

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 over realism of sound synthesis to the performance of source separation, automated music production and in less technical areas, the framework of a listening test can be used to measure emotional response to music or test cognitive abilities.

Technical, interfaces, user friendliness, reliability

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 made some essential features like sample manipulation of audio streams possible [18].

Situating the Web Audio Evaluation Tool between other currently available evaluation tools, ...

... However, BeagleJS [9] does not make use of the Web Audio API,

Selling points: remote tests, visualisation, create your own test in the browser, many interfaces, few/no dependencies, flexibility

As recruiting participants can be very time-consuming, and as for some tests a large number of participants is needed, browser-based tests [18]. However, to our knowledge, no tool currently exists that allows the creation of a remotely accessible listening test.

[Talking about what we do in the various sections of this paper. Referring to [8].]

1.1 Meeting 8 October



Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). Attribution: owner/author(s).

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

© 2016 Copyright held by the owner/author(s).

- Do we manipulate audio?
 - Add loudness equalisation? (test_create.html) Tag with gains.
 - Add volume slider?
 - Cross-fade (in interface node): default 0, number of seconds
 - Also: we use the playback buffer to present metrics of which portion is listened to
- Logging system information: whichever are possible (justify others)
- Input streams as audioelements
- Capture microphone to estimate loudness (especially Macbook)
- Test page (in-built oscillators): left-right calibration, ramp up test tone until you hear it; optional compensating EQ (future work implementing own filters) → Highlight issues!
- Record IP address (PHP function, grab and append to XML file)
- Expand anchor/reference options
- AB / ABX

1.1.1 Issues

- Filters not consistent (Nick to test across browsers)
- Playback audiobuffers need to be destroyed and rebuilt each time
- Can't get channel data, hardware input/output...

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

Name	Ref.	Language	Interfaces	Remote	All UI
APE	[5]	MATLAB	multi-stimulus, 1 axis per attribute	(not natively supported)	✓
BeagleJS	[9]	JavaScript	ABX, MUSHRA		
HULTI-GEN	[7]	MAX		✓	✓
mushraJS		JavaScript	MUSHRA		
MUSHRAM	[19]	MATLAB	MUSHRA		
Scale	[6]	MATLAB		✓	✓
WhisPER	[2]	MATLAB			
WAET	[8]	JavaScript	all of the above		

Table 2: Table with interfaces and which toolboxes support them

Interface	HULTI-GEN	Scale	WhisPER	WAET
MUSHRA (ITU-R BS. 1534)	✓			✓
Rank scale	✓			✓
Likert scale	✓		✓	✓
ABC/HR (ITU-R BS. 1116)	✓			✓
-50 to 50 Bipolar with Ref	✓			✓
Absolute Category Rating (ACR) Scale	✓			✓
Degradation Category Rating (DCR) Scale	✓			✓
Comparison Category Rating (CCR) Scale	✓		✓	✓
9 Point Hedonic Category Rating Scale	✓		✓	✓
ITU-R 5 Point Continuous Impairment Scale	✓			✓
Pairwise Comparison (Better/Worse)	✓			✓
APE style				✓
Multi attribute ratings	✓			✓
AB Test	✓			✓
ABX Test	✓			✓
“Adaptive psychophysical methods”			✓	
Repertory Grid Technique (RGT)			✓	
(Semantic differential)			(✓)	

2. ARCHITECTURE

WAET utilises the Web Audio API for audio playback and uses a sparse subset of the Web Audio API functionality, however the performance of WAET comes directly from the Web Audio API. Listening tests can convey large amounts of information other than obtaining the perceptual relationship between the audio fragments. WAET specifically can obtain which parts of the audio fragments were listened to and when, at what point in the audio stream did the participant switch to a different fragment and what new rating did they give a fragment. Therefore it is possible to not only evaluate the perceptual research question but also evaluate if the participant performed the test well and therefore if their results are representative or should be discarded as an outlier.

One of the key initial design parameters for WAET is to make the tool as open as possible to non-programmers and to this end the tool has been designed in such a way that all of the user modifiable options are included in a single XML document. This document is loaded up automatically by the web page and the JavaScript code parses and loads any extra resources required to create the test.

The specification document also contains the URL of the audio fragments for each test page. These fragments are downloaded asynchronously 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 as a ‘use once’ object which, once the buffer has been played, the buffer must be discarded as it cannot be instructed to play the buffer 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.

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. Therefore the most widely supported media file is the wave (.WAV) format which can be accepted by every browser supporting the Web Audio API. The next best supported audio only formats are MP3 and AAC (in MP4) which are supported by all major browsers, Firefox relies on OS decoders

and therefore its support is predicated by the OS support.

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.

A slightly technical overview of the system. Talk about XML, JavaScript, Web Audio API, HTML5. Describe and/or visualise `audioholder-audioelement-...` structure.

Which type of files?

Streaming audio?

Compatibility?

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 web server so that subjects 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. Note also that 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
- Randomly pick specified number of audioholders
- Calibration
- Functionality to participate multiple times
 - Possible to log in with unique ID (no password)
 - Pick ‘new user’ (need new, unique ID) or ‘already participated’ (need already available ID)
 - Store XML on server with IDs plus which audioholders have already been listened to
 - Don’t show ‘post-test’ survey after first time
 - Pick ‘new’ audioholders if available
 - Copy survey information first time to new XMLs
- Intermediate saves
- Collect IP address information (privacy issues?) → geo-related API?
- Collect Browser and Display information

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. A review of existing listening test frameworks was undertaken and presented in Table 2. 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.

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.

While implementing all of these interfaces, 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 functions 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.

A screenshot would be nice.

Established tests (see below) included as 'presets' in the build-your-own-test page.

5. ANALYSIS AND DIAGNOSTICS

It would be great to have easy-to-use analysis tools to visualise the collected data and even do science with it. Even better would be to have all this in the browser. Complete perfection would be achieved if and when only limited setup, installation time, and expertise are required for the average non-CS researcher to use this.

The following could be nice:

- Web page showing all audioholder IDs, file names, subject IDs, audio element IDs, ... in the collected XMLs so far (`saves/*.xml`)
- Check/uncheck each of the above for analysis (e.g. zoom in on a certain song, or exclude a subset of subjects)
- Click a mix to hear it (follow path in XML setup file, which is also embedded in the XML result file)
- Box plot, confidence plot, scatter plot of values (for a given audioholder)
- Timeline for a specific subject (see Python scripts), perhaps re-playing the experiment in X times realtime. (If actual realtime, you could replay the audio...)
- Distribution plots of any radio button and number questions (drop-down menu with 'pretest', 'posttest', ...; then drop-down menu with question 'IDs' like 'gender', 'age', ...; make pie chart/histogram of these values over selected range of XMLs)
- All 'comments' on a specific audioelement
- A 'download' button for a nice CSV of various things (values, survey responses, comments) people might want to use for analysis, e.g. when XML scares them
- Validation of setup XMLs (easily spot 'errors', like duplicate IDs or URLs, missing/dangling tags, ...)

A subset of the above would already be nice for this paper. Some pictures here please.

6. CONCLUDING REMARKS AND FUTURE WORK

The code and documentation can be pulled or downloaded from code.soundsoftware.ac.uk/projects/webaudioevaluationtool.

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

Use [18] as a 'checklist', even though it only considers subjective evaluation of audio systems (and focuses on the requirements for a MUSHRA test).

[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. BeagleJS: 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. Attikisson. 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 evaluation of audio source separation algorithms. In *UK ICA Research Network Workshop*, 2006.