PERSONAL WEBMELODY: CUSTOMIZED SONIFICATION OF WEB SERVERS

Maria Barra *Tania Cillo* *Antonio De Santis*
Umberto Ferraro Petrillo
Alberto Negro *Vittorio Scarano*

Dipartimento di Informatica ed Applicazioni
Università di Salerno
Baronissi (Salerno), 84081 Italy

{marbar,umbfer,alberto,vitsca}@unisa.it

Teenie Matlock

Paul P. Maglio

IBM Almaden Research Center
650 Harry Rd., NWE-B2
San Jose, CA 95120 USA

{tmatlock, pmaglio}@almaden.ibm.com

ABSTRACT

This paper presents Personal WebMelody, a sonified web server that informs its administrator of both normal and abnormal operation through background music. It allows customization and full integration of system-generated music representing web server activity with external music sources (audio CD, MP3, etc) selected by the administrator.
Our sonification technique works by associating MIDI or WAV sound tracks with web server events. In an attempt to enable the webmaster to listen to such system-generated music for a long period without becoming fatigued, we introduce the opportunity of mixing an external music source with system-generated music. In this way, the administrator can *hear* the status of the web server while listening to his or her preferred music.
We present an empirical study that shows how our web server sonification can convey useful information efficiently.

1. INTRODUCTION

World Wide Web servers are growing in size, complexity and workload. For popular servers, millions of requests are often served everyday. These requests are logged to files so that they can later be analyzed to find information for tuning and malfunctioning. Although web servers usually offer the capability to choose the kind of information that is logged, given the workload, these logs can be enormous. Analysis is greatly helped (and in some cases even made possible) by programs that perform statistics on massive amounts of log data. It is common, therefore, to monitor a web server off-line, either on a regular basis or when prompted by evident malfunctioning. But for web servers that are used as a company asset, for instance, to showcase products and to support web-based commerce, timely monitoring of server behavior and access patterns is critical, as it could very well mean the difference between making money and losing money. Thus, it is important to have an effective and efficient way to monitor in *real-time* the behavior of the server so that prompt action can be taken to fix a malfunctioning server or to fine-tune the system.
In this paper, we present Personal WebMelody, a sonified Web server that allows customization and full integration of system-generated music with external music sources (audio CD, MP3, etc.). As will be discussed, experiments conducted with the previous version of WebMelody **[2]** suggest that music can convey peripheral information **[19]** efficiently. Though it remains to be shown that such system-generated music can indeed be listened to for a long period without inducing mental fatigue in the listener, Personal WebMelody is meant to minimize such potential limitations by integrating system-generated music with personal musical selections. To convey server information, our system indicates server state by changing the volume of the personal music source and by mixing in additional sounds represented by MIDI tracks.
We believe Personal WebMelody offers an effective and efficient way for real-time monitoring of the server. By real-time monitoring, we mean that the system provides feedback to the webmaster of a web server event within a short, fixed time interval, δ. Note that ordinary log analyzers cannot achieve such real-time monitoring as they generally rely on log-file information that is aggregated and assembled in a variety of ways. Even using log-analyzers periodically (e.g., every hour) still has the drawback that information must be explicitly requested[1] by the webmaster. Our system solves this problem by pushing to the administrator a constant flow of information about the status of the web server. Though on the one hand, real-time monitoring ought to be useful and effective because it enables constant awareness of the server's state, on the other, it can only be effective if the user interface is not too intrusive, constantly interrupting the webmaster. Because Personal WebMelody relies on sounds --- music in particular --- to convey information to the webmaster, we believe it is critical to offer the user a choice of musical representations of server behavior. Thus, our system is fully configurable, using customizable sounds to represent server behavior and integrating personal external audio sources.
This paper is organized in four parts. First, we provide an overview of the Personal WebMelody system including its motivations and related approaches. Second, we detail the architecture and implementation of the system, including the features that make Personal WebMelody an extension of our previous system **[2]** and specific technical problems we faced. Third, we describe results of empirical studies on users that suggest that the sorts of sonified web servers we describe are

[1] A possible exception is the capability to send alarm messages via messages on cellular phones (via SMS) or faxes but they are used only to inform of evident and severe malfunctioning (server and/or connection down).

not too distracting yet can convey useful and timely information. Finally, we conclude with lessons learned and directions for future work..

2. MONITORING WEB SERVERS THROUGH SONIFICATION

Personal WebMelody is meant to convey peripheral information efficiently through sound, by associating web server events with specific musical events. In this section, we describe related approaches to sonification, and the state of the art in web server monitoring tools.

2.1.1. *Music as peripheral information*

Because web server administration has the general character of a background task, our objective was to build a system that informs in the periphery. In this case, *peripheral information* (i.e., information not critical to primary task performance but potentially helpful or useful nevertheless [19]) is delivered through sound, as sound does not take up precious screen space yet it can effectively convey certain kinds of status [4],[7],[8].
There were several reasons for choosing to use musical sound over non-musical sound. First, our intention was to provide auditory indications of both working and non-working states. Using sound to signal that the system is operating normally (in addition to using sound to indicate that the system is operating abnormally) requires a fairly constant auditory stream. Because periodic background sounds can be distracting (e.g., [19]), we decided to try background music instead, as it can be less distracting (e.g., [22]). Second, music has been found to be more aesthetically pleasing than other types of background noises (such as synthetic voices or computerized beeps) and even preferable to a totally quiet office [26]. Moreover, musical cues can be easily tailored or customized to individual tastes. In addition, our intention was to provide background information that would be easily distinguished from ordinary office and computer alerts, such as the ringing of phones and beeps of computers.
Finally, our system allows to use also external sound sources to convey information as well. Our guess is that by manipulating such sources, for example, changing the audio output volume[2] or mixing a source with expressive sound cues, it is possible to push information to the user while at the same time allowing the user to listen to preferred music. In this way, the user can effectively receive information describing the status of the server for a long period of time.
Research on auditory icons and “earcons” focuses on how to add sounds to computer interfaces so as to simplify or otherwise facilitate interaction [5],[11]. In such cases, sounds presented with visual icons and physical actions have been shown to speed responses and decrease errors [11]. This sort of sonification tends to use individual sounds rather than music—and then only when the sounds are in coordination with visual cues. Along similar lines, others have created musical sonifications to help users understand complicated data sets, but again only when in coordination with visual information (e.g., [16]).
By contrast, Cohen’s ShareMon system [8] is a clear antecedent of our sonified Web server, as it conveys status information in the background using subtle sounds. Cohen developed a repertoire of reasonable metaphorical mappings between file server events and sounds, such as a login on the server mapped to a knocking sound. Though conceptually appealing and anecdotally judged to be useful, because the evaluation of the ShareMon was done in a qualitative rather than controlled and quantitative way, very little can be learned about the effectiveness of this kind of peripheral display.

[2] Note that care is required in the choice of adequate background music as specified later in paragraph 3.1.1.

2.1.2. *Techniques for monitoring Web servers*

As mentioned, web servers are usually monitored by analyzing log files that contain information about each HTTP request, its consequent response, and any errors that arise. According to the Common Log File Format [18] each row in the file has the following format: `remotehost remotelogname authuser [date] "request" status bytes.`
Many servers allow log data to be stored in a proprietary format. For example, an Apache module, `mod_log_config`, allows the Webmaster to configure the information to be logged. Recently, *Extended Log File Format* was proposed [13], which allows capturing a wider range of information that might be useful for detailed demographic analyses.
Because of the high workload of popular web servers, log files often grow to hundred of megabytes. This makes analysis possible only with tools (known as *Log Analyzers*) whose goal is to present data aggregated and assembled so that they effectively represent a snapshot of the behavior and functioning of the server over a given time interval. There are several popular Log Analyzers, as Analog (http://www.statslab.cam.ac.uk/~sret1/analog/), Wusage (http://www.boutell.com/wusage) and Access Watch (http://accesswatch.com/). These can be used to discover many parameters and statistics that can be visualized through graphical interfaces. For example, a typical log analyzer output might show which pages on a web site are the most popular, which countries (i.e., domains) most accesses are coming from, which pages outside the site contain broken links to pages on the website, and so on. Among the most versatile Log Analyzers is WebTrends (http://www.webtrends.com), which allows the creation of filters to monitor several parameters together. Nevertheless, these filters are limited to either “Include all the following condition” or “Exclude all the following conditions” without regular expression matching.
The desire to integrate several views of the log file has motivated some to add a third dimension to the graphical displays: 3Dstats (http://www.netstore.de/Supply/3Dstats/) analyzes log files to generate three-dimensional VRML models of the server's load by month, by day, and by hour. With a VRML browser, the webmaster can "walk" through the scene to examine the bar chart from different points of view. The Avatar Visualization System [23] combines 3D visualization with real-time monitoring by providing a projection of the dynamic behavior of the web servers. In particular, the system represents the real-time performance data for a NCSA's web servers [14], displaying and analyzing time varying data with the Avatar virtual environment.
Regardless of aggregation method or display format, real-time monitoring represents an important advantage in observing the behavior of a web server. Some products do provide (to some extent) real-time features through graphical displays and alarm mechanisms. But real-time monitoring is generally limited to severe errors (e.g., connection down, server down), and to

consequent messages sent through e-mail, pagers or faxes. For instance, AlertSite (http://www.alertsite.com) checks the server up to 12 times per hour and sends an alert message (via e-mail or pager) when the system does not work correctly (including slow connections). Systems such as @Watch (http://www.atwatch.com) simulate a browser that visits the site to check links. Another external tool, WatchWise (http://www.watchwise.com), requires that each page to be monitored include a tag that forces the browser to send information to the WatchWise server.

2.1.3. Comparison with Personal WebMelody

- Personal WebMelody is meant to be non-intrusive yet informative. Using sounds to represent events in the server and allowing their integration with external audio sources (whose volume follows the workload of the server), our system can usefully supplement information provided by log analyzers without unnecessarily averting the webmaster's focus on daily activities.
- Personal WebMelody offers true real-time monitoring. The system does not rely on the analysis of log files and, therefore, it is able to push to the clients a stream of information that renders the server status within a small, fixed time interval, and without an explicit request from the webmaster (server push).
- Personal WebMelody is complete. Because it does not rely on log files, it can monitor *all* the HTTP headers that are exchanged between client and server, not only those that are sent to the log files.
- Personal WebMelody is versatile. It can be completely configured by the webmaster: boolean operators and regular expressions can be used to build arbitrarily complex patterns to monitor. Moreover, the webmaster can choose an external sound source to monitor the web server behavior.
- Personal WebMelody combines true monitoring and alarms. Rather than pushing information to the webmaster only in case of severe errors (as some products do) the system can push a stream of information that can be used not only for alarms but also for "keeping an eye" on arbitrarily complex access patterns.

3. PERSONAL WEBMELODY

Our sonified web server was implemented using a three-level distributed architecture. Events that can be monitored include load, throughput, and almost any attribute of individual requests and responses. Our system is flexible because it can be configured to associate individual or group web server events with specific musical tracks, allowing an administrator to customize and monitor the web server's state. There are real-life situations in which such a real-time tool can help a web administrator. For example, by monitoring web server behavior via sonification the administrator is able to

- Detect "denial of service" attacks, in which the server is bombarded by too many requests to handle all at once. In this case, music can easily indicate that the load or throughput has increased and reached the maximum for a long period.
- Determine that the server has gone down by silence, as there would be no activity to generate music.
- Discern different types of requests; for example, determine where requests come from (by matching the top-level-domain of the client), or discriminating accesses by different browsers. If two different conditions are to be monitored on the same HTTP header, it might be useful to associate two opposite tracks (same notes but by different instruments, for example) with the monitored events so that the administrator can pick out which is the most frequent.
- Monitor access to restricted (e.g., by user authentication) directories. In addition, by cross-monitoring (with a complex event) unsuccessful authentications, the administrator can also know if many users are providing bad credentials.
- Recognize the different error types and the HTTP method of the request. For example, it is possible to monitor "many" broken links in a newly created section of the web server by creating a complex event with HTTP error 404 ("File not found") with pathnames in a given directory and an internal HTTP-Referer (i.e., links followed by a page from the same server).
- Associate load peaks with other monitored events. In fact, by musically recognizing the co-occurrence of several tracks at once, the administrator might be able to associate a heavy load period with accesses to a particular directory, providing some possible explanation in real-time of unusual load peaks.

Easy configuration and control over the musical tracks allow for efficient personalization of the musical representation of web server behavior according to personal taste, cultural heritage, workplace (physical environment, noisiness, hardware limitation), and so forth. Of course, our sonification technique is not meant to substitute for off-line analysis. Rather, it is meant to complement log analyzers by giving administrators immediate access to what is going on. Prompt recognition of server activity is obtained through the human ability to focus attention on a particular background sound when needed. At the same time, we hope to leverage on the human ability to recognize composite musical patterns [9].

3.1. Architecture

We have developed Personal WebMelody to satisfy a number of architectural requirements. First, the system must be easily configurable since it is well recognized that personal characteristics of users influence how they hear and feel about the information represented by sounds. To this end, the system should also allow the mixing of the sounds generated by the sonification of the server with an external sound source chosen by the user (e.g. MP3 files, audio CD). Second, the architecture must be distributed, as a popular web server is probably not a simple workstation located in the webmaster's office, but more likely a cluster of machines physically located in a computer room-. Because audio devices are cheaply available for PCs or workstations, it is natural to think of the output as being played by a workstation different from the server. A distributed architecture offers the possibility to re-distribute the musical information to different personal workstations so that different people can be monitoring the same server. Third, the system must be portable and, therefore, the "player component" should be able to run easily on several different platforms.

3.1.1. Representing Web Server Behavior

How much server information can be effectively conveyed through music? We believe that there are a small number of different coordinates that can be merged into music: the limit depends on the musical ability of the webmaster and the degree of attention that he or she is willing to devote to the sound.
We distinguish three categories of web server events that can be effectively associated with sounds:

1. Information describing the workload of the server. It is important to recognize workload peaks during certain periods of the day and be able to sense that the system is running. A normal arrival rate of requests should be heard as a (non-disturbing) background sound and the absence of it can really be very expressive, indicating that the server is off, or that the connection with the rest of the Internet is down. On the other side, if we want the users to keep their favorite background music while monitoring the behavior of the server, we can convey information through the manipulation of the audio output volume by associating high workload to louder volumes. Note that in choosing the musical track to represent load, the user should choose something that would be appropriate as background music. For instance, something that is rather smooth in volume and that sounds good when played at low-volume would make reasonable background music and thus could effectively be used by Personal Webmalody to represent load information.
   It is important to note that log analyzers offer statistics based on the throughput, that is, number of served HTTP requests within a time slice, as log files maintain information about the time of completion of requests. Our real-time monitoring system can receive more precise information about current load expressed in terms of pending requests. The webmaster can choose to monitor either throughput or load during configuration. In some cases, it can be significant to monitor the load rather than the throughput, for example, for heavily accessed servers with large data files or complex cgi-bin scripts that may require additional access to other servers (e.g. DBMSs). High load might indicate, for example, bandwidth limitations (resulting in slow connections toward clients) or other malfunctioning.
2. Information reporting "severe errors" in web server activity should be immediately recognizable (alarm sounds). In case we are using an external sound source, such errors should be represented through easily distinguishable sound cues played with a louder volume. In this category, we find, for example, server errors (i.e., HTTP responses with status code 5xx) and client errors (i.e., HTTP responses with status code 4xx) when there are more than a predetermined threshold. Of course, a severe error is that the server is reported to be not accessible, which can be due to either the server being actually down or network problems.
3. Information denoting the normal behavior of the web server. In this category, we find most of the situations to be monitored. In this case, we associate each event with a single audio track, such as the main part of a musical theme. If we are playing a background sound track, we can report this information using sounds having a musical structure that is neutral with respect to the usual and conventional musical themes. Using this approach, the way the user perceives the original sound source is not affected by the mixed sound cues. For example, one would like to monitor requests from certain domains, or requests for files in a certain directory, or redirections (HTTP status code 3xx), or combinations thereof.

3.1.2. Components

The architecture of our prototype consists of three components: the sonification Apache module **mod_musical_log**, the Collector server, and the WebPlayer application (see Figure 1):

- The sonification module is run inside the Apache server, it intercepts each HTTP request and response received by the web server. First, it sends the Collector information about the arrival of a request. When the request is served, relevant it sends additional information to the Collector according to the rules for filtering request and response parameters.
- The Collector is the middle level component of our architecture. It acts as the conductor of the sonification process. The Collector buffers events provided by the server, parses them and then instructs the remote WebPlayer about the sounds to be played. It is in the position to monitor load and throughput of the server, as well as analyze the events that must be played only if a threshold is passed. Finally, a command string specifying the sounds to be played is produced and sent to the WebPlayer.
- The WebPlayer produces audio output. At startup, the WebPlayer downloads several sets of sound files from the network using the information provided by the **mod_musical_log** through the Collector. The loaded sound files are then played according to the command strings periodically received from the Collector. Optionally, the user can choose to mix the sounds used as representation of server's behavior with an external audio source.

3.1.3. Apache module mod_musical_log

Apache, currently the most popular WWW server (see http://www.netcraft.com/survey), allows for modules to be added, enabling functionality to be plugged in by local web administrators. We have developed the **mod_musical_log** module, which is hooked into request processing to intercept all information available at the beginning and at the end of processing (i.e., during **post_read_request** and **logging** phases of processing).
These data are gathered from the server according to a set of sonification patterns specified by *Channel* descriptions. These include a set of rules describing which information is to be monitored and how this information is to be translated into sounds. When an HTTP request or response matches one of the patterns, a new remote event is generated and then sent to the Collector. More precisely, channels are defined using our new **<MusicalLog>** container directive in the **httpd.conf** file.
A Channel definition includes a description (or name), the location of sound tracks (MIDI or WAV), and sonification directives that associate web server events with musical events. In addition to using the Channel definitions itself, the **mod_musical_log** module also sends this information to the Collector, which needs to know how to map events to sounds.

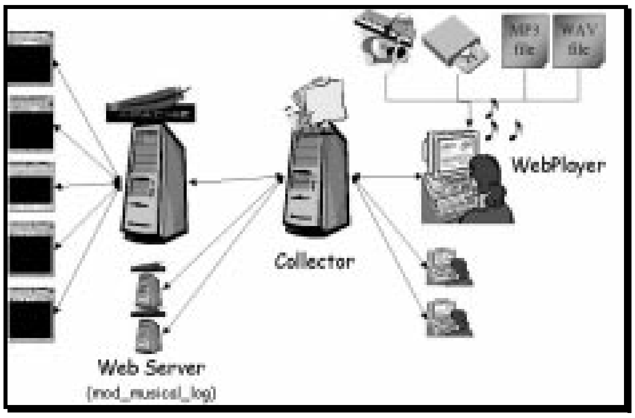

Figure 1 *The sonified web server is composed of three components, an Apache module that sends events to an event Collector, which in turn can provide aggregated events to the WebPlayer application that also gets the input of an external audio source.*

Our current implementation of the sonification module supports three types of events:

1. Simple events, generated by **mod_musical_log** when an HTTP request or response exactly matches one of:
   - HTTP status codes;
   - HTTP headers, divided in RequestHeaders (that include general-headers, request-headers, and entity-headers) and ResponseHeaders (that include general-headers, response-headers, and entity-headers);
   - HTTP methods of a request;
   - HTTP version of a request;
   - Request-URI of a request;
   - Remote host (or client IP address).
2. Complex events, generated by composing simple events with logical AND and NOT.
3. Workload events, generated based on responses served or requests pending. Using these events, the Collector can know the exact number of requests currently being processed by the server, which is a measure of load. Each time the server receives a new request, during the **post_read_request** phase, **mod_musical_log** sends a "debit" event to the Collector. When a request is served, it sends a "credit" event to the Collector that can correctly evaluate both throughput and load.

We briefly comment now this example of configuration.

```
<MusicalLog>
   MusicalPlayer "neverland.dia.unisa.it"
   MusicalPlayer "localhost"
   EventsCollector wonderland.dia.unisa.it:7000
   StatisticsFrequency 30
  <Channel>
    ChannelName "demo"
    ChannelDescription  "Example"
    SampleType Midi
    SampleSet "http://isis.dia.unisa.it/son.mid"
    TracksNumber 16
    Throughput [1-6]  5 10 1
    RequestHeader User-Agent "MSIE" 7
    RequestUri "/~vitsca/" 8
    StatusCode "404"  9
    StatusCode "5.."  10
    <ComplexEvent>
        RequestHeader User-Agent "Mozilla"
        RequestHeader User-Agent ! "MSIE"
        RemoteHost "(\.it)$"
        Track 11
    </ComplexEvent>
    <ComplexEvent>
        RequestHeader Authorization  ".+"
        StatusCode  "401"
        Track 12 10 5
    </ComplexEvent>
  </Channel>
 </MusicalLog>
```

After information about location of distributed components (in order to be able to authorize accesses), a channel called "demo" is defined. A musical log configuration directive can include several channels and the Webmaster can select (by the WebPlayer) the channel he/she is more interested to.

Throughput is monitored by tracks 1 to 6: a new track is played if the throughput grows by 5 units per time interval (set to 10) where the sliding temporal window is moved each second to the right.

Then we define several simple events and associate them to tracks. Track 7 is used to monitor accessed by Internet Explorer, track 8 for accesses to home directory of user `vitsca`, and track 9 and 10 monitor several kind of HTTP error codes, respectively, 404 and any server error, i.e. 5xx by using the regular expression "5..".

Then we define two complex events. Track 11 monitors accesses by Netscape[3] coming from italian geographic top-level-domain. The three conditions are connected by a logical AND and also boolean NOT is used as "!" in the second condition. Then, frequent (i.e. more than 10 in 5 seconds) failed authentication (with wrong credentials) are monitored by track 13 by using the request header used for authentication and the failed authentication response.

3.1.4. *Collector*

In our architecture, the Collector lies between the web server and the WebPlayer application. In this way, the Apache module can simply provide statistics on the workload and analysis of events within some threshold based on the output of all the children processes of the Apache server. Moreover, by introducing a middle layer, it is easy to (a) manage multiple players, (b) use the Collector to receive data from other sources (servers of any type), and (c) have the player work on different Channels coming from different servers. The Collector processes batches of events according to thresholds specified in the configuration file. In particular, it collects events for the interval **StatisticsFrequency**, which means that an event that happened on the server at time *t* is sent at most *StatisticsFrequency* seconds later to the WebPlayer.

The Collector deals both with the sonification modules and the WebPlayer applications. The communication protocol between the sonification module **mod_musical_log** and the Collector operates in two main phases:

1. Connection initialization and configuration: At startup, the Collector waits for a connection request from a remote sonification module. Upon successfully establishing a connection, the Collector receives information describing the Channels.
2. Simple/Complex/Load Events Transmission: The Collector receives event notifications from the **mod_musical_log** module For each received event, the

[3] Because of the way IE presents itself to the server in the User-Agent field by using also the string "Mozilla ", we must exclude the string "MSIE" to succesfully select Netscape.

Collector checks if the event has a threshold. If there exists a threshold for the event, the Collector checks if in the current timeslice (sliding temporal window) the counter associated with the event has reached the threshold. In such a case, the related sound request is generated. If the event has no threshold, the sound request is also generated. The load is evaluated by the Collector by adding or subtracting a load counter each time a "debit" or "credit"message is received. In case of seriously malformed requests (e.g., URI too long), no "debit" is sent to the Collector because the core process of the server directly invokes the standard module **mod_log_config** and then our **mod_musical_log**. Therefore, we must be sure that the Collector decreases the load counter only for requests whose "debit" message was received.

The protocol used to talk with the WebPlayers is simpler than the one used with the sonification module. After a successful connection, the Collector uploads to the WebPlayer the description of the available Channels and the information about what sound set is to be downloaded. The WebPlayer receives sound requests about the selected Channel from the Collector. If the user chooses a different Channel, then the WebPlayer asks to the Collector to receive data about the selected Channel.

The Collector provides data to the WebPlayer on the form of a batch of requests coded as bit strings. The size of the batch depends on the duration of the time slice used by the Collector. For each second in the time slice, the Collector batches as many bit strings as the number of available Channels. Because the WebPlayer supports up to 16 distinct audio-channels, each bit string holds exactly 16 bits. The string related to a Channel indicates what audio-channel must be played for that Channel. Inside a string, each bit indicates whether the corresponding track should play.

3.1.5. WebPlayer

When the Collector knows exactly how the next time slice of events is to be musically represented, it can then instruct the WebPlayer. The WebPlayer is coded in Java as an application and uses the Java Sound and Java Media Framework API to pilot a required audio device. WebPlayer functionality has been extended (with respect to [2]) to allow users to integrate an external audio source with sounds used to represent server behavior. The main idea is to mix the "server event music" with the user's preferred music, and to use volume and sound cues to inform the webmaster of the status of the web server.

At startup, the WebPlayer downloads several sets of sound files from the network using the information provided by the **mod_musical_log** through the Collector, and permits the user to create a playlist with audio files (also with different audio format) to be used as background music. Volume changes and sound cues are then played according to the command strings periodically received from the Collector.

In fact, the WebPlayer can run in two different modes: "Standard" and "Personal". The first mode offers the same functionalities provided by original WebPlayer (i.e., playing sets of sound files according to the information obtained from the Collector). The second mode includes "Standard" features, and adds the capability for the user to choose an existing playlist or to create a new one to set as background music, which will be played and dynamically modified according to the Collector information.

Several sound cues can be simultaneously played using a multiple *audio channel*[4] sound architecture. When the player module is run, a set of audio tracks is loaded from a remote web server into memory, and each track reserves for itself a unique audio channel. At any time, the user can create a new playlist choosing which audio files include in it. At the moment, WebPlayer can load and play MP3 or WAV files stored in the local file system as a background sound track. While running, the WebPlayer determines what audio channels are to be played according to the information received from the Collector. If the WebPlayer is running in "Personal" mode, audio channels associated with workload events will be muted and workload information will be translated into changes of background music volume. Higher gain level will denote a higher server load. Moreover, all the audio channels will play with a louder volume than the one used to play the background sound track.

The WebPlayer uses a multi-threaded architecture to allow to playback audio without being affected by I/O activities such as the ones required to deal with the remote Collector. The first thread manages the remote connection with the Collector. The second thread provides the user a graphical command console to switch between different Channels and handle playlists. The third thread plays the background sound tracks according to the content of the current playlist (if selected). The final thread plays the required sounds cues and modifies the volume of the audio output depending on the information received from the Collector. The interactions among these threads are assured by the use of synchronized methods. The WebPlayer uses an internal buffer of **StatisticsFrequency** seconds to allow for network delays. In this way, an event occurred at the Web server at time *t* is played at the WebPlayer side *StatisticsFrequency* seconds later.

In summary, we have described the Personal WebMelody architecture and implementation. The Apache module passes information to the Collector based on Channel definitions that specify what sort of web server events are interesting. The Collector aggregates these events and eventually maps them to specific sounds and music according to the Channel definitions. In the end, the Collector instructs the WebPlayer application to mix specific musical events with the personal music the web administrator has selected.

4. SONIFICATION STUDY

There are two potential problems with providing an administrator with sonic indicators of web server status: (a) sound, especially music, might be so distracting that the administrator's overall productivity drops; (b) specific sounds that indicate specific web server events might be difficult to pick out and attend to, especially if the administrator is engaged in other ongoing tasks. We set out to investigate whether either of these might in fact be problems in practice.

We view our sonified web server as providing information that is usually not critical to an administrator, who distribute attention among a host of tasks. More precisely, following Maglio and Campbell [19], we define *peripheral information*

[4] We use the term audio channel to denote the channels used by the audio device to play sounds, whereas we use the term *Channel* to mean the information channel that is provided (through the Collector) to the WebPlayer.

as information that is not central to a user's current task, but provides the user the opportunity to learn more, to do a better job, or to keep track of less important tasks. We view peripheral informing as imposing extra-task demands on a user's cognitive resources. In the context of the sonified web server, this means an administrator attends to a primary task, such as reading an online document or editing text, but occasionally something else, such as a flurry of activity on the server or a network outage, becomes important and is attended to.

The goal of our study was to explore the effectiveness of our sonified web server in a controlled setting. To determine whether the music played by the sonified server is distracting, whether it is informative, and whether there are tradeoffs between the two, we used a dual-task paradigm in which participants performed a primary text-editing task while at the same time monitoring music that simulated the sonified web server (roughly following the method described in [19]). We measured the degree to which music distracted performance in an editing task, and we measured memorability of musical events in a post-experiment test.

4.1. Methods

4.1.1. Participants

Twenty-eight science undergraduate students at the University of Salerno volunteered to participate. All were native Italian speakers and only one was a trained musician.

4.1.2. Design

We used a 3 (editing task/listening task/dual task) x 2 (presentation order) design. Presentation order varied between participants, and task varied within participants. Each participant edited text in two conditions, once with instructions to listen to the background music (Emphasis Music), and once with instructions not to listen to the music (Background Music). In addition, each participant also listened to music without editing (Music Only).

4.1.3. Materials

The materials consisted of three passages for text editing, three sound tracks (one for training, one for the Emphasis condition, and one for the Background condition), a music questionnaire (for testing the informativeness of sound tracks), and a final questionnaire.

Texts. For text editing, we chose three movie reviews of films by Massimo Troisi, a popular Italian director and actor. The texts were of similar length and of easy reading level. We introduced 51 errors into each, including triple letters instead of double letters, Italian gender agreement of articles problems, and subject-verb agreement problems.

Soundtracks. Our goal was to provide sounds that go beyond simple alarms. These were designed as a compromise between providing information and being distracting, while at the same time creating pleasant background music. All three soundtracks used improvisational jazz styles to insure that in all cases no melodic sequence would be familiar to the participants. In addition, jazz style is largely unpredictable, lending itself to representing the spontaneous and unpredictable nature of web server activity. Jazz-like chords have been used effectively to represent several coordinates to be monitored for parallel programs [10].

The rhythmic base of the piece served to represent increasing and decreasing load. A discrete drum beat represented minimum (i.e., zero) load and a heavy "Slap Bass" represented maximum load. Accordingly, intermediate levels were achieved by adding bass and drum tracks. Two solos were featured, one with "Electric Guitar" and the other with "Trumpet". These represented two opposed events, namely access by one of the most popular browsers, Netscape and Internet Explorer. Hand claps and piano chords represented severe errors and warning errors, respectively.

*Music Questionnai*re. The music questionnaire consisted of eleven multiple-choice questions, followed by a table summarizing the associations of sounds with events. For load, participants were asked to indicate the frequency (for example, how many times the load reached the maximum/minimum), the duration (i.e., number of short/long periods of maximum/minimum load), and the frequency of contemporaneous occurrences of maximum load and "severe errors". For opposite events, participants were asked about the dominance of one of the opposed events with respect to the other, and the frequency of alternation during the first or second half. For warning and severe errors, participants were asked to identify a range of occurrences and the frequency in the first and second half.

Final Questionnaire. To obtain information about reading and musical skills, we asked the participants to fill out an final questionnaire, rating their own reading skills and indicate the high school attended and the final grade obtained. To determine a musical profile of the participants, we also asked whether they usually studied or worked with music in the background, and whether they were musicians, what instruments they played, and so forth.

4.1.4. Procedure

One key manipulation was the instructions in the Emphasis condition versus the instructions in the Background condition. That is, these two conditions differed only in that in the Emphasis condition, participants were told that they had to pay attention to both the editing task and the musical events, as there would be a test on the musical events at the end, whereas in the Background condition, participants were told not to pay attention to the music. In both cases, participants were instructed to do as much the text editing as possible.

For the Music Only condition, participants were told to listen carefully to the music, as there would be a test on musical events at the end. Prior to the Music Only Condition, participants were given brief training (20 minutes) on the identification of musical events. During this training, they learned to associate sounds to proper names (e.g., "severe error") and also to recognize maximum and minimum load. Before the Music Only and Emphasis conditions, participants read the music questionnaire for one minute. During Music Only and Emphasis conditions, students were given a time mark in the middle of the period so that they could distinguish between the first and the second half.

Participants were split into two groups (A and B) of equal size to balance order of presentation. Each participant first practiced text editing for ten minutes without any music playing in the background. Participants in Group A then performed text editing with Background Music for ten minutes, followed by Music Only for ten minutes and the music test, followed by text editing in the Emphasized Music condition for ten minutes and the final music test..

After editing practice, Group B listened to the Music Only condition for ten minutes followed by the music test, then text editing in the Emphasis Music condition ten minutes and the music test, followed finally by text editing in the Background Music condition for ten minutes. The dependent measures were number of correct edits on the text-editing task and number of correct responses on the music tests.

4.2. Results

Unusable data resulting from technical problems or because a participant did not follow directions were discarded[5]. In the end, we were left with data for 18 participants, 8 in Group A and 10 in Group B.

Data were analyzed as follows. First, we performed a 2 x 2 analysis of variance (ANOVA) on the editing data, with the within participants factor Background vs. Emphasis and the between participants factor presentation order. No effect was obtained for either factor on editing performance; for Background vs. Emphasis, $F(1, 16) < 1$, and for Group, $F(1, 16) = 1.35$, $p = 0.26$. Overall, the mean number of edits in ten minutes was 26.1.

Second, we performed a 2 x 2 ANOVA on the results of the music test, with the within participants factor Music Only vs. Emphasis and the between participants factor presentation order. There was no effect of order, $F(1, 16) = 2.09$, $p = 0.17$, but there was main effect of experimental condition, $F(1, 16) = 5.79$, $p < 0.05$. The mean number of correct answers in the Music Only condition was 6.56, and in the Emphasis condition, 4.72. There was no interaction effect, $F(1, 16) < 1$.

Finally, comparison of the information from the questionnaire and the results did not reveal any notable variation in terms of reading or musical skill. For instance, the one skilled musician appeared to perform equally as well as the other participants.

4.3. Discussion

Overall, the results suggest that background music does not distract users from their primary task, and at the same time can effectively convey information. In particular, we found no decrease in amount of text editing (primary task) when participants were explicitly instructed to pay attention to the music compared to when participants were explicitly instructed not to pay attention to the music. Nevertheless, we also found a small (28%) decrease in the amount of information our participants extracted from the music in the dual-task as compared to the single-task conditions. Of course, it is not surprising to find that performance suffers when people performed two tasks compared to when they performed only one task. It is a little surprising, however, to find that the primary task (text editing) did not suffer at all, and that the secondary task (server event identification) suffered only a little.

Though we followed the same basic paradigm as Maglio and Campbell [19], our results differ dramatically from theirs. Using text-editing along with a visual information monitoring task, Maglio and Campbell found a decrease in text-editing performance under dual-task as opposed to single-task conditions, and at the same time, found no effect on memorability (or informativeness) of visually monitored information. The present study obtained the opposite results: namely, no decrease in text-editing performance under dual-task as opposed to single-task conditions, but a decrease in memorability of auditorially monitored information. One major difference, of course, between the methods used in the two studies is that ours employed two different modalities (visual plus auditory) whereas theirs employed a single modality for the two tasks (visual). In addition, Maglio and Campbell's "peripheral" task was a reading task, a skill that all participants have presumably mastered. In the present study, the peripheral task was musical event identification, which is not likely to be a skill that any of our participants have mastered. (Recall that only one participant in the current study reported being a musician.)

Other studies have found evidence that intermodal task-sharing (i.e., tasks that mix different modes of interaction, such as visual and audio information) is more successful than intramodal ones (i.e., when only vision is involved) [6]. In fact, a multiple resource theory of attention (see, for instance, [24]) explains why this might be the case. Because different modalities share fewer common operations and representations, it is reasonable to suppose that processing in different modalities can occur simultaneously without interfering with one another.

5. CONCLUSION AND FUTURE WORK

We described Personal WebMelody, a system for the real-time monitoring of a web server by sonification.

Our sonification technique works by associating MIDI or WAV sound tracks to events describing the behavior of the web server. The experiments we have conducted have shown that this system allows an efficient and non obtrusive monitoring. For these reasons, it can be considered an ideal complement to standard, off-line techniques of monitoring and tuning of web servers. In order to face the problems that can arise while using such a system for a long period of time like, for example, a mental fatigue in the user, we have proposed a solution that allows a user to listen a preferred music in background while using our system. In this way, we convey information to the user changing the volume of the personal music source and by mixing in additional sounds.

It remains to be proved that our sonification technique is still effective for a long period of time, even if used with a personal background musical source. In this sense, we are planning further experiments to investigate the behavior of our system when used for a long time.

Besides the possible use of this system as a monitoring tool, the technique we have used can have interesting applications with respect to the accessibility issue: the sonification allows sight-impaired persons to effectively be informed of the behavior of the server or any other computer program.

From a technical point of view, the system we have developed is able to deliver the sonification of an Apache web server through the use of portable Java applications. Multiple users can be connected at the same time while hearing different representations of a same web server. In order to improve the accessibility of our system, we are considering the possibility to encode as standard sound streams the output of the sonification and then publishing them using the HTTP protocol. In this way, a user can access the sonification of a

[5] In those cases, the participants did not save the modified questionnaires on the computer and therefore we did not find usable data.

web server by just pointing its browser to the URL published by our system. Moreover, we plan to extend our architecture to support multiple sources of sonification events. This features can be useful when monitoring the activity of multiple instances of a same service (e.g. cooperative web servers) or when monitoring different services (FTP servers, large DMBSs) as well as networks. Finally, we are studying an extension of audio monitoring with graphical output based on abstract shapes and colors whose variations over the time provide synthetic information about the web server.

Acknowledgments. We thank our participants for their enthusiastic help in completing the study. We also thank Chris Campbell, Andreas Dieberger, and Shumin Zhai for helpful comments on a draft of this paper.

6. REFERENCES

[1] Apache Software Foundation (2000). Apache Web Server. Available at http://www.apache.org.

[2] Barra, M, et al. (2000) "WebMelody: Sonification of Web servers." In *Poster Proceedings of the Ninth Annual World Wide Web Conference* (WWW9). Amsterdam (Holland).

[3] Barra, M, et al. (2000) "MMM: Multi-Modal Monitoring of Web Servers". Submitted for publication.

[4] Blattner M., Greenberg, R.M., and Kamegai, M., (1992), "Listening to turbulence: An example of scientific audiolisation," in *Multimedia Interface Design*, (Blattner, M., & Dannenberg, R.M., eds.) , Chapter 6, pp. 87 - 102, Wokingham: ACM Press

[5] Blattner, M. M., Sumikawa, D. A., & Greenberg, R. M. (1989). "Earcons and icons: Their structure and common design principles". *Human-Computer Interaction*, 4, 11-44.

[6] Brown, M.L., Newsome, S.L. and Glinert, E.P. (1989). "An experiment into the use of auditory cues to reduce visual workload." In *CHI '89 Proceedings*, pp. 339-346. New York, NY, USA: ACM.

[7] Buxton, B. (1989). Introduction to this special issue on nonspeech audio. Human-Computer Interaction, 4, 1-9.

[8] Cohen, J. (1994). "Monitoring background activities." In G. Kramer (Ed.), *Auditory display: Sonification, audification, and auditory interfaces.* Reading, MA: Addison-Wesley.

[9] Deutsch, D. (1986). "Auditory pattern recognition." In K. R. Boff, L. Kaufman, & J. P. Thomas (Eds.), *Handbook of human perception and performance* (Vol II, pp. 32.1 – 32.49). New York: Wiley.

[10] Francioni, J. F., L. Albright, and J. A. Jackson. (1991) "Debugging parallel programs Using sound", in *Proceedings of the ACM/ONR Workshop on Parallel and Distributed Debugging,* 68--73. Reading, MA: ACM Press/Addison-Wesley, 1991.

[11] Gaver, W. (1986). "Auditory icons: Using sounds in computer interfaces." *Human-Computer Interaction,* 2, 167-177.

[12] Bill Gaver (1991). "Effective Sounds in Complex Systems: The ARKola Simulation", In *CHI '91 Proceedings* ACM.

[13] Hallam-Baker P.,Behlendorf B.. "Extended Log File Format." World Wide Web Consortium Working Draft 960323. http://www.w3.org/pub/WWW/TR/WD-logfile.html.

[14] Katz, E. D., Butler, M., and McGrath, R. (May 1994). "A Scalable HTTP Server: The NCSA Prototype", in *Proceedings of the First International WWW Conference.*

[15] Lemmens, P. M. C., Bussemakers, M. P., & de Haan, A. (2000). "The effect of earcons on reaction times and error-rates in a dual-task vs. single-task experiment." In *Proceedings of International Conference on Auditory Displays* 2000.

[16] Lohda, S. K., Whitmore, D., Hansen, M, & Charp, E. (2000). "Analysis and user evaluation of a musical-visual system: Does music make any difference?" in *Proceedings of International Conference on Auditory Displays* 2000.

[17] LoevstrandL. (1991). "Being selectively aware with the Khronika System". In *Proc. of 2nd European Conf. on Computer-Supported Cooperative Work* (ECSCW'91) Sept. 24-27 Amsterdam, NL. Kluwer Academic Publishers, pp.265-278.

[18] Luotonen A. "The Common Log file Format." http://www.w3.org/pub/WWW/Daemon/User/Config/Logging.html.

[19] Maglio, P. P. & Campbell, C. S. (2000) "Tradeoffs in the display of peripheral information", in *Proceedings of the Conference on Human Factors in Computing Systems* (CHI 2000).

[20] McCrickard, D. S. (2000). "Maintaining information awareness in a dynamic environment: Assessing animation as a communication mechanism". Doctoral dissertation. Georgia Institute of Technology.

[21] Netcraft (August, 2000). Netcraft web server survey. Available as http://www.netcraft.com/survey.

[22] Pool, M. M., van der Voort, T.H.A., Beentjes, J. W. J., Koolstra, C. M. (2000). "Background television as an inhibitor of performance on easy and difficult homework assignments." Communication Research, 27, 293-326.

[23] Scullin Will H., Kwan Thomas T., & Reed Daniel A. (1995). "Real-time Visualization of World Wide Web Traffic," in *Proceedings of Symposium on Visualizing Time-Varying Data*.

[24] Vickers P., Alty, J.L.. "Musical Program Auralisation: Empirical Studies". In *Proc. of 6th International Conf. on auditory Display.*P.R. Cook, Ed. Atlanta GA: International Community for Auditory Display, April 2000, pp.157-166. ISBN 0-9670904-1-5.

[25] Wickens, C. D. (1992). "Engineering psychology and human performance "(2nd Edition). New York, NY: Harper Collins.

[26] Young, H.H., & Berry, G. L. (1979) "The impact of environment on the productivity attitudes of intellectually challenged office workers." *Human Factors*, *21*, 399-407