International Workshop on Web Intelligence and Virtual Enterprises 2 (2010) 1–9
IOS Press

Information System For Mapping Wearable Sensors Into Healthcare Services: Application To Dietary Habits Monitoring [1]

Timothe Faudot [a,*], Guillaume Lopez [a,b] and Ichiro Yamada [a,b]
[a] *The University of Tokyo, School of Engineering, Hongo 7-3-1 Bunkyo-ku, Tokyo, Japan*
E-mail: {timothe, guillaume, yamada}@lelab.t.u-tokyo.ac.jp
[b] *Japan Science and Technology Agency, CREST*

Abstract. We have been developing a system aiming at providing patients but also healthy people a way to monitor their eating habits from environmental sound data collected by mobile devices and processed online. The collected sound data is stored on a server and can be shared with registered researchers looking for samples to build and test new processing algorithms, which can be uploaded into the system and used by physicians as analysis tools to provide the patients with useful feedback. Such a system provides the research community with a rich and ever-growing database of sound samples; it also provides physicians a way to follow their patients and generate personalized feedback reports by selecting which algorithms to combine and apply to each patient. In this paper, after presenting the social and professional needs for such a system, we first describe its composition. Then, we present in detail the main technical and organizational points of the current system, and end by discussing about the technological issues and challenges for further development.

Keywords: Health Information Monitoring, Healthcare Community, Wearable Physiological Sensors, Metabolic Syndrome, Information Processing Platform

1. Introduction

Following the Ministry of Health, Labor and Welfare of Japan to promote home services aiming at preventing diseases to happen rather that curing them, we developed a system where small sensors worn by the patients send their data over a network (Internet) where this data gets processed and customizable reports are generated for the patients to see. This preventive kind of medicine, while having the objective of providing a better life for the users (patients or healthy people), also gives the opportunity of accelerating healthcare related research, as the massive amount of data that can be collected in such a system can also be shared amongst researchers and doctors on the web for them to develop new ways of detecting and preventing diseases.

Though such system using wearable sensors to collect health information has already been studied as in [7] [8], we wish to push the concept farther as virtually any data processing algorithm could be inserted in our system, not only mastication counting. Also some more web-oriented approaches have been implemented as in [5] where images of meals are the principal data source. The system we propose has been designed to accept audio files as data that will be processed. We have began generalizing it to other kind of data types but this paper will focus only on audio. Though the goal system aims at a management of various human life style diseases prevention using sensed data, we decided to first design the architecture and implement the prototype of the system for a specific ap-

[1] This research was supported by the research fund for Core Research for Evolutional Science and Technology (CREST),granted by Japan Science and Technology agency (JST).
[*] Corresponding author.

plication field: eating habits using sound information. In this paper we first introduce our system architecture, then the different usages for each person involved, the challenges imposed by such a system and finally the future trends that we would like to implement. Also, in the rest of this paper we use the term "patient" for both ill or healthy people who just want to monitor their health as the system can be helpful to both.

2. System overview

The system is composed of mainly two parts: the sub-system composed of the mobile devices that all patients own and the online sub-system for processing the collected data. The mobile devices sub-system is used for sound data collection and is used by the patients as a way to keep track of their health style habits evolution and receive advices and reports. This kind of device is highly interactive and allows the patient to really play a part in his treatment, which leads to more trust and better feedback or advices from the both the doctors and the patients.

The processing part of the system is more complicated as it is what glues everyone involved in it. Its main purpose is to process the uploaded data by the patients and provide them with reports in which the current progress, goals, advices and remarks are indicated. It must also provide an interface for the doctors to follow their patients and another interface for the researchers to help them develop new algorithms. This processing system demands a high technical expertise and while it is quite common in nowadays web-services we are dealing with medical data and reports that influence the daily life of people so we must take extra care in the development. Processing is good, but it is only a single node of our system, we also want to provide the research world with data that is diverse and easily classifiable, searchable and downloadable. We will now expose the different parts of the whole system.

2.1. patients, researchers, doctors

There are three types of users: patients, doctors and researchers. Each has a different usage of the system as follows:

- Patients
  - Record sound data during meals
  - Receive reports

Fig. 1. bluetooth sensor

- Researchers
  - Download/use the available uploaded sounds to develop new algorithms
  - Upload their developed algorithms to the online server for usage in reports
- Doctors
  - Assign developed algorithms to patients regarding their needs
  - View and edit reports of followed patients

2.2. Sensor

To get the sound of meals activity, we use a Bluetooth headset as in figure 1 that records the internal sound of the body via one ear. The headset is composed of two microphones (one internal, one external) and while we could use both channels, for the sake of simplicity at first we use only the internal sounds. Using the external channel would allow us to better understand the context in which the patient is in: inside, outside, alone, at a restaurant, at a station, etc. Also it could be used to provide some filtering to clean the noises that could appear in the internal channel as it is not 100% soundproof. Hence our recordings are done in AMR-NB [1] 16 bits 8000Hz mono. This corresponds roughly to 70kb per minute of recording.

2.3. Stored sound data

Once the amr_nb audio data is uploaded to the server, an acknowledgement is made to the mobile device and the data is converted for space matters on the server using a lossless codec (Flac), for which the bit rate becomes approximately 245KB per minute. In ad-

dition to storing the raw data as a file on a hard drive, some meta-data is uploaded by the patients following the privacy settings they have set. The uploaded meta-data is for the moment only the precise position via GPS (or network triangulation if unavailable) of the patient when the recording started. Once the data is converted on the server the associated waveform and spectrogram images are automatically generated.

2.4. System openness

The concept of sharing the uploaded data from patients and making it available to researchers to help them in their findings is one of the key points of such a system. If well done, a relatively large database could be built in decent time and could help this field of research where samples are not so numerous. For an open system, we wanted open components hence everything is based on open-source software or programming languages: the Android platform [2] , java [3] and python [4] languages, ffmpeg [5], Flac [6], django [7] / piston, etc. By making an intense usage of web technologies, the system is more widely accessible than if it was only developed for a specific hardware architecture. The web is based on an open architecture, with global storage and global access capabilities, which is what we desired.

2.5. System interest

Often when doing analysis on raw data researchers are confronted with a lack of such data, this system allows them to work on an ever growing set of annotated data which they can filter and use as they will. Doctors can also follow in real time the activity of their patients, get recommendations for some in case of long inactivity or abnormal behavior detected, that leads to the patients being hopefully better followed. At last patients are provided with a fun and easy to use system which allows them to be the main actor in their treatment, close interactions with the doctors can lead to a better understanding of the bad eating habits they could cure. This whole system aims at putting together people with different needs but as we saw, making them work together is profitable to all of them.

3. System usage

3.1. Patients

We use a commonly available mobile platform for the development of the application aimed at patients: the current version of the application runs on Google's Android OS version 1.6 and upper. We choose to use mobile phones as a personal data collection mean as almost everybody owns one and have it with him also most of the time. The choice of Android was guided by the openness of the system and while in Japan it is not yet as developed as in other countries, its rapid growths should come quickly as every operator announced at least one model in their next line-up.

In order for our application not to be obtrusive for the patient in the everyday usage of the phone, long running tasks such as recording data and upload/download of current patient status and files are executed in the background as services or processes. The application consists of four main functions:

- Status viewing: this screen shows the current patient status, data related to his account and social-networks related information.
- Recording: on this screen the patient can start and stop a recording session. While recording he also have access to several mini-games to help him get better eating habits.
- Data synchronization: This is done automatically every day, asking for the permission of the patient to upload the locally stored data to the web server.
- Report viewing: this can be accessed from the patient's device home screen as a folder grouping all the available reports.

Each patient is attributed a personal RSS/Atom feed, to which updates are pulled from the server. The android application polls this feed regularly and download the new reports available. We store the reports contents in a system/data agnostic way (json) so that when requested, the report can be generated on the fly to better match the request. For instance if a link to a report is clicked directly from the personal feed via a web browser (figure 2) an html version of the report is generated and sent as output. But if the same is done from an android device, then the raw data is sent and is further processed locally to use the display widgets available on the device that better fits in this context than html does as send in figure 3. We currently store the reports contents as a collection of algorithm results which are json formatted output of the algorithms.

Such a system could be improved by using directly the recently emerging NoSQL [8] databases such as CouchDB [9] or MongoDB [10] for instance. Such document oriented databases would allow the system to provide a more flexible alternative for the researchers to develop their algorithms as they would be less re-

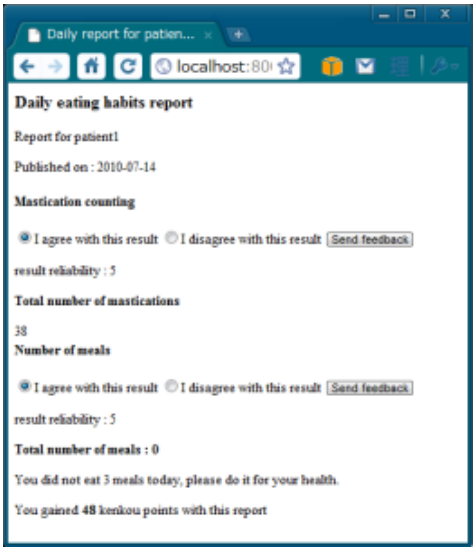

Fig. 2. HTML report

Fig. 3. report on android

stricted to some extent to a certain format and would permit a more easily scalable and upgradable system.

3.2. Doctors

The doctors have access to a web interface on which they can see the current status of the patients and the delivered reports. They can view the available algorithms developed by the researchers and create report templates with those which once created can be assigned to patients. Of course the same template can be used for different patients and will produce different reports. Templates are just a collection of algorithms that share a common timeline; that is algorithms that can be run on the same set of data. Cron [11] processes will be launched by the server every day, week and months to generate the related reports for each patient.

3.3. Researchers

3.3.1. Selecting sounds

Researchers have access to a web interface (figure 4) to get batch of sounds that are interesting to their research. They can filter the sound database following some defined criteria (associated tags, geographical location, duration, date and time the data was recorded, specific patient) and download the whole selection of sounds as a compressed archive. We remind that sounds are stored in a lossless compressed format (Flac), the action of compressing files is only here to simplify the transfer process, not to decrease the size of the transfer since the gain is merely perceivable in such a case. Indeed the compression ratio is around 95%, while the time needed to create the compressed files is much more important to us for performance matters.

Given the raw and meta-data available, various tags such as "8000Hz" or "evening meal" for instance are automatically associated with each sound. A cloud visualization of such tags is available and allows the researchers to focus their research on samples that are specific and/or numerous and to view the main tendencies of the sounds in the database.

3.3.2. Developing algorithms

Researchers can then develop algorithms following the provided guidelines, which are mostly restrictive about the capabilities of the server and security issues. An API is provided only in python for the moment while including other languages should be done pretty easily as the interface for the input and output of the algorithms is already defined. The output interface is a json dictionary which is defined like this:

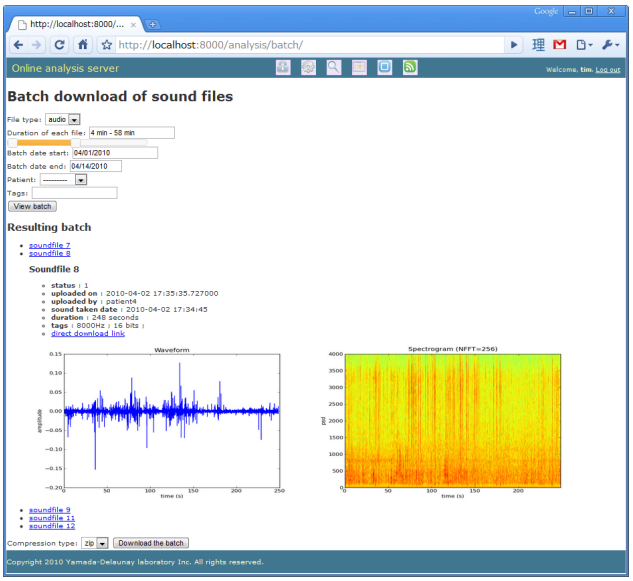

Fig. 4. batch download interface

Listing 1: json output interface

```
{
  "title": "title of the algorithm result",
  "reliability": 2/5
  "sections": [
    {
      "title": "title of the section",
      "body": "text content of the section
          body",
      "images": [list of generated image
          names],
    },
    {
    ...
    }
  ],
  "points": total number of points associated
       with the report,
  "links": ["http://www.localfood.com",
            "http://news.japan.com/foodmart/"
                ]
}
```

Outputting json is trivial in most of the common languages used nowadays, so using other languages to develop algorithms is not really a problem, as long as the server supports it.

Listing 2: Call method for the algorithms

```
>python script_file.py path_file1 path_file2
    ...
```

The input interface is defined simply as passing the list of paths to the audio files to be analyzed as a list of arguments to the program. For python scripts in gives: Researchers can test their algorithms on their computer and once they give satisfactory results they can upload it to the server, from which doctors will be able to use them in reports. Researchers are given a set of raw audio files and some parameters as parameters of their algorithms, they can transform this low level data into more high level abstracted results. The already developed algorithms concern mainly mastication counting and meal time analysis [4]. A lot of previous papers insisted on the predominant role of mastication count regarding good eating habits [2] [3], that is why the first algorithms we developed were related to this. Other algorithms involve more the meta-data associated with the sound files, not the raw sound themselves, for instance an algorithm can check that the patient takes 3 meals per day, takes his time to eat, eats around the same time each day and do not eat at the middle of the night for instance. Please refer to the next section on Programming Interfaces to see how the researchers can develop their algorithms.

3.3.3. Security

While not our main subject of concern at this time, we must think about the security and integrity of the stored data while running code from untrusted source. The activity of every algorithm is monitored and as the researchers who registered to the server will have to provide a proof of identity and affiliation with a research organism, the subject of malicious algorithms is not as present as should real unknown persons upload their algorithms. Running non-trusted code on the server can be done safely using a restricted user account, having access to only special parts of the operating system (sandboxed). We implemented this using chroot jailing techniques. We tested the jail by uploading algorithms that would download viruses and try to execute them or trying to erase the whole file system, both proved ineffective against the jailing.

Concerning the connexion to the system by the patients, we use the new standard-to-be OAuth in its 1.0a specifications. This way we avoid storing the patient's user name and password directly on the mobile phone, and can severe any access previously authorized without changing the password of the patient in case he loose his device for instance. The Twitter and Facebook connections are also done using their proper OAuth authentication protocols.

4. Community

The concept of community is omnipresent in our system for instance if the patient wants to we take advantages of already existing social networks and send automatic updates to those. As an example the patient status updates can be sent to Twitter[12] and Facebook[13]. We extended the concept of community farther, as our system puts in interaction patients, researchers and patients (Figure 5). Regarding our previously described usages of each user type we have those main communities interacting with each other:

- Patients <-> Doctor
- Doctor <-> Researchers
- Researchers <-> Researchers

We can also in a lesser measure have interactions between patients and researchers as the patients can provide direct feedback concerning the developed algorithms. When they view their reports a form is included in it and they can tell if they agree or not with each result. Lastly and obviously communities of users of the same type can also exist, let it be externalized in an existing SNS with automatic updates for patients or via the server web interface. The geographical position of the data can also be available following the privacy preferences of the patients, some more local communities can be built easily, while this functionality is yet to be developed it can be used to develop more in-real-life interactions between patients. Unfortunately the community aspect is not fully implemented at the moment and most of the interactions are limited to simply sending e-mail, writing comments on specific parts of the system (uploaded file, algorithm), or sending feedback as described before.

To keep the patients active and willing to use the application, several methods can be developed. For instance, a personal evolving status is attached to each patient, containing data such as found in games like RPG (Role Playing Game) (figure 6). The more the patient uses the application the more "experience points" he gets to gain levels, unlock achievements and so on.

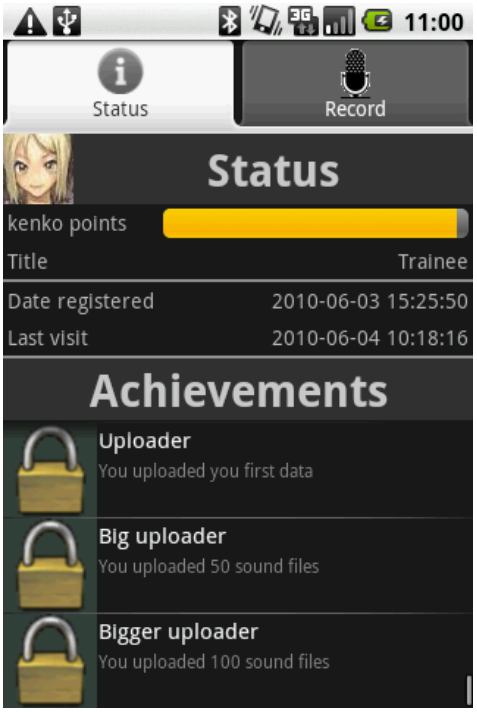

Fig. 6. patient status on android

5. Application Programming Interfaces

Two APIs are available to use with this system. One is for the researchers to manipulate the provided audio files as is, that is to be able to read Flac encoded files, and format the output of their algorithms following the defined interface. They can choose to use it or not, for the moment it is only available in python, as a glue language it still can be used to call many other languages functions and hence benefit from the provided API as well.

Another API is here for accessing the data stored on the server. A web interface is provided to filter the whole database and work on the samples that are of interest to one's current research, but one can nevertheless choose to use this API to query, download, and upload sounds or algorithms. The API is Restful based on HTTP and queries can be made in xml, json, yaml or pickle, the default being spoken by the server is json. It is currently used by the application running on the mobile phone to synchronize data between the devices and the server.

Listing 3: Example of request using the Restful API

```
curl http://server.com/analysis/api/
    soundfiles/1/
> GET /analysis/api/soundfiles/1/ HTTP/1.1
> Authorization: Basic cGF0aWVudDE6cGF0aWVudA
    ==
> Host: server
> Accept: */*
>
< HTTP/1.0 200 OK
< Date: Tue, 16 Feb 2010 08:12:55 GMT
```

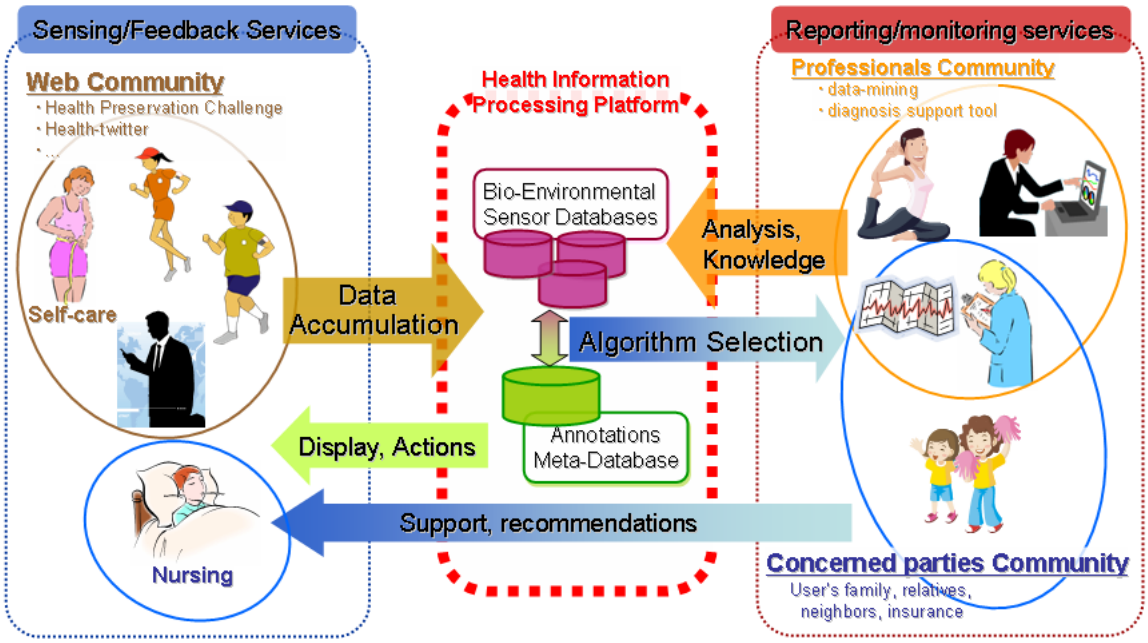

Fig. 5. Physiological information processing platform community use cases

```
< Server: WSGIServer/0.1 Python/2.6.4
< Vary: Authorization
< Content-Type: application/json;
<
{
  "status": 1,
  "sound_taken_date": "2010-02-16 14:40:14",
  "sound_file": "uploads/soundXX.flac",
  "public_url": "uploads/soundXX.flac",
  "spectrogram": "soundXX_spec.png",
  "waveform": "soundXX_wf.png",
  "uploaded_by": {
    "username": "patient1"
  },
  "location": {
    "provider":"network",
    "latitude":"35.71433461428571",
    "longitude":"139.76295114285713",
    "altitude":"0.0",
    "accuracy":"165.0"
  },
  "duration": 8,
  "id": 1
}
```

(This program simply counts the number of files that where passed to it, and returns it well formatted to the calling server process)

Listing 4: Example of usage of the python API for running algorithms of the server

```
import sys
import api
#logs are written to the std. error stream
log = sys.stderr
#output is written to the std. output stream
output = sys.stdout
soundPaths = sys.argv[1:]
#a result object to be filled
r = api.Result(
  title="Audio files counting", #main title
  reliability=5, #(5:top, 0:worse)
)
s = api.Section(
  title="Number of files",
  body="Number of sound files used : %d"%len(
      soundPaths)
)
r.addSection(s)
#number of kenkou points attributed
r.addPoints(len(soundPaths)*10)
#flushing result as json to the output
output.write(r.toJson())
#returning a successful exit code
sys.exit(api.RESULT_OK)
```

6. Data processing

6.1. Processing estimations

We tried to extrapolate the needs in term of processing power and availability for our system, here are some: The test machine was a Virtual machine on VirtualBox running Linux Ubuntu 9.10, four cores at 2.5GHz were allocated with 3.5GB of RAM. The software running was python 2.6, django 1.2, apache 2 with mod_wsgi, the database used was mysql 5.1. Given the previous numbers, the following reflects what charge the system should expect on average.

1. Number of patients: 100
2. Number of files uploaded by a patient for each day: 2.5 [1]
3. Length of the files uploaded: 30 minutes (2.77MB amr_nb)
4. Number of reports for each patient: 3 (daily, weekly, monthly)
5. Number of algorithms for each report: 3
6. Running time for an algorithm on a sound file: 15 seconds
7. Conversion of a sound file to Flac: 2 seconds
8. Generation of the waveform and spectrogram: 2 seconds
9. Space needed to store a 30 minutes Flac encoded sound file: 11MB
10. Waveform and spectrum images of a sound file on disk: 600KB

A patient uses everyday (2)*(5)*(6)+(2)*(7)+(2)*(8) ≙ 120 seconds of running time. A hundred patients would use 204 minutes, or a little less than four hours. Four hours of constant processing can be done, but our server is used for other applications such as the sounds search system for researchers which itself consumes some precious computation power and resources. Concerning the space needed for storing the data, we can expect (2)*((9)+(10))≙30MB per day per patient. For 100 patients this number quickly rises to 3GB per day in total. Such storage can only be envisaged using the capabilities of cloud storage services such as Amazon S3. A quick calculation gives us the price of transferring each month's data to be at least 18$US. The double if we consider transferring back the files for processing by the algorithms which is most probable.

[1] three meals per day but the non-constancy of the patients makes this number drops

That being said, for our system to handle more than a mere hundreds of patients, we will need to shift to more robust processing architecture as described in the next chapter. We also did some load testing and tuning of our Apache processes to see how many users at the same time we could serve. The test bench was a python script allocating a pool of processes to which was mapped a list of sounds ranging from 4min to 120min. Each process uploaded a randomly selected sound and its associated meta-data using curl. 18 sounds were processed on average in 1 minute. If we try to upload around 40 sounds simultaneously (around 20 hours of sound) the overhead of conversion on the server makes it run at full capacity, making the responses to the connected clients really slow and swapping, which renders the conversion process awfully slow (around 15 minutes).

For this first prototype, we chose the default Apache configuration and the static media were served using the same Apache instance as the dynamic content which is something to proscribe in a real environment. We do more tests from now on while switching to Lighttpd for static content delivery, and also experiment with FastCGI which should reduce drastically the overhead and provide a better isolation of our instances hence a better security.

6.2. Next challenges

Such a system will involve lot of processing and can be done in a massively parallel fashion, as the reports generated are for one and only one patient. Having said that, the current trend in massively distributed computing for processing data can be achieved using the usage of the Cloud, and a MapReduce [1] type programming framework allowing the dispatch of algorithms to multiple files on different servers of the cloud. Our current system runs on a single machine, and will be quickly saturated as the data available and the number of patients to generate reports for becomes large. We will need to migrate to the previously cited technologies in order to withstand the growth of the system. Our next vision of the system is based on Amazon Elastic Computing Cloud (EC2/S3) and their use of Hadoop Map Reduce [14]. Using those cloud technologies will also fix our current security issues on virtualizing the file systems of our own servers, which can be quite a hindrance setting-up and maintaining.

Also, we previously introduced in [6] the concept generating different kinds of graphical user interface (GUI) content depending on the temperament of the

user. We can implement it for the reports display by dynamically modifying the generated templates and adapt them to the mobile phone GUI, though such a framework is still to be developed on this kind of devices.

7. Conclusion

We introduced our system architecture, the different usages for each involved users, and described in detail its main technical issues. The current system uses only sounds to provide the patients advices related to their habits and also to provide researchers a varied and rich database upon which to develop new algorithms. One future goal is to generalize this system to not only sounds and eating habits but to other lifestyle habits for which one could need advice. The current system can be ported to support new kind of sensors and data without too many changes, for instance a sensor for constant monitoring of blood pressure is currently being developed at our laboratory, such a sensor is now being integrated into the system.

References

[1] J. Dean and S. Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*, OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004.

[2] M. Furuta and H. Miyako, *Epidemiological observations regarding the significance of chewing habits for health*, Japanese Journal of Psychosomatic Dentistry, Vol.14, No.2, 1999, Page.113-134.

[3] M. Furuta and H. Miyako, *Observation of matters related to chewing habits*, Japanese Journal of Psychosomatic Dentistry, Vol.14, No.2, 1999, pp.97-111.

[4] H. Zhang, G. Lopez, M. Shuzo, J.J. Delaunay, I. Yamada, *Meal Sound Analysis Using Artificial Neural Network*, 2010 Mechanical Engineering Congress, Nagoya, Japan, September 05-09 2010.

[5] K.Kitamura, T. Yamasaki and K. Aiazawa, *Food log by analyzing food images*, Proceeding of the 16th ACM international conference on Multimedia, 2008, pp.999-1000.

[6] G. Lopez and I. Yamada, *New Healthcare Society Supported by Wearable Sensors and Information Mapping based Services*, 1st International Workshop on Web Intelligence and Virtual Entreprise (WIVE'09). CD-Rom proceedings of the 10th IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2009, Thessaloniki, Greece, October 7-9, 2009.

[7] J. Nishimura and T. Kuroda, *Eating habits monitoring using wireless wearable in-ear microphone*, International Symposium on Wireless Pervasive Computing, 2008, pp.130-133.

[8] M. Shuzo et al., *Wearable Eating Habit Sensing System Using Internal Body Sound*, Journal of Advanced Mechanical Design, Systems, and Manufacturing, vol. 4, no. 1, pp. 158-166, 2010.

Notes

[1] http://en.wikipedia.org/wiki/Adaptive_Multi-Rate
[2] http://www.android.com
[3] http://java.sun.com
[4] http://www.python.org
[5] http://ffmpeg.org
[6] http://flac.sourceforge.net
[7] http://www.djangoproject.com
[8] Non-relational, distributed and horizontal scalable databases
[9] http://couchdb.apache.org/
[10] http://www.mongodb.org/
[11] Time-based job scheduler in UNIX-like OS
[12] http://www.twitter.com
[13] http://www.facebook.com
[14] http://hadoop.apache.org/mapreduce/