4. DESIGN EVALUATION

As the aim of this present master thesis was to realise an audio application and to use it to study how to exploit audio for Nomadic users, thus an evaluation of the application designed for this purpose was conducted and will now be described.

4.1 <u>Observations and possible scenario</u>

In order to being able to evaluate the application designed, a practical test that interacted with the programs was needed. A reason why the application was tested, was to find out its good characteristics, and to find areas where the application needs an improvement. An example scenario of using the manager and the clients is also explained.

4.1.1 <u>Good characteristics</u>

Some of the points that were considered positive after testing are the following:

- The application is considered *easy to use*, as most of the processing is hidden from the user. The user need not know details of the behaviour of the manager and the player, since both of them receive the data they need to operate from other programs, rather than from the user. Since the user_interface client shows the user the available functions, he/she only has to enter a number to indicate the desired command, and the data requested (if any). For an example of how written notifications can be heard as audio notifications, the user only needs to start alert_generator followed by the text that is to be output.

- Thanks to the client user_interface, the *user* is able to *directly interact* with the manager *in the construction* and modification *of the playlist*. This fact was considered important in the test, since the user can easily and very directly control the way in which the playlist is built, and, when desired, terminate both manager and player.

- The digital *sound*, that the SmartBadge 4 CODEC provides, was considered to be of *high quality*, comparable with MP3 or CD players.

- In the test, the advantages of having *audio notifications*, and using SmartBadge 4 as an *audio-only* wearable device were appreciated. Audio-only wearable devices allow the user to move about or have his/her hands free, while still receiving notifications, and listening to music.

4.1.2 Characteristics needing improvement

The aspects of the application that were considered in need of improvement after testing are the following:

- When there are MP3 files included in the playlist, the output of these files only starts after an appreciable *delay*. The client player makes a function call to *mpg123*, and, after that, two fixed delays occur. The first delay comes from opening /etc/sound/dsp. This always takes approximated 15 seconds to open. The second delay comes from the opening of the file itself, taking another approximated 15 seconds. The start-up should be improved, specifically since the majority of digital songs are stored in the form of MP3 files, and this delay can be really annoying.

- *Not all* the existing *types* of *audio files* are handled in the application. Although two of the types, MP3 and wave files, are the most commonly used formats for storing digital music, the client user_interface will only accept files of the types handled, thus the user is restricted in the types of audio files he/she can listen to with the present application. Some additional formats should be added or SOX[37] could be used for conversion.

- In the application designed, *no information* about the *user's environment* is acquired as an input to the application. Therefore, no intelligent behaviour of the wearable device is demonstrated by this application. However, the present application (along with other work being done with the SmartBadge 4[13], [18], [19]), is easily extended to accept context information. This would improve the wearable device's behaviour.

- As the SmartBadge is just a prototype for experimentation, it does *not* have a *display* (there is no monitor) or keyboard for inputting data. This restricts the desired usage and mobility of this application, since a connection to a computer provided with a display and keyboard is needed when using both user_interface and alert_generator clients for the necessary input and output of information.

- The *termination* of the client *user_interface* always makes the *manager terminate*. This was useful for the current application, since it was interesting having a way to terminate the manager and the termination of user_interface was considered the best way of doing it. However, in a future improvement of the application adding more clients, it is not convenient that the manager terminates when the user_interface terminates, since there would be other clients sending data to the manager.

- In all the application, when a request is sent, a response is expected, but the *loss* of that *response* is not contemplated. For example, when the player requests the manager to delete the song which is going to be played, it waits for the response from the manager. But if that response is lost, the player waits forever and the audio file will not be output. Therefore, adding *timeouts*, *retries* after a established time, and *checking* of the datagram received is clearly convenient for enhancing the reliability and scalability of the application.

4.1.3 Example scenario

A possible scenario that could occur when using this application is the following:

- David, a student working on his master thesis in a department at KTH, Stockholm, learns that he can use his SmartBadge and the application created for this thesis to listen to some songs (which he has earlier downloaded to his linux computer), while he works. He is also interested in finding out how audio notifications would be like. He starts the manager, and the player, and then, he starts the client user_interface (as explained in section 3.8). He enters command number 1 and then enters all the songs he wishes to hear for the rest of the morning. Once he has entered the first song, the player begins to output it. After a while, he remembers that he wanted to use the client alert_generator in order to test how notifications would sound like. As a simple test, he enters the string “David is such a clever guy” and following the next thing he hears is this message read by a rather human sounding voice. Over the course of the rest of the morning he tries out all the possible commands, and before going to lunch he loads up enough audio content so that he will have something to listen to during his lunch break. He decides that after lunch, since the audio alert from text messages worked so well, he will see if he can get the badge to read his new incoming mail and deliver the latest instant messages from his friends.

4.2 New functions and scenario

After experimenting with the application, some ideas about new functions to improve the present design occurred. These functions and new scenarios will be explained in this section.

4.2.1 New functions

The new functions that can contribute to an improvement of the application are the following:

- A function that would enable use of *playlists already built*. The client user_interface would have an additional command to select an existing playlist. The songs in that playlist would be copied to the /tmp directory in the SmartBadge 4, where the server manager and the player are able to access them. The manager would have to check if there are any elements in the list, and if so, would add the already built playlist after the existing elements of the manager's list. Another possibility would be to add a *priority* to the playlist the user would like to include in order to be able to listen to it first, even if there were more elements in the manager's list.

- Another possible command to add (specifically when utilising existing playlists) would be a *shuffle* option. If this command has been given, the manager will insert new playlists in a shuffled order. This is similar to the shuffle option on many CD and MP3 players.

- A possible modification of the client player would be to replace the function get_delete_first_song() with two functions: *get_first_song*(), *delete_song_played*(). The client player would ask the manager to delete the audio file once it has been played and not before. That way, the audio file to be played stays in the playlist, and is deleted only when the manager knows that it has been played. This modification in the player affects alert_generator, since the priority (set to play the notification at the next available time) would have to be changed in order to insert the notification after the first element, which is the one still being output.

- Another modification that could improve the application is extending the manager to be able to accept a request to *output* an audio file *at* or *before* a *particular point of time*. The way in which user_interface and alert_generator send the information to the manager would have to change, since it would be a time instead of a priority that determines when to schedule a particular audio file. An easy way of implementing this for user_interface and alert_generator could be setting the priority parameter to 0 when the audio file is requested to be played at a particular point of time. The manager would then include the audio file in the appropriate place in the list by comparing the time_to_play of the other elements of the list. It would also have to check that this audio file is placed in the correct place when it adds or deletes new files from the playlist. With this new procedure, alert_generator could be also set to tell the manager whether it uses priority or time for including a file in the playlist. Adding the possibility of inserting an element *after* a particular time would be easy once the above addition has been made.

- A function that would *delete more* that one audio file at a time. This function would be useful if the user would like to delete all the elements of playlist after the current one, for instance. It could also delete the files after the Nth one, or delete a single audio file but in all his occurrences in the playlist.

- A function for *pausing* the *application* without having to terminate it. That would be useful when the user does not desired to listen to any audio file for a while. This function obviously needs another for *re-starting* the application again, at the same point it was when paused, but checking the times.

4.2.2 <u>Example scenario</u>

Having returned form lunch, David decides to put off making his own enhancements. However, he has found out that the application has already been enhanced with several new functions, so David decides to try out these new enhancements. He has learned that he can now listen to pre-built playlists, thus he does not have to come up with his only selection of songs for that afternoon. He has heard that Maria is making playlists available from her web site, so he uses his web browser to navigate to this site and discovers that she has made a playlist to introduce listeners to Flamenco Guitar music from Andalucia. So he downloads this playlist and his manager starts to insert the songs in the playlist and, as soon as it has the first one, the player starts. After listening to a couple of songs he decides that it would be interesting to listen to a mix of contemporary guitar pieces (which he also noticed on Maria's site), mixed in with these classical pieces. So he loads the new playlist and requests the manager to shuffle the new playlist into his existing playlist.

As time passes, he realises that he needs something to wake him up after lunch and decides to switch to rock music for a while. So he saves his current playlist and decides that he can return to it after his coffee break.

During his coffee break he learns from Maria that one of the new features is being able to schedule an audio file to be played at a specific time of the day. He learned that this is how she reminds herself when it is 15 minutes before her first class of the afternoon-- with a short cut of Bach's Fugue in D Minor played on the Great Organ in the Church of Our Lady of the Incarnation of Marbella.

Returning from his coffee break he returns to the playlist of guitar music that he was listening to. While listening to this music, he decides he is going to wake himself up early the next day with some of his favourite rock songs. So he creates a new playlist, which includes the rock songs that he needed in the morning to wake him up and he schedules this playlist to start playing at 5:30 the next morning. He decides that after getting up, he will want to listen to some more relaxed music, so he inserts a playlist he created two days ago just following his morning wakeup selections. He also decides that, rather than being late to meet Maria for lunch, he will create an audio notification using alert_generator with the string: "Time to go to lunch" and sets its time_to_play to be 12:30 the next day. Just to make sure he schedules another audio alert, for 12:40 to say "Are you trying to starve Maria? She has been waiting for ten minutes!".

He is quite surprised the next morning when loud rock music starts playing at 5:30, but after a little while he remembers that he arranged this to make him up early to get to his class at 8:30. Following class he starts his two hour shift (10:30-12:30) working at the help desk. He listens to his selection of easy listening songs while he tries to figure out how to remove a virus from a user's portable computer. He is concentrating quite hard on solving the problems with this computer and does not notice that time is passing. Fortunately just at 12:30 he gets an audio alert reminding him about his lunch appointment with Maria.

He is very happy to be on time for lunch with Maria, he deletes the rude reminder that he scheduled for 12:40. He has even managed to get to the cafeteria in time to get his favourite selection, blini with löjrom (which usually disappears quickly). He is starting to really appreciate this application that lets him listen to the songs he likes while giving him the audio announcement reminding him to go to lunch on time. During lunch he pauses the application while chatting with Maria. At the end of the lunch he decides that he better stop listening to songs and concentrate on the audio lecture notes he has from the class he missed last week. So he deletes all the pending entertainment elements of the playlist after the currently playing song and inserts the audio lecture notes file into the playlist. But before really focusing on the lecture notes, he remembers that he has promised to call his mother to ask what he should bring over for dinner (as it is his turn to bring the food). Therefore he adds two audio notifications: one to phone his mother - to be output at 16:00, when his mother is for sure to be at home, and another notification to leave at 18:00 so that he can stop at the deli on his way to his mother's house. After a long afternoon of studying he is really happy to heard the audio notice that it is time to go to dinner. He is very happy with all of the improvements in the application.

4.3 Additional clients and scenario

As a result of the testing, several additional clients were considered useful for improving the application. These clients and an example scenario will be described in this section.

4.3.1 Additional clients

The additional clients considered useful for a future improvement of the application are the following:

- A client that would provide *speaker recognition* to the application, using one of the already existing tools. This client would receive a voice-audio file as an input, and would decide who is the user cut of a group of users whose voices are registered. Then, it would pass this information to the manager, thus the manager could possibly insert an already built playlist for this specific user. If while using a playlist, this client detects a different user, the manager could change the playlist to the new user's selections.

- A client that would provide the application with *speech recognition*. This client would wait for audio commands and parameters from the device's input. The expected audio would include the possible commands set for the user_interface, such as “enter a file in the playlist” or “save the playlist”. This client would recognise these utterances, and initiate the processing that user_interface should do. Using this client, the application will have an all audio user interface eliminating the need for a text based user_interface client. The resulting application would be a real audio-only application. Of course, the client user_interface could be retained by the application in case a visual interface was also desired.

- A client dedicated to *capturing audio*. This client would be continuously listening to the device's audio input, and could send to the speaker recognition and speech recognition clients the audio that it captured in order for them to be able to do their processing.

- A client providing information about the user's environment, so called *context information*. This client could decide whether the user is in a meeting or in his office, for instance, by analysing and evaluating the audio input that the application is receiving. It could also decide whether to connect to the internet via one of the several available interfaces, or not connect at all if the personal user's playlist already has enough elements in it. Sean Wong's client for providing *location awareness*[19] could also be easily added to the present application.

- If the device is connected, then a client *checking* the arrival of *e-mail* or *SMS* could be useful. This client could send these e-mails or SMSs to the alert_generator, and the user could hear these notifications instead of having to read them on a screen. At the same time, it could set the priority or time_to_play to those e-mails or SMSs according to the sender. The application would maintain a data base in which the user specifies the priorities for the people with who he/she usually communicates.

4.3.2 Example scenario

David has now been using the previously improved application for several weeks. When he arrives at the department one morning, Maria asks if he will test the new clients she has just recently added to the application. To do this, he begins by separating the people with whom he usually communicates into groups. He gives a high priority to his brother Mark, his girlfriend Ana, and his thesis supervisor Magnus, and indicates that he will want to hear the audio notices concerning e-mail or SMS from any of them, no matter where he is (i.e., an important meeting) or what he is doing (i.e., concentrated at work or engaged in a conversation). To the rest of people he usually communicates with, he gives lower priorities, thus notices concerning communication with them will not interrupt him in situations in which he does not want to be disturbed. He also indicates that he wishes to hear a maximum of ten notifications in one hour, so the application should consider this limit when generating audio notifications. He also indicates a number of context in which he does not want to receive notifications (unless they are of the highest priority): meetings in the department and conversations that have already lasted more than 5 minutes.

After all this, he selects one of his playlists (which contains his desired notifications for going to lunch and home) using the visual user interface, since he is in front his desktop computer and finds it easier and faster than using the speech interface. He starts working and after a while decides to try the new speech user interface to add some new songs he thinks would stimulate his mind for his current work.

At 12:25 he gets an audio alert reminding him that he has a project meeting with his supervisor in the conference room next door to the cafeteria at 13:00. He finishes eating and goes to the meeting. However, just before the meeting ends, he receives an SMS from his girlfriend. Since he had set a high priority to her SMS message, her SMS is converted to audio and played in his right ear. In this message, she asks him if he would like to go to the cinema with her. He indicates that he has a reply and sends a quick "Yes".

When the meeting finishes, he sends her a longer answer saying that he would love to go and asks her to meet him in the department. As he knows that she is also using a SmartBadge 4, he subscribes to her presence at the department.

Over the course of the afternoon, he receives three e-mails from three of his friends who have arrived in town for the weekend; as he was expecting them, he had added them to his priority list for today, but at a somewhat lower priority. Thus, their e-mail messages were scheduled between songs.

He continues working until he hears an audio notification telling him that his girlfriend has just arrived at the entrance of the department. Immediately after this he hears a notification telling him that it is "time to go home". So he grabs his coat and heads for the entrance, hoping that he will meet Ana on the way to the front door. Just after entering the building, Ana got into a discussion with one of his colleagues and followed him over to the coffee machine just around the corner from the entrance, but not visible from the entrance. Meanwhile, David has started down the stairs and, after five minutes, arrives at the entrance, but there is no sign of Ana. He sends her a note asking if she took the elevator up and sends his location by the entrance. She gets his message and sends a reply saying she will be there in just a minute and adds her coordinates. David sees that she is near the coffee machine and heads in that direction. They meet each other in the hallway and decide to use the exit from the building which is closer to the cinema.

Later this same week, David sends a message to Maria with his experience and in summary says that the latest version of the application is even better than he had imagined.