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

Nicholas Jillings
n.g.r.jillings@se14.qmul.ac.uk

Brecht De Man
b.deman@qmul.ac.uk

David Moffat
d.j.moffat@qmul.ac.uk

Joshua D. Reiss
joshua.reiss@qmul.ac.uk

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

ABSTRACT

Here comes the abstract.

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, as can be seen in Table 1. A review of existing listening test frameworks was undertaken and presented in Table 1. HULTI-GEN [7] is a single toolbox that presents the user with a large number of different test interfaces and allows for customisation of each test interface. The Web Audio Evaluation Toolbox (WAET) 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, it does not require any progamming experience as 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 [1].

Web Audio API has important features for performing perceptual tests including sample level manipulation of audio streams [18] and the ability for 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 advanced 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 [18] can enable participants in multiple locations to perform the test. However, to our knowledge, no tool currently exists that allows the creation of a remotely accessible listening test.

Both BeaqleJS [9] 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 [8] which at the time of its inception was capable of running a listening test in the browser from an XML configuration file, and storing an XML file as well, with one particular interface. We have now expanded this 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 which 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. Specifically, in Section 2 we cover the general implementation aspects, with a focus on the Web Audio API, followed by a discussion of the requirements for successful remote tests in Section 3. Section 4 describes the various interfaces the tool supports, as well as how to keep this manageable. Finally, in Section 5 we provide an overview of the analysis capabilities in the browser, before summarising our findings and listing future research directions in Section 6.

2. ARCHITECTURE

Although WAET uses a sparse subset of the Web Audio API functionality, its performance comes directly from using 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 obtain which parts of the audio fragments were listened to and when, at

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

Web Audio Conference WAC-2016, April 4–6, 2016, Atlanta, USA

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

Toolbox	**APE**	**BeaqleJS**	**HULTI-GEN**	**mushraJS**	**MUSHRAM**	**Scale**	**WhisPER**	**WAET**
Reference	[5]	[9]	[7]		[19]	[6]	[2]	[8]
Language	MATLAB	JS	MAX	JS	MATLAB	MATLAB	MATLAB	JS
Remote		(not native)		✓				✓
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			✓					✓
Degredation 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						✓		

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 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 called the specification document and can be designed either by manually writing the XML (or modifying an existing document or template) or using our included test creator. These are standalone HTML pages which do not require any server or internet connection and help a build the test specification document. The first (test_create.html) is for simpler tests and operates step-by-step to guide the user. It supports media through drag and drop and a clutter free interface. The advanced version is for more advanced tests where raw XML manipulation is not wanted but the same freedom is required (whilst keeping a safety net). 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 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. This is used in the test to perform loudness normalisation without needing to edit any audio files. Equally the gain can be modified in either editor using an HTML5 slider or number box.

The specification document contains the URL of the audio fragments for each test page. These fragments are downloaded 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, the XML information 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, which is implemented in the standard as a 'use once' object. Once the bufferNode has been played, the bufferNode must be discarded as it cannot be instructed to play the same bufferSourceNode again. Therefore on each start 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. 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 slider is not indicative of the final volume exiting the speakers and therefore its use should only be considered in a lab condition environment to ensure proper behaviour. Finally the gain nodes allow for cross-fading between samples when operating in synchronous playback. Cross-fading can either be fade-out fade-in or a true cross-fade.

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 accpeted 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) [17]
  - Multiple stimuli are presented and rated on a continuous scale, which includes a reference, hidden reference and hidden anchors.
- Rank Scale [12]
  - Stimuli ranked on single horizontal scale, where they are ordered in preference order.
- Likert scale [10]
  - Each stimuli has a five point scale with values: Strongly Agree, Agree, Neutral, Disagree and Strongly Disagree.
- ABC/HR (ITU-R BS. 1116) [16] (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 comparison. There is also a provided reference
- Absolute Category Rating (ACR) Scale [14]
  - Each stimuli has a five point scale with values: Bad, Poor, Fair, Good, Excellent
- Degredation Category Rating (DCR) Scale [14]
  - Each stimuli has a five point scale with values: (5) Inaudible, (4) Audible but not annoying, (3) slightly annoying, (2) annoying, (1) very annoying.
- Comparison Category Rating (CCR) Scale [14]
  - Each stimuli has a seven point scale with values: 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 [13]
  - 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 [15]
  - Each stimuli has a five point scale with values: (5) Imperceptible, (4) Perceptible but not annoying, (3) slightly annoying, (2) annoying, (1) very annoying. There is also a provided reference.
- Pairwise Comparison (Better/Worse) [4]
  - A reference is provided and ever stimulus is rated as being either better or worse than the reference.
- APE style [5]
  - Multiple stimuli on a single horizontal slider for inter-sample rating.
- Multi attribute ratings
  - Multiple stimuli as points on a 2D plane for inter-sample rating (eg. Valence Arousal)
- AB Test [11]
  - Two stimuli are presented at a time and the participant has to select a preferred stimulus.
- ABX Test [3]
  - 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 choice 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. To this end, there is an 'Interface' object which includes object prototypes for creating the on-page comment boxes (including those with radio or checkbox responses), start and stop buttons with function handles pre-attached 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 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
- Box plot, confidence plot, and scatter plot of rating values
- 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

[Some pictures here please.]

6. CONCLUDING REMARKS AND FUTURE WORK

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

[Talking a little bit about what else might happen. Unless we really want to wrap this up.]

[18] gives a 'checklist' for subjective evaluation of audio systems. The Web Audio Evaluation Toolbox meets most of its given requirements including remote testing, crossfading between audio streams, collecting browser information, utilising UI elements and working with various audio formats including uncompressed PCM or WAV format.

[What can we not do? 'Method of adjustment', as in [18] is another can of worms, because, like, you could adjust lots of things (volume is just one of them, that could be done quite easily). Same for using input signals like the participant's voice. Either leave out, or mention this requires modification of the code we provide.]

7. REFERENCES

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

[2] S. Ciba, A. Wlodarski, and H.-J. Maempel. Whisper – A new tool for performing listening tests. In *126th Convention of the Audio Engineering Society*, May 7-10 2009.

[3] D. Clark. High-resolution subjective testing using a double-blind comparator. *Journal of the Audio Engineering Society*, 30(5):330–338, 1982.

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

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

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

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

[8] 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.

[9] 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.

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

[11] S. P. Lipshitz and J. Vanderkooy. The great debate: Subjective evaluation. *Journal of the Audio Engineering Society*, 29(7/8):482–491, 1981.

[12] G. C. Pascoe and C. C. Attkisson. The evaluation ranking scale: a new methodology for assessing satisfaction. *Evaluation and program planning*, 6(3):335–347, 1983.

[13] D. R. Peryam and N. F. Girardot. Advanced taste-test method. *Food Engineering*, 24(7):58–61, 1952.

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

[15] I. Rec. Bs. 562-3,'subjective assessment of sound quality'. *International Telecommunications Union*, 1997.

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

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

[18] 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.

[19] 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.