# WEB BASED PSYCHOACOUSTIC RESEARCH ON COCHLEAR IMPLANT PATIENTS

Vaerenberg Bart[1], Verschooten Eric[2], Bracke Peter[3]
[1]vaerenberg.bart@student.ha.be
[2]e.verschooten@ha.be
[3]peterb@abionics.fr

June 6, 2007

***Abstract*—PACTSweb is a web based research platform for offering psychoacoustic tests to cochlear implant patients. Its purpose is the facilitation of the repeated testing of human subjects required to produce meaningful results in the field of psychoacoustic research. The platform provides a variety of common psychoacoustic tests and the ability to manage and extend the collection of tests contained in the framework. PACTSweb offers a level of adaptability to researchers without the need of any programming experience and pursues maximum usability through user-friendly interfaces. The system is easily deployed and globally accessible by means of a web service. The platform's responsibilities range from the generation of stimuli and the setup and execution of test scenarios to the management of test results.**

## I. INTRODUCTION

Psychoacoustics studies subjective human perception of sounds. Because of this subjective sensation, assessing sound perception is only possible through experiments with humans. This raises the need for an experimental setup that allows testing subjects in a controlled way so that the results can be reproduced. This experimental setup should preferably be versatile and user-friendly at the same time [1]. Psychoacoustic testing is an important tool in the development of next-generation cochlear implants. The knowledge of human sound perception is used to improve speech and sound processing algorithms within cochlear implant devices. Until now various platforms have been developed to establish this kind of test infrastructure. But because of the tedious and repeating nature of the test scenarios and the limited portability of these systems, it has always been an intensive task for both patient and researcher to conduct the experiments in a controlled manner. By levering the test platform to a web based implementation, availability of the system is greatly increased while test results are centralized and easily accessible to researchers.

## II. TECHNOLOGY SURVEY

### A. PSYCHOACOUSTIC TESTS SOFTWARE

The most common psychoacoustic tests are identification and discrimination tasks. An identification test consists of the presentation of a series of stimuli. The subject tries to identify each stimulus. The subject's responses to the stimuli result in a confusion matrix. Each entry of such a matrix indicates the number of times a particular stimulus evoked a particular response. This matrix illustrates the subject's identification capabilities. A discrimination task searches the threshold of the subject's ability to discriminate two stimuli. Each trial a series of stimuli is presented out of which the subject tries to pick the differing one. The discrimination test results in a psychometric curve converging to the subject's threshold (see Figure 1). Other psychoacoustic tests include balancing and ranking tasks.

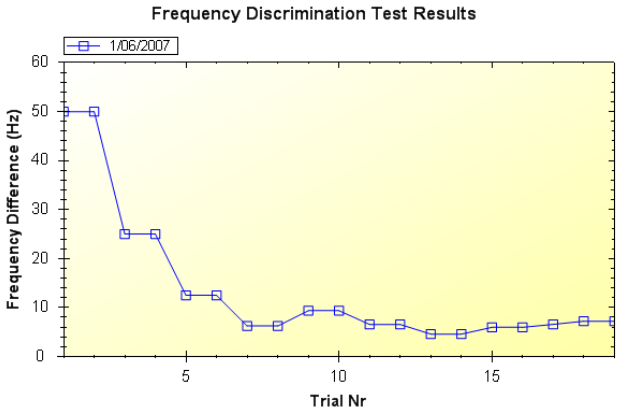


**Fig. 1:** Output of an adaptive discrimination task

A number of applications providing psychoacoustic tests exist. APEX is a research tool for conducting psychophysical experiments. It was developed at the Catholic University of Leuven. The APEX personal computer software reads a text file which specifies the experiment and the stimuli, controls the experiment, delivers the stimuli to the subject through a digital signal processor board, collects the responses via a computer mouse or a graphics tablet, and writes the results to the same file [2]. PACTS (Psychoacoustic Test System) is another framework for conducting psychophysical experiments and was developed at the University of Antwerp. It contains adaptive threshold procedures, ranking, balancing and closed-set identification experiments, with support for both acoustic and direct electrical output. PACTS is an XML based Windows application suffering from laborious deployment and versioning issues. PACTSweb is the successor of PACTS[1] and differs from the latter in its web based implementation and the fact that it makes use of an RDBMS for storing all data. Moreover PACTSweb is a completely newly designed system to maximize functionality, usability, reliability, performance, supportability and security.

### B. WEB APPLICATION DESIGN

Modern web applications that have the features and functionality of traditional desktop applications typically transfer the processing necessary for the user interface to the web client but keep the bulk of the data (i.e. maintaining the state of the program, the data, etc.) back on the application server [3]. They run in the secure environment

[1]PACTSweb uses the psychoacoustics engine written by Filiep Vanpoucke, author and creator of PACTS.

of a web browser and do not require software installation. PACTSweb is designed according to this paradigm. All business logic and data storage is located at the server. Business logic is implemented in Microsoft .Net assemblies and Microsoft SQL Server technology is used for data storage. Presentation logic is realized through ASP.NET 2.0 web pages, HTML, JavaScript and Adobe Flash. User interfaces are loosely coupled to the business logic and all components are designed keeping maintainability and adaptability in mind.

## III. SYSTEM SPECIFICATIONS

### A. *FUNCTIONALITY*

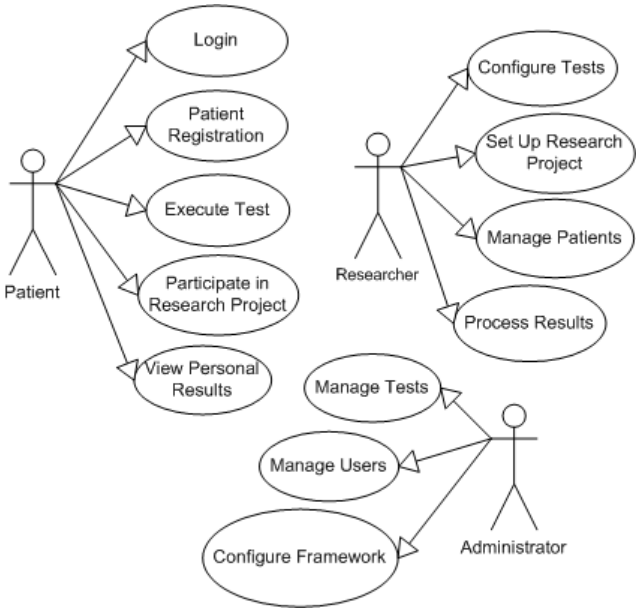


**Fig. 2:** Use Case model

Patients make use of a web interface through their internet browser to view their personal data and test results. An overview page lists all available tests and research projects the patient can participate in. Patients can execute tests for personal revalidation or by participation in a research project. Every user has to register and log in to make use of PACTSweb. Researchers can set up new projects which contain a series of psychoacoustic tests offered to a set of patients during a period of time. They have the ability to view and process gathered test results. An administrator is able to define new types of psychoacoustic tests and alter existing test procedures. He also has the authority to configure general and technical features of the framework. Musical instrument identification tests and a frequency discrimination tests are already built in the framework. Researchers can set up specific instances of these test types and offer them to patients. Other test types can easily be defined and instantiated in the framework. A use case diagram of the global functional requirements is shown in Figure 2.

### B. *ACCURACY AND PERFORMANCE*

For psychoacoustic tests to produce meaningful results, a number of aspects need to be taken into consideration. Stimulus creation and presentation should be reproducible in order for the test to maintain its value. To meet this requirement PACTSweb uses lossless coding algorithms to generate and process audio signals. In addition the context in which the subject is presented with the stimuli should be predictable and independent of external coincidences. By its web based constitution PACTSweb inevitably runs into some serious issues here. The internet, by its packet switched implementation, cannot guarantee any foreseeable throughput or delay. In psychoacoustics various temporal conditions lead to different perceptions [4]. It is for that reason an accurate timing is essential when a sequence of interrelated stimuli is presented (e.g. during a discrimination task the pause between stimuli cannot depend on accidental network congestions). Furthermore, due to the disconnected nature of web browsers and the HTTP protocol, once the server has sent his response to the client, it is in essence impossible to determine exactly what is happening at the client side. A variety of web browsers and system configurations exist to make a uniform and anticipated execution of test procedures even harder. To overcome these obstacles PACTSweb uses Adobe Flash[2]. The Flash player is a widely distributed multimedia engine for web based applications and is supported by all major browsers and operating systems. It offers a solution to the timing issue by its exact frame based playback system and has the ability to preload contents in a controlled way so lag by network instability can be avoided.

## IV. APPLICATION ARCHITECTURE

Data access logic, business logic and presentation logic components are implemented in loosely coupled layers.

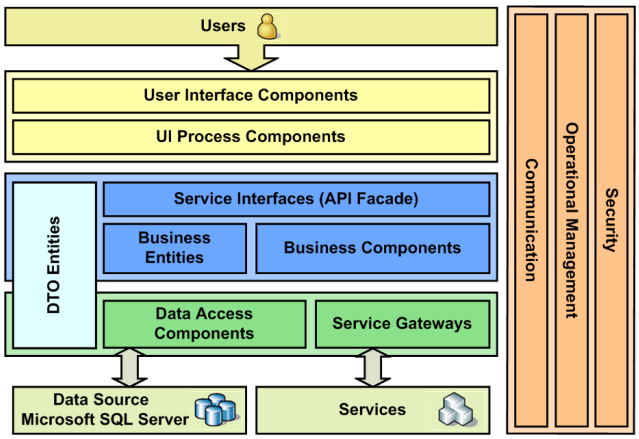


**Fig. 3:** Global application architecture

Description of the components depicted in Figure 3:

- **User interface components:** ASP.NET pages enabling the user to interact with the application. They are responsible for data formatting, producing output to the user and receiving and validating input from the user.
- **User process components:** User interaction often follows a predictable process. To manage and synchronize user actions this process logic is contained in specific user process components. They define the order of possible actions and keep track of the current state of the user process. This way this logic is not contained in the user interface components themselves and the same user interface engine can be used to serve multiple user interfaces.
- **Business components:** PACTSweb requires a number of components that implement business rules, tasks en logic. Their responsibility is to transform and transport data from and to adjacent layers in a manner conforming the test scenario's business logic. These data could be originated from the database or from other services like a stimulus generator.
- **Service gateways:** When business components need functionality not inherent to the business logic they contain, they will address external services. To make sure this happens in a controlled manner they will act on service agents. These components act as translating entities to manage communication between both ends.
- **Service interfaces:** In order to make the application core as reusable as possible, it is important to expose its functionality

[2]Adobe Flash, formerly Macromedia Flash, a vector based multimedia and application player suitable for creating rich internet applications and streaming video and audio.

through a uniform interface towards different objects willing to make use of this core functionality. Service interfaces define contracts concerning communication, formatting, protocols, security and exception handling.

- **Data access logic components:** The data access layer transforms data manipulation and retrieval requests from business components to database specific commands. This way data storage and access is implemented independently from the business logic, thus improving adaptability and portability.
- **DTO entity components:** Data Transfer Objects encapsulate data in the application. They represent the object oriented equivalent of the data store and are used to transport data trough different layers. The DAL is responsible for filling these entities with data retrieved from the database and persisting any changes made to the data they contain back to the database.
- **Business entity components:** These components represent real world concepts of the application domain. They contain specific business oriented data and logic and are used to communicate among business components. These complex entities are as opposed to the DTO objects not exposed outside the business logic layer.
- **Security Policy:** Security policies should be consistently enforced on each layer. Security policies realize concepts of authentication, authorization, secure communication, auditing and profile management.
- **Operational management Policy:** Realizes tasks of configuration, management of meta data, error handling, monitoring, etc. These policies should also be enforced throughout the system, adaptively implemented into each layer.
- **Communication Policy:** Defines rules for the different components of the application to communicate with each other. It realizes things like synchronicity, formats and protocols.

## V. DATA STORAGE AND ACCESS

While the existing PACTS application stores both experiment design and test results in XML files, PACTSweb aims at centralizing all data in a single relational database store built on Microsoft technology. SQL Server 2005 Express Edition is a fairly small footprint free database engine service that can be installed on any current desktop or server Windows operating system and addresses all functionality PACTSweb requires: it ensures overall data consistency while dealing with concurrency, replication, security and performance issues.

### A. *DATABASE DESIGN*

All core data for PACTSweb to function properly is accommodated in a relational model of 35 tables. Each table has been normalized [5] to a degree where data consistency is being enforced by foreign keys and check constraints. All necessary CRUD[3] functionality is exposed through parameterized stored procedures. Table data contain definitions of test procedures and stimuli, patient data, test results and management metadata. A number of views supports retrieving meaningful results and reports from the database. As throughout the application the main design goals are scalability and adaptability. Hence all test procedures are defined as scenarios with a variable set of parameter definitions. To allow such flexible scheme to be reflected in the static structure of a relational model, parameter definitions are stored in a separate table and linked to the scenario they apply to and to the data type they represent. Scenarios are managed by administrators and instantiated by researchers into test setups by assigning values to their parameters. Again these parameters are stored in a separate table and constrained by their corresponding definition. The scenario for closed set identification can be instantiated into a setup for musical instrument identification for example, as well as it can be set up for a speech recognition test. Similarly stimuli are defined by their characteristic features (amplitude, phase frequency, duration, ramp up and ramp down) which can be extended to any set of parameters (e.g. an ADSR[4] envelope or SNR value). Moreover stimuli can either be loaded statically (from binary data in the database or from a path to a file on disk) or generated dynamically at runtime. Either way both acoustic and electrical signals can be defined. Dynamic stimuli use different signal type identifiers (e.g. pure tone or narrow band noise) to be generated correctly while static stimuli are assigned to groups like speech or music to get selected into a particular test setup's stimulus pool. The model for persisting stimuli is illustrated in Figure 4.

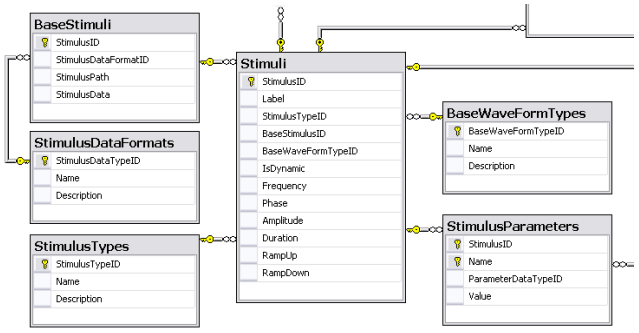


**Fig. 4:** Extract from the database schema

### B. *DATA ACCESS LAYER*

A common problem in designing data based applications is the object-relational (O/R) gap between the scalar values contained in tables of the relational database model and the object oriented programming environment in which data is encapsulated in complex logical entities having their own methods and properties to validate, manipulate and expose these data. PACTSweb fills this gap through the use of .netTiers, a set of templates for use with the CodeSmith generator. Many O/R mappers and DAL code generators exist, but .netTiers with its feature-rich enterprise architecture, web application oriented utilities and Microsoft driven implementation makes a fit data access layer for PACTSweb. Consequently the DAL base architecture is built upon the Microsoft Enterprise Library Application Blocks, a set of libraries that provide proven solutions to common development challenges such as data access, logging, caching and user interface processes. By implementing these patterns the DAL inhibits following features:

- Database implementation independent data access through pluggable providers following the ADO.NET 2.0 provider model.
- An entity provider (Table Module) for each table and view in the database, responsible for exposing CRUD operations on its particular underlying structure and hydrating entity objects.
- Generic data access by executing custom SQL statements against Microsoft's Data Access Application Block.
- A central mechanism, called the DataRepository, for instantiating and loading different providers at runtime and retrieving and saving data. The DataRepository is essentially a singleton facade object into the DAL exposing the individual entity providers.
- An optimistic concurrency mechanism based on timestamp columns in tables.
- SQL injection detection on generic data access methods.
- A transaction manager supporting all common isolation levels to ensure data consistency.

[3]Create, Read, Update and Delete: the four basic functions of persistent storage.

[4]Attack, Decay, Sustain, Release: describing changes in a sound over time, including alterations in a sound's amplitude, frequency and timbre.

### C. DTO ENTITY LAYER

While the DAL's responsibility consists of retrieving and manipulating data from and to the underlying data store, another level of abstraction is needed to bridge the O/R gap. For that reason, the data originating from the DAL are encapsulated into entity objects that are used throughout the application. Each entity is a significant type (e.g. a stimulus, a patient or a test setup) that is essential for the processing of data in the business layer and the presentation of these data through the user interface. The entity framework maps the relational data coming from the database to a logical object graph for use in an object oriented environment. For example a test scenario entity will hold, in addition to its own properties, an administrator entity identifying its designer and (if deep loaded by the DAL) a collection of test setups that have been instantiated from it. PACTSweb combines the Table Module and Data Transfer Object (DTO) patterns [6] to realize lightweight entities that are able to travel through the many tiers while still maintaining a loosely coupled entity layer, independent on any data provider. While PACTSweb entities do not contain real business logic they do keep functionality other than just holding data to be transported across layers:

- All entities inherit from the EnityBase class in which the entity's lifecycle is defined. This lifecycle differs from the .Net CLR object lifecycle and is tracked through the EntityState property which holds the entity's current status. Whenever an entity is created, retrieved from the database, deleted or one of its properties has been modified, its state will automatically change to reflect the new status (Added, Unchanged, Deleted or Changed respectively). This way software components handling these entities do not have to concern themselves about tracking changes. Calling the corresponding provider's Save method in the DAL suffices to persist all changes made since the entity was retrieved from the database.
- By implementing the IEditableObject, IComponent, INotifyPropertyChanged and IDataErrorInfo interfaces from the System.ComponentModel namespace, each entity can be treated as a .Net Component which gives them design-time support. As a consequence they have the ability to be added to the toolbox of Visual Studio, be dragged and dropped onto a form and be manipulated on a design surface or used as databindable object datasources.
- A validation rule engine providing a mechanism to validate entities against a set of rules able to contain virtually any validation logic by means of using custom delegates.
- The entity layer holds a number of optimization constructs. The EntityCache wraps the Enterprise Library Caching Block which can be configured to cache entities instead of going down to the database with each query. The EntityManager implements an entity factory and a weak referenced object store, thus enabling a tracking system. In case the application is handling a high volume of entities, which many are of the same record, this tracking system returns the same entity object for all references until that entity is persisted to the DataRepository.
- The handling of a collection of entities, be it for processing by business logic or formatting towards user interfaces, occurs by means of a specialized list called the TList. The TList is a full featured collection of entity objects. It makes full use of the new generics feature of the .Net 2.0 framework and has the advantage over arrays by its dynamic capacity and over ArrayLists and other Collections by its strongly typed nature. A TList inherits from System.ComponentModel.BindingList and implements the IBindingListView, IBindingList, IList, ICloneable, IListSource, ITypedList, IDisposable, IComponent, IRaiseItemChangedEvents and IDeserializationCallback interfaces, giving it many of the functionality that applies to entities, transferred to the level of the enclosing list. Tlists exhibit advanced filter, sort, shuffle and find algorithms, can also be used as typed datasources for databinding and have the ability to expose their content as an array or a DataSet. As all entities are serializable to native binary code and to XML, TLists can also be serialized by using the EntityHelper class. On serialization of a TList, its entire contents, including all referenced child entities in its entity nodes will be serialized into one corresponding file or stream.

## VI. BUSINESS LOGIC

### A. PLATFORM MANAGEMENT

The intention of PACTSweb is to offer a remotely accessible platform for setting up and executing psychoacoustic experiments. One variety of target users includes cochlear implant wearers wishing to occasionally examine their current progress of revalidation by carrying out individual tests and comparing their accumulated results through time. Another type of users consists of research teams wishing to inspect the effect of different speech and sound coding algorithms on the auditory performance of human subjects. For this kind of usage a more structured way of offering experiments is at hand. Research teams have the ability to create projects containing a set of related experiments. These projects are aimed at a selected group of patients and can be offered during a predefined period of time. To increase privacy, a patient's personal data are only accessible to himself after successfully logging in to the system. When performing a test only relevant data, like the configuration of devices (i.e. cochlear implants) the subject is using during participation in the test, will be collected. Each research team can manage their patients by assigning them to groups. A patient can belong to several groups and multiple groups can be assigned to a project. Projects are by default inaccessible to users outside its target audience. However, the researcher owning the project has the ability to request making his project publicly available by setting a flag. A PACTSweb administrator then has the choice to comply with this request or to refuse it. The same logic applies to individual test setups. Researchers are able to configure their own experiments and subsequently have the option of making them public. This way test setups and results can be shared among different research groups. The actual management functions are handled in software by the ProjectManager and UserManager classes, two controllers responsible for servicing the ASP.NET researcher and administrator pages. They expose an API into both business logic and data layer and implement the necessary authorization rules. Please refer to Figure 6 to get an overview of the internal workings of the PACTSweb software system.

### B. EXPERIMENT DESIGN

Since PACTSweb is designed to maximize scalability and adaptability, experiments are defined as test scenarios that can be configured to pursue satisfaction of a researcher's divergent needs. Currently PACTSweb hold 2 scenarios: a closed set identification task definition and an adaptive discrimination procedure definition. It is the responsibility of PACTSweb administrators to manage and maintain these scenarios. They are able to add new scenarios provided that they carry through the logic needed to process these scenarios correctly. This task requires a certain knowledge of the internal workings of the framework and the programming skills to implement the logic. Along with scenario creation a set of parameter definitions is assigned to it. This set can contain virtually any aspect of the scenario the administrator prefers to make variable. When test setups are instantiated from a scenario, values are assigned to its parameters. In order to effectuate these values during a test, it is important that each parameter is processed correctly. This means that every parameter definition should come with a processing component containing the necessary algorithm logics. When running a test, its setup's parameters are parsed and processed by the scenario's logic.

This approach allows researchers to setup a speech pitch detection test [7] using the same scenario as when configuring an adaptive frequency discrimination task (see Figure 5 for an example of possible output with visual feedback enabled).


**Fig. 5:** Part of the discrimination test user interface

For orchestrating the execution of a test procedure controller classes expose simple methods to the UI components. These classes inherit from the TestManager base and combine the logic needed from the Audio, Psychoacoustics and Data namespaces to instruct the FlashBuilder to generate the desired UI component. TestManagers maintain state between requests and ensure a scenario is run through correctly.

### C. STIMULUS CREATION

An essential part of the system involves the creation and handling of stimuli. Because of the limited knowledge of the exact workings of the human brain and in particular its perception of sound, the only way to measure the auditory capabilities of human beings is to subject them to psychophysical tests [8]. These tests all follow a basic common pattern: a stimulus is presented and the subject's response to this stimulus is recorded and evaluated. The stimulus-response pair, and in some cases the context in which the stimulus was presented, is the only information researchers have to carry out their analysis. For that reason it is of utmost importance that stimuli are created accurately and can be reproduced while analyzing output of psychophysical experiments [9]. When it comes to acoustic stimuli, the signal presented is a sound wave. For this sound wave to be presented to the subject exactly as it is intended, PACTSweb uses standard PCM encoded WAV data. Despite all modern audio coding techniques, there is no way to achieve a significant compression without loss of information. This loss of information, even though its consequences are often inaudible to the average human auditory system (e.g. a 320 kbps MP3 encoding of an everyday piece of music), is unacceptable in the field of psychoacoustics. After all many of these popular codecs themselves rely on psychoacoustic phenomena to achieve the compression ratios that they proclaim [10]. The use of these algorithms would lead to an unpredictable construction of the stimulus for presentation and no way to re-construct the originally intended signal. Although its web based nature craves some sort of audio compression, it is for that reason PACTSweb generates and processes audio signals in 44100 Hz 16bit uncompressed PCM encoding by default. The software components responsible for signal generation and processing are located in the Audio namespace which is, as all packages, built into a separate assembly. Internally the audio namespace uses RIFF parsers to read WAV data from disk and load them into its own AudioSignal class, basically a wrapper around Microsoft DirectSound's WaveFormat. Signals generated at runtime are created by classes implementing the IGenerator interface (e.g. a sine wave generator or a noise generator). All signals, generated or loaded from disk, can be post processed by IProcessor implementations for altering specific characteristics.

### D. PSYCHOACOUSTICS

The Psychoacoustics namespace contains the core logic of psychophysical experiment procedures. A number of classes provide algorithmic constructs that represent well-known psychophysical strategies. The Rover randomly scales amplitude in a certain dynamic range. The StairCaseSeeker implements an adaptive threshold seeking algorithm using either a continuous or discrete iteration variable and the TrialSequencer is used for determining the sequence of trials to be presented to the subject.

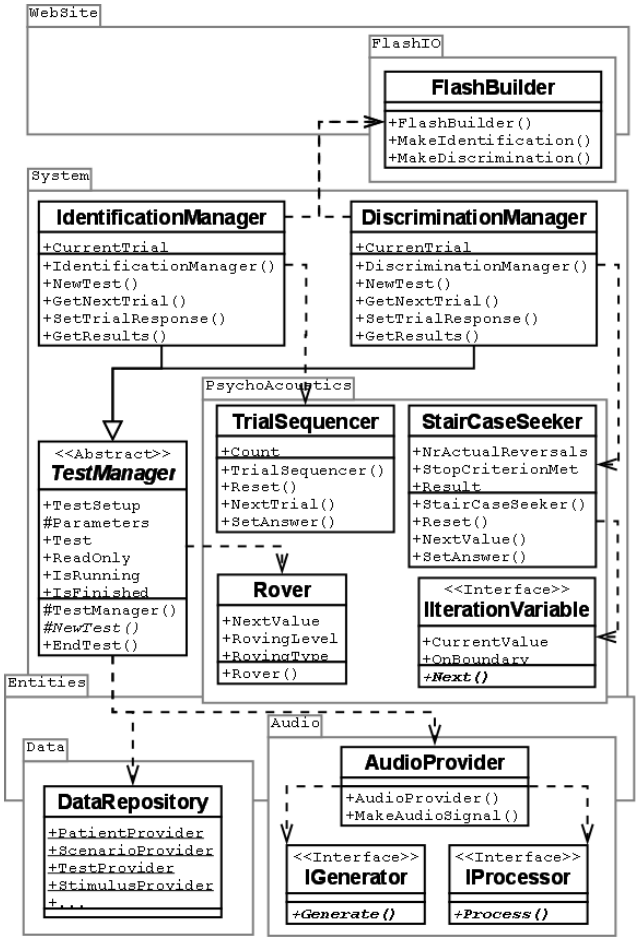


**Fig. 6:** Simplified class diagram of the system core

## VII. PRESENTATION

To be globally accessible, it is crucial for the user interface to be supported by all major browsers and operating systems. However, the only thing all web browsers understand is basic HTML (and most often also JavaScript). This fact imposes major restrictions on the design of the presentation layer. Also a careful consideration should be made which logic to run on the server and which is to be executed on the client. Moving too much processing to the server leads to a stiffer user experience caused by the overhead of posting back every interaction to the server, reducing the application responsiveness. On the other hand charging the client with increased responsibility demands more application logic to comply with the limited execution environment of a web browser which inherently results in a design of lesser adaptability and limited scalability.

### A. IIS AND ASP.NET 2.0

To expose its functionality to remote clients PACTSweb uses the ASP.NET 2.0 framework. It offers the infrastructure to build dynamic web content and lies within the scope of the technology used. PACTSweb makes use of a number of built-in features and server controls. The ASP.NET membership provider is adopted to handle

authentication and authorization. All metadata for this role based security model is stored in a separate schema on the database server. A site map, skins, themes, and master pages administer a uniform layout and user experience across the web site. For displaying test results graphically, PACTSweb uses ZedGraph, a library written in C# for creating flexible 2D line and bar graphs (see Figure 7).

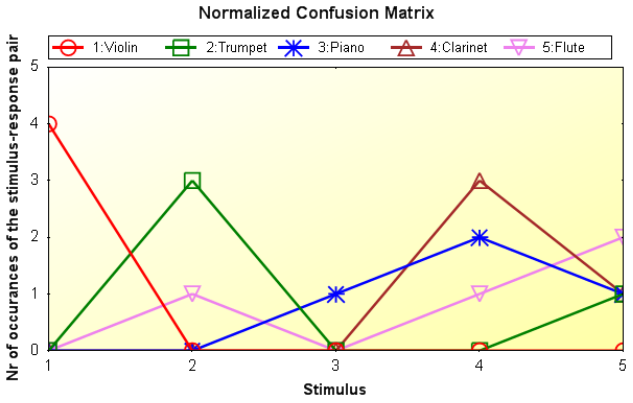


**Fig. 7:** Output of a closed set identification

ASP.NET applications run within IIS, Microsoft's web server, which processes requests and basically generates HTML pages according to the application's logic to be sent back to the client. This means that while all the benefits of server side scripting and processing result in greatly enhanced dynamics, the client, receiving plain old HTML, is totally unaware of this. Consequently it is the server's task to maintain state of the application on a per-user basis. PACTSweb accomplishes this by using a combination of viewstate and querystrings and saving the TestManager object responsible for orchestrating a test scenario to the session object after each request.

### B. AUDIO ON THE WEB

Despite all technological progress in the field of web based applications, services and multimedia in particular, web browsers still remain pretty archaic programs when it comes to audio playback. The web browser's ability (or better: the lack of it), to play audio in a controlled manner, revealed itself to be one of the major challenges during the development of PACTSweb. Web browsers basically remain ordinary HTML parsers, they have no audio processing capabilities. Every attempt to play audio in a web page results in some kind of plug-in or external helper application being loaded to handle the actual playback. This means an immense variety of browser-plug-in configurations is present with today's internet users. To start with, Internet Explorer's Active X plug-in system is incompatible with the system Firefox is based on. As being two of the most popular browsers this incompatibility cannot be ignored. Common audio playback engines like Windows Media Player, Quicktime, and RealPlayer all have their specific implementation for either of the major browsers. Now as long as implementation is independent of their behavior this should not pose a real problem. However it does mean that users must first install the correct plug-in. Often this leads to an undesired and uncomfortable user experience[5]. Different techniques using JavaScript and dynamic HTML to control these audio players have been tested. None of them offered the functionality PACTSweb needs.

### C. ADOBE FLASH PLAYER

The Flash player features a number of advantages if compared to previously discussed audio engines. Its omnipresence being one of the most noteworthy. Flash Player is installed on 98% of Internet-enabled desktops worldwide. Also it offers rich multimedia and vector based animation possibilities and is equipped with its own ActionScript programming model, while at the same time it is extremely lightweight and loads swiftly. The drawback however is that Flash player uses its own precompiled proprietary format. It is impossible to feed the player a plain audio file. Generally, to develop Flash applications a specialized and often commercial IDE (Integrated Development Environment, e.g. Adobe Flash Professional) is used, where the application design and source code are compiled into an SWF file. Using the object oriented ActionScript in combination with the SWLiveConnect interface highly customizable multimedia applications can be created. By passing variables into the Flash player from the surrounding web page environment at runtime, a precompiled SWF file's behavior can be controlled to a high extent. Regardless of all these features Macromedia (nowadays Adobe) neglected to incorporate the ability to load uncompressed audio at runtime. Flash player only allows this for MP3, other formats can only be used if precompiled into the SWF file. This shortcoming makes the use of Flash in its traditional way unsatisfactory for the purpose of presenting uncompressed, dynamically generated audio.

### D. JAVA APPLETS

After all this adversity in search for a suited audio engine, one tends to fall back to programming Java Applets. Indeed this alternative exhibits the freedom of implementing any custom logic and behavior as long as it fits into the sandbox or is approved by the user. However, the availability of Java's Virtual Machine in today's web browsers is considerably lower compared to the Flash player's. Its startup time is significantly higher and the JVM has the reputation of being not the most stable runtime environment when hosted in a web browser. On top of that, the standard JRE features only a minimal audio API, forcing the user to download additional packages when the application needs to handle audio in a controlled way. These aspects in combination with the fact that the Java platform is not really the most friendly environment to develop customizable, animated user interfaces against, justify Java Applets only as a last resort for realizing PACTSweb's presentation logic.

### E. RUNTIME SWF GENERATION

Were it not for its inability to load uncompressed audio at runtime, the Flash player would be the perfect tool to implement PACTSweb's user interfaces. That's why a number of strategies have been looked into to work around this shortcoming. A technique to generate raw WAV data from within a flash application exists and is based on the new ActionScript 3.0 model. Its Loader and ByteArray classes allow for SWF bytecode to be created in memory. The Flash application itself would then be able to create and dump raw audio samples in an empty SWF skeleton. Thereupon this in-memory construct could be accessed and controlled by the initial flash application that created it. This approach however, raises some questions. The responsibility for audio creation would be moved to the client side and pre-recorded audio signals would not be handled by this system. Additionally ActionScript 3 along with the new IDE from Adobe were still in beta release during development of PACTSweb. To avoid using another commercial IDE and the ActionScript language all together, an alternative strategy has been conceived. PACTSweb now generates its own SWF bytecode at runtime, straight from a C# .Net assembly, omitting any compiler or external program. It is the FlashBuilder class in the FlashIO namespace that is responsible for this task. Internally it uses the SwfDotNet library which has been extended for use with PACTSweb. The FlashBuilder puts together the Flash application frame per frame, line per line and sample per sample, all conforming with the SWF file format specification. The SWF format is a binary tag based description of vector based graphics, transformations, multimedia and actions. All actions are stack based, meaning for every instruction the programmer needs to first push its arguments on the stack, and after execution pop the result

[5]Try running Windows Media Player embedded in Firefox

back off, and so on. This may sound like a laborious task in today's software development. Nonetheless this going down to the bytecode level opens up a vast amount of possibilities. PACTSweb now is able to generate any kind of signal and embed it in virtually any type of interface. Also this means the presentation logic is contained at the server, in the same environment as the rest of the application, which again results in an increased level of maintainability.

## VIII. CONCLUSIONS

With the web based implementation of a psychoacoustic test system, PACTSweb brings the infrastructure for conducting psychophysical experiments out of the closed environment of research centres. Patients are given the opportunity to test themselves and their progress during revalidation from within their own homes. For researchers to study the behaviour of certain sound processing algorithms and the perception of sound they induce with their patients the system is built to be highly configurable and extendable. This way a variety of procedures and interactive experiments can be set up to reflect the researcher's own methodology. PACTSweb maximizes control over the actual execution of an experiment through a frame based mechanism that allows for accurate timing and ensures reproducible presentation of stimuli. Results are gathered and persisted in a central database, giving researchers the ability to analyze and process them in an orderly manner. The output of experiments can easily be shared among researchers without compromising the patient's privacy. By offering both patient and researcher an infrastructure to investigate the experience of sound, PACTSweb's aspiration is to support and stimulate the development of next-generation cochlear implants.

## REFERENCES

[1] Laneau and Boets, "A flexible auditory research platform using acoustic or electric stimuli for adults and young children," 2004.
[2] Geurts and Wouters, "A concept for a research tool for experiments with cochlear implant users." 2000.
[3] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture."
[4] D. Nelson and Swain, "Temporal resolution within the upper accessory excitation of a masker," 1996.
[5] E. Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, 1970.
[6] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*. Prentice Hall PTR, 2001.
[7] Laneau, "When the deaf listen to music: pitch perception with cochlear implants." 2005.
[8] B. Edwards, "Signal processing, hearing aid design and the psychoacoustical turing test," *International Conference on Acoustics, Speech, and Signal Processing*, 2002.
[9] Cabrera, "Psysound:a computer program for psychoacoustical analysis," 1999.
[10] E. Larsen and R. Aarts, "Audio bandwidth extension. application of psychoacoustics, signal processing and loudspeaker design." 2004.

If you wish to cite this paper, please use following code:
Bart Vaerenberg, Eric Verschooten and Peter Bracke, Web Based Psychoacoustic Research On Cochlear Implant Patients, Masters Thesis, Department of Industrial Sciences and Technology, University College of Antwerpen, Belgium, July 2007