# Personal DJ, an Architecture for Personalised Content Delivery

Adam Field, Pieter Hartel * Wim Mooij †

## ABSTRACT

Automated Personalised Audio is a relatively new concept, currently making its debut on the Web. Personalised audio relies on the existence of information about the music (music metadata) and information about the users (listener profiles). By gathering profile information, personalised audio systems attempt to select appropriate content for each user. This paper introduces the *Personal DJ* architecture for personalised audio. An evaluation of the concept is presented on the basis of data gathered from user tests. These tests were performed with a prototype developed from this architecture using simple mood based music metadata. [1]

**Keywords:** Personalisation, Radio, Moods, User Evaluation

## 1. INTRODUCTION

Music delivery systems fall into two broad categories, content purchasing and audio broadcasting. In the case of content purchasing, the consumer pays for specific music (e.g. CDs, cassette tapes, LPs) to build up a collection of personal favorites over which he/she has complete control. Broadcast audio (e.g. radio, TV, Internet radio) provides more content, but at the cost of limited consumer control (consumers choose radio stations, not music). Personalised audio attempts to bridge the gap between the two. Our hypothesis is that by matching information about the users (listener profiles) with information about the music (content metadata) it should be possible to automatically generate good playlists for individual listeners. In this paper we test this hypothesis.

The Smart Radio Project has looked at moving Personalised Audio into the consumer electronics sector. The idea is to provide radio appliances (both portable and non-portable) that would download content from varied and distributed sources (digital radio, Internet radio, remote storage, CDs, etc.). This appliance would create a playlist of music, programming and advertising based on the preferences of the listener. The *Personal DJ* architecture has been developed as a part of the Smart Radio Project.

The present paper discusses our architecture, and an experiment with an implementation of the architecture. The architecture was designed with a heterogenous, chaotic, distributed network (such as the Web) in mind. The current implementation, in Java, is well suited for further development into a Web application, and future work will be looking into running an experiment over the WWW.

## 2. RELATED WORK

In June 2000, *Newsweek* published an article titled "Hitting the Right Notes" [15] which contained an overview of Internet based music recommender systems. The overall tone of the article suggested that the author did not think much of the performances of the systems specifically mentioned (`Listen.com`, `Mubu.com`, `Moodlogic.com` and `Launch.com`), even though each site approaches the problem with different methods (e.g. collaborative filtering [1], categorization by professional DJs, digital signal analysis [11]). The author suggested that these recommender systems will remain a novelty until they can successfully "refer users to music that they'll like".

Stone's opinion agrees with a Web survey [4] we are currently performing. We found that the technology involved in automatic selection is immature but promising.

Work in personalisation on the Internet seems to be based mainly around text retrieval. Personalised retrieval systems, such as the Krakatoa Chronicle [5], a personalised newspaper, analyse user behavior to keep the profiles up to date [13]. Many of these systems operate on a combination of collaborative and content-based filtering [1, 3].

Recent developments in personalisation and automatic categorisation of multimedia include a system for the construction of personalised TV news programs [8] based on manual categorization and automatic keyword extraction. Systems also exist that attempt to recognise music based on audio input [2].

MIT has been developing an audio wearable computer [14], which has some interesting time based personalisations, interacting with the user and delivering messages, emails, etc. at appropriate times.

Customized Internet Radio(CIR) [6] is an application that schedules content retrieval from multiple Web radio stations based on a user configurable schedule. Based purely on streaming stations, the scheduling seems to be based on simple time slots (e.g. play BBC from 9am until 10am, then

*Dept. of Electronics and Computer Science, University of Southampton, UK. Current affiliation Dept. of Computer Science, University of Twente, The Netherlands. email : {field,pieter}@cs.utwente.nl

†Irdeto Access BV, Hoofddorp, The Netherlands, email WMooij@irdetoaccess.com

[1]This work was supported by Irdeto Access BV, The Netherlands


*WWW10*, May 1-5, 2001, Hong Kong.
ACM 1-58113-348-0/01/0005.

MP3.com until midday), with some provision for station unobtainability. Personalised on a 'large grain' set of content (whole programs rather than single songs).

The *Personal DJ* looks at single pieces of content (single songs and adverts), however there is no reason that the architecture and profiling could not deal with this 'large grain' data.

## 3. SYSTEM OVERVIEW

The *Personal DJ* resembles a real DJ performing the tasks of finding and playing the appropriate music tracks for the intended audience. A consumer informs the *Personal DJ* of the type of music it needs to generate, including a genre mix, a mood mix and environmental details. The *Personal DJ* will then generate a music stream matching the consumer's preferences. It is then possible for the consumer to evolve the selection by giving feedback to the *Personal DJ*, which will adjust the music preference profile and the corresponding playlist schedule. This adaptive behavior replaces the detailed and time consuming process of preference specification.

The key innovation of *Personal DJ* is that it can make selections from both local and distributed music archives (e.g. CD's, DVD's, MP3 files, MiniDiscs, Internet resources) or make automatic transitions between streamed music instances (e.g. FM radio, Web radio) or any combination of these. This makes it an enabling technology for a wide music delivery platform.

The *Personal DJ* system architecture assumes a plurality of music delivery infrastructures accessible to a music player device. A diagram of a music player connected to multiple music delivery infrastructures is given in Figure 1.

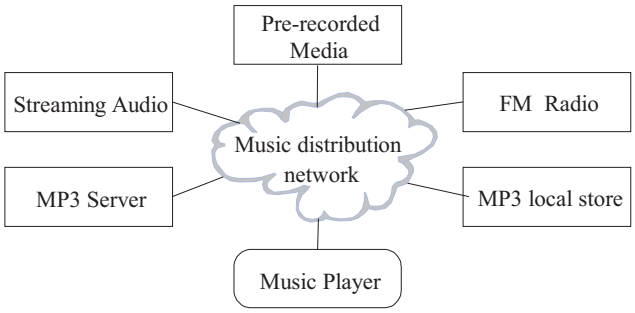


**Figure 1: Music Delivery Infrastructure**

The aim of the *Personal DJ* is to make optimal choices between content from various sources including network servers, local disk, and radio broadcast. Both the user and the content provider(s) control the scheduling objectives directly or indirectly, and the relevant business rules (e.g. Digital Millennium Copyright Act [10]) are maintained throughout.

## 4. ARCHITECTURE

The *Personal DJ* operates on an enhanced audio format that consists of digital music, and metadata (information about the music). The metadata could include information such as artist, song title, genre, mood, timing information and IPMP (Intellectual Property Management and Protection) usage rules. The *Personal DJ* processes the components of the audio format to automate the music selection, scheduling, fetching and play-out. It has some internal databases to control this process.

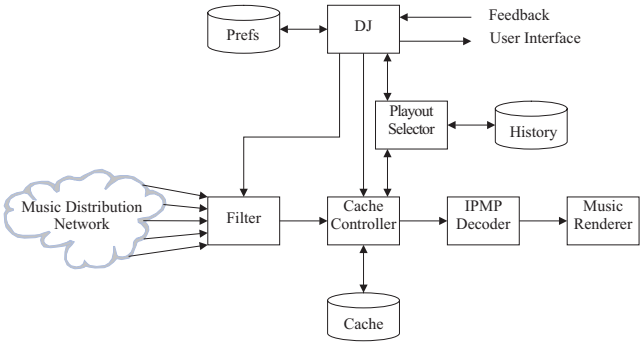


**Figure 2: Diagram of System Architecture**

Figure 2 shows the system architecture. The following is a brief explanation of the main data flows in the system:

- Content – Flowing horizontally from the Music Distribution Network, a piece of music will flow through the Filter, Cache Controller, IPMP Decoder finally being converted into sound by the Music Renderer.
- Metadata – Entering the system from the Music Distribution Network, relevant Content Metadata is stored in the Cache Controller, and made available to the Playout selector for examination. When a piece of music is selected, the associated metadata is passed up to the DJ for User Interface control.
- User Profile – Stored in the Prefs database, the user profile is broken into three sets of criteria based on the volatility of the data. These criteria (Figure 3) are used by the Filter, the Cache Controller and the Playout Selector to assist with different stages of the filtering process.

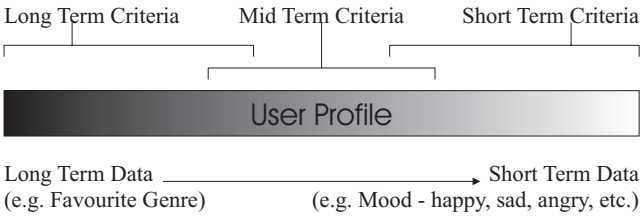


**Figure 3: User Profile and Criteria**

A brief description of the modules of the system follows.

- The Filter Module – The Filter acts as a network interface, informing the cache controller of relevant available content and delivering the content on request.
- The Cache Controller Module – The Cache Controller is responsible for managing the Cache and ensuring that a good variety of music is always available to the Playout Selector.

- The Playout Selector Module – This module builds the playlist by selecting music from the Cache based on information that could potentially be continuously changing (e.g. mood of listener, IPMP conflicts, play order suitability).

- The DJ Module – This module acts as an overall manager of the system. Processing user feedback, it maintains a user preference database with which it provides data to the filter, playout and cache controller modules to enable decisions to be made about music to store and play. It also controls the user interface, showing relevant visual output for each piece of content.

- The IPMP Decoder Module – If any audio needs decoding due to IPMP conditions of use, the IPMP decoder ensures that this is done. Note that the decoding of audio is performed as late as possible, ensuring that content is encrypted and secure for as long as possible.

- The Music Renderer Module – The audio output of the system. Converts the digital stream or audio file (e.g. RealAudio or MP3) into actual sound that can be heard by the listener. Also maintains all play-out settings (e.g. volume, treble, bass, etc.).

## 4.1 Functional Overview

To give a better understanding of the system, here follows three descriptions of the system performing different tasks.

- Normal Play – The Filter is continuously downloading and evaluating metadata. Any item of metadata that matches the long term criteria is passed to the Cache Controller. The Cache Controller evaluates it based on the mid term criteria. Depending on the current state of the available music (cached files, available streams and remote files) the cache controller will make a decision about replacement (it is likely that an item of content will be removed from the cache once played). The Playout Selector makes a play choice from the available music, taking into account any scheduling requirements that may exist, the IPMP data, the history and the short term criteria. This selection is sent to the IMPM decoder and playout modules for rendering.

- Feedback Processing – On receipt of feedback, the DJ performs an analysis of the history and feedback then modifies the user preference appropriately. Any updates are then filtered down the system through the use of the criteria. If the system is not performing optimally in the long term, the Long Term Criteria and Mid Term Criteria would be adjusted. For short term changes, the Short Term Criteria and Mid Term Criteria would be modified.

- Content Handling – The Cache Controller holds metadata for items on the cache, streams about to become available and reliably downloadable files. The data held would have to include a reliability rating, and/or retrieval time, which would be taken into account by the Playout Selector and Cache Controller. In the case of a desirable item of music with a low reliability rating, it would be requested and cached. Once on the cache, it would be the same item of music with a perfect reliability rating. However, a long download time would lead to the possibility that the music is not a good match by the time it is available. The Cache Controller would then have to make a decision regarding the probability of it becoming playable in the future. If algorithmically possible, some kind of look-ahead content predictor could be employed at this level to assist.

- Caching – Note that the size of the cache determines the scope of the long, short and cache criteria – in the extreme, with no cache, the filter is given all criteria, and only streamed audio can be handled. In an ideal system with a good cache, the cache criteria would store the 'mid term' criteria, taking the shorter term data from the long term set and the longer term data from the short term set. This would enable the cache to store specific content that fits the current mood, but also enough general content to be able to handle a change in the short term criteria.

# 5. PROTOTYPE

To evaluate the *Personal DJ* concept we have implemented a prototype based on a simplified architecture using music metadata but no user profiling.

## 5.1 Metadata

We created a simple metadata schema to represent our music based on moods. Each song has a vector containing 3 'moods'. Angry, Chill and Upbeat were chosen as they represent between them a large amount of music, and are easily understood. Each is represented by an integer in the range 0 to 3. These can be thought about in two different ways (as the user or as the music):

- When I feel [Angry/Chilled/Upbeat] I [would/would not] listen to this music.

- I think this music [is/is not] [Angry/Chill/Upbeat] in itself.

The 0 to 3 scale was designed with the following in mind:

0. Not at all suitable to this mood
1. Not particularly suitable, but acceptable if no other option
2. Suitable
3. Highly Suitable / Perfect

Each song has a rating for each mood, e.g. "Hand In My Pocket", by Alanis Morissette was rated at 2 in Upbeat, 0 in Chill and 1 in Angry (see Figure 5). These ratings are based purely on our opinion of the music. A fielded system would require professional or panel ratings instead. The scale and the number of moods were kept as simple as possible to avoid an overly complex system. A 3 mood, 4 level compromise between complexity and functionality was thought to be sufficient as the basis for a working prototype.

## 5.2 Prototype Architecture

Figure 4 shows the architecture used when constructing the prototype. The differences from the *Personal DJ* architecture (see Figure 2) resulting from simplifications are outlined below:

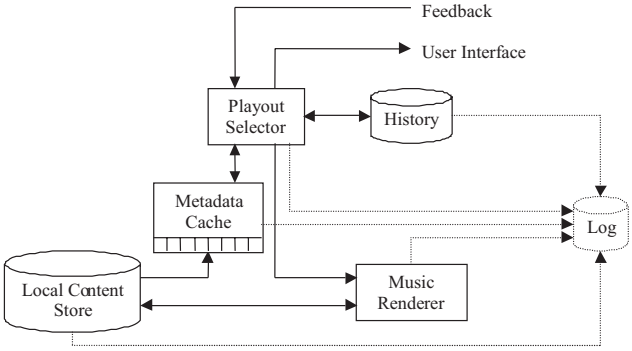


**Figure 4: Prototype Functional Architecture**

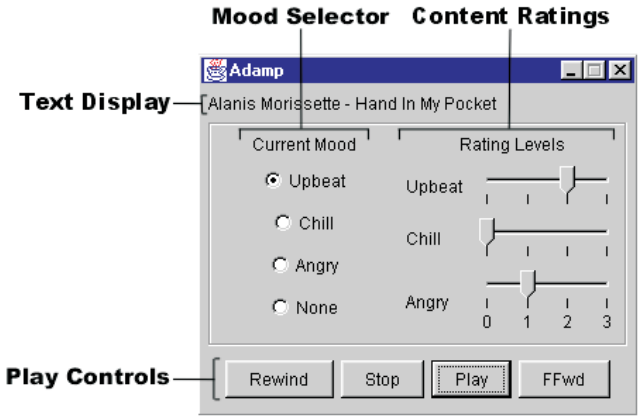


**Figure 5: Prototype User Interface**

- The functionality of the DJ module has been moved into the Playout Selector module as the metadata is simple and the task of matching preference (selected mood) to metadata (content mood levels) trivial.
- The Music Distribution Network is implemented by the Local Content Store, a directory of music and metadata on the hard disk of one of our machines.
- No content Cache is required as the music is stored locally.
- The Metadata Cache provides a small set of metadata for the Playout Selector to process. This module implements both the Filter and the Cache Controller.
- The Playout Selector informs Music Renderer of choice, and Music Renderer fetches content from Local Content Store.
- There is no requirement for an IPMP Decoder as the prototype is a closed system.

To track the behavior of the system, explicit logging of all noteworthy events (e.g. choices of songs, user feedback, errors, etc.) was added.

## 5.3 User Interface

The user interface was designed to be simple, functional and intuitive. Figure 5 shows the design.

The functionality of components of the interface are detailed below:

- Text Display – Displays information about the current song.
- Play Controls – Allows the user to control the playout.
- Mood Selector – Sets the current mood for the choice of content.
- Content Ratings – Shows the levels of the current song, and allows them to be changed.

## 5.4 Prototype Behaviour and Selection Algorithm

During normal play (see Section 4.1), as soon as a song ends, the Metadata Cache fetches and caches 30 songs from the Local Content Store. If a song with a rating of 0 in the current mood is fetched, it is rejected, and another song is fetched (this is repeated until a song with a higher rating is found, or the fifth song is picked with a 0 rating). A check is also made to ensure no song is held twice in the cache. The size of the Metadata Cache is dependent on the amount and quality of the content store. The more appropriate music there is, the smaller the sample must be. A size of 30 was arrived at by 'tuning' the software experimentally until good play was achieved.

Once the cache is filled, a song is chosen based on the rating of the song in the current mood, and the place in the history. A song with a high rating that was played 75 songs ago would take precedence over a song with a low rating that hadn't been played. The Playout Selector then sends the path of the content to the Music Renderer and the name, artist and ratings to the User Interface.

## 5.5 Implementation

We implemented the prototype in Java using the Java Media Framework API (`java.sun.com/products/java-media/jmf/index.html`) which provides classes for playing MP3 files. Swing (`java.sun.com/products/jfc/tsc`) was used for the user interface and we used the joeshmoe mpegjava classes (`www.joeshmoe.com/mpegjava/`) to read the ID3 tags (`www.id3.org`) of the MP3 files we used. These classes and APIs were all freely downloadable and well documented.

The prototype ran smoothly on a 700 MHz Pentium III computer with no breakup in the playback when performing tasks such as opening and running a browser. Playback breakup did occur when the software was run on a 150 MHz Pentium II if browsing a computationally intensive Web site (e.g. site with Java programs or Macromedia Flash graphics). Playback quality, and system capabilities was of major concern when designing the user evaluation.

# 6. USER EVALUATION

An assessment was performed on the prototype for the following reasons:

a. To evaluate mood based content metadata

b. To study the feasibility of preference generated playlists

c. To gain an understanding for the way people listen to music.

d. To evaluate the architecture and software implementation.

We gathered a group of 20 subjects and, under controlled conditions, observed them using the prototype.

## 6.1 Subject Demographics

All of our test subjects were between the ages of 15 to 35 for the following reasons:

- there are a large number of people in this age group in and around a university;
- people of a similar age seem to have some common musical tastes. It would have been difficult to provide enough music for a group with disparate musical taste without the task taking an unreasonable amount of time.

## 6.2 Content Preparation

To produce a pleasant mix that would appeal to the majority of subjects the content chosen was specifically picked to be non-offensive and relatively easy to listen to. We gathered a selection of 750 'middle of the road' pop songs likely to be acceptable for the group of subjects. We then evaluated and graded them in terms of the three metadata mood criteria. While there were many that scored highly in one or more moods, there were a small number that were assigned 0 or 1 in all three moods, making them unlikely to be picked; however, simple performance evaluation showed that this number provided acceptable listening conditions in all moods over an extended period.

## 6.3 Rating Mechanism

To get feedback from the user, a small extension was added to the prototype specifically for the evaluation. At 10 minute intervals, after the current song has finished, the window shown in Figure 6 will pop up (with an audio prompt) onto the screen. The rating slider can be dragged to a value and the OK button records that value in the log. If ignored, the rating box simply waits to be noticed, issuing audio prompts every 10 minutes.

The ratings implement one of two user feedback methods. This mechanical feedback method runs during the test, to constantly collect evaluations from the user.

# 7. EVALUATION PROCEDURE

The following describes many of the primary considerations and factors that were taken into account when designing the evaluation.

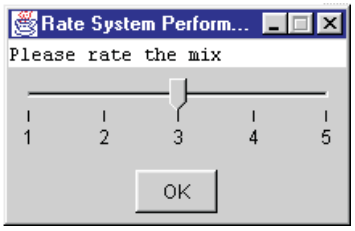


**Figure 6: Prototype Rating Window**

## 7.1 Environment

To control the environment as much as possible, the test was run in an office with the door shut and a 'do not disturb' sign on the door. Only the subject and the observer were present for the duration of the test. To vary the test as little as possible, the same room and the same observer were used for all tests. We decided that listening to music should be treated as a 'background' task, as anecdotal evidence suggested that most people listen to the radio while doing something else in the 'foreground' (e.g. driving, eating breakfast, jogging). We chose Web surfing as the 'foreground' task as it's simple, entertaining, intellectually light, easy to observe and easy to control.

## 7.2 Duration

The time chosen for the test was 90 minutes; it was thought that this would give a reasonable number of ratings (from 7 to 9) without taking an unreasonable amount of the subject's time.

## 7.3 Subject Instructions

The users were given the following instructions:

- To treat the prototype as a radio – As the subjects may or may not have been aware of Internet radio or computer audio, this metaphor was given to help the user understand the main function of the software.
- To treat the mood selections as different stations – Another analogy used to explain the function of the moods. The subject was encouraged to change moods as often as they wanted to.
- To rate the music as and when the rating box appeared – A point was made of informing the user that the rating should reflect how they felt the music had been overall since last rated. The rating levels were available to the subject at all times in the form of a large-print sheet placed next to the computer.
- To use the Fast Forward button if they didn't like the song – Simply to keep the user happy and relaxed, and avoid forcing them to listen to hated music.
- To surf the Web – The chosen 'foreground' task. The subject was asked to avoid Web sites with musical content as this would affect the test.
- To stop or pause the music if they left the room for any reason.

## 7.4 Rating Levels

The following rating levels were presented to the user:

1. Very Bad,
2. Bad,
3. Neutral,
4. Good,
5. Very Good.

This kind of scale seems to be standard in questionnaires dealing with measurements of opinions [7], and was felt to be adequate in this case.

## 7.5 Post-Test Questionnaire

After listening to the music produced by the prototype for 90 minutes, the user was asked four open-ended questions to attempt to get more detailed feedback than was available purely from the log.

- How much were you aware of the music?
- Overall what did you think of the music?
- How much attention was devoted to the Web?
- What did you think of the user interface?

The user was then given the opportunity to make any comments they wished to.

This questionnaire gathered a second set of user feedback, to help confirm the results of the rating boxes. Designed to be as open-ended as possible, the questions were intended to gather as much information as the subject wanted to convey about their behavior while performing the test.

This structured interview, coupled with the periodical Likert scale feedback and the logging of events (see Section 6.3), was designed to gather a balanced set of data [12] [9, Chapter 7].

# 8. USER RESULTS

After performing the experiment on twenty subjects, a number of trends emerged. Firstly we will look at the behavior of the user as captured by the rating and logging mechanism of the prototype:

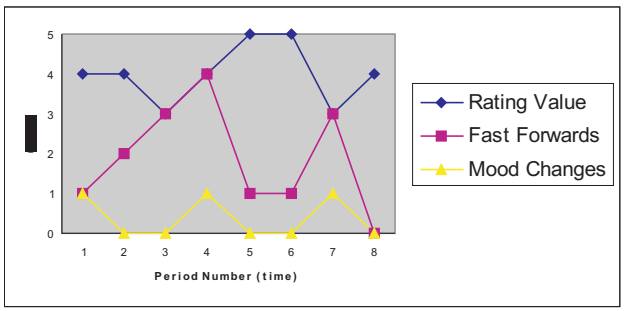


**Figure 7: Chart of captured user data**

## 8.1 Captured Data

Figure 7 shows user input over the 90 minutes of the test, taken from actual test data of a subject, chosen as it illustrates several points nicely. The graph shows the value of each rating as a function of time. Each rating has two associated values; the number of fast forwards and the number of mood changes generated by the subject during the rating period. The following can be seen from the graph:

- Rating Levels – There is a relationship between the number of fast forwards and the level of the rating. As a general rule the rating levels are high when the number of fast forwards is low (and vice versa). This trend is apparent for 11 out of 20 subjects (with little or no significant trends appearing in the remaining 9 subjects).

- Mood Changes – A relationship between the number of fast forwards and the number of mood changes also seems to be apparent. Figure 7 shows that the user set the moods three times (once at the start, and then two changes at periods 4 and 7). The two mood changes occurred during periods which also contained high numbers of fast forwards. This trend is apparent for 12 users, again without significant 'disagreement' from remaining 8 subjects.

The fact that there is an anti-correlation between rating levels and the number of fast forwards could lead to the conclusion that the more unhappy a listener is with the music, the more songs he/she will skip. With this in mind, we may be able to use the number of fast forwards as an indication of overall unhappiness (rather than a specific rating level), although this assumes that fast forwarding is as convenient to the user as it is with the prototype.

A second, interesting fact, is that the majority of users preferred to fast forward rather than change moods. In this case (Figure 7), the mood was set 3 times, and there were 15 fast forwards. A ratio of 1:5 is not unrepresentative of the other users. Changing moods, being somewhat analogous to changing stations on a radio, in the prototype is simple, but users still preferred to fast forward past songs they didn't like, rather than change moods.

## 8.2 User Feedback

After sitting the test, each user was asked 4 open ended questions (see Section 7.5) to add to the data gathered by the prototype logging and the Likert ratings. The answers often overlapped into the domains of the other questions. In these cases we have included these responses in the comments for the relevant questions. The following is a selection of the most relevant answers given:

The question that provided the most useful information was the first (user awareness of music). Those that claimed the music 'faded into the background' (6 subjects) also commented that when music they didn't like came up, they were immediately aware of the music (e.g. "If I don't like a song, I notice immediately" and "Only [aware] when I didn't like the track"). The 14 users that claimed they were "Always aware [the music] was there", 4 also made this comment. The third question was originally intended to support the answer to this (people aware of the music wouldn't be concentrating too much on the Web), but it ended up providing

its own useful responses, as well as corroborating question 1 in all but 1 case.

The second and third questions gave some information on the way the people were listening to music. To question 2 many people gave relatively 'lukewarm' responses (e.g. "Some of it wasn't my style but I enjoyed most of it" or "Well, very mainstream. I like more diversity actually") but three subjects made the point that when they got music they didn't like they fast forwarded.

In the fourth question (user interface) a few people commented that it was like "Listening to a CD player and fast forwarding to songs you like". This may confirm that fast forwarding is an indication of unhappiness. Almost everyone commented that the interface was simple, functional and basic ("Bit boring really", "Fairly OK, fairly standard", "Just a normal interface, isn't it"). Although 4 people commented that there should have been more moods.

Overall the comments made led to the conclusion that the subjects were comfortable with both the concept and the interface (the only parts of the system visible) of the prototype (see Section 6 items b and d).

## 9. CONCLUSION AND FUTURE WORK

The analysis of the observational and mechanical data (see Section 8) could lead to three 'rules of thumb', each one confirmed by at least 50% of the subjects. Together these give us some idea of how people listen to music (see Section 6 item c).

i. When people hear music they do not like, they take more notice (See Section 8.2).

ii. People do not skip music they enjoy (See Section 8.1).

iii. People don't change moods with music they enjoy (See Section 8.1).

When people hear music they do not like, their initial reaction is to fast forward, followed by changing moods if they do not hear acceptable music within a reasonable number of fast forwards (e.g. 4 ffwds). We believe that people appreciate having these two levels of choice. This makes our selection mechanism different from radio, where changing the station is the only option.

The simple content mood metadata used in the prototype seemed to be sufficient in this test. Given that we were working with a small group of similar people, and the music was selected to fit this group, no definitive conclusions can be reached regarding automated personalised audio. However, the fact that most subjects gave high ratings consistently (only 3 subjects gave an overall mean rating of 3 or less) could suggest that personalisation is possible given carefully prepared content. Further testing would need to be conducted on a more sophisticated system to discover if larger vectors of content metadata could support sensible content choices in an environment supporting a wider variety of musical tastes.

The functional success of the prototype, although a simplified form of the architecture, seems to indicate its suitability for personalised audio (see Section 6 item d). A more sophisticated prototype will need to be constructed to test the architecture at a more detailed level.

As far as we know, all Internet based radio systems use genre based categories as opposed to mood based. We believe we have shown that mood based categorization is a viable alternative. This method could improve listener profile personalisation based on an analysis of the music and user behavior. The functional success of the prototype, although with a simplified form of the architecture, indicates its suitability for personalised audio. A more sophisticated prototype will need to be constructed to test the architecture at a more detailed level.


## Acknowledgement

We thank all of our volunteers for their support. The advice of Dave DeRoure, Hugh Glaser, and Stevan Harnad has been appreciated greatly.


## 10. REFERENCES


[1] M. Balabanovic and Y. Shoham. Fab: content-based, collaborative recommendation. *CACM*, 40(3):66–72, 1997.

[2] S. G. Blackburn and D. C. DeRoure. A tool for content based navigation of music. In *6th Int. Conf. on Multimedia*, pages 361–368, Bristol, UK, Sep 1998. ACM Press, New York.

[3] W. W. Cohen and W. Fan. Web-collaborative filtering: Recommending music by crawling the web. In *9th Int. World Wide Web Conference*, page Paper266. ACM New York, May 2000. `www.www9.org`.

[4] A. N. Field, P. H. Hartel, and W. Mooij. A review of internet audio. Declarative Systems & Software Engineering Technical Reports DSSE-TR-2001-01, University of Southampton, 2001.

[5] T. Kamba, K. Bharat, and M. C. Albers. The krakatoa chronicle - an interactive, personalized newspaper on the web. Technical Report 95-25, Graphics, Visualisation and Usability Center, Georgia Institute of Technology, USA, 1995.

[6] V. Krishnan and S. G. Chang. Customized internet radio. In *9th Int. World Wide Web Conference*, page Paper353. ACM New York, May 2000. `www.www9.org`.

[7] R. Likert. A technique for the measurements of attitudes. *Archives of Psychology*, 140, Jun 1932.

[8] B. Merialdo, K. T. Lee, D. Luparello, and J. Roudaire. Automatic construction of personalized TV news programs. In *7th Int. Multimedia Conf.*, pages 323–331, Orlando, Florida, Oct 1999. ACM Press, New York.

[9] J. Nielsen. *Usability Engineering*. Academic Press, London, 1993.

[10] U. S. Copyright Office. *Digital Millenium Copyright Act*. 1998.

[11] S. Pfeiffer, S. Fischer, and W. Effelsberg. Automatic audio content analysis. In *4th Int. Multimedia Conf.*, pages 21–30, Boston, Massachusetts, Oct 1996. ACM Press, New York.

[12] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey (eds.). *Human-Computer Interaction*. Addison Wesley Longman, 1994.

[13] H. Sakagami and T. Kamba. Learning personal preferences on online newspaper articles from user behaviors. In *6th Int. World Wide Web Conference*, pages Paper142 – TEC155. Apr 1997. `http://decweb.ethz.ch/ WWW6/`.

[14] N. Sawhney and C. Schmandt. Nomadic radio: Scaleable and contextual notification for wearable audio messaging. In *SIGCHI Conference on Human Factors in Computing Systems*, Pittsburgh, Pennsylvania, May 1999. ACM Press, New York. `http:// nitin.www.media.mit.edu/ people/ nitin/ NomadicRadio/`.

[15] B. Stone. Hitting the right notes. *Newsweek*, Jun 2000.