ArchCollect: A set of Components directed towards web users' interaction

Joubert de Castro Lima, Tiago Garcia de Senna Carneiro, Edgar Toshiro Yano, Rodrigo Martins Pagliares, Júlio César Ferreira and João Bosco Mangueira Sobral.
joubert@facic.fuom.br - FACIC/FUOM
tiago@dpi.inpe.br - Federal University of Ouro Preto
yano@comp.ita.br - IEC/ITA
pagliares@yahoo.com.br - FAS/UNIPAC
juliof@bol.com.br - Federal University of Ouro Preto
bosco@inf.ufsc.br - Federal University of Santa Catarina

Abstract

This paper describes an example of a system focused on web users' interactions, called ArchCollect. One JavaScript component and five Java components gather information coming only from the user, independing on the web application that will be monitored and on the web server used to support it. Collecting information directly from the client improves the portability of this software and its capacity to deal with many web applications in a Data Center at the same time, for instance. The ArchCollect relational model, which is composed by several tables, provides analyses, regarding factors such as purchases, business results, the length of time spent to serve each interaction, user, process, service or product. In this software, data extraction and data analysis are performed either by personalization mechanisms provided by internal algorithms, or by any commercial decision-making tools focused on services, such as, OLAP, Data Mining and Statistics, or by both.

***Keywords*:** Interaction pattern, web users' interactions, components, relational model, decision-making tools.

1. INTRODUCTION

Interaction, in this work, is a general term used for classifying specific events that were emitted by users in any sort of application. These events are classified by clicks on elements on a page of an application. These elements are buttons, links and banners, the last one used particularly for commerce applications.

In the electronic commerce, the users' interactions analysis area has been largely studied [1, 2, 3, 4, 5, 6, 7, 17]. There are many commercial tools available for this area [8, 9, 10, 11, 12, 13, 14, 15, 16]. All these tools extract the initial data from different kinds of servers and, as [2] explains, the majority of the commercial tools typically keep count of hits, rank the most popular pages requested and tell where the user came from, the length of time each page was viewed and the page from which the user entered the site and from which the user exited.

Generally, web usage mining tools or decision-making tools like the ArchCollect, have four main steps, which are: 1. Collecting data from servers or from clients or from both. 2.Transforming these data into a meaningful set of data, which contains the information necessary to start the diagnostic. 3. Loading this transformed information into databases or data structures. 4. Extracting this stored information using tools that have OLAP, Data Mining and Statistic services, or recommending web pages also based on the stored information.

The purpose of this article is to show an example of software directed towards web users' interactions, which has low coupling to the monitored application, and also has a general collecting mechanism. This software enables future experiments, once the basic ideas have been implemented and tested.

The low coupling is obtained once all the information, necessary to the ArchCollect, is collected directly from the web client. The ArchCollect ***user component,*** inserted by a parser software into the HTML or XML code of the existent application, collects all the necessary information and sends it to the ArchCollect ***collecting component.*** This last component sends this information to the ArchCollect ***transforming component***, which filters it and calls the ArchCollect ***loading component*** to store this filtered information into a set of tables called the ArchCollect relational model. In this relational model we have the final interaction pattern, ready to be used.

Data extraction and data analysis are performed either by personalization mechanisms provided by internal algorithms, or by any commercial decision-making tools focused on services, such as, OLAP, Data Mining and Statistics, or by both.

The rest of this article is structured as follows: In section 2 the related works are emphasized and compared to the ArchCollect. Section 3 explains, with details, the ArchCollect components, their low coupling to the existent application and the ArchCollect relational model. In section 4, the experiments are characterized. Section 5 presents the experiments results with one example of the visualization component. Finally, in section 6, the conclusions are made and the future works are proposed.

2. RELATED WORK

Some works, such as [1, 2, 3, 4, 5, 6, 7, 17], were proposed to deal with data extraction under many different perspectives, in other words, to illustrate analyses offered to administrators (*Sites Modification*, *Systems Improvement*, *Business Intelligence*) and offered to the application users (personalization).

Such projects have as mechanism or internal logic, the possibility of storing data into data structures, typically graphs, or into relational models consisting of some tables.

Projects such as ECI, WebLogMiner use relational models and tools with OLAP, Data Mining and Statistics services for extraction. Projects such as Shahabi use graphs where each node is a page and each track (graph arrow) corresponds to one link clicked by the user. The overlapping of the graphs generates the similarities. Projects such as WUM bring, at each node of the graph, a set of the users who have passed through the page. Similar paths are identified by this mechanism.

Table 1 summarizes each work characteristics using the criteria proposed by [3]. The ArchCollect advantages are: its low coupling to the monitored application, its general collecting mechanism, and the correlation of the data stored into the relational model.

The low coupling is obtained by the mechanism used to insert the user component code into the web page sent to the client, by the user component collecting mechanism and by the duplication component. A parser software that automatizes the process makes the insertion, and the duplication component separates the information necessary to the monitored application from the one necessary to the ArchCollect. Any other mechanisms of the related works cause high coupling, once they collect their information from servers and create applications that depend strongly on the analysis mechanism.

The user component was made for collecting properties of a clicked element on a page. As it works collecting properties, these can be any sort of properties, in other words, subjective properties, such as the average age of users, or essential properties, such as the interaction name. The amount of information is also irrelevant for this mechanism, once the number of properties is defined by the monitored application.

For the rest of the quoted works, it is hard to obtain such flexibility, once the collection is made from many different sources and any change can bring huge efforts to find new information.

It is important to remember that both the kind and the number of properties in an element affect directly the quality of the collected information and, consequently, the capacity of analyses offered by the ArchCollect, although this is under the responsibility of the monitored application.

The duplication component collects the length of time that each requisition of a certain client spends to be answered by the monitored application servers and by the ArchCollect servers. The correlation of this data with others collected by the user component brings extra information about business versus performance.

Project	**User**	**Coupling (App. Monitor)**	**Application Focus**	**Data source**	**Usage data**	**Emphasized subjects**
WebSifit	Multi	High (unique application)	General	WEB Server	Usage/ Content/ Structure	-----
ECI/IBM	Multi	High (unique application)	General	WEBServer/ User	Usage/ Structure	Purchases and business results
SpeedTracer	Multi	High (unique application)	General	WEB Server	Usage/ Structure	----
WUM	Multi	High (unique application)	General	WEB Server	Usage/ Structure	-----
Shahabi	Multi	Medium (unique application)	General	User	Usage/ Structure	-----
ArchCollect	Multi	Low (multi application)	General	User	Usage/ Structure	Purchases, business results and performance.

Table 1. Related works and their characteristics.

3. ARCHCOLLECT AND ITS COMPONENTS

The ArchCollect is composed by seven components as shown in Figure 1. The four phases, already described in the introduction, are implemented looking for simplicity, for the software scalability and, finally, for the development of components capable of supporting loads in the order of 100 million of hits a day. The components internal communication is made by the TCP protocol, allowing the support of any application layer protocol.

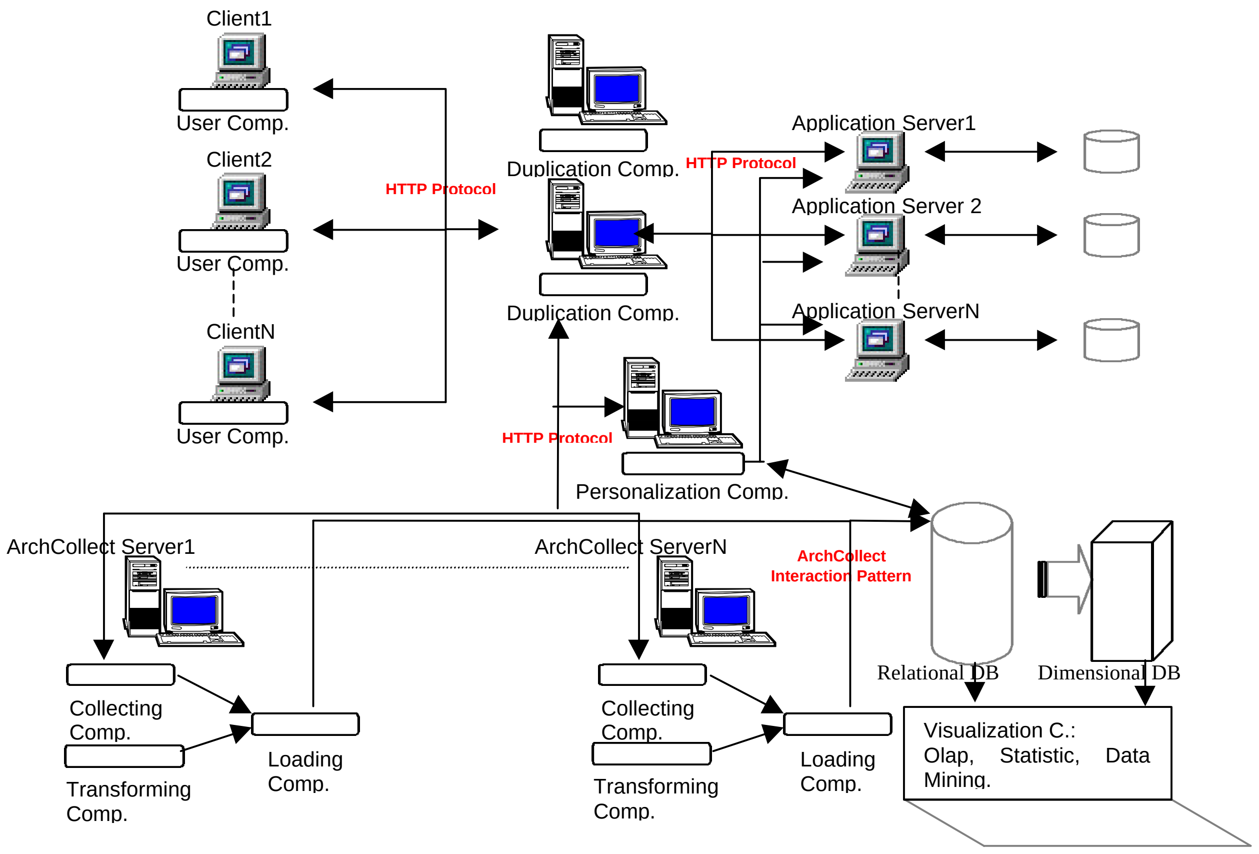

Figure 1 :Operational environment showing one possible deployment of the ArchCollect architecture.

3.1. USER COMPONENT

Every element in a web page has its name and its identifier. The user component defines these two properties or, if necessary, other properties to give semantic meaning to the interactions. This last situation can happen in monitored applications implemented with XML elements.

Besides these page elements properties, called semantic issues, the user component collects the interaction complete date and time so that the local time where the interaction took place can be obtained. For specific elements in commerce applications, the user component also collects the price and the quantity related to the product, service or process that has been acquired. The page where the element has been clicked is also collected. At last, the client cookies and some semantic issues are collected, if they exist.

The information is stored in a text element in each page and sent together with each request in the ArchCollect interaction pattern showed in Figure 2. The user component is a set of JavaScript functions that are inserted by a parser software, implemented as one activity of the duplication component.

```
pattern ::= date“+”page“+”element[“+”product]“+”session”+”semantic
date    ::= day“+”month“+”year“+”hour“+”minute“+”second
element ::= name“+”id
product ::= quantity“+”price
session ::= userIP[[“+”sessioncookie]“+”persistentcookie]“+”entranceID
semantic ::= age association”+”rent association”+”layout association”+”etc.
```

Figure 2. ArchCollect interaction pattern.

3.2. DUPLICATION COMPONENT

The ***duplication component*** must be operating on the same host(s) and port(s) where the original web server should be running. Thus, all HTTP requests sent to the original application are firstly received by this component and replicated to the web server, which should have been moved to another host(s) or port(s). In other words, the duplication component implements the *Proxy* pattern[18].

The first task of this component is listening to the port(s), already predefined, looking for users' interactions. When a request arrives at the duplication component, this sends an identical request to the original web server and a request with its first line modified to the ArchCollect server. This line is modified so that the collecting component is instantiated in the ArchCollect server. The duplication component can deal with applications that use both, GET and POST HTTP methods.

Unfortunately, this component will also receive all the requests from all the users and not only the interactions that are relevant to the ArchCollect. Because of this, the component first filters and duplicates only requests related to the application services, for example, ASP, JSP, CGI, PHP pages.

The ***duplication component*** also registers the elapsed time between the moment when a request arrives and: 1) the moment when the application response arrives, 2) the moment when the collecting component response arrives, and 3) the moment when the HTTP response has just been sent to the user browser.

Once the duplication component receives the monitored application server response, it can insert a block of code at each page (parser software) and it can also create a XML document from a HTML document (wrapper softwares). After these tasks, the duplication component sends the response to the client that initiates the process.

3.3. COLLECTING COMPONENT

The tasks proposed for the ***collecting component*** are:

1) Identifying the user, i.e., verifying if the ArchCollect cookies do exist on the HTTP request.
2) If the ArchCollect persistent cookie does not exist:
   a. the collecting component assumes that this request came from a new user, whose information is sent to the transforming component and added to the relational database. The HTTP request header information is sent to the *loading component*;
   b. the session and persistent cookies are created, and the session.`entranceID` field of the interaction pattern is set to "1".
3) If the ArchCollect session cookie does not exist, but the persistent cookie does exist:
   a. the collecting component assumes that this request came from a new session, from a known user, and then the session cookie is created, and the session.entranceID field of the interaction pattern is set to "1".
4) If the ArchCollect session and persistent cookies do exist:
   a. the collecting component assumes that this request came from a known session and known user, and the session.entranceID field of the interaction pattern is set to "0".
5) Sending the interaction pattern to the transforming component.
6) Sending the current cookies to the duplication component.

3.4. TRANSFORMING COMPONENT

Because user information needs to be stored in a relational database, a specialized component is required for some specific activities such as extracting, transforming and loading data from the collecting and duplication component. This component has to identify the data about unique users' sessions so that they can be stored into the ArchCollect relational model.

The first task is to process the data coming from the collecting component, storing each line of the buffer into the relational database using the *loading component*.

The second task is to identify the session. Using a simple rule, it is possible to establish the information of the user entrance in the application: if the field *session.id* from a line in the interaction pattern is set to '1', then it is that line that corresponds to the entrance page of the user session.

To find the last interaction of each user session, i.e., the point where the user exits the monitored application, it is necessary to process the whole database, looking for the register that has the most recent date in the session.

3.5. LOADING COMPONENT

The ***loading component*** has the purpose of receiving the information from the ***transforming component***, and storing it, according to the ArchCollect architecture final interaction pattern, into the relational database.

Considered the persistence layer, the ***loading component*** allows the integration between the ArchCollect logical model and the relational model. The architecture business layer is separated from the transaction layer resulting in low coupling among the architecture modules, achieving better performance, once such component is instantiated on demand and with better transactional integrity.

3.6. VISUALIZATION COMPONENT

Once the interactions are correctly stored into the relational models, it is necessary to define some extraction tools to answer business questions.

The relational model was translated to a dimensional model. The solution is based on the separation of the data that is used for decision-making, called dimensions, from the operational data, called fact table. This kind of data is stored at data webhouses.

The data webhouse is responsible for data storage and management, while the OLAP services, for example, convert the stored data into useful information for decision-making [19].

The ***visualization component*** can be implemented in any data webhouse tool and OLAP or Data Mining or Statistic services off-the-shelf.

3.7. PERSONALIZATION COMPONENT

Web commerce applications deal with anonymous users who arrive, emit interactions and leave the application. Important information with grain of granularity in the order of the users' interactions is obtained by the components that were described until now. Also, in order to re-pass these modifications online to the commerce application user, it was proposed the ***personalization component***.

The ***personalization component*** task is to generate, in the beginning of its execution, in background and periodically, the initial profiles. The interaction pattern stored into the relational model is used to establish the initial profiles. Once this task is done, the online process begins.

The second task is the profile sophistication, assuming new profiles and increasing the existing ones. The interaction pattern illustrated in figure 2 has a subpart called semantic. Based on this subpart the online profile is sophisticated.

It is not the intent of this article to detail this component, once the ideas are not implemented and tested yet. Problems with performance in the second task persist. We based our ideas in the ***Yoda*** [20] project. Once this component is ready it will require an individual work.

3.8. RELATIONAL MODEL

The relational model is considered the ArchCollect architecture core, which reflects the ArchCollect architecture final interaction pattern.

The purpose of this article is to describe an example of interaction analysis architecture, and no metrics were used to establish the relational model. We just emphasize general subjects that all corporations will be interested in. In the future works a detailed study will be done classifying the set of metrics that the ArchCollect will adopt.

An example of an ArchCollect relational model is illustrated in the appendix. This example is used to store all the tests information.

4. EXPERIMENTS CHARACTERIZATION

The system was analyzed by an experiment composed by three computers that are interconnected by a hub Ethernet 10/100Mbps. One of the computers executes the benchmark Microsoft Web Stress Tool that implements many threads, that continuously, sends POST requisitions on ASP pages – Active Server Pages of a web application stored in another computer that executes the IIS server – Microsoft Internet Information Server 5.0.

The third computer executes all the ArchCollect architecture components and the JWS server – Java Web Server 2.0 used by the architecture ***collecting component*** for servlet instantiation. Table 2 shows the configuration of each one of the computers.

COMPUTERS	**Web Stress Tool**	**IIS**	**ArchCollect**
Processor	Pentiun II, 350 MHz	Pentiun II, 350 MHz	Athlon , K7 650 MHz
Memory	128 MB SDRAM	128 MB SDRAM	256 MB SDRAM
Hard Disk	IDE,7200 rpm	IDE,7200 rpm	IDE, 7200 rpm
Network	Ethernet 10/100 Mbps	Ethernet 10 Mbps	Ethernet 10/100 Mbps
Operational System	Windows NT 4.0	Windows 2000 Professional	Windows NT 4.0

Table 2. Computers used in the experiments.

The experiments where made in two scenarios. In the first scenario, only two computers are connected to the hub and the Microsoft Web Stress Tool generates HTTP requests directly to the IIS server with no interference of the ArchCollect architecture. In the second scenario, the three computers are used and the Microsoft Web Stress Tool generates HTTP requests to the ArchCollect architecture server that collects and analyzes the requests and, in parallel, re-passes them to the computer that executes the IIS server.

4.1. SERVICE LEVEL DEFINITION AND PERFORMANCE METRICS CHOICE

The ArchCollect architecture must be able to collect the information related to an interaction in a finite and shortest time interval. Every time this interval is exceeded, the information is rejected and a timeout is raised. The greater it is this timeout value, the greater it will be the number of simultaneous users, the greater it will be the number of active threads and the greater it will be the use of resources as memory and CPU. The smaller it is this timeout value, the greater it will be the number of timeouts and the smaller it will be the quantity of collected information. Therefore, there is a clear relation between the desired service quality and the computational performance (internal resources) necessary to reach this quality. This parameter can be optimized as the system workload increases or diminishes, but in our experiments it was used the value of 60 seconds. The performance metrics [21, 22] chosen to analyze the collected interactions service are: the total response time observed by the user and the system throughput.

4.2. PARAMETERS THAT INFLUENCE THE EXPERIMENTS

Many parameters influence the experiments, some of these parameters are characteristics of the system itself, such as the quantity of the servers RAM memory or the processor speed and the transmission speed of the connection between the clients and the server. Other parameters depend exclusively on the workload, such as the number of simultaneous users, the number of new users that arrive at the system per time unit and the think time. The experiments were made with the purpose of knowing only the impact of changes in the parameters: number of simultaneous users and number of new users per time interval. The values used for these parameters in the experiments were: number of simultaneous users using the system, which was equal to 50, 100, 125, 150, 175 and 200; and number of new users arriving at the system in a time interval of 10 minutes, which was equal to 150, 200 and 300. The think time was established as 10 seconds, an average time for an ASP page used in the experiment to be received by a user in the Internet.

All the experiments were made with a Microsoft Web Stress Tool, configured to simulate 56 Kbps connections between the clients and the web server.

4.3. CHARACTERIZATION OF THE WORKLOAD

The workload was characterized by a set of HTTP requests using the POST method on 10 ASP pages stored into the IIS server. Each one of the pages was associated to a different product and each request sent to the IIS server has informed the amount of this product that was bought. It was supposed a site where 20% of the users accesses at least, up to the third page of the site, 30% up to the seventh page of the site and 50% of the users accesses up to the tenth page. All the pages had the size of 12885 bytes.

For each one of the proposed scenarios, experiments were made where the number of simultaneous users had varied and the rate of new users remained constant, and experiments where the opposite had occurred, resulting in a total of 2x(6x1+1x3)=18 experiments. Each experiment lasted, on average, 10 minutes and was repeated 10 times, so that the results average could be collected. The total time used in the execution of the experiments was, at least, equal to 30 hours.

5. PRESENTATION AND ANALYSIS OF THE RESULTS WITH AN EXAMPLE OF ArchCollect VISUALIZATION COMPONENT

Figure 4 presents some samples of the response time observed by a web user, when the number of simultaneous users varies from 50 to 200 users and the rate of new users arriving at the system remains in 20 users/minute. The ArchCollect architecture had its best performance when the number of simultaneous users was equal to 50 users and the response time was equal to 3,9 seconds. Up to 100 users, the response time increases linearly as the number of users increases, but from this limit, the response time seems to have an exponential growth as the number of users increases.

The IIS server response time remains relatively the same for all the experiments. This figure also shows that, in the second scenario, the response time observed by the user increases in an order of magnitude when compared to the response time observed in the first scenario. The ArchCollect architecture has increased 3.2 seconds on the average response time observed by the user, and at most 15.6. seconds

Figure 5 shows how the service and waiting time of the monitored application and of the ArchCollect architecture behave when the number of simultaneous users increases and the rate of new users arriving at the system remains constant and equal to 20 users/minute. When the number of simultaneous users is over 150, it is observed that the waiting time of the ArchCollect architecture becomes bigger than its service time and starts to grow quicker, confirming the saturation state of the architecture.

Figure 6 shows that the throughput observed by the web user grows linearly as the number of simultaneous users increases up to a limit of 100 users. Over this value, the system comes to a saturation state where the ArchCollect throughput tends to a limit of 7 requisitions answered per second, approximately 604.800 interactions a day, and the IIS server throughput continues growing linearly. This discrepancy can be justified by the fact that the ".asp" pages that compose the workload do not have any ASP code.

6. CONCLUSIONS AND FURTHER WORKS

This article presented an architecture for collecting and analyzing users' interactions. The low coupling to the application that is going to be monitored enables the ArchCollect to deal with many web applications at the same time.

Components with specific functions allowed the development of a tool that keeps sufficient information to answer questions about sales, business results and performance results, for example. We implemented and tested the initial ideas of this architecture.

All the results of the ArchCollect version 1.0 show that the software can operate monitoring small sites, once the throughput is small and the response time increases significantly when 150 or more simultaneous users access the monitored application. Version 2.0 can monitor medium or big sites. Preliminary tests show that the ArchCollect can deal with 350 simultaneous users increasing the throughput with a reasonable response time.

The work is being extended in many directions. Some experiments, using different numbers of ArchCollect servers, must be made to observe the architecture scalability.

The data mining and OLAP services must be coupled to the statistics visualization, ending the possibility of data extraction from a unique relational model.

The personalization components are just one path for the understanding of the collected interactions. Web recommendation algorithms or communities creation algorithms will be able to bring a huge contribution for the understanding of the interactions, and then to provide a more sophisticated user behavior profile.

Nowadays, only the waiting time and the service time for each interaction on the ArchCollect servers and on the web application servers are analyzed. These times show how much it costs to carry out a certain service or process. A better architecture internal performance analysis allows knowing which component, specifically, behaviors as the bottleneck of the whole architecture.

The ArchCollect implementation may be modified to support a bigger number of simultaneous users, to decrease its response time and to increase its throughput so that these values may be compared to the values presented by the IIS web server. Two alternatives to obtain this situation are to implement more efficient strategies in our component algorithms or to attempt to distribute these components among many hosts connected by a communication network using the J2EE architecture model.

Another point that can be studied is the adequacy of the current ArchCollect architecture to the interactive TV models or telephony models. Information about the target public, their interactions and these interactions relation to a service or product, has to be established.

7. References

[1] M.S.Chen, J.S.Park, P.S.Yu, " Data Mining for Transversal Patterns in a Web Environment", Proc. of 16th International Conference on Distributed Computing Systems, 1996.

[2] S.Gomory, R.Hoch, J.Lee, M.Poldlaseck, E.Schonberg, "Ecommerce Intelligence : Measuring, Analyzing, and

Reporting on Merchandising Effectiveness of Online Stores”, IBM Watson Research Center, 1999.
[3] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, Pang-Ning Tan, “Web usage minig : Discovery and applications of usage patterns from web data”, SIGKDD, January, 2000.
[4] Myra Spiliopoulou and Lukas C Faukstich. WUM: A web utlilization miner, EDBT Workshop WebDB98, Valencia, Spain, 1998.
[5] Kun-Lung Wu, Philip S Yu, and Allen Ballman. SpeedTracer: A web usage mining and analysis tool, IBM Systems Journal, 37(1), 1998.
[6] Cyros Shahabi, Amir M Zarkesh, Jafar Adibi, and Vishal Shah. Knowledge discovery from users’ web-page navigation, Workshop on Research Issues in Data Engeneering, Birmingham, England, 1997.
[7] O.R.Zaiane, M.Xin, J.Han. Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs, Proc. of Advances in Digital Libraries Conference, 1998.
[8] Andromedia Inc´s Aria, http://www.andromedia.com
[9] DoubleClick Inc, http://www.doubleclick.com
[10] Engage Technologies Inc´s, http://engagetechnologies.com
[11] IBM Corp´s SurfAid, http://surfaid.dfw.ibm.com
[12] Marketwave Corp´s Hit List, http://www.marketwave.com
[13] Media Metrix, http://www.mediametrix.com
[14] net.Genesis´net.Analysis, http://www.netgenesis.com
[15] NetRating Inc., http://www.netratings.com
[16] Straight UP!, http://www.straightup.com
[17] Kun-Lung Wu, Philip S Yu, and Allen Ballman. SpeedTracer: A web usage mining and analysis tool, IBM Systems Journal, 37(1), 1998.
[18] Gamma, E. et.al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley. 1995.
[19] Sakhr Youness, “Professional Data Warehousing with SQL Server7.0 and OLAP Services”, 2000.
[20] Cyrus Shahabi, Farnoush Banaei-Kashani, Yi-Shin Chen, and Dennis McLeod, “Yoda: An Accurate and Sacalable Web-based Recommendation System”, In the preceedings of the Sixth Internacional Conference on Cooperative Information Systems, Trento, Italy, September 2001.
[21] Menascé, Daniel A. & Almeida, Virgilio A.F., "Capacity Planning for WEB Performance - Metrics, Models & Methods", Prentice Hall, PTR, 1998.

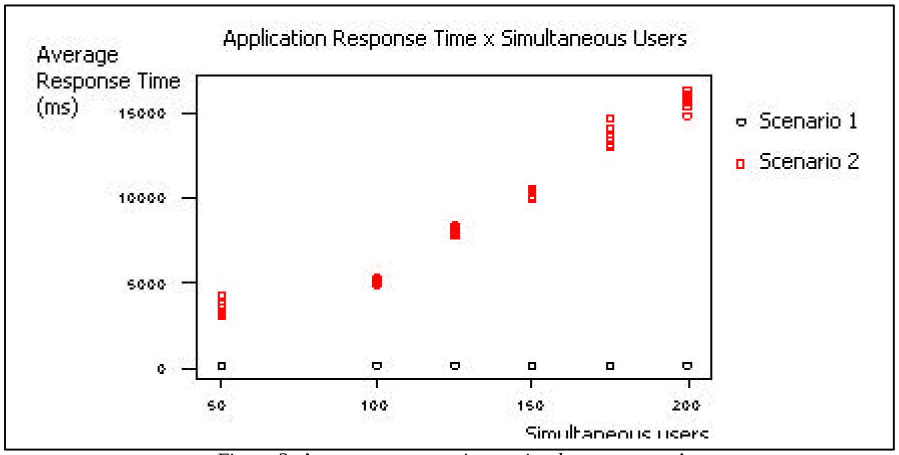

Figure 3. Average response time x simultaneous users’.

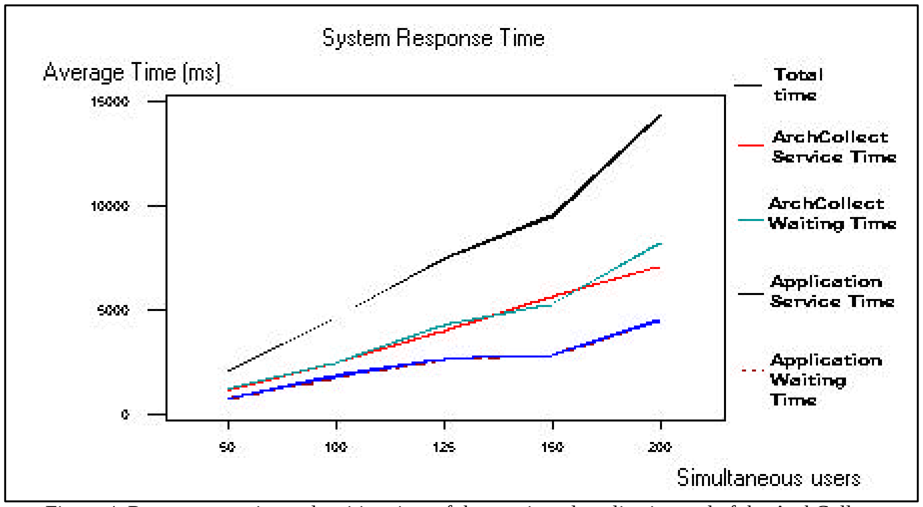

Figure 4. Response, service and waiting time of the monitored application and of the ArchCollect architecture.

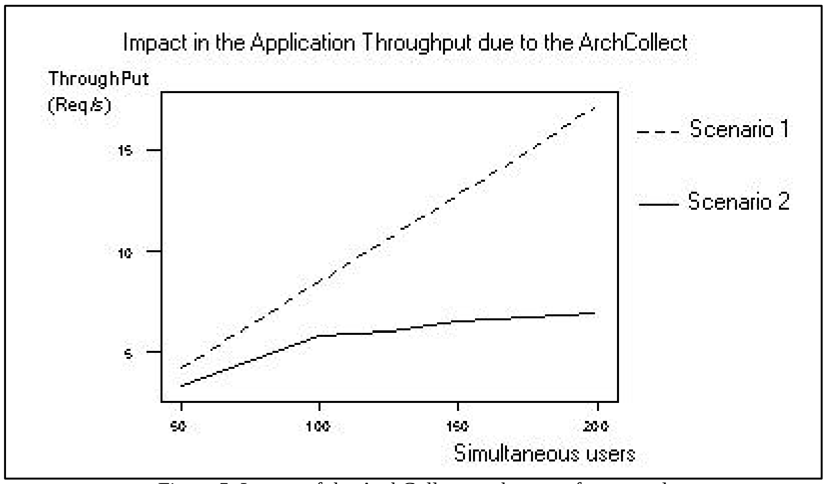

Figure 5. Impact of the ArchCollect on the rate of answered requisitions observed by the server.

[22] Jain, R.. "The Art of Computer Systems Performance Analysis - Thechniques for Experimental Design, Measurement, Simulation, and Modeling", John Wiley & Sons,Inc., 1991

Appendix

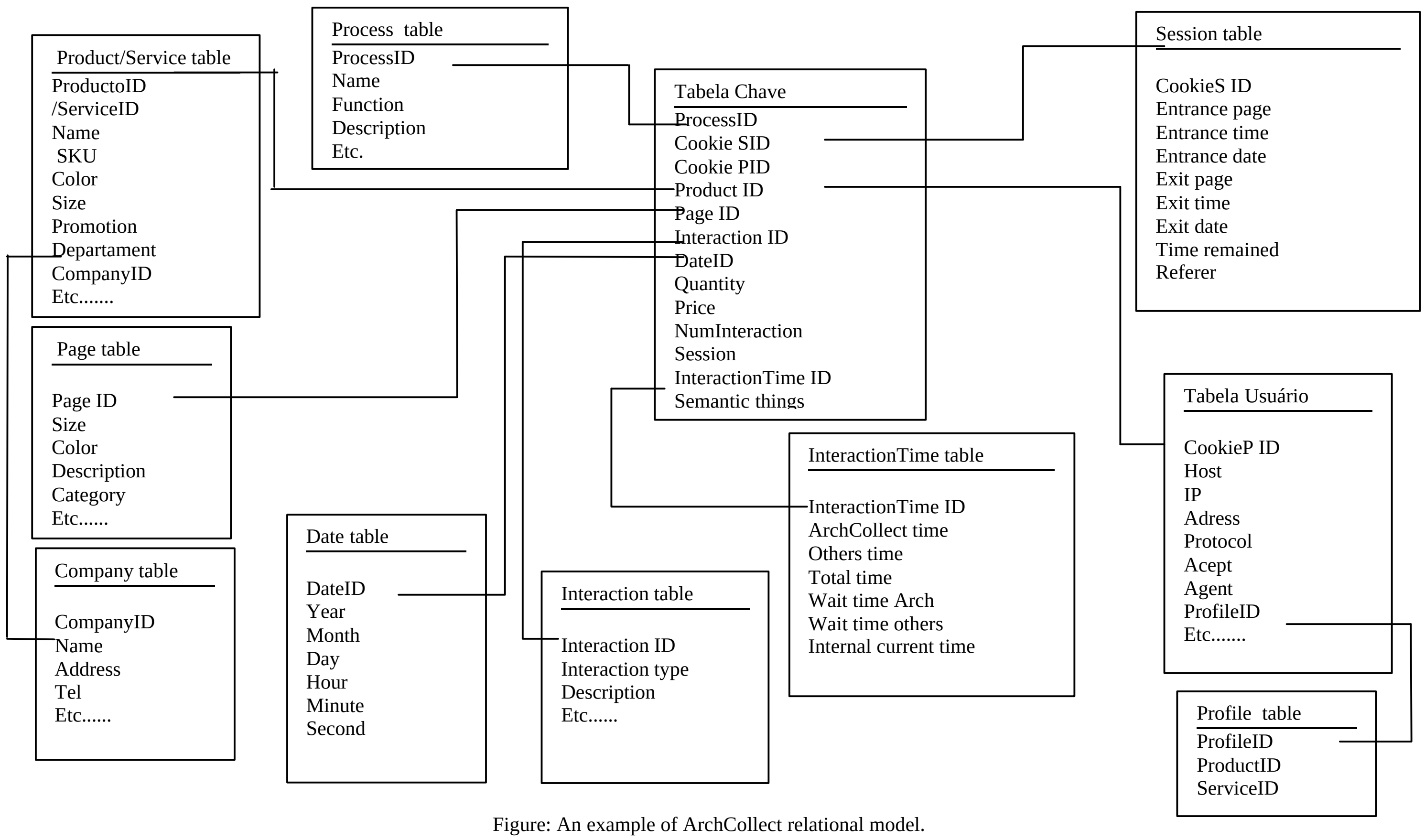

Figure: An example of ArchCollect relational model.