How to use Web Analytics for Improving Web Applications

Overview
As soon as a Web application has been deployed into the production environment, typically by the end of its development, some or all of its features are available to the users. By observing and analyzing the application usage over a certain period of time, it is possible to extract users’ behavior patterns. Using these patterns to modify the application increases the chances to best meet users’ expectations and thus improve their satisfaction.

In this IBM® Redpaper, we first introduce Web analytics: the context, the main concepts, and the common characteristics of the available tools. Then we explain how to exploit these tools to improve various points in the development and the quality assurance of a Web application, namely client compatibility, user navigation, robustness, bandwidth sizing, development, load-test plans, maintenance, and fat-client extensions. Finally, we describe a few problems and give some insights for solutions.

This paper focuses on J2EE™ Web applications, because J2EE is the programming model implemented in IBM WebSphere® Application Server. Yet, the methods presented can be applied to any Web application technology, such as ASP, PHP, or CGI.

References are marked [xxxx] and are provided in “References” on page 42.
Introduction

Hundreds of Web analytics tools are available on the market. This paper focuses on their common concepts and characteristics. Some screen captures come from WebTrends Log Analyzer, a complete and popular Web analytics tool. Nevertheless, most other Web analytics tools generate similar results, and the explanations given here can be applied to any other tool that supports the same features.

Web analytics definition

Web analytics—also referred to as Web metrics, e-Metrics, or e-analytics—is the science of Internet audience measurement and analysis. It deals with the consultation of Web sites, the gathering of Web usage data, the computation and presentation of metrics, and the exploitation of the results, in order to improve the satisfaction of Web site objectives.

History

Web analytics started about 10 years ago with the emergence of the Internet, when its audience started to grow significantly and other media were showing increasing interest and concern for the World Wide Web.

Web analytics tools have been designed to gather and parse Web servers log files, compute metrics from the extracted data, and generate text reports. With technologies’ evolutions, new data-gathering methods have been developed and reports have been enhanced with tables and graphics (Figure 1).
Evolution of the World Wide Web during its existence can be categorized into three phases (Figure 2).

1. **Document Web**
   - Static Web sites: HTML documents
   - Web servers
   - HTTP protocol
   - Browser

2. **Application Web**
   - Dynamic Web sites: generated HTML
   - Application servers
   - Business logic (Java, CGI)
   - Transactions, distributed processing

3. **Service Web**
   - Web services
   - Generated XML
   - SOAP, WSDL, UDDI
   - Transactions initiated by program

_Figure 1_ A Web analytics diagram

_Figure 2_ Evolution of the World Wide Web
Web analytics can generate results for any kind of Web site:

1. **Static Web sites**: Web analytics tools have been designed from the ground-up to analyze the consultation of static Web sites. Web analytics tools are traditionally used for such Web sites, and literature abounds in explaining how to interpret the reports in this context.

2. **Dynamic Web sites**: Web analytics tools are used less for dynamic Web sites because in this case most of the reports are useless, incomplete, or must be interpreted differently. This Redpaper aims to clarify which reports can be useful and how they can be exploited to apply a beneficial retrospective effect to activities of the Web application development process.

3. **Web services**: What today’s Web analytics tools can bring into this context is a question that remains to be answered. Some insights are discussed in “Web services” on page 41, along with some Web-services requirements for Web analytics products.

### Web usage data

In this section, we introduce terminology and definitions for Web usage data, which is the metadata handled by Web servers.

Figure 3 shows an HTTP transaction between an HTTP client and an HTTP server.

**Figure 3**   HTTP transaction

**Note:** An HTTP client is more commonly known as a Web client, and an HTTP server is more commonly known as a Web server.

A Web client designed for human users is called a Web browser. Examples of Web browsers are Netscape Navigator and Microsoft® Internet Explorer.

Examples of Web servers are IBM HTTP Server, Apache HTTP Server, and Microsoft Internet Information Server (IIS).
HTTP transaction
An HTTP transaction has the following characteristics:

- IP address of the HTTP client machine; this can be resolved to the fully qualified DNS name
- User ID, if the user has performed HTTP authentication
- Time at which the server finished processing the request
- HTTP method (GET, POST, ...)
- Requested URI
- Protocol and protocol version, practically, HTTP 1.0 or HTTP 1.1
- HTTP status code sent back to the client
- Response size, in bytes
- Referrer, which is the URL the client reports having been referred from
- User agent, information that the client browser reports about itself. This information usually includes:
  - Browser name
  - Browser version
  - Operating system the browser runs on

More details about Web usage data can be found in [IET04, APA04a, APA04b].

Terminology
In the sections that follow, we also use several terms with a particular meaning:

- **Page**—Text or HTML or XHTML file in a Web site
- **Static page**—Information page existing before it is requested to the Web server
- **Dynamic page**—Information page generated in real time according to the HTTP request, by the application server
- **Hit**—HTTP request to a file
- **Consultation** or page view—HTTP request to a page
- **URL/URI**: see Figure 4

![URL](http://www.piggybank.com/TransferForm.do?debitAccount=222)

Figure 4 URL/URI
General Web analytics terminology can be found in [Cof01] and in most Web analytics tools.

**Data-gathering methods**

The capability, comprehensiveness, and timeliness of a Web analytics solution depend on the methods used for capturing Web usage data [Moe03]. In this section we introduce several methods, all of which are viable options to use—either alone or in combination—along with an indication of their main benefits and drawbacks.

A prioritized list of Web analytics requirements should be defined and agreed on, to help decide which method or combination of methods best answers the requirements with sufficient details and accuracy in the appropriate time frame.

Note that to produce Web analytics reports for a given period of time, Web usage data must have been gathered during the entire period.

Web usage data can be gathered at several locations in the Web environment (Figure 5):

- The client browser that displays the Web pages.
- The Internet Service Provider (ISP) network carrying the data flow between the Web server and the client browsers.
- The Web server.

![Figure 5   Web usage data gathering locations](image)

Different methods may be used to gather Web usage data in each of these locations [Mal01a]:

- Single-pixel—Web pages are instrumented by adding about 500 bytes of HTML and JavaScript™ code that sends information to a Web-analytics server.
Network monitoring—A network-monitoring application *sniffs* the packets carried by the network wire. This method cannot monitor encrypted transmissions and is too CPU-intensive for high-traffic Web sites.

Server monitoring—A plugin is added into the Web server, getting and filtering information about events through an application programming interface. The plugin introduces risk into the Web server.

Web server logging—The HTTP server is configured to produce *log files*. For each HTTP request, a *log line* is added into each of the log files.

All methods gather approximately the same data and can achieve similar results.

**Log formats**

In Web server logging, data is logged by the Web server to files in one or more file formats. As Web server software has evolved, so has the variety of logging options and logging implementations. Some of the most widely spread log file formats are [Mal01b]:

- NCSA Combined log format
- NCSA Separate log format (3-log format)
- NCSA Common log format (access log)

Combined log format (Example 1) combines all the Web usage data described in “Web usage data” on page 4.

**Example 1  Combined log format log line**

```
62.201.74.134 - - [29/Aug/2003:22:37:54 +0200] "GET /DisplayAccounts.do HTTP/1.1" 200 2972 "http://www.piggybank.com/MainMenu.do" "Mozilla/4.0 (compatible; MSIE 5.01; Windows NT)"
```

Separate log format combines all the Web usage data but splits the data into three log files in the following formats:

- Agent log format—agent data (Example 2)
- Referrer log format—referrer data (Example 3)
- Access log format (common log format)—the rest of the data (Example 4 on page 8)

**Example 2  Agent log format**

```
Mozilla/4.0 (compatible; MSIE 5.01; Windows NT)
```

**Example 3  Referrer log format**

```
http://www.piggybank.com/MainMenu.do -> /DisplayAccounts.do
```
Example 4  Common log format (access log format)


Information in the Separate log format is the same as in the Combined log format, but data in the different log files are uncorrelated and may not exactly match.

IBM HTTP Server logging is configured by default with Common log format. This format contains only a subset of the Web usage data. Therefore some Web analytics reports cannot be generated from Common log format log files.

To produce all the reports, a complete log format such as the Combined log format is needed in the Web server logging configuration. In Apache and IBM HTTP Server, this is done in the conf/httpd.conf configuration file (Example 5).

Example 5  Combined log format configuration

LogFormat "%h %l %u %t "%r" %>s %b "%{Referer}i" "%{User-Agent}i"" combined
CustomLog logs/combined.log combined

In IBM HTTP Server, this can be configured using the IBM HTTP Administration Server console (Figure 6).

Figure 6  IBM HTTP Server Administration: logging configuration panel
The Web server logging configuration should be done as soon as possible, ideally before the Web server is running in production, so that all user requests are logged properly.

The PiggyBank case

For the sake of illustration by example, we introduce PiggyBank, a simple case of a Web application.

Model

Be a small city somewhere in the world...

In the city, the currency is the dollar. The smallest monetary unit is the full dollar. There are no cents\(^1\).

The city has one local bank: PiggyBank. Most of the city inhabitants have one or more accounts with PiggyBank. That defines them as PiggyBank customers (Figure 7).

![Person Account diagram](image)

Figure 7 PiggyBank model class diagram

PiggyBank has a computer system that manages customers’ accounts. This system is available online for customers, through a Web interface called PiggyBank Home-Banking System.

\(^1\) This hypothesis is made for the sake of simplicity.
Features
The PiggyBank Home-Banking System allows customers to view the balance of their accounts and transfer money from one of their accounts to any other PiggyBank account (Figure 8).

![PiggyBank use case diagram](image)

User navigation
When a customer logs into the system, a welcome screen is displayed (Figure 9).

![Welcome screen](image)

On top of the welcome screen, a menu navigation bar asks the user to select one of the available features.
The *Display Accounts* feature is made of one screen, which displays a list of the customer's accounts and their respective balances (Figure 10).

![Figure 10  Display accounts screen](image)
The *Transfer Money* feature is more complicated and is made of three screens. In the first screen the customer selects in a list the one of his accounts he wants to transfer money from. In the second screen he enters the credit account and the amount to be transferred to it (Figure 11).

![Figure 11 Transfer money screens](image)

When the customer clicks *Transfer*, the system performs the money transfer between accounts and displays a result window to summarize the transaction (not represented in Figure 11).
Figure 12 summarizes the navigation in the system.

![Diagram of PiggyBank user navigation]

For the sake of clarity, Figure 12 does not show that all screens include the menu navigation bar.

PiggyBank can be implemented using the Struts framework [Wah03]. In this PiggyBank-Struts implementation the URIs used to communicate between the browsers and the application are shown in Figure 13.

![Diagram of PiggyBank-Struts]

In PiggyBank-Struts, each window is associated with a different URI. This choice impacts the meaning and usefulness of some Web analytics reports, as explained in detail in “Examining the reports” on page 19.
The Web analytics process

This section describes the Web analytics process applied to Web applications.

Roles:
- Server administrator
- Application user
- Web analytics tool administrator
- Developer

Activities:
- Deploy site
- Configure and administrate Web analytics tool
- View Web analytics reports
- Define Web application updates
- Update Web application

Artifacts/deliverables:
- Reports
- Web site

Products/tools:
- Web content management tool
- Web analytics tool
- Web server/application server
As illustrated in Figure 14, the Web analytics process is composed of the following phases:

1. Application users use the Web application; information about this usage is gathered (see “Data-gathering methods” on page 6).

2. Web usage data is analyzed by the Web analytics tool (see “Analysis process” on page 16) to produce reports; the tool is configured and administered by the Web analytics tool administrator.

3. Developers analyze these reports to define Web application updates (see “Examining the reports” on page 19). Updated Web application is tested and deployed.

Then the process cycles to phase 1.

Figure 14  Web analytics process lifecycle
**Analysis process**

Inside the tool, the analysis and report generation process has four steps (Figure 15):

1. Gather Web usage data.
2. Parse data, and eventually store the data to a database and retrieve previously parsed data from the database.
3. Analyze data, and eventually store the results.
4. Generate and distribute the reports (see “Troubleshooting” on page 18).

![Figure 15  Analysis process](image)
**Distributing the reports**

Typically, the Web analytics tool provides a Web interface with a list of available reports, along with a calendar to select the time frame (Figure 16).

![WebTrends Log Analyzer report-viewing interface](image)

**Figure 16**  WebTrends Log Analyzer report-viewing interface

Reports usually contain a graphic, a table of numbers, and text explaining the meaning of these.
To distribute the reports to the report viewers, several methods can be used (Figure 17):

- Print reports on paper and give them to the appropriate persons.
- Generate the reports as PDF, Word, or Excel® documents and send them as e-mail attachments.
- Send an e-mail with a hyperlink to or the URL of the report-viewing interface.
- Insert a hyperlink to the report-viewing interface on the company intranet.

**Figure 17  Distributing reports**

**Troubleshooting**

Every report is generated for a given time frame. If no report is available for a certain time frame, possible reasons are:

- The report generation for the corresponding time frame type (daily, weekly, monthly, quarterly, yearly) is not enabled in the configuration. Enable it and restart the analysis process.
- The analysis process is ongoing and the data for the time frame has not been analyzed yet. The solution is to wait for analysis process to be completed. Check the analysis progress status.
- The log files that have been analyzed contain no data for the time range. If this sounds wrong, check the log files content:
  - If no log line is present for the time frame, check the Web server logging configuration (see “Log formats” on page 7).
  - If log lines are present for the time frame, check the tool configuration. Also look for an error in the result status of the analysis process.
Examining the reports

The next sections show a number of interesting reports and explain how to analyze and exploit them to improve various aspects of a Web application, namely client compatibility, user navigation, robustness, bandwidth sizing, development, load-test plans, maintenance, and fat-client extensions.

When activities of the Web application development process can benefit from feedback and how to apply the feedback will be explained in detail later (Figure 18).

Reports hereby presented have been generated from computer-generated log files, designed to be realistic and illustrative.

Browser technologies

In intranets, the set of installed browsers is usually known, controlled, and homogeneous. In this case it is easy to develop a Web application that uses client technologies that are compatible with all users’ browsers.

For Internet applications, client browsers are usually diverse, unknown, and uncontrolled. In these cases it is more difficult to write a Web application that runs on all users’ browsers. To optimize the number of users who can use the application, it is important to know their browsers.

Web analytics reports can show what proportions of each browser family have connected the Web application (Figure 19), and within each family the proportions of each version (Figure 20).
Figure 19  Top browsers

<table>
<thead>
<tr>
<th>Browser</th>
<th>Visits</th>
<th>%</th>
<th>Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Microsoft Internet Explorer</td>
<td>386</td>
<td>50.13</td>
<td>2,418</td>
</tr>
<tr>
<td>2. Netscape</td>
<td>149</td>
<td>19.35</td>
<td>901</td>
</tr>
<tr>
<td>3. Mozilla</td>
<td>111</td>
<td>14.42</td>
<td>879</td>
</tr>
<tr>
<td>4. Opera</td>
<td>59</td>
<td>7.66</td>
<td>389</td>
</tr>
<tr>
<td>5. Other Netscape Compatible</td>
<td>16</td>
<td>2.08</td>
<td>103</td>
</tr>
<tr>
<td>6. UP Browser 4.0</td>
<td>11</td>
<td>1.43</td>
<td>74</td>
</tr>
<tr>
<td>7. Konqueror</td>
<td>10</td>
<td>1.30</td>
<td>77</td>
</tr>
<tr>
<td>8. Lynx</td>
<td>8</td>
<td>1.04</td>
<td>46</td>
</tr>
<tr>
<td>9. Mosaic 2.0</td>
<td>7</td>
<td>0.91</td>
<td>39</td>
</tr>
<tr>
<td>10. Others</td>
<td>7</td>
<td>0.91</td>
<td>39</td>
</tr>
<tr>
<td>11. ia_archiver</td>
<td>6</td>
<td>0.78</td>
<td>44</td>
</tr>
<tr>
<td>Total</td>
<td>770</td>
<td>100.00</td>
<td>4,609</td>
</tr>
</tbody>
</table>

Figure 20  Top browsers by version

<table>
<thead>
<tr>
<th>Browser</th>
<th>Version</th>
<th>Visits</th>
<th>%</th>
<th>Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Microsoft Internet Explorer</td>
<td>6.0</td>
<td>152</td>
<td>19.74</td>
<td>919</td>
</tr>
<tr>
<td></td>
<td>5.5</td>
<td>84</td>
<td>10.91</td>
<td>634</td>
</tr>
<tr>
<td></td>
<td>5.01</td>
<td>61</td>
<td>7.92</td>
<td>398</td>
</tr>
<tr>
<td></td>
<td>5.0</td>
<td>47</td>
<td>6.10</td>
<td>309</td>
</tr>
<tr>
<td></td>
<td>4.01</td>
<td>10</td>
<td>2.34</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td>1.</td>
<td>11</td>
<td>1.43</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>6.0b</td>
<td>7</td>
<td>0.91</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>5.14</td>
<td>6</td>
<td>0.78</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Other</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
</tr>
</tbody>
</table>
This information helps Web designers to have an idea of the target browsers they design the Web site for [Joh01]. A browser support chart (Table 1) indicates what technologies are supported by each browser: JavaScript, DHTML, CSS, layers, DIV tag, and so forth.

**Table 1 Example of a browser support chart**

<table>
<thead>
<tr>
<th>Browser/technology</th>
<th>JavaScript</th>
<th>CSS</th>
<th>DHTML</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer 6</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Internet Explorer 4-5</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Internet Explorer 3</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Netscape 6+</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Netscape 4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Netscape 3</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Opera 3-4</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Lynx</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

This gives a more precise idea of what versions of the technologies are supported by the visitors’ browsers. This detailed report is especially useful when a decision has to be taken about a technology that is:

- Supported by only certain browser versions: XML
- Differently supported by different versions of the same browser: Java™ applets

If the Web site pages use a technology that is not 100% supported, we can measure the percentage of visitors who do not see the pages correctly, and decide whether it is worthwhile to redesign the unsupported pages, or develop an alternative version of these pages with a supported technology.

[Joh01] establishes a list of browser technologies that require testing on several client configurations.

**Note:** The information in these reports is provided only if the Web server records the agent information. See “Log formats” on page 7.
Broken links

Users may send requests to URIs that are not recognized by the application. In this case, the server returns a *HTTP 404 error*. When a user receives such an error page, he may be discouraged and stop using the application, especially in his first request. (This can be detected as explained in “Entry points” on page 23.)

These 404-type errors can be monitored with a Web analytics report (Figure 21).

![File Not Found Errors](image)

<table>
<thead>
<tr>
<th>File Not Found Errors</th>
<th>Hits</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files Not Found and Referring URL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. /welcome.html</td>
<td>10</td>
<td>100.00%</td>
</tr>
<tr>
<td><a href="http://www.partner.com/partners.html">http://www.partner.com/partners.html</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

*Figure 21  Unrecognized URIs*

**Note:** The referral page information in this report is provided only if the Web server records the referrer information. See “Log formats” on page 7.

Some possible solutions are as follows:

- Fix the broken link or links in the referral page; if the referring URL belongs to an external Web site, contact the external Web site webmaster.
- At a popular unrecognized URI insert an HTML page with a meta tag that redirects to a more appropriate resource, for example to the welcome page:

  ```html
  <meta http-equiv="Refresh" content="0;URL=/index.html">
  ```

- In Web servers the default Error 404 page is rough. Configure a custom error page as explained in “Error page configuration in IBM HTTP Server” on page 37. Tune the page look to match the rest of the application. In the page, try to help the user to satisfy his need alternatively and mention the helpdesk phone number if any.
Entry points

When a user enters a Web application, several problems may arise:

- The user is not logged in. This is not a problem if application security is configured to redirect unlogged users to a login page [Wah03, p348]. In this case a successful logon seamlessly redirects the user to the expected response page.

- The user enters the application in the middle of a use case with wrong HTTP parameters. This leads to an error page, which is not a convenient way to welcome a user. Modify the controller component code to check the referrer and display a more appropriate page.

- The entry point does not exist any more in the Web site; this case boils down to “Broken links” on page 22.

The list of entry points into the Web application can be found in a report as shown in Figure 22.

![Top Entry Pages](chart.png)

<table>
<thead>
<tr>
<th>Pages</th>
<th>Visits</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <a href="http://www.piggybank.com/index.jsp">http://www.piggybank.com/index.jsp</a></td>
<td>542</td>
<td>67.50%</td>
</tr>
<tr>
<td>Total</td>
<td>803</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

*Note:* The information in these reports is provided only if the Web server records the referrer information. See “Log formats” on page 7.
In Figure 22 we can see no entry in the middle of a use case, which is good. We can also see that about a third of the entries are not the welcome page, which is fine because unlogged users are redirected to the login page.

### Exit pages

Sometimes users leave the application from pages that are not intended to be exit pages. For example, the logout screen is an appropriate exit page, while the screen in the middle of a use case can be the sign of a disappointed user. The list of exit pages (Figure 23) can be interpreted in terms of customer behavior and lead to an improving change in the application.

**Figure 23  Top exit pages**

In Figure 23 we can see that users mostly stop using PiggyBank after displaying their account balances or finishing a transfer operation, which are acceptable exit pages. The second page users leave the application from is the transfer form, which denotes an interrupted use case. Maybe some customers forget to have the credit account number at hand and interrupt the transfer operation to find it; therefore the application might be improved to offer the customer a list of his favorite credit accounts.

**Note:** A user is defined as leaving the application when, after a hit, the user does not connect any more for a time longer than the session termination period, the default value of which is usually set to 30 minutes.
User navigation analysis

The report in Figure 24 shows what paths are followed by the visitors through the application.

<table>
<thead>
<tr>
<th>Top Paths Through Site</th>
<th>Paths from Start</th>
<th>Visits</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Entry Pages</td>
<td><a href="http://www.piggybank.com/index.jsp">http://www.piggybank.com/index.jsp</a></td>
<td>533</td>
<td>67.73%</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.piggybank.com/MainMenu.do">http://www.piggybank.com/MainMenu.do</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="http://www.piggybank.com/DisplayAccounts.do">http://www.piggybank.com/DisplayAccounts.do</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="http://www.piggybank.com/TransferAccounts.do">http://www.piggybank.com/TransferAccounts.do</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="http://www.piggybank.com/TransferForm.do">http://www.piggybank.com/TransferForm.do</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="http://www.piggybank.com/Transfer.do">http://www.piggybank.com/Transfer.do</a></td>
<td>114</td>
<td>14.49%</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.piggybank.com/DisplayAccounts.do">http://www.piggybank.com/DisplayAccounts.do</a></td>
<td>60</td>
<td>10.17%</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.piggybank.com/TransferAccounts.do">http://www.piggybank.com/TransferAccounts.do</a></td>
<td>60</td>
<td>7.62%</td>
</tr>
</tbody>
</table>

*Figure 24  Top paths through application*

This report presents a click-by-click view of user behavior. Guidelines to modify the Web application are similar to those explained in “Exit pages” on page 24, with the same objective of improving the users’ experience.

Top navigation paths can also be used to build realistic performance- and load-test plans (see [Sav01] and Chapter 18 in [Wah03]).

**Notes:**
- Abnormal navigation in multiple-step use cases can be detected only if URI format enables users to distinguish the different screens.
- This method is useless if the use cases cannot be distinguished in the URI.
Bandwidth usage

In a typical Web environment, the Web server is connected to a LAN and receives requests from an external network such as the Internet (Figure 25).

![Web environment diagram](image)

Figure 25  Web environment

Because HTTP responses are usually bigger than HTTP requests, most of the Web server traffic is outgoing. This outgoing traffic flows through the Internet router and line. This line is usually a critical resource for the hosting organization, and the bandwidth allocated to the Web server traffic is likely to be limited:

- If the Web server is hosted by an external ISP, bandwidth is a commercial resource and is usually limited for each Web server to a certain amount that depends on the fee.
- If the Web server is hosted within the company network, bandwidth may be limited to prevent Web traffic from disturbing higher-priority network traffic.

When the bandwidth limit has been reached, exceeding client requests are stalled.

Saturation can be detected in a report that shows the hourly number of kilobytes transferred from the Web server to the client browser.
Figure 26 shows the curve of a Web server that suffers no bandwidth saturation.

A similar curve is reproduced in Figure 27 as a green dashed line, and the black line represents the Web server traffic with saturation due to limited bandwidth allocation.

Observe the two horizontal parts of top of the black curve: these show the time period during which the limit is reached. If such a limit is observed in the Web server traffic curve, the bandwidth allocation is likely to be undersized. Check this hypothesis with network administration tools. If it is confirmed, increase the bandwidth of the Web application.
Server errors

Figure 28 report shows for the requests for which the Web server has returned an error code between 500 and 505. Such error codes indicate an internal Web server error.

For example, repeated 500 errors can be caused by uncaught Java exceptions in the Web application.

Web server errors may be critical for Web server or Web application stability and performance. Additional investigation is usually required to track the precise error cause.

Also, it may be interesting to customize the Error 500 page to the Web site general design (see “Error page configuration in IBM HTTP Server” on page 37).

Hourly load analysis

Figure 29 report shows the number of hits for each hour of the day.
This report gives a clear overview of the evolution of the Web site load. The curve in Figure 29 is a typical Web server traffic curve. Several areas can be observed:

1. Night
2. Morning consultations from people who surf at office
3. Lunch time
4. Afternoon consultations from people who surf at office
5. Evening consultations from people who surf at home after a certain hour, when modem/phone communications are cheaper.

The shape of the curve is blurred if the Web server audience is distributed among several time zones.

A table with the corresponding numeric values is also useful (Figure 30).
The Most Active Hour of the Day numeric information (326 in Figure 30) is the peak load of the Web application during the selected time frame.

The average of the 24 numbers—total divided by 24—is 205 in our example, and represents the average load of the Web application during the selected time frame. These values can be used to estimate the corresponding load sizes for the performance and load testing activity (see [Sav01] and Chapter 18 in [Wah03]).

The Least Active Hour of the Day is the period of the day when the fewest users are impacted by Web application updates and maintenance on Web server hardware, software, and content [Joh01].
Organizing use-case development

Activities such as re-engineering, profiling, and debugging are typically concentrated in the end of a project. Practical experience shows projects often end under time pressure. In these conditions, time becomes the critical resource. To save time while optimizing added value of the work, it is essential to organize the work by defining priorities. As a principle says: “when you have many things to do and little time to do it, start with what is the most important.”

For example, in the debugging activity developers fix bugs in the application. To know what bugs have to be fixed, they consult a bug tracking system, where all the bugs are listed and described (Table 2).

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug ID</td>
<td>1</td>
</tr>
<tr>
<td>Status</td>
<td>To be fixed</td>
</tr>
<tr>
<td>Title</td>
<td>Transfer non-number amount</td>
</tr>
<tr>
<td>Use case</td>
<td>Transfer money</td>
</tr>
<tr>
<td>URI</td>
<td>/TransferServlet</td>
</tr>
<tr>
<td>Description</td>
<td>When a non-number string is entered in the amount field, the system returns a wrong error message.</td>
</tr>
<tr>
<td>How to reproduce</td>
<td>Select <em>Transfer Amount</em> in the main menu&lt;br&gt;Select any debit account&lt;br&gt;Enter a valid credit account&lt;br&gt;Enter ‘ABC’ in the <em>Amount</em> field&lt;br&gt;Click <em>Submit</em></td>
</tr>
</tbody>
</table>

Note: The bug tracking system can vary from a simple file edited in a word processor to complex bug tracking system software, such as BugZilla. More information about BugZilla can be found at [http://www.bugzilla.org](http://www.bugzilla.org).

When the application is deployed and is tested by the testing team or by the end users, bugs start to be reported and the bug list starts to grow. When the number of bugs to be fixed becomes overwhelming for the developers, use case priority lists help the developers to know what to do first, while delivering the highest added value. Also, they get a clear path to follow, which is encouraging.
Use-case usage as a priority variable

In general the most-used use case is considered as the most important one and deserves the highest development priority [Joh01]. In the same way, the other use cases can be assigned valued priorities. From these use case priorities, it is possible to order the developers’ tasks while optimizing their added value.

**Note:** This simplification does not take all priorities into account and should be refined according to other actual, imperative priorities. For example, legal or sensitive factors can increase the importance of a use case.

A story that happened to the author illustrates this. It occurred in a European-country public administration, where a custom ERP application was being developed. The application has dozens of use cases, among them these two:

- Electronic scheduling: Several times a day employees click check-in and check-out buttons as soon as they respectively start or stop working. This data was used by managers to control and analyze employee availability.
- Workflow: Employees weekly enter their activities, along with the number of hours spent on each activity. This data was used to compute the employees’ payment. Also, in the country the law says administration employees must get paid before the end of each month.

 Needless to say the second use case, although it was not the most used, had better be running first and fine!! It is an example in which legal and sensitive factors prevail.

To know the usage of each use case, different Web analytics reports can be used. The choice of the report depends on the format of the URIs that are used to communicate between the browser and the server:

1. If each use case is mapped to exactly one URI, use a report that shows the top URIs.
2. If each screen of a use case is mapped to a different URI, group the URIs for each use case and compute the hits for each group; this can be done manually, or automated if the tool supports the feature.
3. If all use cases are mapped to a single URI, standard Web analytics reports cannot help.

In any case, another solution to obtain the use-case usage is to manually instrument the application code (see “HTTP parameters” on page 40).
One use case - one URI

If each use case is mapped to exactly one URI, it is possible to know the use-case usage from a report that shows a list of the most-viewed URIs (Figure 31). The example in Figure 31 does not apply to PiggyBank-Struts but to another version of PiggyBank, called PiggyBank-Servlets, that maps each use case to a Java servlet. PiggyBank-Servlets is presented in detail in [Wah03].

![Top Pages by Views](image)

**Figure 31**  Top URIs

<table>
<thead>
<tr>
<th>Pages</th>
<th>Visits</th>
<th>Views</th>
<th>%</th>
<th>Avg Time Viewed</th>
<th>Avg Time to Serve</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <a href="http://www.piggybank.com/TransferServlet">http://www.piggybank.com/TransferServlet</a></td>
<td>721</td>
<td>2,583</td>
<td>52.36%</td>
<td>00:00:14</td>
<td>0</td>
</tr>
<tr>
<td>2. <a href="http://www.piggybank.com/DisplayAccountsServlet">http://www.piggybank.com/DisplayAccountsServlet</a></td>
<td>624</td>
<td>1,266</td>
<td>25.66%</td>
<td>00:00:14</td>
<td>0</td>
</tr>
<tr>
<td>3. <a href="http://www.piggybank.com/MainMenuServlet">http://www.piggybank.com/MainMenuServlet</a></td>
<td>542</td>
<td>542</td>
<td>10.99%</td>
<td>00:00:14</td>
<td>0</td>
</tr>
<tr>
<td>4. <a href="http://www.piggybank.com/index.jsp">http://www.piggybank.com/index.jsp</a></td>
<td>542</td>
<td>542</td>
<td>10.99%</td>
<td>00:00:14</td>
<td>0</td>
</tr>
</tbody>
</table>

**Total** | 2,429 | 4,833 | 100.00% | 00:00:14 |

**Attention:** In the report of Figure 31, the term *page* is misleading because it is used in place of the term *URI*. In a static Web site, the confusion does not matter, because each URI is associated with exactly one page. In a Web application, a URI can display several pages, so in this context the term *page* in the figure is not correct and should be read *URI*.

This report can be used to organize use-case development if the URI exactly identifies one use case. In these URI formats, each use case is associated with one URI. Thus each request to a use-case step is one hit to the use-case URI. Therefore, the report in Figure 31 shows the number of times a step of a use
case is called, with the results grouped by URI (that is, by use case). To know the number of times a use case is used, we divide the number of hits displayed in the report by the number of steps, as represented in the following equation:

\[ \text{Use-case usage} = \frac{\text{hits}}{\text{steps}} \]

In this formula, \( \text{hits} \) is the number of hits for all the screens of the use case and \( \text{steps} \) is the average number of screens the user has to get through to complete the use case.

This result is only approximate because of two disruptive factors:

- A use case is not always completed by the user.
- Some steps can be repeated because of errors.

If we note the factors \( e_1 \) and \( e_2 \) and introduce them into the equation, we can see that they balance each other:

\[ \text{Use-case usage} = \frac{\text{hits} + e_1 - e_2}{\text{steps}} \]

For the sake of simplicity we consider that the two factors are roughly equal and use the first equation result as the use case usage.

From the numbers in Figure 31 we can conclude the most-used use case in PiggyBank-Servlets is Transfer Money.

**One use case - several URIs**

If several URIs are associated with one use case (for example, if each screen has its own URI as in PiggyBank-Struts), the use-case usage must take into account the hits of each URI. One way of doing this is to manually add the number of hits for every one of the URIs that comprise the use case. For real-world applications with many use cases and screens per use case, this task can become tedious. For such applications, it is convenient to automate the addition of the hits by grouping URIs into URI groups (Table 3) and using a report that shows the hits of each URI group (Figure 32).

<table>
<thead>
<tr>
<th>Use case</th>
<th>URIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Accounts</td>
<td>DisplayAccounts.do</td>
</tr>
<tr>
<td>Transfer</td>
<td>TransferAccounts.do</td>
</tr>
<tr>
<td></td>
<td>TransferForm.do</td>
</tr>
<tr>
<td></td>
<td>Transfer.do</td>
</tr>
</tbody>
</table>
By definition, the number of hits for a URI group is the sum of the hits for every URI in the collection. The use-case usage can therefore be computed by the equation:

\[ \text{Use-case usage} = \frac{\text{URI Group hits}}{\text{steps}} \]

From the numbers in Figure 32 we conclude that the most-used use case in PiggyBank-Struts is Transfer Money.

The URI-grouping feature is supported by some tools only. In WebTrends Log Analyzer the feature is called content groups. The groups can be defined using regular expressions (Figure 33).
Client operating systems

If along with the Web application a fat-client application (local security proxy, off-line services, installers, and so on) is planned to be developed, and deployed on the users’ machines, it may be interesting to know what proportion of operating systems run on these machines. This information can be obtained from the use of the Web application, in a Web analytics report as shown in Figure 34.

![Top Platforms by Visits](image)

### Top Platforms

<table>
<thead>
<tr>
<th>Platform</th>
<th>Visits</th>
<th>%</th>
<th>Views</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>158</td>
<td>20.52%</td>
<td>996</td>
</tr>
<tr>
<td>Windows 2000</td>
<td>116</td>
<td>15.06%</td>
<td>723</td>
</tr>
<tr>
<td>Windows NT</td>
<td>111</td>
<td>14.42%</td>
<td>715</td>
</tr>
<tr>
<td>Windows 98</td>
<td>110</td>
<td>14.29%</td>
<td>693</td>
</tr>
<tr>
<td>Windows XP</td>
<td>94</td>
<td>12.21%</td>
<td>557</td>
</tr>
<tr>
<td>Others</td>
<td>63</td>
<td>8.18%</td>
<td>334</td>
</tr>
</tbody>
</table>

**Figure 34  Top platforms**

**Note:** The information in this report is provided only if the Web server records the agent information. See “Log formats” on page 7.

**Anticipate the future**

Estimating the future evolution of a Web application can help to anticipate the needs. While impact of exceptional external factors is better estimated by human evaluation, the computer can help to estimate the future evolution of a Web application in “normal” circumstances.

Every Web analytics report can be generated for several time frames. Select the maximum time frame. On the basis of the curve, visually estimate the future evolution.
To accurately predict future application usage, export the numbers from reports to files. The data can then be submitted to predictive computations on the basis of mathematical models, as explained in [Dhy02a, Dhy02b, Dhy03].

Design and authoring efforts to update the application can be predicted using Web engineering metrics and models [Men01].

**Error page configuration in IBM HTTP Server**

In IBM HTTP Server, a different error page can be configured for each HTTP error code (Figure 35).

![Error page configuration in IBM HTTP Server](image)

*Figure 35  Error page configuration in IBM HTTP Server*

Example 6 shows corresponding entries in the Web server configuration file.
Web Analytics

**Example 6  Error page entries in Web server configuration file**

- ErrorDocument 404 /FileNotFound.html
- ErrorDocument 500 /ServerError.html

**What tool to use?**

Hundreds of Web analytics tools are available on the market [Fri03]. There are few differences between them. The main choice criterion is the Web usage data gathering method.

Screen captures in this paper were taken with WebTrends Log Analyzer. This tool has been used to test all the methods provided in the paper, and supports all the cited features. Chapter 19 in [Wah03] explains in detail how to use WebTrends Log Analyzer for Web applications.

IBM offers two tools:
- Outsourced: IBM SurfAid™ [IBM04]
- For in-house usage: Tivoli® WebSite Analyzer [Dom01a, Dom01b]

The choice between outsourced and in-house solutions is discussed in [Mal01a].

**Problems**

Some problems prevent Web analytics tools from delivering optimal added value. In this section, we present a few problems and give insights for solutions.

**Internal hits**

After the Web application is deployed, some requests are sent to the Web server by the internal development team (especially the testers). These hits should not be considered in data analysis. Some Web analytics offer the feature to exclude groups of client machines from data analysis. In WebTrends Log Analyzer, this feature is called *Intranet Domains*.

**Data volume**

Large log files cause several problems:
- Writing logs into a big log file slows down the Web server performance. A solution is to rotate the log file periodically, usually daily. To avoid repeated
administration overhead, the rotating task can be included in the server operating system task scheduler.

- Disk storage can become insufficient. Compress the log files. The compression ratio for log files is 95% to 98%. The commonly used compression algorithm is GZIP, which is supported by most platforms (including Windows® and UNIX®) and most tools, and is more free of rights than the ZIP algorithm.

- The duration of data analysis is an exponential function of the number of log lines. Use analysis time frames as short as possible.

Example: If the Web server is configured with the NCSA combined log file format, every request to a file makes the log file grow by about 200 bytes. If the Web server receives 2,000,000 hits a day, logging requires 400 MB disk storage a day, or 120 GB a year. Compression brings this number down to a reasonable size: about 4 GB a year. This takes days to be analyzed by a Web analytics tool.

It is possible to reduce the log file size by deleting the log lines for images and CSS (Cascading Style Sheets). This can reduce the data volume by an order of magnitude depending on the number of images in the screens. A drawback is that some reports are not correct any more; for example, all hits must be analyzed to produce the report in Figure 26 on page 27.

**IP multiplexing**

In some organizations, browsers do not connect directly to Web sites, but pass on the requests to a proxycache (Figure 36).

![Proxycache Diagram](image)

*Figure 36  Proxycache*
When a Web server receives a request from a proxycache, it knows only the IP address of the proxycache. The IP address of the requesting machine is hidden from the Web server. Therefore, all the browsers using the same proxycache are seen by the Web servers as a single machine. This blurs the reports that exploit the IP addresses, such as Figure 24 on page 25.

A solution to resolve IP multiplexing is to track client machines with an identifying cookie [Tom04]. This solution is implemented in the single-pixel data-gathering method.

**HTTP parameters**

In standard Web analytics tools, HTTP parameters are not considered. A few specific tools on the market try to exploit information in HTTP parameters but fail to deliver a generic solution.

For needs that are not covered by basic Web analytics software features, the best solution remains code instrumentation. Code instrumentation consists of developing custom logging inside the Web application. An interesting IBM framework for code instrumentation is ALS (Analytic Logging Service) [Ngu01].

**User privacy**

User privacy is a sensitive topic. Users are usually willing to share information for better service. However, users also typically prefer that they be asked for their permission to send them marketing e-mail or to share their contact information with partner companies. If a site provides a privacy statement documenting the intended uses, and gives users an e-mail address for comments, users can determine whether the policy is acceptable.

To start, decide how to use the information that is recorded about the users. Then write a privacy statement based on the users' point of view, and make that statement available on the Web site.

A World Wide Web Consortium (W3C) project called *Platform for Privacy Protection* (P3P) [W3C04] is helping to define and standardize the policies for data collection and the legitimate uses of this data.
Web services

Web server logging has not been designed to gather Web service usage data. To remedy this situation, a draft list of requirements has been submitted in a public note from IBM to W3C Web services working group. Main requirements of this list are the logging of:

- RPC type
- Encryption type
- Method signature
- SOAP-engine performance information

The note suggests using the W3C Extended Log Format, which is already recognized by Web analytics products. Though it is clear that a new, standard, Web services logging format must be defined, it is not so clear that this log format must be implemented by the Web server or by the SOAP engine. The answer to this question probably lies in the location where the information is more easily available.

Other requirements have been introduced to log the data exchanged between the client and the server, back and forth. This requirement exists for all HTTP requests that use the POST method, and are not specific to Web services.

Summary

In this paper, we introduced Web analytics and the sample PiggyBank application.

We examined several Web analytics reports and presented methods to modify the production environment, schedule Web application maintenance, design pages with best-supported Web-client technologies, deploy best-supported stand-alone clients, prioritize re-engineering, debugging, and profiling, extract users’ average behavior, fix broken links, investigate critical server errors, and anticipate future evolution of a Web application. We applied these methods to PiggyBank.

Examining Web analytics reports where URIs are involved and studying the impact of URI formats, we have seen that avoiding using HTTP parameters in URIs to control application and use case flow improves the usefulness of the reports.

Finally, we gave some insights for solutions for internal hits, data volume, IP multiplexing, HTTP parameters, and user privacy.
References

[APA04a] Apache HTTP Server documentation, log files:
  http://httpd.apache.org/docs/logs.html

[APA04b] Apache HTTP Server documentation, log files configuration:
  http://httpd.apache.org/docs/mod/mod_log_config.html

  http://jiad.org/vol1/no2/coffey/


[Dom01a] S. Dominick et al., Introduction to Web Analytics Using WebSphere Site Analyzer, 2001, WebSphere Developer Domain:

[Dom01b] S. Dominick et al., Advanced Web Analysis Reporting using WebSphere Site Analyzer, 2001, WebSphere Developer Domain:


[IBM04] IBM SurfAid Analytics:
  http://www.ibm.com/surfaid/

[IET04] Fielding, et al., Hypertext Transfer Protocol -- HTTP/1.1, IETF RFC 2616:
  http://www.ietf.org/rfc/rfc2616.txt

[Mal01a] A. Malicinski, S. Dominick, T. Hartrick, *Measuring Web traffic, Part 1: Getting to know your audience is the key to online success — make the numbers work for you*, 2001, IBM developerWorks:


   http://www.redbooks.ibm.com/abstracts/sg246931.html

   http://www7b.boulder.ibm.com/wsdd/techjournal/0106_nguyen/nguyen.html


[W3C04] W3C P3P, Platform for Privacy Preferences:
   http://www.w3.org/P3P/

   http://www.redbooks.ibm.com/abstracts/sg246993.html
About the author

Jean-Pierre Norguet has been working three years with IBM on application development, as team leader and coach. He holds a Ph.D in Computer Science from Universite de Bruxelles (ULB, Belgium). His areas of expertise and publication include the complete application development lifecycle and Web analytics.

Acknowledgements

Contributions to this Redpaper have been received from the following persons:

- Ueli Wahli, IBM International Technical Support Organization
- Jonas Andersen, IBM Global Services
- Esteban Zimanyi, Université de Bruxelles—http://cs.ulb.ac.be/
- Joël Cannau, Université de Bruxelles—http://cs.ulb.ac.be/
Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

© Copyright International Business Machines Corporation 2006. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

IBM® SurfAid™ WebSphere®
Redbooks (logo) ™ Tivoli®

Other company, product, or service names may be trademarks or service marks of others.