begin searching a dynamic resource in a specialized site [5] and continues visiting one or more sites depending on the results of the search. In general, the main challenges when modeling current web workloads are: (i) how to model the users' dynamism, (ii) how to represent the different roles that users play in the web, and (iii) how to model the continuous changes in the users' behavior [6].

In a previous work [7,8] we introduced a first approach to model the dynamism of the web workload based on users' behavior models. This paper extends the previous model to characterize the web workload generation and distribution, taking into account the dynamism in web users' behavior, the load balance among different hosts, and also the definition of different generation patterns. Furthermore, a dynamic workload generator has been implemented to show a practical application of this model, which has been illustrated through a case study.
generator. Section 5 includes an illustrative example. Finally, Section 6 draws some concluding remarks.

2. Motivation and related work

From early in the web, different studies attempting to model users’ behavior have been published in the literature. This topic is an important challenge for web performance evaluation because the increasing popularity of new web applications and services produces continuous changes in the users’ navigation patterns, which makes it difficult to design workload models. The main aims of these models are: (i) to reproduce client’s workload in order to evaluate the performance of web systems, and (ii) to test web applications and services according to the behavior of their potential clients.

One of the first studies describing web users’ behavior is the work done by Pitkow et al. [9], which describes different types of web users’ behaviors according to the results obtained from a set of questionnaires and interviews. In other studies they analyze log files from the client site [10,11] and the server site [12], trying to empirically understand the dynamics of the web and its users by devising smart algorithms [13]. Another effort to model users’ behavior appears in the context of web applications design, where a navigation model is required to organize the contents in order to guarantee a successful access by users (i.e., avoiding wrong navigations) [14–16]. Nowadays users’ navigation patterns are widely used to support dynamism in users’ behavior models for adaptive hypermedia applications [17,18]. Users’ models can also be used when studying the relation among web navigation metrics (measures of lostness and successfulness on dynamic web navigations) [19,20].

When doing performance evaluation studies, web systems or web applications are mainly assessed by using either traces or parametrizable workload models. Traces log the sequence of HTTP requests and commands issued by real users and received by a web application or a web server during a period of time under given conditions. Traces are obtained in a specific environment (e.g., server process speed or network bandwidth) where a specific application is run. This means that if any system parameter varies, the resulting trace might differ. Consequently, trace-based models are not flexible enough and are not appropriate to model changes in client’s behavior. On the other hand, parametrizable workload models are abstractions of the real workload that reproduce users’ behavior, hiding those characteristics that are not relevant for a particular study. The model represents a set of clients using a particular web application. It is accurate enough when it reproduces client’s behavior and the web application performs as if real clients were navigating. They are designed to define sequences of HTTP requests similar to real sequences. In order to model different scenarios, workload models must be configured by setting the corresponding input parameters, which specify the characteristics of the workload to be defined.

Some efforts have been addressed to represent dynamic users’ behavior in workloads characterization studies [21–24]. However, user dynamism is complex by nature, and results are still far from being precise and satisfactory enough.

Using the users’ behavior models in the web workload characteristics will provide more accurate tools to model and evaluate web performance. In this sense, Menascé et al. [25] introduced the Customer Behavior Model Graph to be used as input to a workload model of an e-business site.

Workload generators are software products based on workload models that are designed and implemented to generate HTTP requests. Some studies [26,27] confirm how difficult it is to generate representative web requests, especially when trying to model some particular characteristics in a dynamic web site, and how these characteristics impact on the client’s behavior. Workload generators are flexible tools for tuning or performing capacity planning studies but, to the knowledge of the authors, current generators only present a partial capability to model users’ dynamic behavior [7,8].

These shortcomings motivated us to propose a new workload model called Dweb (Dynamic web workload model), which allows to represent the dynamism of web users, introducing users’ dynamic behavior models in the workload characterization. Dweb has been implemented in a new workload generator called GUER-NICA (Universal Generator of Dynamic Workload under WWW Platforms).

3. Modeling dynamism in Web 2.0: the Dweb model

The main objective of the proposed Dweb model is to accurately define the dynamism in the new web (i.e., Web 2.0). Three main concepts allow Dweb to model dynamic workload: navigation, workload test and workload distribution. The two former characterize the dynamism in workload definition by means of web users’ behavior models. The latter concept represents the physical distribution of users in the web.

3.1. Navigation concept

The navigation concept was introduced in a previous work [7]. Basically, a navigation defines the dynamic users’ behavior while interacting with the web. Let us illustrate this concept with an example. Assume a typical navigation where a user looks for specific information. This kind of navigation typically begins running a web searcher to make a query. Then, if the query is successfully resolved (i.e., a list of results is provided) it is likely that the user will either (i) visit the first site of the list, or (ii) refine the search (to reduce the resulting list). On the other hand, if the response time is longer than a given value, the query might be canceled. Notice that each user request depends not only on the response itself but also on its associated characteristics (e.g., response time length or content amount).

Fig. 1 depicts a tree representing this navigation, which consists of four main steps:

(1) The user requests the main page of the web searcher, Google site, in the example.
(2) Once the page is displayed, the user fulfills the form during a given time, namely, the users think time. To model this time, different approaches have been used in the existing workload generators [28,29]. In this case, this time has been defined by a Gaussian distribution with 3500 ms average and 1500 ms standard deviation.
(3) After that time, the user makes the desired query.
(4) Finally, the obtained results, if any, are analyzed. This step is represented by using two branches in the tree. If the search provides results (left branch) the user analyzes them, accesses the first site shown, and finishes the navigation (black dot). However, if no results are provided, the navigation finishes (right branch).

The navigation concept is not just limited to human users, since it can be further applied to any web client, e.g., software automatons. As automatons follow a given pattern, they are easier to model than users.

Formally, we define a navigation N as a sequence of n URLs of HTTP requests where each visited URL depends on the previously visited one; that is:
\(N = \{url_1, url_2, \ldots, url_n\} /\forall i = 2 \ldots n : url_i \) depends on url_{k} content for \(k < i\).

where url_{i} refers both to the content associated with the resource \(i\) and too its derived information; e.g., response time length or user satisfaction level.

### 3.2. Workload test concept

The workload test concept used in this work was introduced in [8]. A workload test consists of the set of navigations launched during the simulation process. These navigations can be concurrently executed since they can be issued by independent threads representing parallel navigations of a given user or different users. As mentioned above, this concept also models automatic processes interacting with web contents (e.g., content aggregation processes).

Formally, we define the workload test \(T(C, \varphi : C \times C \rightarrow N, I)\), where:

- \(C = \{n_1, n_2, \ldots, n_k\} \) with \(n_i \in N\). \(C\) is the set of navigations.
- \(\varphi : C \times C \rightarrow N\). \(\varphi\) is the function which provides the next navigation to be executed.
- \(I\) represents the number of users or navigations to be issued in parallel.

The workload test has been implemented as an automaton where the nodes represent navigations and the arcs indicate the transitions between navigations. The arc weight is the probability to take such arc. A node represents a given user’s behavior and the arcs change among user’s behaviors.

Fig. 2 shows the graph of a straightforward workload test modeling two different users’ behaviors: searcher and surfer. Searcher behavior refers to the users who start navigations with a query in a given searcher engine, while surfer behavior models the users that navigate the web by following direct hyperlinks [12]. The graph shows how a typical user who begins their web session by looking for specific information might continue the navigation. After the searching is performed (left node), the user has three options: (i) to make a deeper search (i.e., remaining in the same node) with a given probability (25%), (ii) to change his or her behavior (i.e., moving to the right node), or (iii) to finish the navigation. Notice that the arc direction in the graph has no constraint, that is, users can change their behavior in any way (they can move towards, move back, or remain unchanged).

### 3.3. Workload distribution concept

Finally, the workload distribution refers to a set of workload tests that are concurrently executed by one or more users (generators) in different nodes or in different generation machines when simulating the web client’s behavior. This concept focuses on distributing the execution of simulated users among the client nodes of the WWW.

Formally, a workload distribution \(D(T, G, \gamma : G \rightarrow T)\) defines:

- The workload tests to be executed, which are \(\{t_1, t_2 \ldots t_k\} \) with \(t_i \in T\).
4. GUERNICA

GUERNICA [7] is a workload generator with the ability to precisely generate the dynamic workload of Web 2.0 by implementing the Dweb model. The generator was developed as a result of the cooperation among the Web Performance Research Group (Polytechnic University of Valencia), Intelligent Software Components, and the Instituto Tecnológico de Informática; thereby, bridging the gap between academia and industry.

GUERNICA, as shown in Fig. 3 is a software made up of three main applications: workload generators, performance evaluator and performance tests planner. Each application permits to carry out the workload test process through four main phases (see Fig. 4):

1. Defining the client behavior (users or automatons) by using navigations as defined in Section 3.1.
2. Defining the workload of the target site by using workload tests and workload distribution as defined in Sections 3.2 and 3.3, respectively.
3. Executing the workload tests gathering performance statistics.

4.1. GUERNICA applications

GUERNICA consists of a set of applications which work concurrently in the different phases of a web performance evaluation project. There are three main applications: planner, generator, and probe.

The planner application controls and plans the workload test. Its goals are: (i) to define the test case, (ii) to plan test cases execution, (iii) to monitor on-line results (generated by distributed clients), and (iv) elaborate and combine distributed results.

Generator applications generate workload for stressing purposes. They are not required to be executed in the same machine as the planner. The planner configures generators and orders their execu-
tion. Then each generator executes the associated navigations and notifies the results to the planner, which performs the corresponding graphical representation. Alternatively, a generator must stop its work when the planner asks for it.

The probe client application is aimed at evaluating, from the user’s point of view (e.g., response time), the major functionalities of a given web application while it is stressed by the workload generators.

Results collected by generators and probe clients are given to the planner, which groups, classifies and reaches a consensus among them in order to obtain a uniform set of results. These results can be represented in different formats.

In addition to these applications, there are two external assistant applications: CARENA and SemViz. CARENA is a Mozilla plugin that captures users’ navigations. Then GUERNICA converts CARENA navigations to its internal representation. For 3D representation purposes, GUERNICA represents the results in an external format based on an ontology that SemViz (implemented as an applet) can read and show graphically. SemViz visualizes knowledge based on ontologies using 3D technology.

4.2. Workload generation models

The workload distribution concept allows to define different distributed workload models. These models are used by a planner (main process) to coordinate a set of secondary processes that generates the final workload. Fig. 5 shows the three main workload generation models: ideal distributed model, advanced distributed model, and basic model.

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 locates the planner, the generators, and the probe client in different machines. The main drawback of this model 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 considered 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 model will introduce noise in the tests. This model can be used as a first tentative in the workload generation to identify performance bottlenecks in web applications. Nevertheless, ideal or advanced approaches are required for a better performance evaluation.

4.3. Architecture

The proposed tool has been written in Java language, so it runs independently of the execution platform. Fig. 6 shows a block diagram of the different modules implemented in the GUERNICA core.

The three main GUERNICA applications (i.e., generators, probe, and planner) have been implemented by using web services technology. A web application, namely the web planner, uses these services to provide the GUERNICA functionalities. As mentioned above, SemViz was integrated in the web planner as an applet, and CARENA as a Mozilla extension.

The core module defines the navigation and the workload test engines that the generators will use to reproduce the dynamic workload. The navigation engine defines an API which allows to control automatic navigations depending on the contents of the previous requests. Currently, the core supports the iSOCO – GETsee technology2 to implement the navigation API, but it can operate with any technology fulfilling this API.

4.4. Main features

GUERNICA supports, totally or partially, the main features introduced in well-known workload generators proposed in the open literature, as well as the capability to represent users’ dynamic behavior.

Fig. 4. Testing phases in GUERNICA.
Regarding the totally supported features, the generator presents distributed architecture (distributed models) based on web services. That is, GUERNICA allows to distribute the generation processes among different nodes which emulate users working on different machines. The generator implements the Dweb model that offers important capabilities. For instance, it permits to represent the structure of the web application business logic and to model users’ dynamic behavior as well as its continuous changes. Dweb model introduces some client variables to parameterize the user behavior (e.g., the user think time by means of a Gaussian distribution). GUERNICA also organizes the workload in categories or types, each of them modeling a given user profile by applying the workload test concept. Workload tests can be used as web application functional tests.

Concerning the GUERNICA features partially supported, it can be mentioned its ease of use (e.g., CARENA to define user navigation), the service 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). Finally, it should be noted that a significant part of the code (the applications and the core packages that do not use GETsee technology) has been written under an open source license.

Further details about GUERNICA features and a study comparing them against other generators can be found in [8].

5. Case study: modeling dynamic workload

This section presents a case study where the dynamic workload has been modeled to evaluate the performance of a real web application. The objective of this section is not to present a detailed performance evaluation but to show how GUERNICA can be used in this kind of studies.

The evaluated application is QUOTES, an advanced solution for electronic negotiation given by iSOCO, which supports: the standardization of the products and services acquisition process, the standardization of quotation requests in the market to identify

---

Fig. 6. GUERNICA architecture.

Fig. 7. QUOTES application.
new suppliers, the comparison of contract conditions, or the increase of competitive conditions.

Fig. 7 shows the user interface of the application, which includes user management, messenger system between users, product management, negotiation management, etc.

In this section we analyze its performance by measuring the throughput and the response time, both in normal and overload conditions.

5.1. Test case definition

The performance test defined represents a typical day-by-day working session with QUOTES. The number of users who navigate the application according to the navigations explained below range from 1 to 25.

- **Messages console consulting** (Fig. 8a): A user accesses the home site and waits for a time (5000 ms). Then he or she logs in the site and waits again for another specific time. Finally, the user surfs to the message console, finishes the navigation after a thinking time (5000-ms mean Gaussian distribution) and logs out.

- **Negotiations console consulting** (Fig. 8b): A user accesses the home site and waits for a time (5000-ms mean Gaussian distribution), she or he logs in the site and waits again for a time. Then the user navigates to the messages console. If the console contains negotiation messages, the user accesses a detail (DOCUMENT_DETAIL) and logs out after a thinking time (Gaussian distribution as 5000 ms). If the console does not have negotiations, the user goes to the logout page.

Fig. 8. Messages and negotiations console consulting navigations.
The users’ behavior does not change: a user navigates into the messages or the negotiations console and finishes his or her navigations. Hence the workload test must define subsets of users according to their types of navigation. Simulations assume that all users are always navigating the site, repeating the selection and execution of a navigation. We obtained results widely varying the number of repetitions and observed that with 50 repetitions the generated workload is significant.

Fig. 9 shows the workload test indicating that one half of the users navigate into messages console, and the other half into negotiations console.

Hardware constraints (only two machines were dedicated to the performance study) forced us to choose the basic model generation previously described, where all workload tests were run in the same machine. The test measures the throughput and the response time. Notice that we do not need a probe client for a specific navigation because we are interested in the global site performance.

5.2. Test case implementation

GUERNICA was implemented by using a model language based on XML labels to define navigations, workload tests and workload distribution.

Listing 1 shows the content of the XML file containing the navigation of the negotiations console consulting. It defines a navigation that accesses a parameterized host, logs in with user and password params, goes to the negotiations console, accesses a negotiation detail (if possible) and finally logs out. PCA-Plugin is the XML file that defines, by using the GETSee technology, the dynamic user’s behavior.

A set of global statistics is obtained, e.g., the total GUERNICA execution time or the navigation time. For each HTTP request, the test execution produces distinct statistics, e.g., the GET or POST HTTP method, the time for establishing the connection, the transfer time, the user think time, the content size, accessed URL, and the successfulness when making the connection (i.e., Stablished). The TimeStamp defines the instant when the request is resolved, and it is used to obtain the throughput index.

Listing 2 presents the XML file that defines the workload test corresponding to the automaton of Fig. 9. It shows how the user starts with a navigation in the messages console or in the negotiations console and then finishes his or her activity. The test defines a loop of 50 repetitions and a number of 25 users. Therefore, it creates 25 execution threads, each one executing 50 times the automaton.

Finally, Listing 3 shows the XML file that defines the workload test planning. This file configures GUERNICA in the basic model gen-

---

Listing 1. Negotiation console consulting navigation XML file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Navigation id="quotes-navigation-negotiation_console">  
  <InputData>
    <Param name="client" value="xxx"/>
    <Param name="host">
      value="http://becaril.bcn.isoco.net:8080/quotes"/>
    </Param>
    <Param name="user" value="cl"/>
    <Param name="password" value="cl"/>
  </InputData>
  <ExecutionCode>
    <PCA-Plugin name="GoConsoleFrame1.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 name="TimeStamp"/>
  </StatisticsConfiguration>
</Navigation>
```

---
eneration with only a workload generator process (generator-1) in a single machine.

5.3. Test case results

Once the performance test has been run, we obtain the traces of conducted navigations and their HTTP requests, each having a different outcome.

Figs. 10 and 11 show the average response time and the throughput obtained, respectively. As expected, response time increases with the number of requests. Concerning the throughput metric, it also rises until a given threshold bound, which is reached with 20 users. After that point, the throughput decreases with the number of users.

6. Conclusions

The evolution of the World Wide Web from hypermedia information repositories to hypermedia distributed applications and web services architectures has involved new features in the current and incoming Web 2.0. Web applications mainly focus on users who are potential clients, by providing an experience closer to desktop applications. One of the most important mechanisms to achieve this goal is the dynamism present in the web content.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<WorkloadTest id="quotes-workload-test">
  <UsersNumber>25</UsersNumber>
  <Loop count="50"/>
  <Loop forEver="false"/>

  <NavigationGraph>
    <InitialNavigations>
      <InitialNavigation id="quotes-navigation-negotiation_console" probability="0.5"/>
      <InitialNavigation id="quotes-navigation-messages_console" probability="0.5"/>
    </InitialNavigations>
    <NavigationTransitions>
      </NavigationTransitions>
  </NavigationGraph>
</WorkloadTest>
```

Listing 2. QUOTES workload test.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<WorkloadPlannerConfiguration>
  <PlannerIdentifier>quotes</PlannerIdentifier>
  <WorkloadGenerators>
    <WorkloadGenerator>
      <Id>generator-1</Id>
    </WorkloadGenerator>
  </WorkloadGenerators>
  <WorkloadTestAssignations>
    <WorkloadTestAssignment testId="quotes-workloadtest" generatorId="generator-1"/>
  </WorkloadTestAssignations>
</WorkloadPlannerConfiguration>
```

Listing 3. QUOTES workload distribution.
(e.g., personalized contents or publicity) which matches the dynamism of users and automaton behaviors in the current web. This feature is the main shortcoming when modeling real web workload.

We have analyzed the characteristics of a representative subset of the state-of-the-art users' behavior models, focusing on those able to represent the user dynamism of the current web, specially in the workload characterization. Unfortunately, at present none of them can represent this dynamism completely.

In this paper we have proposed the Dweb workload model to represent more precisely the web dynamism. Our model is based on three main concepts: navigation, workload test and workload distribution. The two first make use of web users' behavior models to characterize the dynamism of the workload. The last one represents the physical distribution of users in the web. By considering these concepts, the workload generation model has been used to implement a new workload generator called GUERNICA, whose main feature and contribution is its capability to model the users' dynamic behavior. All this has been illustrated in a working example.

Acknowledgments

The authors wish to thank Emilio Gil, Javier Sevilla, Tomás Cusso, Francisco J. García, José Miguel Pardo, Salvador Martín and Carlos Pardo from ISOCO; Ester Palacios and Marga Nacher from Instituto Tecnológico de Informática, and Bernardo de la Ossa and Ingrid J. Niño from Polytechnic University of Valencia, for their valuable help and support in the GUERNICA Project.

This work has been partially supported by the Spanish Government Grant (CICYT TIC2001-1374-C03-02 and FIT-340000-2004-236), IMPIVA Grant (IMIDTD/2004/92, IMIDTD/2005/15) and Spanish Ministry of Education and Science and the European Investment Fund for Regional Development Grant (TSI 2005-07876-C03-01).

References