GuineaPig – A generic subjective test system for multichannel audio

Jussi Hynninen
Laboratory of Acoustics and Audio Signal Processing
Helsinki University of Technology, Espoo, Finland
hynde@acoustics.hut.fi
http://www.acoustics.hut.fi/~hynde/

Nick Zacharov
Nokia Research Center
Speech and Audio Systems Laboratory, Tampere, Finland
Nick.Zacharov@research.nokia.com

Abstract

The GuineaPig test system has been developed as a generic, software based, multichannel audio system for subjective testing. Based around the SGI hardware platform, the system provides testing with several test categories, including standardised tests, providing currently up to 8-channel output and graphical UIs for testing. The GuineaPig provides experimenters fast access to subjective testing.

1 Introduction

The GuineaPig (GP) test system has been developed in co-operation with *Helsinki University of Technology, Laboratory of Acoustics and Audio Signal Processing* and *Nokia Research Center*. The system has been evolved as a very generic test platform for performing a wide range of subjective audio tests, whilst eliminating a lot of the complexity of setting up such experiments.

In recent years subjective testing methods have been popularised and formalised in the field of audio by the likes of Gabrielsson [1], Toole [2, 3], and Bech [4, 5]. In the field of telecommunications and perceptual coding, for example, a variety of subjective evaluation methods are employed as a means of final evaluation. Each standard employs a slightly different method of evaluation, each of which must rigidly adhered to. Some of the common standards encountered in these fields are illustrated in figure 1. Typically these procedures can be very time consuming and complex if performed manually, as frequently occurs. Whilst manually performed subjective test are practical when infrequent, for those

performing such tests on a regular basis, the manual preparation can become tiresome. Manual preparation can also lead to error especially when tests are large. Furthermore, it is often desirable to employ complex experimental design tools to improve the quality of the experiment. Block designs ([6], pp. 439–482) are examples of such tools, which are preferably performed automatically by computer, to avoid experimental error.

In the past computer based subjective test systems have been designed to perform certain and limited test categories or to slave other audio-visual playback hardware. The aim of the GP test system was to overcome limitations of test procedure and to simplify the test hardware to a bare minimum. The GP test system thus consists of a Unix based computer, digital audio output hardware, DAC's and reproduction transducers, as illustrated in figure 2.

The GP test system has been developed to run on the Silicon Graphics, Unix based, platform which offers a wide range of tools for high quality real-time multichannel audio and video support.

This article aims to introduce the background of audio test paradigms around which the GP test system has been developed and describe the systems architecture and functionality with examples.

2 Subjective test paradigms

It is well known that there is a wide range of subjective testing methods and associated standards in existence. In this section we shall introduce a few of the common test paradigms and methods employed within the audio industry that are supported by the GP platform.

2.1 The single stimulus paradigm

In the simplest form of subjective test the listener is subjected to a single sample to evaluate in isolation. Grading is performed on an interval scale without any references. Typically, this method is only applied when it is not possible to compare systems or samples simultaneously for example when comparing the sound quality of two rooms. The method is also quite widely employed in the telecommunications industry and referred to as the MOS (mean opinion score) test [7]. As no direct comparison is possible in this situation, there is a risk there will be fairly high inter listeners error variance. Furthermore, it is not possible to assess any interaction information between samples, as there is no information as to whether or not the listener employed the same scaling for each sample.

Whilst simple means and error variance measures are typically employed for telecommunications tests, the data can be analysed or thoroughly with the analysis of variance (ANOVA) model [1].

2.2 The paired comparison paradigm

The method of pair comparisons has been employed for over a century and is common in the field of psychometrics. An excellent study of the method and associated analysis tools is provided by David [8]. The method has widespread acceptance due to it's simplicity and well proven analysis methods.

The pair comparison task simply consists of presenting the listener with two samples from which the superior/inferior is to be chosen. In practice this is a simple task which is easy for listeners to learn and comprehend, even when unfamiliar with subjective testing. The method is significantly slower than the single stimulus, particularly if full permutation set are employed. Incomplete block design methods can be employed to improve the efficiency whilst not loosing interaction information ([6], pp. 439–482).

The analysis of such experiments can be performed with tools such as the student t-test [9] and interaction information can be accessed with more complex non-parametric methods such as the Bradley-Terry model ([8], pp. 61–71).

2.3 The scaled paired comparison paradigm

The scaled paired comparison paradigm is an extended category of the basic paired comparison discussed above, which introduces the use of interval scales to grade each of the samples. Typically the method employs a single identical rating scale for each sample in the form of a MOS or degradation mean opinion score (DMOS) scales [7]. This type of test is common in the field of telecommunications and is also employed in audio testing [10]. This method is also found in the field of visual testing where it is referred to as the *double stimulus continuous quality* method *Recommendation ITU-R BT.500-8* [11].

An alternative method is to employ the comparative mean opinion score (CMOS) or comparison category rating (CCR) scale as found in *Recommendation ITU-T P.800* [7]. In this case there is only a single scale employed to rate both samples. The scale allows for both positive and negative ratings thus allowing for comparison

Data can be thoroughly analysed with analysis of variance (ANOVA) models. Multi-dimesional scales can be analysed in depth employing Multivariate ANOVA (MANOVA) method.

2.4 The ABX paradigm

The ABX paradigm was developed by Clark [12, 13], provides a helpful extension to the pair comparison paradigm. The method consists of presenting the listener with three samples, A, B and X.

The aim is for the listener to select which sample of A or B is identical to X. The benefits on this paradigm lie in the fact that the task is simple and that there is a clear reference to compare against. Furthermore, the test allows for a very quick and easy assessment of

listener reliability, by simply studying the number of correct answers. However, this test is slightly more time consuming than simple paired comparison methods.

The test can be considered limited to unidimensional studies.

2.5 Triple stimulus hidden reference paradigm

The method of the *double-blind triple stimulus with hidden reference* as standardised in *Recommendation ITU-R BS.1116-1* [14], is a practical extension of the ABX method to include interval scales.

This method provides the listener with three samples: Ref, A and B. Compared to the ABX method, it employs the ITU-R 5-point impairment scale. The listener must first select which sample is different to the reference and grade this sample only. The other sample must be given the maximum grade (i.e. 5).

Whilst this method is quite complex and non-intuitive, it is aimed at so called *expert* listeners who have been trained. The method benefits from the use of a reference and grading scales, providing detailed information on the tested systems. The hidden reference method also provides for a rapid check on listener reliability. The method is not intended for fast and dirty testing, but is well suited to high quality, small impairment tests.

A t-test is employed to post-select suitable listeners for the main analysis of results which is performed with an ANOVA.

2.6 The two-alternative-forced-choice (TAFC or 2AFC) paradigm

The category of two-alternative-forced-choice methods are widely employed in psychometrics for the use of parameter or threshold estimation and are considered a subset of the sequential experiment class of procedures. Various forms of TAFC procedures exist which are described in detail in Levitt's paper [15], considered a definitive summary of the topic.

The simple up-down or staircase method has been implemented to estimate the equivalence of two stimuli. However, the test may also be configured to perform absolute threshold experiments with only one stimulus (e.g. audiometry experiments).

The method basically consists of presenting the listener sequentially with the reference and the test stimuli. The listener must judge whether or not the test stimulus is, say, louder, for example. Based upon a positive response, the following stimulus level is decreased (or increased following a negative response). This procedure continues until 6-8 reversals have occurred. A run is defined as the series of steps in one direction. Based upon this data, the overall parameter estimation can be made from estimating the 50% level (X_{50}), for example.

It should be noted that to ensure that this method is correctly conducted a few rules must be applied, which have been implemented within the GP test system, including the:

- starting level,
- number of runs,
- reversal rules,
- stimuli step size.

The method is relatively efficient, but careful choice of step size is required to ensure efficiency and quality of results.

2.7 The rank order paradigm

The rank order method ([8], pp. 104–130), ([16], pp. 691–700) can be employed when three or more samples are to be compared. The most simple form of the rank order test is that of the paired comparison, where 2 samples are compared and a selection made upon which is found to be superior (or inferior). Whilst this test type is common, the more general rank order procedure does not find much favor. The procedure consists of asking the subject to arrange the samples in order intensity or degree of the tested attribute.

This procedure benefits from its simplicity and the lack of assumptions:

- the task is very simple to comprehend and requires little training or instruction,
- there is no need to understand or interpret a scale,
- there are few assumptions regarding the data type and the data's distribution. Data need not be normally distributed or are there any assumptions regarding the perceptual separation between samples,
- a large number of sample can be considered in a relatively brief test. Whilst certain authors [17] consider there are some perceptual limitations in this respect, the test method itself is not restricted,
- complete or incomplete block designs may be employed,
- data handling and analysis are simple.

To perform such a test a few assumptions are required:

- All products must be evaluated prior to judgement. This may thus lead to sensory fatigue when many sample are tested,
- the direction of ranking must be specified (i.e. which sample is better).

In practice, whilst this test can provide some knowledge of the rank order of samples, it is not possible to provide an absolute rating of quality. This is perhaps one of the reasons why this method lacks favor.

2.8 Scaling

All of the method introduced above employing scales can employ a wide range of scaling techniques. Although the issue of scaling would appear to be trivial at first glance, it is in fact a complex field in psychometrics. The scale can be considered as the visualisation of a persons perception of the quality space and thus its presentation and interpretation are very important. Whilst quite often within the audio industry we employ unidimensional rating scales (e.g. the MOS scale), we are in fact often rating the overall perception of an multidimensional event. As a tool for providing input to design process, this is not always sufficiently informative. Many authors in audio research have studied this aspect and the use of multidimensional scales is gaining increase support [18, 19]. When considering the use of a scale, some of the following issues should be considered in depth and with care.

- Type (Ordinal, interval, ratio, category (or nominal)),
- resolution,
- anchors (name and placement),
- absolute scale length,
- uni- or multi- dimensional.

Clearly, to be able to perform a wide range of multidimensional subjective tests, would require a lot of flexibility in how to define each of the scale to fulfil the requirements listed above. Many of these aspects are supported by the the GP test system and are discussed in section 5.1 and 6.4 and illustrated in figures 3.

For a full discussion on the use of scales, the interested reader is referred to ([20], pp.11–81), ([21], pp. 53–56), ([22], pp. 66–94).

3 Architecture

The general structure of the GP system is shown in figure 4. The main modules consist of the *sound player*, *subject user interfaces*, *test engine*, *configuration files* and *results processing*. The GP system is written in Java [23, 24, 25] except the sound player that is mainly written in C. Although there are no plans to port GP to other systems, the subject user interfaces benefit greatly from Java's cross-platform portability.

The *sound player* (sec. 4) handles the audio output of the GP system. The sound player is written in C to get access to real-time performance and SG's digital media libraries. The sound player is controlled with a Java front-end module that the rest of the GP system uses.

Subjects use a graphical *subject user interfaces* (sec. 6) to give their answers and to select samples to play. GP is designed such that subject user interface windows can be opened

on many different platforms including smaller and silent devices. Any Java-compatible terminal can be used as subject's answering panel.

The *test "engine"* (sec. 5) runs and manages the test. It reads needed information from *test configuration files* and initializes the test and subject UIs. In general, it plays the samples selected by the experimenter for comparison in fixed sequence or selected by the subject, collects subjects' answers and records them. The test engine allows multiple subjects to give answers at the same time but not to select samples independently. In the future our aim is to run 4 independent tests in parallel, thereby shortening the amount of time required for each experiment.

The *results processing* (sec. 7) takes the raw result files generated by the test engine and converts them to human and computer readable tabular format for analysis by other tools.

4 Sound player

The GP's audio output is handled by the *sound player* (SP). Figures 5 and 6 show the general structure of the SP. It is a simple program that plays sound samples directly from hard disk and mixes them together in real time. By playing samples directly from hard disk, the memory usage of the player is quite small even with very large samples. The player is written in C to get access to real-time performance and SG digital media libraries. The player is controlled by a higher-level Java front-end module that is used by the GP system. The GP's player itself is not locked to any sample rate, number of channels, device or interface type or sound file format. It relies on SG's standard audio libraries for most of such things.

The player uses *Silicon Graphics Audio Library (AL)* that provides a uniform, device-independent programming interface to real-time audio I/O on all Silicon Graphics workstations. On SG systems a variety of audio output interfaces are available, from regular analog stereo outputs to digital 8-channel 24-bit optical ADAT outputs. A wide selection of sample rates are also available. However, some output devices may not offer all possible sample rates (such are digital outputs) but at least the most common standard rates are usually supported (32kHz, 44.1kHz and 48kHz). GP's player always uses floating point calculations when processing audio data. Also, AL uses 24-bit precision internally.

Also for audio file access, the *Audio File Library (AF)*, that provides a uniform programming interface to standard digital audio file formats, is used. Currently formats supported by the library include AIFF, AIFF-C, WAVE, MPEG1 audio and many others.

4.1 Sound player's Java-module

A Java-module is used to control the sound player. It hides the implementation details and the low level details of the C-module. Java classes are provided for using the player in an object-oriented fashion. Classes are provided for the sound player, virtual players, sound samples, volume levels and sound player events.

4.2 Virtual players

Virtual players (VP) allow partitioning the output channels of a player (or actually the output device) into smaller sections. For example, a player with an 8-channel ADAT output port can be divided to four stereo players (see fig. 6). Each VP acts the same way as the original player. The outputs of the samples that are attached to a VP are automatically redirected to the channels of the original port that are allocated for the selected VP. Each VP can also have its own output volume level. Virtual players are still to be fully integrated and tested.

4.3 Player's output level

The player has a digital *output level* adjustment that is performed after samples have been mixed together before the audio signal goes out of the machine. Each VP can have their own output level. In GP the output level is usually set to the MCL level (sec. 5.6).

4.4 Sample's volume level control

Each sample has a fader that is used to control the volume level of the sample. The level of each channel can be controlled independently. Level change can be immediate or a fade can be used. Fade's length and type are selectable (amplitude-linear or decibel-linear). Faders are also used for cross-fading with parallel switching (sec. 5.4).

Each sample also has a static *sample level calibration factor* than can be used to calibrate the levels of a set of samples so that their levels are aligned, for example. By setting the calibration factor for a sample, there is no need to edit and scale the sample files themselves.

4.5 Mixing

The signals from any number of samples are mixed together to form the output signal. Each sample's channel can be mapped to any one output channel of the player (or virtual player) with their own strength factor (see *channel mapping* in fig. 5). This would allow simple downmix of a 5.1 signal to stereo or a stereo sample to mono output, for example. In the future, a full mixing matrix for each sample can be implemented when it becomes necessary.

4.6 Drop-out detection

The player *detects drop-outs (framedrops)* that may occur if the system is heavily loaded. These drop-outs are only reported.

4.7 Delay and latency

The *delay and latency* of player's operations can be measured and adjusted. By default the player calculates the output signal in blocks of 4096 sample frames (approx. 93ms when using 44.1kHz sample rate). The calculation is also double-buffered to lower the risk of a drop-out caused by random CPU load peaks. However, this doubles the delay and with other overheads the delay will be a little over 200ms. The delay can be shortened by decreasing the audio block's length with a test configuration parameter.

The delay affects operations that are to take effect immediately, for example, the subject presses a button to play a sample. If an operation is scheduled in advance, some operations can be made to take effect at some point of time with sample frame accuracy. So far only sample's start can be scheduled in advance. Most operations return a sample-frame time stamp that tells the time when the effect of the operation will arrive at the electrical output on the machine. This time stamp can be used to calculate the latency of the operation or the actual time and date of the effect.

4.8 Operations in synchrony

Several samples can be started (or stopped) so that they start (or stop) at exactly the same time. Cross-fade operation also operates on several samples at the same time so that the faders' level slopes start at the same time for all samples.

5 Tests (test engine)

The GP system does not have any rigidly defined test types. Most test types are basically the same with different answer types or different parameters. Only more special test types need some additional code for them. The GP system includes example configurations to show how to implement many standard tests.

Tests are designed by writing test parameters in simple text configuration files. No graphical tools for creating test exist yet.

For running a test, a simple graphical tool is used. It allows to select a playlist file, set session ID and to add test subjects for testing (local or remote). Figure 7 shows the stages of a test.

5.1 Test types

Included in the GP system, there are test types and examples that can be used to implement the following tests:

Single Stimulus A Test in which a single stimulus signal is played and then graded.

A/B Simple paired comparison where two samples are played and the superior or inferior sample is selected [8] (fig. 8).

A/B/X Three samples are played. The listener is to select which sample, A or B, is the same as X [12, 13] (fig. 9).

A/B/Ref Three samples are played. Samples A and B are graded against the reference (Ref).

A/B Scale Paired comparison with a scale for each sample. Also referred to as *double stimulus continuous quality method, ITU-T BT.500-8* [11].

A/B Scale, Fixed Reference The subject gives grades on how samples compare to each other.

A/B Scale, Hidden Reference The subject gives a grade for both samples on a scale specified by the test creator. One of the samples A or B is the hidden reference.

TAFC (Two alternatives forced choice) Two samples are played altering some parameter until the subject can no longer hear the difference between the samples [15].

Rank order N samples are played and the listener grades them in rank order of intensity with a pop-up menu [16] (fig. 10).

5.2 Test items

Test items define the parameters of individual test cases used in a test. For example, a test item for an A/B test has two parameters, A and B, which are the ID labels of the samples that are compared. Test items are also used to store the subject's answers to questions as well as other information (see sec. 7.1). Test items are defined by writing item parameters in a text items-file. Figure 11 provides an example of test item definition.

Test items are loaded by copying a *test item template* for the new item and then setting the parameter values from the parameters defined in test items file. Any number of additional templates can be defined. The templates and items form a tree-structure hierarchy. The test's *default template* is the root of the tree and it usually defines the names of the parameters that there are in each test item. For more specialized tests, the tester can override the default template with additional parameters. Also, if a value is not defined for a parameter in a test item (a leaf of the tree), the value is looked for in the item's template (which again looks for the value from its template if it is not set).

5.3 Playlists

Tests can be easily created to conform to the needs of block designs, with easy implementation of efficient *balanced complete/incomplete block designs* experiments, for example ([8], pp. 83–103), ([6], pp. 439–482). Such designs can be created with commercially available software (e.g. SPSS Trailrun) and imported into GP by use of playlists. A

playlist defines which test items are to presented to the subjects and in which order. It is a text file containing the test item IDs to be presented one per line. All test items that have been defined need not be presented. The GP system automatically looks for a playlist-file based on the session's ID. If no playlist is not defined for a test session, all test items are presented.

5.4 Sample sequence

Most tests are configurable to allow for either fixed sequential playback, as with a tape, or for free switching between samples.

Fixed playback sequence is user-definable with optionally user-defined pauses between samples (see example in fig. 12).

With free switching the subjects play the samples as many times and in any order they like. Normally when the subject selects a sample, the currently playing is stopped and the new sample starts to play from the beginning of the sample. Switching can be done also in parallel, where all samples that are compared are playing at the same time but all but one are silenced. When the subject selects another sample, currently playing sample is switched to the new selected sample with a cross-fade (see fig. 13). The length of the cross-fade and the type of the fade (amplitude-linear or decibel-linear) are configurable.

The subjects can give their answers only after all samples have been listened to (fixed sequence has been played or subject has listened to all samples at least once with free switching). This behaviour can be overridden with a configuration parameter.

5.5 Answering time limit

An *answer time limit* can be set for grading a test item. When the answer time limit is used, an indicator is shown in the subject's window that shows how much time is still left before the item times out (fig. 14d). If a timeout occurs, the item is marked as "timed out". Then the item's answers are stored and the test proceeds to the next test item. If no time limit is set, the subjects can take as much time as they need to give their answers. See fig. 15 for configuration example.

5.6 MCL level

The *most comfortable listening level* (MCLL or MCL level) can be fixed or the test subject can select a comfortable level within a range defined by the experimenter. See fig. 16 for a configuration example. Whilst this type of control maybe be convenient in some tests, it should be included as part of the analysis to avoid level bias effects.

5.7 Multiple subjects concurrently

The GP system allows testing multiple subjects at the same time using remote terminals (sec. 6.9). Currently the system supports any number of subjects giving their answers each on their own terminal. This is useful with fixed playback sequence tests.

GP does not yet support several independent subjects at the same time (each subject could do test independently at their own speed). These features will soon be added and it will then be possible to, for example, divide an eight-channel output port to four stereo ports (with virtual players, sec. 4.2) and four subjects could do the test independently at the same time.

6 Subject's user interfaces

The subjects use a graphical user interface (UI) panel to control the test and to give their answers to the questions (with a mouse). Also test status information can be displayed. Figure 14 shows an example of a subject UI. The UI panels does not actually have to be graphical, any class that implements an interface for subject UIs can be used. That way it would be possible to use simpler, non-graphical interfaces for testing. For example, a module could be written that uses a panel of buttons to give answers. Currently, all the subject UIs in the GP are graphical[1].

The subject UI's components fall into three main categories:

- *Question* components are used to give answers (sec. 6.1).
- *Control* components allow the subject to control some part of the test (sec. 6.2).
- *Monitor* components provide status information about the test (sec. 6.3).

The experimenter can define unlimited number of subject UI components. Any custom component can be used as long as it implements simple subject UI component interfaces. Currently all components are graphical, implemented using the Java's AWT (the same as with the subject UI panel mentioned earlier). Many of the graphical component parameters outlook's are user-configurable, such as labels, fonts and colors (where applicable).

The subject's UI is specified in its own configuration file (simple, text format). The file defines what UI components to include and their parameters. Also other parameters can be included, such as the title of the window, the size of the window (can be fixed to fixed dimensions) and fonts. Two graphical tools are provided for testing UI configuration files and selecting fonts.

[1]Subject UIs are implemented with Java's Abstract Window Toolkit (AWT). In the future, the Java's newer window toolkit "Swing" will probably be used.

6.1 Questions

The question components are used to give answers. When a subject makes a selection, adjusts the grade on a grade component, etc., an event is sent to the test engine. The event contains the question ID of the question component and the answer the subject gave. The test system then records the subject's answer for that question in the test item.

Java objects (`java.lang.Object`) are used as answers generated by the question components. That way any kind of answer types, how simple, complex or special they may be, can be used. The GP system does not care what the answers are actually, it just logs them. More complex answer types may require a custom formatting plug-in module to export the answer for analysis with other tools (see sec. 7 for results processing and custom formatting of special answer types).

6.2 Controls

Control components are used to control some parameters in a test. Currently used controls that are handled automatically for all tests are a sample-play control (sec. 6.7) to play samples, volume level control to set MCL level (sec. 5.6) and button (sec. 6.8) in TAFC-test. Any number of additional controls can be added but they also require writing additional code to handle them. Special tests are made by sub-classing and extending test classes to catch the control messages and then handling them. In the future a better way should be added to allow adding custom plug-in modules to handle different controls without need to extends default test classes. That way adding multiple different controls would be easier and more flexible.

6.3 Monitors

Monitor components provide some status information about the test to the subjects. Currently monitor components are used for two things. The sample play controller also acts as a monitor by showing which sample is currently playing (this is more informative when a fixed sequence is used instead of free switching). Another is an item status monitor that shows how many test items have been completed out of a total number.

6.4 Scales

A generic configurable scale component *GradeBar*[2] is provided that can be used to create many kinds of numeric scales. The component can be used both as a question and a control component. Configurable parameters include:

- The *minimum and maximum* values of the scale.

[2]The name derives from the Java's AWT-component `java.awt.ScrollBar` that is used to implement the scale grading component.

- *Number of decimals* that is used in the answer.
- Whether to *show or hide value* to the subject.
- *Adjectives* can be associated with ranges of values.
- The *initial value* can be set to a fixed value or an automatically generated random value can be used.

A simple extended version *FiveGrade* implements a continuous grading scale with "anchors" derived from the ITU-R five-grade impairment scale given in *Recommendation ITU-R BS.1284* [26]. This versions simply sets the scale parameters accordingly and adds adjectives associated with ranges of values.

Another extended version *TenGrade* implements a ten-grade answering component. The scale goes from 0 to 10 with one decimal and shows the grade symbolically also ("Very unclear", "Rather unclear", "Midway", "Rather clear", "Very Clear"). For example, *Recommendation ITU-T P.910* [27].

Both *TenGrade* and *FiveGrade* components are actually not necessary to implement these scales, they are available mainly as a shortcut. Both can be implemented with the *GradeBar* component by setting the scale's range and adjectives manually, as illustrated in fig. 3.

The *VolumeGradeBar* is another extension to the normal *GradeBar* that gives Volume objects (instead of plain numeric values) as answers by converting GradeBar's numeric answers to Volume-object class.

6.5 Multiple-choice

For making multiple-choice questions, the *CheckBoxChoice* component is provided. It presents a set of choices to the subject who selects one of the choices as the answer using mutually exclusive check-boxes. Any number of choices can be added, each with their own label.

6.6 Rank order

The *RankOrder* question component is used to rank a set of labels (that usually correspond to samples) into an order according to some criteria [16]. Any number of labels can be added. Each label is given a rank using a pop-up menu. The component can be configured to allow or disallow ties and whether to allow incomplete ranking (not all labels have been ranked) as an answer. A special formatter is provided for results processing (sec. 7) to customize the printout format of the rank-order answer.

6.7 Sample-play

The *PlayPanel* control component is used by the subjects to play samples during a free switching test (sec. 5.4). It is also used in both free switching and fixed sequence tests to display which of the samples is currently playing. The panel highlights the button of the sample that is playing. Figure 14a shows an example.

6.8 Button

The *Button* is a simple button control. It sends an event when the button is pressed. Currently button is used in TAFC-test.

6.9 Remote terminals

The GP system allows testing several subjects concurrently. Remote terminals can be used as subject UIs in addition to local terminals (the workstation's console). A local terminal is running in the same process as the test system. A remote terminal runs on a different process than the test system on usually a different host. Remote subject terminals (clients) and the test process (server) communicate using sockets over a LAN (TCP/IP).

As subject UIs are written in Java, any Java-capable terminal (workstation, PC, laptop, etc) can be used. When testing the experimenter first starts a remote terminal server and waits for connections to it. Client terminals are started by running a client program that connects to the server. The server sends the subject UI's code to the client and the client starts it. In both communication end points, wrappers are created that emulate the subject UIs as local terminals, hiding the network between them.

7 Test results processing

The analysis of the test results is not a part of the GP system. GP's role is to gather the data and export it for analysis by any statistical analysis package (e.g. SAS, SPSS, Excel).

Each test session produces a session log file. The log contains copies of the test items presented to the subjects during the session with the answers given by the subjects to the questions (sec. 7.1). The log files are stored in the *Java's Serialization* [28] format which is not readable by any statistical analysis packages. A conversion tool is used to convert the log files into a more readable tab delimited text format. The output format can be customized with configuration files and command line options (sec. 7.2). The simple text format is also easy to process further with many UNIX's common text processing tools, such as Perl or AWK.

7.1 Exported information

Information that can be exported for each test item includes:

- The test item's *item ID*.
- The *session ID* of the test session.
- The *subject ID* of the subject to whom the item was presented to and who gave the answers.
- *Item start time* when this item was presented (both time and date is stored).
- *Item duration*, how much time the subject used to grade this test item.
- *Number of sample switches* the subject made between samples during this item. Actually this saves the how many times samples were played.
- The *item parameters* of this test item, such as the sample IDs of the parameters A and B (for example).
- The *Answers* to the questions shown to the subject in the subject's window.

In addition some per-session information can be included:

- *Starting time* of the session (both time and date are stored).
- *Ending time* of the session (both time and date are stored).
- Session's *MCL level* (sec. 5.6) used for that session.

See example in fig. 17.

7.2 Output customization options

The results output format can be customized with configuration files and/or command line options. Customization options include:

- *Select fields* to print.
- *Order* of the fields.
- *Formatting* of a field. For example, the format used to print a number (number of decimals, date format, etc.) Built-in formatters are included for numbers, dates are used to format numbers and dates, volume levels and rank-order (sec. 6.6). Custom plug-in formatters can be easily added, just extend Java's text formatting

classes[3]. Custom formatter is usually needed if special answer types are used with more comples custom question components.

- Simple *filtering* based on item, subject or session IDs. For example to only print the answers given to some specific test items.

- *Sub-fields* allow you to split a single parameter or answer into multiple fields. For example, you could split the item starting time/date information (which is actually a single Java's date object) into separate time and date fields.

8 Future plans

The list of more important plans, visions and wish-list items for future versions of GP include:

- Multiple audio devices support to allow more than eight channel output. For example, by using two ADAT ports, we could play samples with 16 channels or with hardware tests. Multiple audio ports will be sychronized.

- Parallel testing with independent terminals/subjects: Multiple subjects could be taking the test at the same time (as is possible now with fixed sequence tests) but with their own playlist with their own time (several free switching tests at the same time).

- Switch from Java 1.1 (current) to Java 2, also UIs converted from from AWT to Swing (already used with some tools).

- Real-time and/or offline audio filtering for sound player.

- Video support: Video would be played from hard disk in sync with sound samples.

- Graphical tools for creating and configuring tests.

9 Acknoledgements

The authors would like to thank Vesa Välimäki and Jyri Huopaniemi for their continuous support and comments on this project.

We would also like to thank the funding bodies including Nokia Research Center and Tekes (Technology Development Center of Finland).

[3]Any class that extends Java's `java.text.Format` can be used as a field formatter in the results conversion tool. For example, GP uses Java's text formatting classes `java.text.DecimalFormat` and `java.text.SimpleDateFormat` to format numbers and dates.

References

[1] A. Gabrielsson, "Statistical treatment of data for listening tests on sound reproduction systems," Tech. Rep. Rep. TA 92, Department of Technical Audiology, Karolinska Inst., Sweden, 1979.

[2] F. E. Toole, "Listening tests-turning opinion into fact," *J. Audio Eng. Soc.*, vol. 30, no. 6, 1982.

[3] F. E. Toole, "Subjective measurements of loudspeaker sound quality and listener performance," *J. Audio Eng. Soc.*, vol. 33, no. 1, 1985.

[4] S. Bech, "Training of subjects for auditory experiments," *Acta Acustica*, vol. 1, pp. 89–99, 1993.

[5] S. Bech, "Perception of timbre of reproduced sound in small rooms: Influence of room and loudspeaker position," *J. Audio Eng. Soc.*, vol. 42, no. 12, pp. 999–1007, 1994.

[6] W. G. Cochran and G. M. Cox, *Experimental design.* Wiley, 1992.

[7] ITU-T, *Recommendation P.800, Mehtods for subjective determination of tranmission quality.* International Telcommunications Union Radiocommunication Assembly, 1996.

[8] H. A. David, *The method of paired comparisons.* Oxford University press, 1st ed., 1963.

[9] G. E. P. Box, W. G. Hunter, and J. S. Hunter, *Statistics for experimenters.* Wiley, 1978.

[10] N. Zacharov, "Subjective appraisal of loudspeaker directivity for mulitchannel reproduction," *J. Audio Eng. Soc.*, vol. 46, no. 4, pp. 288–303, 1998.

[11] ITU-R, *Recommendation BT.500-8, Methodology for the subjective assessment of quality of televison pictures.* International Telcommunications Union Radiocommunication Assembly, 1998.

[12] D. L. Clark, "High resolution subjective testing using a double blind comparator," *J. Audio Eng. Soc.*, vol. 30, no. 5, 1982.

[13] D. L. Clark, "Ten years of a/b/x testing," in *Proceedings of the 91*st *Convention of the Audio Engineering Society, Preprint 3167*, 1991.

[14] ITU-R, *Recommendation BS.1116-1, Methods for the subjective assessment of small impairments in audio systems including multichannel sound systems.* International Telcommunications Union Radiocommunication Assembly, 1997.

[15] H. Levitt, "Transformed up-down methods in psychoacoustics," *Journal of the acoustical Society of America*, vol. 49, no. 2, part 2, pp. 467–477, 1971.

[16] H. T. Lawless and H. Heyman, *Sensory evaluation of food.* Chapman and Hall, 1998.

[17] G. A. Miller, "The magical number seven, plus or minus two: some limits on our capacity for processing information," *The Psychological Review*, vol. 63, no. 2, pp. 81–97, 1956.

[18] S. Bech, "Methods for subjective evaluation of spatial characteristics of sound," in *Proceeding of the AES 16th International Conference.*, Audio Eng. Soc., 1999.

[19] J. Berg and F. Rumsey, "Spatial attribute identification and scaling by repertory grid technique and other methods," in *Proceeding of the AES 16th International Conference.*, Audio Eng. Soc., 1999.

[20] J. C. Nunnally and I. H. Bernstein, *Psychometric theory.* McGraw-Hill, 3rd ed., 1994.

[21] M. Meilgaard, G. V. Civille, and B. T. Carr, *Sensory evaluation techniques.* CRC Press, 1991.

[22] H. Stone and J. L. Sidel, *Sensory evaluation practices.* Academic Press, 2nd ed., 1993.

[23] "Java(tm) technology home page." `<URL:http://www.javasoft.com/>`.

[24] "The java(tm) language: An overview." `<URL:http://java.sun.com/docs/overviews/java/java-overview-1.html>`.

[25] "The java language environment – a white paper." `<URL:http://www.javasoft.com/docs/white/langenv/>`.

[26] ITU-R, *Recommendation BS.1284, Methods for the subjective assessment of sound quality - General requirements.* International Telcommunications Union Radiocommunication Assembly, 1998.

[27] ITU-T, *Recommendation P.910, Subjective video quality assessment methods for multimedia applications.* International Telcommunications Union Radiocommunication Assembly, 1996.

[28] "Java's object serialization specification." `<URL:http://www.javasoft.com/products/jdk/rmi/doc/serial-spec/serialTOC.doc.html>`.

[29] ITU-R, *Recommendation BS.1283, Subjective assessment of sound quality - A Guide to existing Recommendations.* International Telcommunications Union Radiocommunication Assembly, 1998.

[30] ITU-R, *Recommendation BS.1285, Pre-selection methods for the subjective assessment of small impairments in audio systems.* International Telcommunications Union Radiocommunication Assembly, 1998.

[31] ITU-R, *Recommendation BS.1286, Methods for the subjective assessment of audio systems with accompanying picture.* International Telcommunications Union Radiocommunication Assembly, 1998.

[32] ITU-R, *Recommendation BS.775-1, Multichannel stereophonic sound systems with and without accompanying picture.* International Telcommunications Union Radiocommunication Assembly, 1994.

[33] ITU-T, *Recommendation P.830, Subjective performance assessment of telephone-band and wideband digital codecs.* International Telcommunications Union Radiocommunication Assembly, 1996.

[34] ITU-T, *Recommendation P.85, A method for subjective performance assessment of the quality of speech voice output devices.* International Telcommunications Union Radiocommunication Assembly, 1994.

[35] ITU-T, *Recommendation P.861, Objective quality measurement of telephone-band (300 -3400 Hz) speech codecs.* International Telcommunications Union Radiocommunication Assembly, 1998.

[36] ITU-T, *Recommendation P.920, Interactive test methods for audiovisual communications.* International Telcommunications Union Radiocommunication Assembly, 1996.

Figures

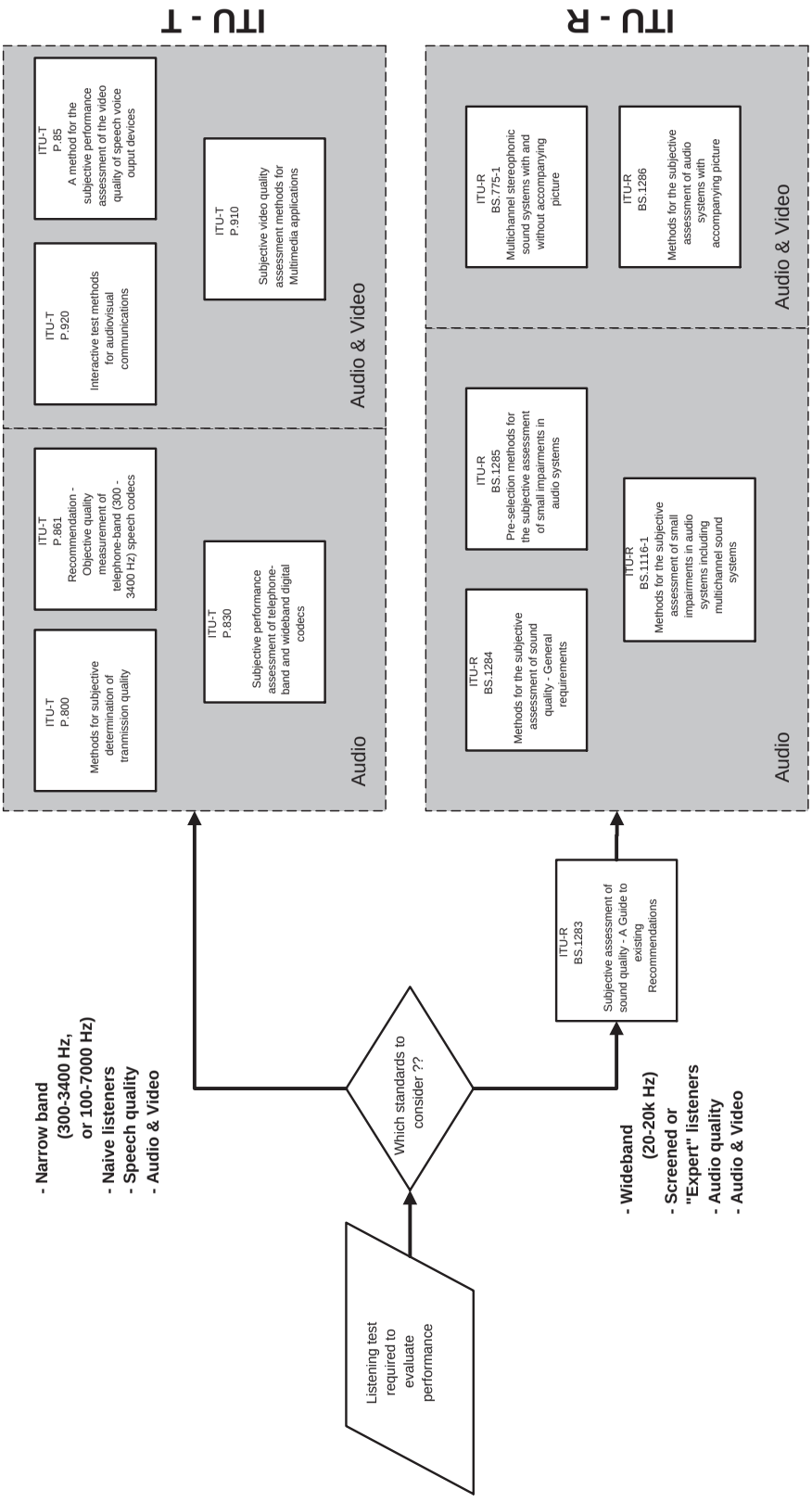

Figure 1: A limited guide to audio listening test standards [7, 14, 26, 27, 29, 30, 31, 32, 33, 34, 35, 36]

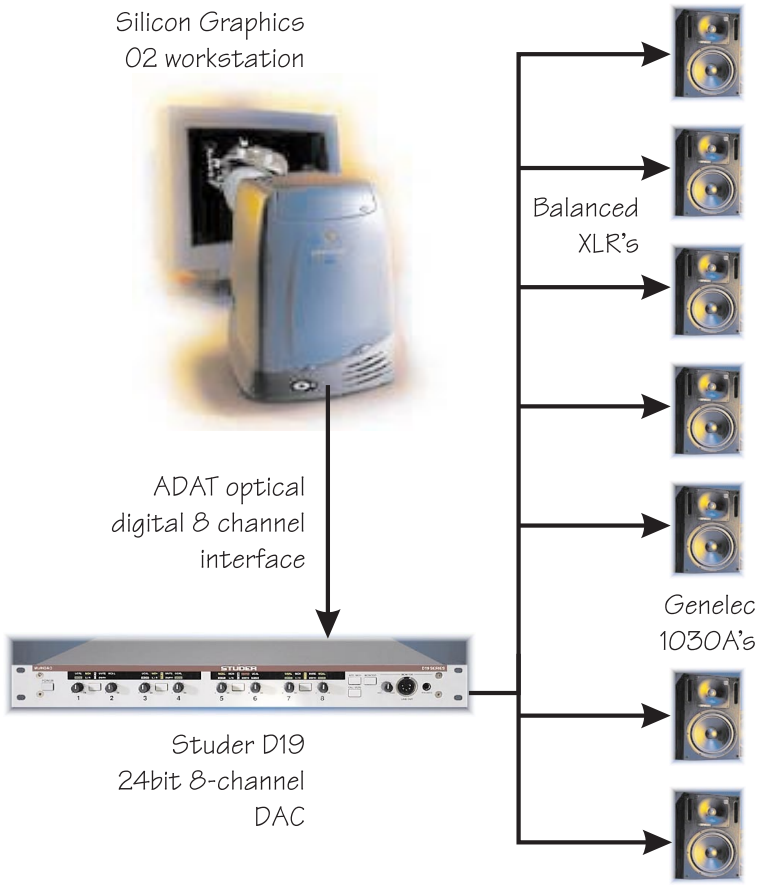

Figure 2: Typical GP test system setup.

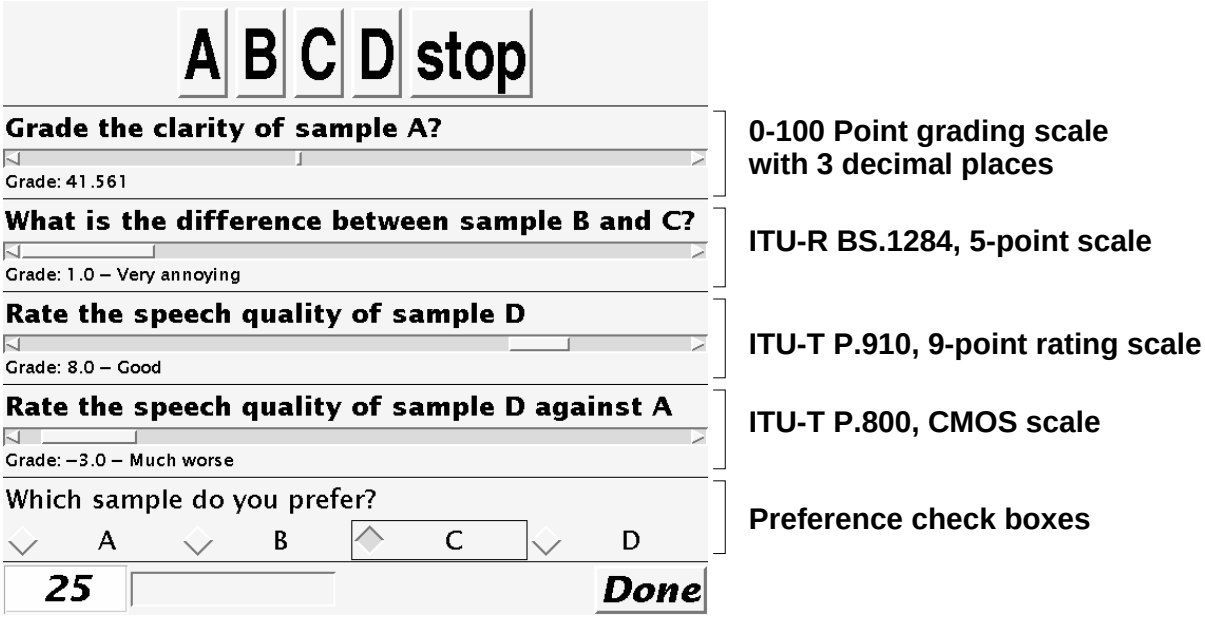

Figure 3: User interface example to illustrate the use of different scales.

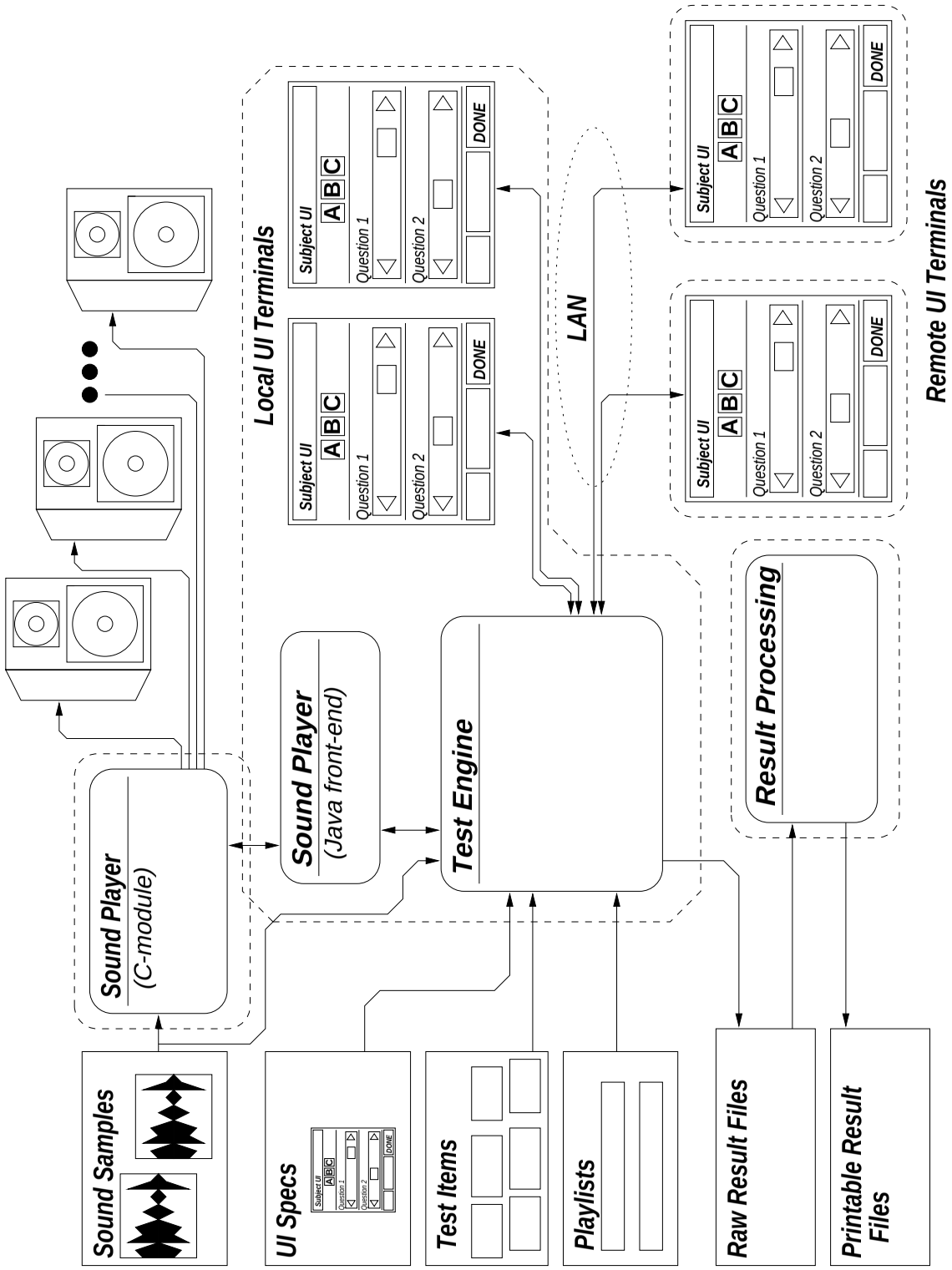

Figure 4: Modules of the GuineaPig system. Dashed lines indicate process boundaries.

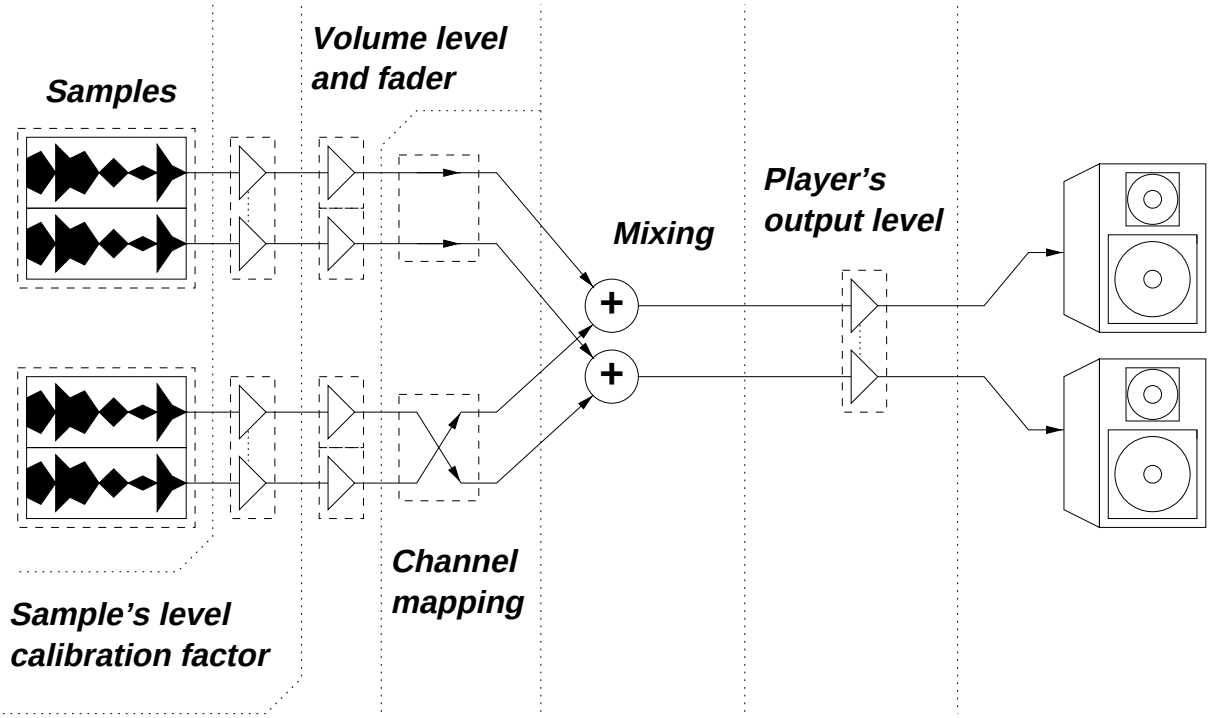

Figure 5: Structure of the sound player. Two stereo samples are played and mixed to stereo output. The lower sample has its left and right channels swapped.

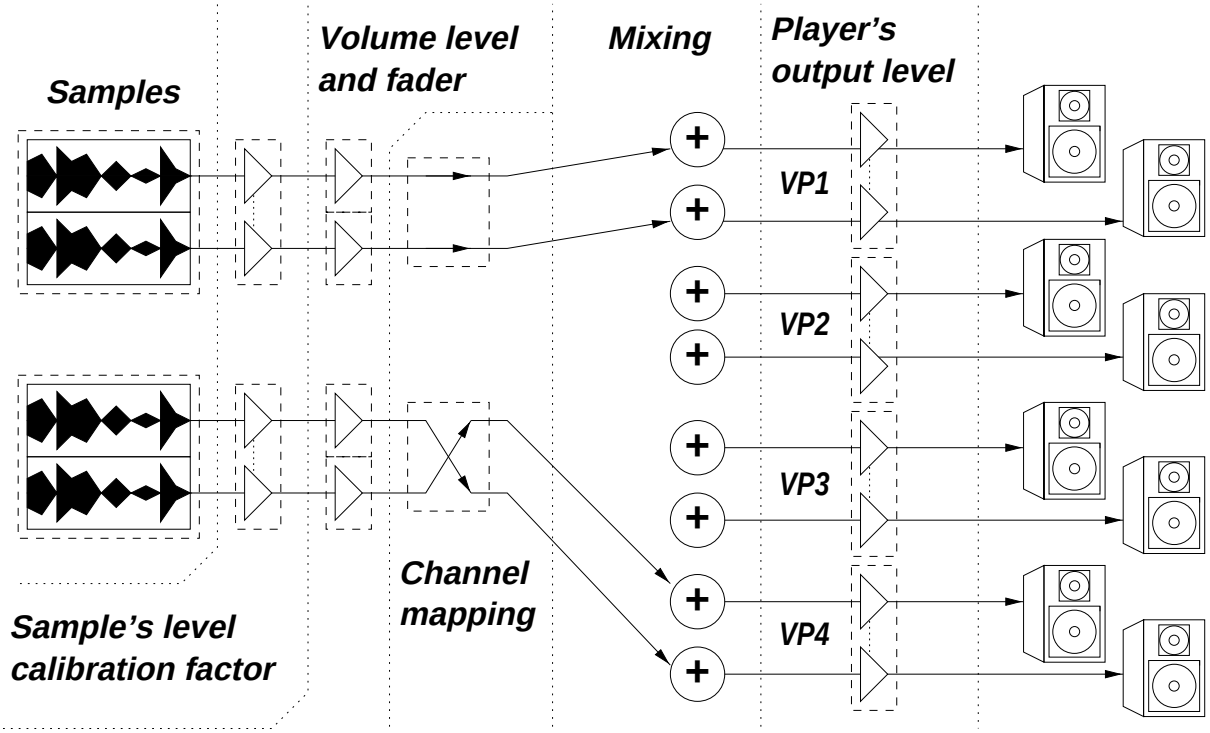

Figure 6: Sound player with four virtual stereo players (VP1-4). Two stereo samples are played. Upper sample's output goes to VP1, lower sample's output goes to VP4. The lower sample has its left and right channels swapped.

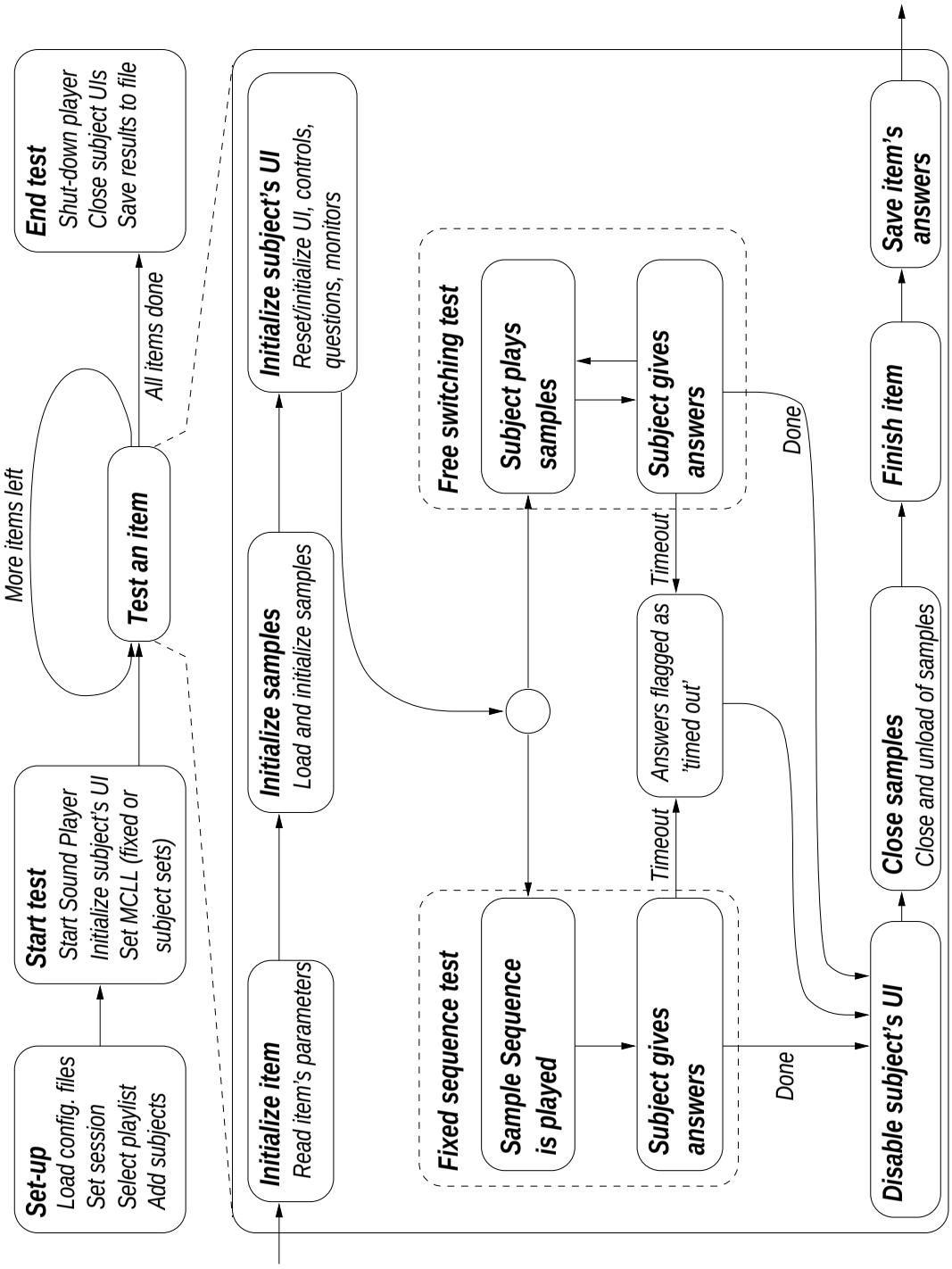

Figure 7: The stages of a test.

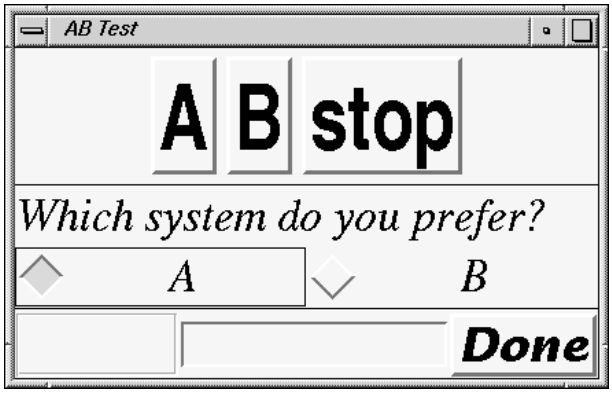

Figure 8: A simple paired comparison user interface.

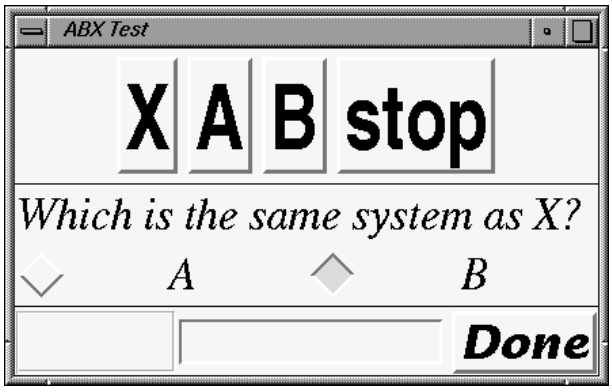

Figure 9: A simple ABX user interface.

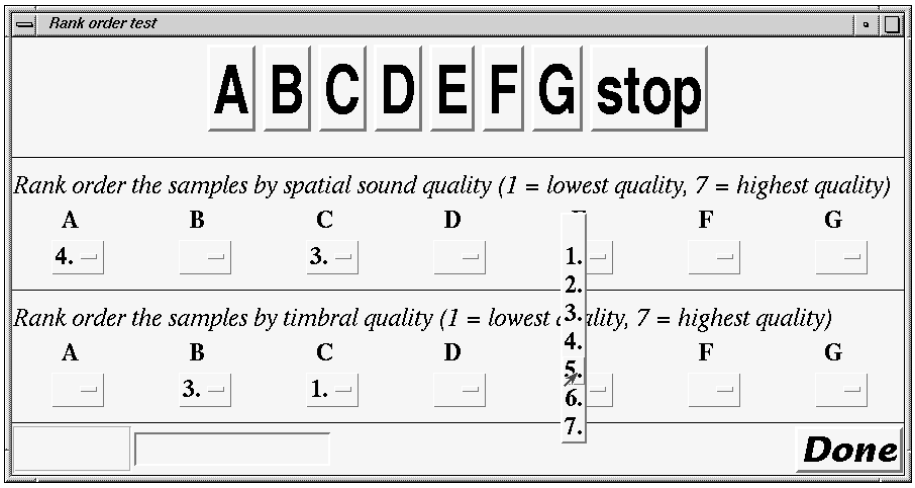

Figure 10: The rank order test user interface.

```
item1.A:        pirr44
item1.B:        pirr32

item2.A:        pirr22
item2.B:        pirr32

item3.A:        pirr8
item3.B:        pirr11
```

Figure 11: Example of a test item definition for an A/B or A/B scale test. Three test items are defined with sound sample IDs as parameters A and B.

```
# Specify fixed sequence
sequenceType:  fixed
# The fixed sample sequence
sequence:      Ref;0.5s,A;0.5s,B;1s,Ref;0.5s,A;0.5s,B
```

Figure 12: Example of a fixed sequence configuration. Samples *Ref*, *A* and *B* are played in sequence with a pause of 0.5 seconds between them. Then a pause of one second followed by the same sequence again.

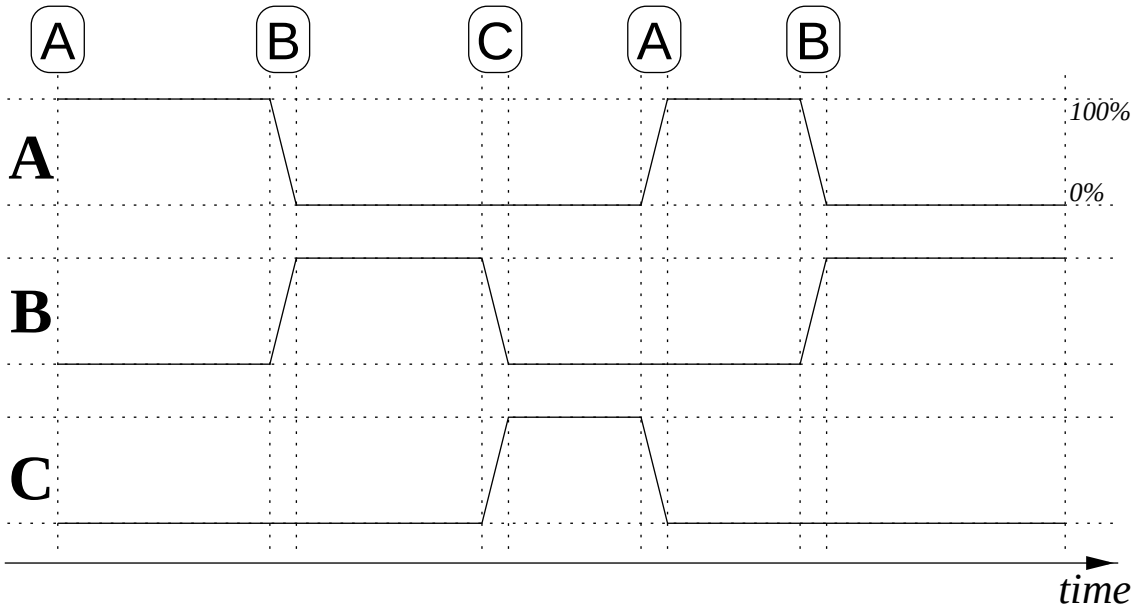

Figure 13: Example of using a crossfade with free parallel switching (sec. 5.4). Three samples A, B and C are compared. All samples are played concurrently but only one has a non-zero volume level. To switch to another sample, the subject presses the corresponding buttons (boxed letters on the top) on the subjects UI panel. The switch is done with a cross-fade (amplitude-linear fade).

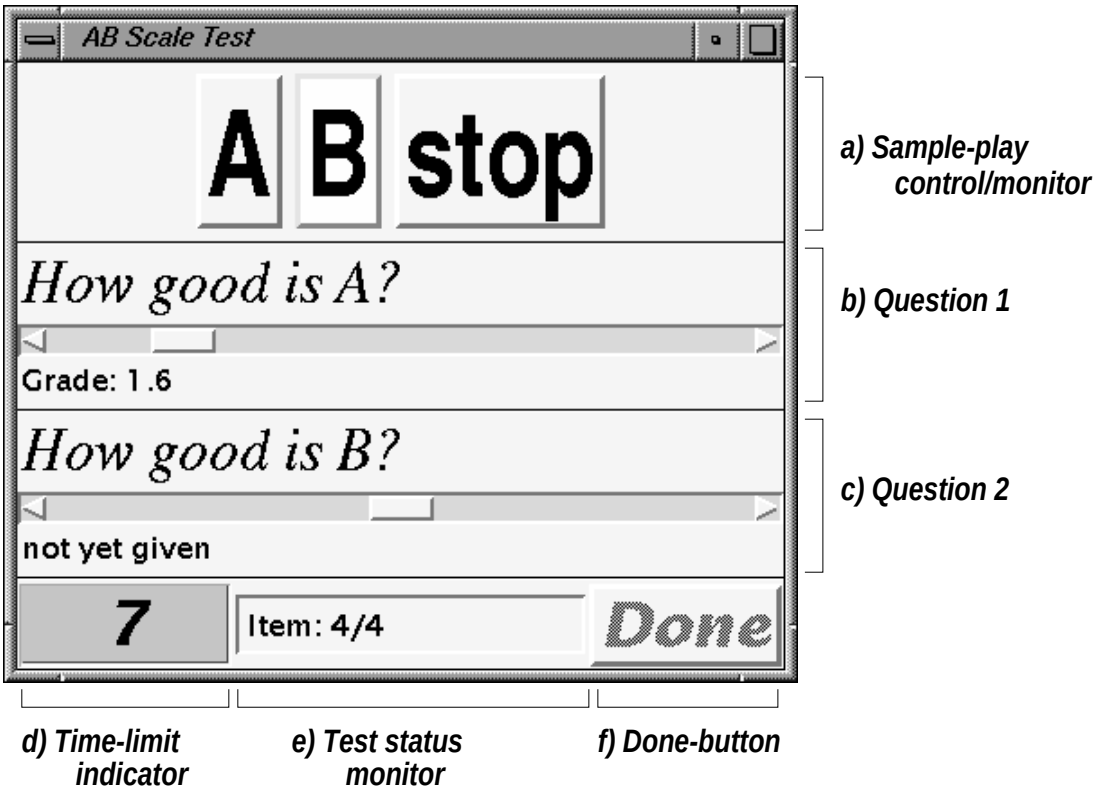

Figure 14: An example of subject's user interface (UI). a) is a sample-play control and/or monitor (sec. 6.7) used to play samples and to show which sample is playing currently. b) and c) are scale questions (sec. 6.4). First question has been given a grade already, second hasn't been answered yet. d) shows how much time (in seconds) there is left to answer (sec. 5.5). e) shows that this is the last item in a set of four. f) Subject presses the Done-button when he/she has completed grading this item. It it currently disabled because question two hasn't been answered yet. It is enabled when all answers have been given.

```
# Item timeout or time limit for answering in seconds.  Set item timeout
# to 0 (or don't set it at all) if you want no time limits
itemTimeout:          10.0
# Warning time before timeout
itemWarningTimeout:   4.0
```

Figure 15: Example of answering time limit configuration. Total time allowed for answering the questions is 10 seconds. When there is less than four seconds to answer, the time indicator on the subject's UI changes from green to yellow to warn that time is about to end.

```
# Subject sets MCLL
MCLL.subjectSetsLevel:  true
# Default (or fixed) MCL level.
MCLL.default:   -20dB
# The limits of listening level the subject can set.
MCLL.max:         0dB
MCLL.min:       -40dB
```

Figure 16: Example of a MCLL configuration. The test subject sets the level. The allowable range the subject can select a level within is $[-40, 0]$ dB with -20dB as the initial or default level.

```
#session id: SES03
#session start time: Thu Feb 25 15:07:06 GMT+02:00 1999
#session end time: Thu Feb 25 15:08:37 GMT+02:00 1999
#session MCLL: 0.0dB
#ItemID SubjID  SesID   Time/s  Switch  A       B       gB      gA
item1   hynde   SES03   15.7    7       pirr44  pirr32  5.6     1.2
item4   hynde   SES03   18.6    3       pirr11  pirr16  6.5     3.6
item2   hynde   SES03   TIMEOUT 8       pirr22  pirr32          7.9
item3   hynde   SES03   14.3    3       pirr8   pirr11  7.0     2.0
```

Figure 17: Example of the output format of the results.