Web Audio Evaluation Tool: A framework for subjective assessment of audio

Nicholas Jillings[2]
nicholas.jillings@mail.bcu.ac.uk

Brecht De Man[1]
b.deman@qmul.ac.uk

David Moffat[1]
d.j.moffat@qmul.ac.uk

Joshua D. Reiss[1]
joshua.reiss@qmul.ac.uk

Ryan Stables[2]
ryan.stables@bcu.ac.uk

Centre for Digital Music, School of Electronic Engineering and Computer Science[1]
Queen Mary University of London
Mile End Road, London E1 4NS
United Kingdom

Digital Media Technology Lab[2]
Birmingham City University
Birmingham B4 7XG
United Kingdom

ABSTRACT

Perceptual listening tests are commonplace in audio research and a vital form of evaluation. Many tools exist to run such tests, however many operate one test type and are therefore limited whilst most require proprietary software. Using Web Audio the Web Audio Evaluation Tool (WAET) addresses these concerns by having one toolbox which can be configured to run many different tests, perform it through a web browser and without needing proprietary software or computer programming knowledge. In this paper the role of the Web Audio API in giving WAET key functionalities are shown. The paper also highlights less common features, available to web based tools, such as easy remote testing environment and in-browser analytics.

1. INTRODUCTION

Perceptual evaluation of audio, in the form of listening tests, is a powerful way to assess anything from audio codec quality to realism of sound synthesis to the performance of source separation, automated music production and other auditory evaluations. In less technical areas, the framework of a listening test can be used to measure emotional response to music or test cognitive abilities.

Several applications for performing perceptual listening tests currently exist. A review of existing listening test frameworks was undertaken and presented in Table 1. Note that many rely on proprietary, 3rd party software such as MATLAB and MAX, making them less attractive for many. With the exception of the existing JavaScript-based tool-

boxes, remote deployment (web-based test hosting and result collection) is not possible.

HULTI-GEN [1] is a single example of a toolbox that presents the user with a large number of different test interfaces and allows for customisation of each test interface, without requiring knowledge of any programming language. The Web Audio Evaluation Toolbox (WAET), presented here, stands out as it does not require proprietary software or a specific platform. It also provides a wide range of interface and test types in one user friendly environment. Furthermore any test based on the default test types can be configured in the browser as well. Note that the design of an effective listening test further poses many challenges unrelated to interface design, which are beyond the scope of this paper [2].

The Web Audio API provides important features including sample level manipulation of audio streams [3] and synchronous and flexible playback. Being in the browser allows leveraging the flexible object oriented JavaScript language and native support for web documents, such as the extensible markup language (XML) which is used for configuration and test result files. Using the web also reduces deployment requirements to a basic web server with extra functionality, such as test collection and automatic processing, using PHP. As recruiting participants can be very time-consuming, and as for some tests a large number of participants is needed, browser-based tests can enable participants in multiple locations to perform the test [3].

Both BeaqleJS [4] and mushraJS[1] also operate in the browser. However, BeaqleJS does not make use of the Web Audio API and therefore lacks arbitrary manipulation of audio stream samples, and neither offer an adequately wide choice of test designs for them to be useful to many researchers.

To meet the need for a cross-platform, versatile and easy-to-use listening test tool, we previously developed the Web Audio Evaluation Tool [9] which at the time of its inception was capable of running a listening test in the browser from

---

[1]https://github.com/akaroice/mushraJS

Table 1: Table with existing listening test platforms and their features

Toolbox	APE	BeaqleJS	HULTI-GEN	mushraJS	MUSHRAM	Scale	WhisPER	WAET
Reference	[5]	[4]	[1]		[6]	[7]	[8]	[9]
Language	MATLAB	JS	MAX	JS	MATLAB	MATLAB	MATLAB	JS
Remote		(✓)		✓				✓
MUSHRA (ITU-R BS. 1534)		✓	✓	✓	✓			✓
APE	✓							✓
Rank Scale			✓					✓
Likert Scale			✓				✓	✓
ABC/HR (ITU-R BS. 1116)			✓					✓
-50 to 50 Bipolar with ref.			✓					✓
Absolute Category Rating Scale			✓					✓
Degradation Category Rating Scale			✓					✓
Comparison Category Rating Scale			✓				✓	✓
9 Point Hedonic Category Rating Scale			✓				✓	✓
ITU-R 5 Continuous Impairment Scale			✓					✓
Pairwise / AB Test			✓					✓
Multi-attribute ratings			✓					✓
ABX Test		✓	✓					✓
Adaptive psychophysical methods							✓	
Repertory Grid Technique							✓	
Semantic Differential						✓	✓	✓
n-Alternative Forced Choice						✓		

Figure 1: A simple example of a multi-stimulus, single attribute, single rating scale test with a reference and comment fields.

an XML configuration file, and storing an XML file as well, with one particular interface. This has now expanded into a tool with which a wide range of listening test types can easily be constructed and set up remotely, without any need for manually altering code or configuration files, and allows visualisation of the collected results in the browser. In this paper, we discuss these different aspects and explore which future improvements would be possible.

2. ARCHITECTURE

Although WAET uses a sparse subset of the Web Audio API functionality, its performance comes directly from it. Listening tests can convey large amounts of information other than obtaining the perceptual relationship between the audio fragments. With WAET it is possible to track which parts of the audio fragments were listened to and when, at what point in the audio stream the participant switched to a different fragment, and how a fragment's rating was adjusted over time within a session, to name a few. Not only does this allow evaluation of a wealth of perceptual aspects, but it also helps detect poor participants whose results are potentially not representative.

One of the key initial design parameters for WAET was to make the tool as open as possible to non-programmers and to this end all of the user modifiable options are included in a single XML document. This document is the specification document and can be designed either by manually writing the XML (or modifying an existing document or template) or using the included test creator. These standalone HTML pages do not require any server or internet connection and help a build the specification document. The first (test_create.html) is for simple tests and operates step-by-step to guide the user through a drag and drop, clutter free interface. The advanced version is for more complex tests. Both models support automatic verification to ensure the XML file is valid and will highlight areas which are either incorrect and would cause an error, or options which should be removed as they are blank.

The basic test creator, Figure **??**, utilises the Web Audio API to perform quick playback checks and also allows for loudness normalisation techniques inspired from [5]. These are calculated offline by accessing the raw audio samples exposed from the buffer before being applied to the audio element as a gain attribute. Therefore the tool performs loudness normalisation without editing any audio files. Equally the gain attribute can be modified in either editor using an HTML5 slider or number box respectively.

The specification document contains the URL of the audio fragments for each test page. These fragments are down-

loaded asynchronously in the test and decoded offline by the Web Audio offline decoder. The resulting buffers are assigned to a custom Audio Objects node which tracks the fragment buffer, the playback *bufferSourceNode*, other specification attributes including its unique test ID, the interface object(s) associated with the fragment and any metric or data collection objects. The Audio Object is controlled by an over-arching custom Audio Context node (not to be confused with the Web Audio Context). This parent JS Node allows for session wide control of the Audio Objects including starting and stopping playback of specific nodes.

The only issue with this model is the *bufferNode* in the Web Audio API, implemented in the standard as a 'use once' object. Once this has been played, the node must be discarded as it cannot be instructed to play the same *bufferSourceNode* again. Therefore on each play request the buffer object must be created and then linked with the stored *bufferSourceNode*. This is an odd behaviour for such a simple object which has no alternative except to use the HTML5 audio element. However, they do not have the ability to synchronously start on a given time and therefore not suited.

In the test, each buffer node is connected to a gain node which will operate at the level determined by the specification document. Therefore it is possible to perform a 'Method of Adjustment' test where an interface could directly manipulate these gain nodes. These gain nodes are used for cross-fading between samples when operating in synchronous playback. Cross-fading can either be fade-out fade-in or a true cross-fade. There is also an optional 'Master Volume' slider which can be shown on the test GUI. This slider modifies a gain node before the destination node. This slider can also be monitored and therefore its data tracked providing extra validation. This is not indicative of the final volume exiting the speakers and therefore its use should only be considered in a lab environment to ensure proper usage.

The media files supported depend on the browser level support for the initial decoding of information and is the same as the browser support for the HTML5 audio element. The most widely supported media file is the wave (.WAV) format which is accepted by every browser supporting the Web Audio API. The toolbox will work in any browser which supports the Web Audio API.

All the collected session data is returned in an XML document structured similarly to the configuration document, where test pages contain the audio elements with their trace collection, results, comments and any other interface-specific data points.

3. REMOTE TESTS

If the experimenter is willing to trade some degree of control for a higher number of participants, the test can be hosted on a public web server so that participants can take part remotely. This way, a link can be shared widely in the hope of attracting a large amount of subjects, while listening conditions and subject reliability may be less ideal. However, a sound system calibration page and a wide range of metrics logged during the test mitigate these problems. In some experiments, it may be preferred that the subject has a 'real life', familiar listening set-up, for instance when perceived quality differences on everyday sound systems are investigated. Furthermore, a fully browser-based test, where the collection of the results is automatic, is more efficient and technically reliable even when the test still takes place under lab conditions.

The following features allow easy and effective remote testing:

PHP script to collect result XML files and store on central server.

Randomly pick a specified number of pages to ensure an equal and randomised spread of the different pages ('audioHolders') across participants.

Calibration of the sound system (and participant) by a perceptual pre-test to gather information about the frequency response and speaker configuration - this can be supplemented with a survey.

Intermediate saves for tests which were interrupted or unfinished.

Collect IP address information for geographic location, through PHP function which grabs address and appends to XML file.

Collect Browser and Display information to the extent it is available and reliable.

4. INTERFACES

The purpose of this listening test framework is to allow any user the maximum flexibility to design a listening test for their exact application with minimum effort. To this end, a large range of standard listening test interfaces have been implemented.

To provide users with a flexible system, a large range of 'standard' listening test interfaces have been implemented, including:

- MUSHRA (ITU-R BS. 1534) [10]
- Rank Scale [11]: stimuli ranked on single horizontal scale, where they are ordered in preference order.
- Likert scale [12]: each stimuli has a five point scale with values: Strongly Agree, Agree, Neutral, Disagree and Strongly Disagree.
- ABC/HR (ITU-R BS. 1116) [13] (Mean Opinion Score: MOS): each stimulus has a continuous scale (5-1), labeled as Imperceptible, Perceptible but not annoying, slightly annoying, annoying, very annoying.
- -50 to 50 Bipolar with Ref: each stimulus has a continuous scale -50 to 50 with default values as 0 in middle and a reference.
- Absolute Category Rating (ACR) Scale [14]: Likert but labels are Bad, Poor, Fair, Good, Excellent
- Degradation Category Rating (DCR) Scale [14]: ABC & Likert but labels are (5) Inaudible, (4) Audible but not annoying, (3) slightly annoying, (2) annoying, (1) very annoying.
- Comparison Category Rating (CCR) Scale [14]: ACR & DCR but 7 point scale: Much Better, Better, Slightly Better, About the same, slightly worse, worse, much worse. There is also a provided reference.
- 9 Point Hedonic Category Rating Scale [15]: each stimuli has a seven point scale with values: Like Extremely, Like Very Much, Like Moderate, Like Slightly, Neither Like nor Dislike, dislike Extremely, dislike Very Much, dislike Moderate, dislike Slightly. There is also a provided reference.
- ITU-R 5 Point Continuous Impairment Scale [16]: Same as ABC/HR but with a reference.
- Pairwise Comparison (Better/Worse) [17]: every stimulus is rated as being either better or worse than the reference.

- APE style [5]: Multiple stimuli as points on a 2D plane for inter-sample rating (eg. Valence Arousal)
- AB Test [18]: Two stimuli presented at a time, participant selects a preferred stimulus.
- ABX Test [19]: Two stimuli are presented along with a reference and the participant has to select a preferred stimulus, often the closest to the reference.

It is possible to include any number of references, anchors, hidden references and hidden anchors into all of these listening test formats.

Because of the design to separate the core code and interface modules, it is possible for a 3rd party interface to be built with minimal effort. The repository includes documentation on which functions must be called and the specific functions they expect your interface to perform. The core includes an 'Interface' object which includes object prototypes for the on-page comment boxes (including those with radio or checkbox responses), start and stop buttons and the playhead / transport bars.

5. ANALYSIS AND DIAGNOSTICS

There are several benefits to providing basic analysis tools in the browser: they allow diagnosing problems, with the interface or with the test subject; they may be sufficient for many researchers' purposes; and test subjects may enjoy seeing an overview of their own results and/or results thus far at the end of their tests. For this reason, we include a
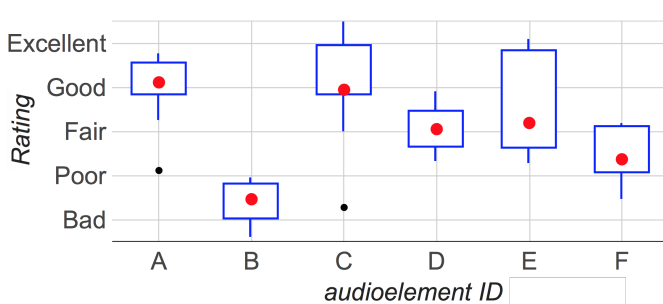
Figure 2: Box and whisker plot showing the aggregated numerical ratings of six stimuli by a group of subjects.

proof-of-concept web page with:
- All audioholder IDs, file names, subject IDs, audio element IDs, ... in the collected XMLs so far (`saves/*.xml`)
- Selection of subjects and/or test samples to zoom in on a subset of the data
- Embedded audio to hear corresponding test samples
- Scatter plot, confidence plot and box plot of rating values (see Figure)
- Timeline for a specific subject
- Distribution plots of any radio button and number questions in pre- and post-test survey
- All 'comments' on a specific audioelement
- A 'download' function for a CSV of ratings, survey responses and comments

6. CONCLUDING REMARKS AND FUTURE WORK

We have developed a browser-based tool for the design and deployment of listening tests, essentially requiring no programming experience and third party software. Following the predictions or guidelines in [3], it supports remote testing, cross-fading between audio streams, collecting information about the system, among others.

Whereas many other types of interfaces do exist, we felt that supporting e.g. a range of 'method of adjustment' tests would be beyond the scope of a tool that aims to be versatile enough while not claiming to support any custom experiment one might want to set up. Rather, it supports any non-adaptive listening test up to multi-stimulus, multi-attribute evaluation including references, anchors, text boxes, radio buttons and/or checkboxes, with arbitrary placement of the various UI elements.

The code and documentation can be pulled or downloaded from our online repository available at code.soundsoftware. ac.uk/projects/webaudioevaluationtool.

7. REFERENCES
[1] C. Gribben and H. Lee, "Toward the development of a universal listening test interface generator in max," in *AES Convention 138*, Audio Engineering Society, 2015.

[2] S. Bech and N. Zacharov, *Perceptual Audio Evaluation - Theory, Method and Application*. John Wiley & Sons, 2007.

[3] M. Schoeffler, F.-R. Stöter, B. Edler, and J. Herre, "Towards the next generation of web-based experiments: A case study assessing basic audio quality following the ITU-R recommendation BS. 1534 (MUSHRA)," in *1st Web Audio Conference*, 2015.

[4] S. Kraft and U. Zölzer, "BeaqleJS: HTML5 and JavaScript based framework for the subjective evaluation of audio quality," in *Linux Audio Conference, Karlsruhe, DE*, 2014.

[5] B. De Man and J. D. Reiss, "APE: Audio Perceptual Evaluation toolbox for MATLAB," in *136th Convention of the AES*, April 2014.

[6] E. Vincent, M. G. Jafari, and M. D. Plumbley, "Preliminary guidelines for subjective evalutation of audio source separation algorithms," in *UK ICA Research Network Workshop*, 2006.

[7] A. V. Giner, "Scale - a software tool for listening experiments," in *AIA/DAGA Conference on Acoustics, Merano (Italy)*, 2013.

[8] S. Ciba, A. Wlodarski, and H.-J. Maempel, "Whisper – A new tool for performing listening tests," in *126th Convention of the AES*, May 7-10 2009.

[9] N. Jillings, D. Moffat, B. De Man, and J. D. Reiss, "Web Audio Evaluation Tool: A browser-based listening test environment," in *12th Sound and Music Computing Conference*, July 2015.

[10] ITURBS Recommendation, "Bs. 1534-1: Method for the subjective assessment of intermediate quality levels of coding systems," *International Telecommunication Union*, 2003.

[11] G. C. Pascoe and C. C. Attkisson, "The evaluation ranking scale: a new methodology for assessing satisfaction," *Evaluation and program planning*, vol. 6, no. 3, pp. 335–347, 1983.

[12] R. Likert, "A technique for the measurement of attitudes.," *Archives of psychology*, 1932.

[13] ITURBS Recommendation, "1116-1: Methods for the subjective assessment of small impairments in audio systems including multichannel sound systems," *International Telecommunication Union, Geneva*, 1997.

[14] ITUT Recommendation, "P. 800: Methods for subjective determination of transmission quality," *International Telecommunication Union, Geneva*, 1996.

[15] D. R. Peryam and N. F. Girardot, "Advanced taste-test method," *Food Engineering*, vol. 24, no. 7, pp. 58–61, 1952.

[16] ITUR Recommendation, "Bs. 562-3,'subjective assessment of sound quality'," *International Telecommunications Union*, 1997.

[17] H. A. David, *The method of paired comparisons*, vol. 12. DTIC Document, 1963.

[18] S. P. Lipshitz and J. Vanderkooy, "The great debate: Subjective evaluation," *Journal of the AES*, vol. 29, no. 7/8, pp. 482–491, 1981.

[19] D. Clark, "High-resolution subjective testing using a double-blind comparator," *Journal of the AES*, vol. 30, no. 5, pp. 330–338, 1982.