International Journal of E-Health and Medical Communications
Volume 11 • Issue 2 • April-June 2020

Monitoring Practitioner's Skills in Pure-Tone Audiometry

Alexander Kocian, Department of Computer Science, University of Pisa, Pisa, Italy
Stefano Chessa, Department of Computer Science, University of Pisa, Pisa, Italy
Wilko Grolman, Brain Center Rudolf Magnus, University Medical Center Utrecht, Utrecht, Netherlands

ABSTRACT

So far, there exists no standard, to evaluate a practitioner's skills in pure-tone audiometry. To narrow the gap, this article presents an artificial patient (AP) emulating various types of hearing impairment. In contrast to other solutions, the AP autonomously listens to real pure-tones in soft real-time, while taking into account elements from psycho-acoustics. The emulated patient profiles are reproducible. New profiles can be easily added. The AP is able to recover from error. In this contribution, the authors develop software requirements specifications and derive a modular system architecture. To analyze the performance, the article proposes a stochastic extension to existing synchronous data flow graphs, taking into account the unbounded nature of the tasks' worst case response time. Maximization and summation over the graph reveals the joint distribution of the response time with first and second central moments corresponding to, respectively, the expected response time and the jitter of the task. The theoretical results have finally been validated by measurements on the target.

KEYWORDS

Artificial Patient, Data Flow Graph, Medical Robot, Real Time

INTRODUCTION

Background

Medical malpractice is a rising concern to all healthcare providers. "Otolaryngologists have not been immune to the acceleration of the malpractice crisis" (Hong, Yheulon, Wirtz, & Sniezek, 2014). Medical errors not only contribute to wrongful death and injury of people, but they add more expenses to an already inflated healthcare cost structure. For the USA alone, it has been estimated that a minority (7%) of all malpractice lawsuits reach trial, with only 18% resulting in a plaintiff verdict, topping USD 100 million in just a 6-year period from 2000-2005 (Hong, Yheulon, Wirtz, & Sniezek, 2014). The situation in the European Union is similar. The two top legal allegations against the defendants are failure to supervise and inadequate training. Focusing on the latter in pure-tone audiometry, the question arises as to what kind of standard is applicable to evaluate the skills of a practitioner? Currently, there is no such available. Quality of medical services depends on knowledge and skills.

DOI: 10.4018/IJEHMC.2020040103
This article, originally published under IGI Global's copyright on April 1, 2020 will proceed with publication as an Open Access article starting on January 25, 2021 in the gold Open Access journal, International Journal of E-Health and Medical Communications (converted to gold Open Access January 1, 2021), and will be distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

Related Work

To develop clinical reasoning skills in pure-tone audiometry while avoiding the costs implied by long test sessions with real patients, several patient simulators were developed over the past years (Round, Conradi, & Poulton, 2009). Among them are *Otis - The virtual patient* (Innoforce Est., 2013), *AudsimFlex* (Nova Southeastern University, 2015), and *Audiology Clinic* (Parrot Software, 2009). To comply with the test procedure in New Zealand, Heitz developed in (Heitz, 2013) the Clinical Audiology Simulator (CAS) complying with the test batteries by the New Zealand Audiology Society. All these patient simulators never make mistakes, as they only simulate hearing thresholds in noise-free environments without considering the patient behavior. To close this gap, an autonomous artificial patient (AP) mimicking the patient's hearing thresholds and psychoacoustics in real-time and real-environments is sought.

There exists a large number of integrated development environments. To design applications that require test, measurement, and control with rapid access to hardware, Laboratory Virtual Instrumentation Engineering Workbench (LabVIEW™) is ideally suited (Vose & Williams, 1986). LabVIEW™ is based on a graphical dataflow environment developed by National Instruments and comes shipped with a real-time module, to create life-critical applications such as that of a full-flight simulator (Keenan, 2008). In such cases, the programs run on stand-alone embedded hardware targets such as Ardence Phar Lap ETS, Wind River Systems VxWorks, or NI Linux Real-Time Operating System (RTOS). When reliability of software is soft-critical such as user experience in e-health applications (Raval, Sakinala, Jadhav, & Karia, 2017), LabVIEW™ code may be executed on a General-Purpose Operating System (GPOS) in soft real-time, bypassing the constraints of the RTOS module. That implies incorporation of optimized C-code as Dynamic Link Library (DLL) within the VI, de-allocating memory for temporary variables inside the DLLs, and fixed-point arithmetic operations for the code (Peddigari, Kehtarnavaz, & Loizou, 2007). The performance of the code could be notably improved with parallel computing. However, a small number of specialized developers has only considered this option, so far.

Data flow models and their corresponding analysis methods can be used to derive a periodic schedule at compile time, guaranteeing hard real-time threads meet their deadline (Lee & Messerschmitt, 1987). To schedule a streaming task across multiple processors sharing memory without contention, the original Synchronous Data Flow (SDF) graph is transformed into a Directed Acyclic Graph (DAG) that not only exploits functional parallelism but also data parallelism (Pino & Lee, 1995). When executed in soft real-time, the task still has the same schedule but different response time, mainly because the worst case response time is *unbounded* as the scheduler meets deadlines only in a stochastic way. Current design practice for minimizing the system latency is to i) either annotate each actor in the SDF graph with the worst case response time obtained by simulations and then, use the properties of the graph, to obtain the worst case response time of the entire task (Bekooij, et al., 2005); or to ii) assign a stochastic response time to the entire task (Maxim, 2013) without exploring the properties of the underlying graphical model. In both cases, the predicted task latency is sub-optimal.

Contribution

In (Kocian, Cattani, Chessa, & Grolman, 2018), the authors proposed a multiple-input multiple-output (MIMO) auditory hearing model containing elements from psychoacoustics, developed a hardware realization, and verified its functionality. This paper presents the software architecture of the AP and evaluates its performance theoretically as well as by measurements. In contrast to other solutions, the proposed is based on a theoretical framework supporting physical as well as psycho-acoustic parameters. Moreover, our AP is able to handle not only ipsilateral but also contralateral air and bone conduction testing. The modular software architecture is built on adaptability, autonomy, consistency, extensibility, reactivity and robustness, hosting general-purpose tasks such as operator interfaces and databases but also time-critical tasks such as audio streaming and actuation. The code

is then optimized for execution in soft real-time on multi-core microprocessor using synchronous data flow graphs. A promising approach, we will follow in this contribution, is to annotate each actor in the SDF graph with a stochastic response time that has unbounded probability density function (pdf), and then use the graph to compute the joint pdf for the response time of the entire task in an efficient way. The first and second central moment of the joint pdf correspond to, respectively, the expected response time and the jitter of the entire task, respectively. Measurement results support the proposed theoretical framework.

METHODS AND MATERIALS

The Software Platform

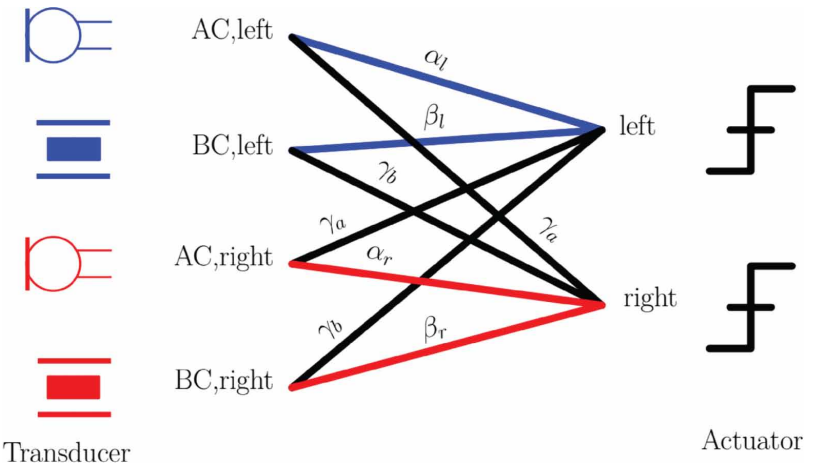
We start with the software architecture, which is based on the human audiometric hearing model in (Kocian, Cattani, Chessa, & Grolman, 2018), synthesizing various types of hearing loss as well as elements from psycho-acoustics. There, it is shown that sound propagation in the human auditory system can be modeled as MIMO array with input acoustic transducers and output actuators, as sketched in Figure 1. The entries of the MIMO array are determined by the left (right) AC thresholds α_l (α_r), the left (right) BC thresholds β_l (β_r) and the AC (BC) interaural attenuation coefficients γ_a (γ_b).

Specifications and Requirements

In user mode, the application software randomly picks a patient profile from the local database and constructs the MIMO hearing model in Figure 1. The software calculates the hearing level of the alleged test tone at the output of the MIMO array and sends control commands to successive actuator just like the auditory nerve fibers provide a synaptic link to the neurons in the auditory brainstem. The software may issue delayed or erroneous control commands with particular probability, as the synapse may fire delayed or erroneously.

The evaluation procedure is organized in sessions and tests. During one session, the same practitioner may perform several tests. At the begin of each test, the AP picks a patient profile from the medical database at random, chooses the right communication port for every hardware device and then, autonomously processes the signals from all the transducers. When the test is finished, the same practitioner may launch another test or leave the session.

Figure 1. A MIMO description of the human auditory system with the loss of signal energy as path metric



In the interactive administration mode, authorized users may monitor the status of the system, to have the earliest warning of failure; calibrate the AP by injecting predefined pure-tones into the acoustic transducers; manage administrator and patient profiles; and visualize the performance of the practitioner.

The software platform has to meet the following requirements:

- **Adaptability:** False alarms, missed detection, and reaction time are emulated dynamically;
- **Autonomy:** The AP is able to carry out its tasks alone without human intervention;
- **Consistency:** The emulated patient profile is reproducible;
- **Extensibility:** New patient profiles can be easily added;
- **Reactivity:** The response time of the AP is upper bounded by the typical reaction time of the normal hearing listener, $\tau_{\max} = 0.19$ seconds (Marshall & Brandt, 1980). Alternately, a time-out shall occur;
- **Real-Time:** To achieve continuous audio streaming without distortion, crackles or even drop outs, audio streaming, signal processing and database query tasks must run concurrently, and allow the data to continuously flow;
- **Memory:** The AP should be able to emulate a large number of different pre-stored patient profiles considering various psychoacoustic aspects, and write the measured hearing levels, background noise, temperature, and hardware calibration sketches on a database from which meaningful reports could be generated. All data has to be secure and accurate besides ACID (Atomicity, Consistency, Isolation, Durability);
- **Backup and Restore:** To recover from failure reliably, the AP must have the ability to backup its state during normal operation and restore its last state after failure;
- **Robustness:** The AP must be able to recover from error.

High-Level Concept

A high-level concept of the software architecture is sketched in Figure 2, highlighting the data flow.

The *Database* module provides a simple API, which allows access to the database connections and drivers. It exposes all functions needed to put, retrieve and modify data. The database consists of patient profiles, calibration data and measurement data.

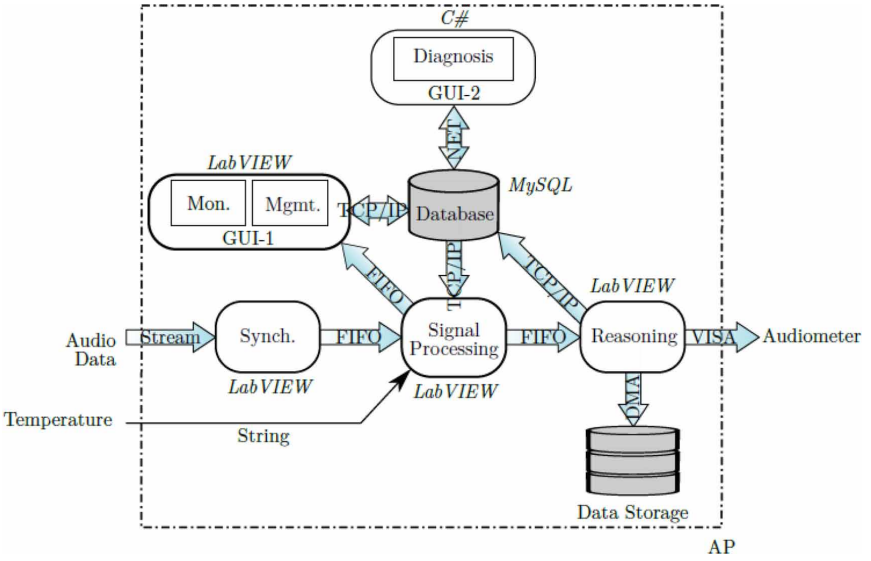
The *Synchronization* module captures and buffers the audio streams arising from the microphones, skull simulators, and background noise. Each synchronized audio stream is forwarded as first-in first-out (FIFO) queue to subsequent signal processing module. Note that the synchronization module has real-time constraints.

The *Signal Processing* module randomly picks a patient profile from the database at the begin of each test by using TCP/IP data socket, processes the synchronized FIFO audio streams in real-time, and ships the calculation result as stereo FIFO queues to subsequent Reasoning module in real-time.

The *Reasoning* module detects the audiometric signal in the received stream and possibly actuates a USB relay, connected to the serial board of the PC. The relay contacts are linked to the audiometer of the practitioner. Whenever the AP detects a valid test tone, log data is forwarded to the database module using TCP/IP data socket. On top of that all audio data is dumped to disk in form of direct memory access (DMA) at the end of each audiometric test.

LabVIEW's internal Graphical User Interface (GUI), dubbed *GUI-1*, includes two attractive features: i) monitoring the audiometric test. The Signal Processing module provides the measurement data using FIFO queue; ii) managing input/output fields for administrator and patient profiles in conjunction with the Database manager using TCP/IP data socket. An additional external GUI, coined *GUI-2*, focuses on analyzing the audiograms obtained by the practitioner. GUI-2 connects to the database within Microsoft's My Structured Query Language (MySQL) connector/Net framework.

Figure 2. High-level software architecture of the proposed AP (Mon.: Monitoring. Mgmt.: Management. Synch.: Synchronization)



For the software implementation, GUI-2 is programmed in C# with Visual Studio (version 12) for the sake of simplicity. GUI-1, the signal processing and reasoning modules are implemented in G, the programming language of LabVIEW™ (version 13). All performance critical parts of the code are realized as DLL that are accessed by LabVIEW™ via the Call Library Function node. All software runs on the general purpose operating system Windows 7.

Shall we follow a thread-based or an event-driven approach for the target application? The debate between the two is an ongoing (van Behren, Condit, & Brewer, 2003). It is worth pointing out that in event-driven systems all handlers are called in the same thread, so that events can only be executed one by one. In thread-based systems, all threads may run concurrently but creating new threads adds overhead in the memory. As a trade-off, we decided to realize the time critical modules Synchronization, DSP, and Reasoning on separate threads using parallel loops but GUI actions and I/O operations event-driven using Event structures.

Database

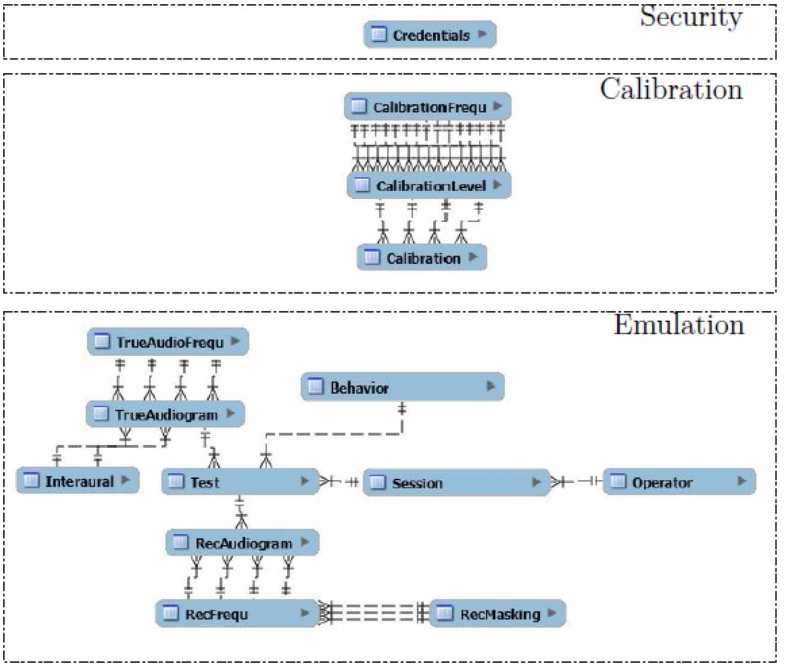
The data in our database is organized according to a database model, which can be relational, or NoSQL. As our data is structured, we follow the former approach. The market provides a large number of software products for relational database management. Top reasons to use MySQL by Oracle under the GNU General Public License are: i) Compatibility with the instrumentation environment LabVIEW™, ii) ACID transaction model, iii) Community Support, and iv) Open Source i.e., free of charge.

The enhanced entity-relationship diagram (EED), showing the relationship between entities of a database and capturing cardinality constraints, can be used to design relational databases. The EED of the database for our AP is shown in Figure 3, indicating three unrelated regions.

Security, containing the credentials of the operator; *Calibration*, accommodating hardware specific reference hearing levels; and *Emulation*, comprising patient profiles, practitioner profiles and measurement results.

We start with the security region. The credentials in Table *Credentials* allow administrator(s) to modify the application software. The credentials are encrypted. For reasons of compatibility with MySQL, we deploy Rijndael's Advances Encryption Standard (AES) algorithm (Daemen & Rijmen,

Figure 3. Enhanced entity-relationship diagram of the proposed AP



2002), known for its very high security. The algorithm is symmetric i.e., the same key is used for encryption and decryption.

The calibration region ensures reproducible hearing test results. The Table *Calibration* contains reference values for a particular hardware, specified by the tuple level L in Table *CalibrationLevel* and frequency f_0 in Table *CalibrationFrequ*.

Finally, the emulation region manages data during an audiometric test. At the beginning of the session, Table *Session* stores information related to the practitioner and the equipment under test. During one session at the beginning of each test, a new patient profile is created based on randomly chosen rows from the Tables *TrueAudiogram*, *Behaviour*, and *Interaural*. The first is related to the true air-conduction and bone-conduction audiograms at frequency *TrueAudioFrequ*. The second comprises elements from psychoacoustics such as false positive/negative response rate and response time. The last table lists transcranial attenuations for AC and BC hearing. The resulting patient profile is documented in Table *Test* along with a link to the logged audio streams. Whenever the AP presumably detects a valid test tone, this table is updated with input hearing loss level and source location. Table *Operator* lists all the authorized users. At the end of the session, the audiograms recorded by the practitioner are migrated to the tables *RecAudiogram*, containing the hearing thresholds at the particular frequencies *RecFrequ* and masking level *RecMasking*. Table 1 lists above entities, their attributes and a brief description.

The database may be installed locally or remotely on a server. To reduce managing costs, we decided to install the database aside the software application on the same machine. Inspecting the query execution plan reveals that only a few data are queried during the audiometric test.

Synchronization

The synchronization thread reads P audio channels from the sound cards and produces digital streams at rate $R = \{R_1, \ldots, R_P\}$. Note that different audio channels operate at different rates. The DSP thread

Table 1. Description of the entities for the proposed AP

Entity	Attributes	Description
Credentials	Username, Password	Login information.
Calibration	ACleft, ACright, BCleft, BCright, Time-stamp	Calibration information for AC and BC testing, at sound level, and octave frequency.
CalibrationLevel	$-20,-10,\cdots,120$ dB	
CalibrationFrequ	$125,250,\cdots,8000$ Hz	
TrueAudiogram	Description, ACleft, ACright, BCleft, BCright, Time-stamp	Patient's true AC and BC hearing thresholds, at octave frequency f_0.
TrueAudioFrequ	$125,250,\cdots,8000$ Hz	
Behavior	FalsePosRate, FalseNegRate, RespTime, Notes, Time-stamp	Patient's psychoacoustic parameters.
Interaural	$125,250,\cdots,8000$ Hz	Transcranial attenuation.
Operator	Name, Surname, Affiliation	Authorized users.
Session	Audiologist, Site, RoomNr, Earphones, BoneOscillator, Notes, Administrator, Time-stamp	Session details.
Test	PathToFile, Session, Audiograms, Behavior, Time-stamp, Notes	Test details.
RecAudiogram	ACleft, ACright, BCleft, BCright, TestID, ReportID, Time-stamp	Recorded audiograms associated with this test, along with min. and max. masking levels at frequency f_0.
RecFrequ	$125,250,\cdots,8000$ Hz	
RecMasking	$125,250,\cdots,8000$ Hz	

consumes this data in chunks at rate R_c. As audio streaming and signal processing operate *asynchronously* at different rates, we have to make a decision on how to buffer the incoming data from the soundcard. Missing buffer deadlines result in signal distortion and drop-outs, causing an erroneous diagnosis by the practitioner. Hence, the buffers are crucial in streaming applications.

The producer and consumer threads may either share data using a common ring buffer, or exchange messages in a distributed memory environment such as first-in first-out (FIFO) mailbox queuing (Martin, 2013). For the sake of simplicity, we follow the former approach.

Suppose the concurrent threads share L fixed-size FIFO buffers of length N forming a ring in the RAM. For $P=1$ producer and $L=2$ FIFO buffers, the producer places a data stream in one FIFO buffer in form of DMA, bypassing the inherently high latency of the operating system, while the consumer processes the data stored in the other FIFO buffer. Once the producer reaches the end of the first FIFO buffer, an interrupt service routine (ISR) is executed, and it will start to fill the second FIFO buffer. As soon as CPU resources are available, the DSP thread reads the data in the first FIFO buffer. As the CPU clock is much faster than the clock in the sound card i.e., $R_c > R_1$, the first FIFO buffer is emptied faster than the second FIFO buffer is filled, preventing buffer overflow, provided the consumer thread is light. When $P>1$ producers write data to the same buffer, every producer has exclusive writing rights, to prevent a race condition. Classical thread synchronization

methods such as mutex locks, condition variables or semaphores cause thread priority inversion aside non-negligible overhead due to locking and thread wakeup. To prevent these setbacks, we follow lockless and wait-free programming. This paradigm is a way of writing thread-safe code such that i) Atomic (single bus) read-modify-write and read-compare-swap transactions guarantee all threads involved in accessing the shared buffer, to never see inconsistent operations. ii) All threads agree on the order in which memory operations occur right down to CPU level, using read-acquire and write-release barriers. In this way, the execution order of an instruction is guaranteed on the local CPU, and so is the visibility order on other CPU cores working on the same bus.

The theoretical latency is directly proportional to the buffer size and inversely proportional to the sampling rate. On top of that, the more producers are active the faster the FIFO buffers are filled. Hence, the minimum and the maximum latency, $T_{\min,\mathrm{FIFO}}$ and $T_{\max,\mathrm{FIFO}}$ have the form:

$$T_{min,FIFO} = \frac{N}{PR_{max}}, T_{max,FIFO} = \frac{N\left(L-1\right)}{PR_{max}} \tag{1}$$

respectively, where $R_{\max} = \max_p R$. The L-size FIFO queue is implemented as array in fixed-point arithmetic (Zeitnitz, 2018).

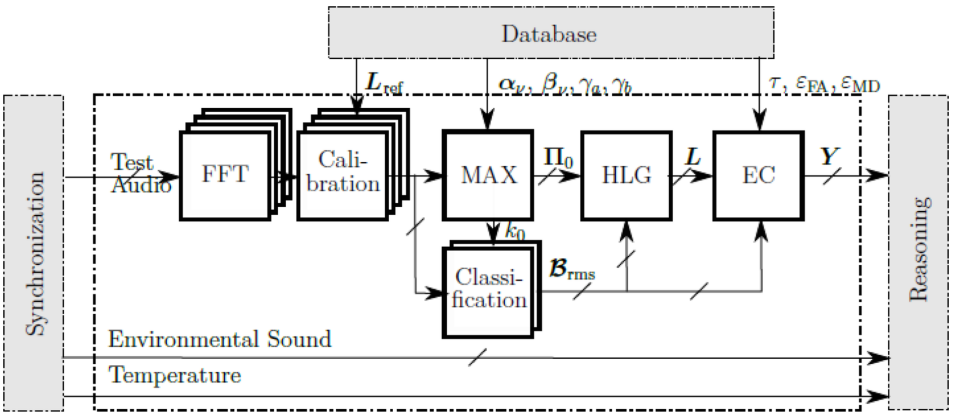
Digital Signal Processing

The logic diagram of the DSP module is depicted in Figure 4 for $P = 4$ producers.

To compute the parameters of the MIMO model in Figure 1, the DSP module needs to load a few tables from the database at the begin of each test, such as *TrueAudiogram*, containing the AC and BC threshold vectors α_ν and β_ν at audiometric frequency ν, $\nu = 1,\ldots,K$, respectively; *Behaviour* comprising response time τ, and response error probability ϵ; and *Calibration* accommodating the hash table L_{ref}.

In pure tone audiometry, the Discrete Fourier Transform (DFT) is ideally suited for the analysis of waves in the time-discrete test signal. The frequency spectrum of the audiometric signal contains N bins, ranging from $-R_i/2, +R_i/2$, $i = 1,\ldots,P$, with frequency resolution R_i/N. When the input size N can be factored into small prime numbers, the Cooley-Tukey algorithm (Cooley & Tukey, 1965) can be used to efficiently compute the DFT with computational complexity of

Figure 4. The Signal Processing module for P=4 producers. HLG.: Hearing Level Generator, EC.: Error Control



$\mathcal{O}\left(N\log N\right)$ operations. Following this approach, the *FFT*-block of our DSP module applies this kind of Fast Fourier Transform (FFT) to each of the P rows of the synchronized $P\times N$ -dimensional input array and generates a $P\times N$ -dimensional output array.

Subsequent *Calibration* block sends the array data to the hash table L_{ref}, to find corresponding calibration results.

The adjusted data is passed to the *MAX* and *Classification* blocks. The MAX block chooses the matrix $\Pi_0 \in \mathbb{R}^{2\times 2}$ with the largest two-norm:

$$\Pi_0 = \max_{k_\nu} \left\|\Pi_\nu\right\|^2, \ for\,any\ k_0 \in \arg\max_{k_\nu} \left\|\Pi_\nu\right\|^2 \tag{2}$$

at center frequency index k_0, where:

$$\Pi_\nu \triangleq \begin{bmatrix} \sqrt{\mathrm{E}_\nu^{(l,l)}\left(\alpha_l,\beta_l\right)} & \sqrt{\mathrm{E}_\nu^{(l,r)}\left(\gamma_a,\gamma_b\right)} \\ \sqrt{\mathrm{E}_\nu^{(r,l)}\left(\gamma_a,\gamma_b\right)} & \sqrt{\mathrm{E}_\nu^{(r,r)}\left(\alpha_r,\beta_r\right)} \end{bmatrix} \tag{3}$$

and $\mathcal{E}_\nu^{(n,m)}\left(\cdot,\cdot\right)$ is the energy along the propagation path from the m-th ear to the n-th synapse, $m,n \in \left\{\mathrm{left}\left(\mathrm{l}\right),\ \mathrm{right}\left(\mathrm{r}\right)\right\}$ at audiometric frequency index k_ν (Kocian, Cattani, Chessa, & Grolman, 2018).

The *Classification* block, in contrast, classifies the received signal based on the root-mean squared bandwidth $\mathcal{B}_{\mathrm{rms}}$ around the optimum frequency index k_0. The entries of the classifier $X \in \mathbb{B}^2$ read:

$$x\left[m\right] = \begin{cases} 1; & \mathrm{Pure-tone} \\ 0; & \left(\mathrm{Masking}\right)\ \mathrm{noise} \end{cases} \tag{4}$$

From the signal matrix in (3) and the classifier in (5), subsequent block *Hearing Level Generator* computes the Hearing Level (HL) vector $L \in \mathbb{R}^2$:

$$L \triangleq 20\log_{10}\left(\Pi X\right) - 20\log_{10}\max\left(\Pi\left(1-X\right), \begin{bmatrix} p_{ref} \\ p_{ref} \end{bmatrix}\right) \tag{5}$$

in dB HL where p_{ref} is the lowest standardized audible sound pressure of 20 μPa (ISO 389-1, 2000).

Finally, the block *Error Control* mimics the psychoacoustics of the patient by distorting the HL vector in (5) with additive white Gaussian noise $N \triangleq N\left(\in_{FA},\in_{MD}\right)$ as a function of false alarm and missed detection probabilities, and by delaying the result according to:

$$Y = \begin{cases} L+N; & \ell \geq \left\lfloor \tau \,/\, T \right\rfloor \\ 0; & \mathrm{otherwise} \end{cases} \tag{6}$$

where τ is the response time of the patient. The result is forwarded as FIFO queue to subsequent reasoning module with adaptive threshold $\Theta\left(\varepsilon_{FA}, \varepsilon_{MD}\right)$.

When switching acoustic signals, the maximizer in (2) tends to produce tentative values until a steady-state is reached. A Finite State Machine (FSM) shall overcome this drawback. Let the state machine $\mathcal{S}$ be the 5-tuple:

$$\mathcal{S} = \left(States, Inputs, Outputs, Update, Init\right) \tag{7}$$

The state space of our state machine comprises three states:

$$States = \left\{Zero, Slope, One\right\} \tag{8}$$

In state Zero, the AP waits for a detectable input signal; in state Slope, the input signal is detected but not yet stable; and in state One, the input signal is stable. Each edge is labeled by an input and an output. The input and output alphabets in (7) are:

$$Inputs = \left\{absent, present\right\} \tag{9}$$

$$Outputs = \left\{keep, update\right\} \tag{10}$$

The input symbols absent and present in (9) correspond to $\left\|\Pi_{\nu}\right\| < p_{ref}$ and $\left\|\Pi_{\nu}\right\| \geq p_{ref}$, respectively, where p_{ref} is the reference sound pressure. For the output symbols in (10), update recomputes Π_{ν}, while keep leaves Π_{ν} unchanged. The Update function in (7) defines the new state and output symbol given the current state and input symbol. Finally, the Init function in (7) defines the initial state Idle indicated by the bold edge. The FSM is sketched in Figure 5.

Adaptive Reasoning

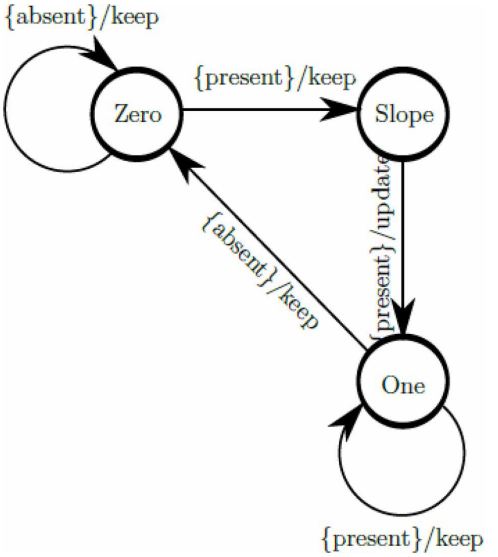
The reasoner classifies each element $Y\left[k\right]$, $k = 1, 2$, of the HL vector in (6) and potentially triggers the actuator. The behavior of the reasoner can be described as FSM, comprising four states:

$$States = \left\{Zero, Idle, Dump, Act\right\} \tag{11}$$

In state Zero, the AP detects the smallest possible hearing threshold $\Theta_{\min}\left[k\right]$; in state Act, the AP activates the USB relay via the National Instrument's Virtual Instrument Software Architecture (VISA) standard; in state Dump, all relevant data is dumped to disk; while in state Idle, nothing is happening. Note that state Zero ensures responsiveness to actual test tones in real environments. It is also worth pointing out that saving data in state Dump occurs after actuation, to minimize latency. Each edge is labeled by an input and an output. The input and output alphabets are:

$$Inputs = \left\{absent, present\right\} \tag{12}$$

Figure 5. State transition diagram describing the behavior of the matrix Π_v in the MAX block



$$Outputs = \{zeroing, actuate, dump\} \tag{13}$$

The input symbols absent and present in (12) correspond to $Y[k] < \Theta(\varepsilon_{\mathrm{FA}}, \varepsilon_{\mathrm{MD}})[k]$ and $Y[k] \geq \Theta(\varepsilon_{\mathrm{FA}}, \varepsilon_{\mathrm{MD}})[k]$, respectively. For the output symbols in (13), zeroing updates the hearing threshold $\Theta(\varepsilon_{\mathrm{FA}}, \varepsilon_{\mathrm{MD}})[k]$. If $\Theta(\varepsilon_{\mathrm{FA}}, \varepsilon_{\mathrm{MD}})[k] < \Theta_{\min}[k]$, let $\Theta(\varepsilon_{\mathrm{FA}}, \varepsilon_{\mathrm{MD}})[k] = \Theta_{\min}[k]$; actuate corresponds to activating the USB relay, while dump is related to saving all relational data in the database and dumping all audio files to disk. For example, the input sequence:

$$(absent, present, present, absent, absent) \tag{14}$$

causes the state response:

$$(Zero, Act, Act, Dump, Idle) \tag{15}$$

as sketched in Figure 6.

Graphical User Interfaces

Figure 7 shows the flowchart for the graphical user interface GUI-1. Starting from the main window, the user may choose between "Operator" menu and "Admin" menu.

The former guides the operator through the audiometric test. The entire test procedure is handled by an invisible tab control. The first tab displays the automatically detected hardware settings i.e., measurement microphones, skull simulator, environmental microphone, relay and temperature sensor along with their serial port numbers. Alternate ports may also be assigned through a dialog box. By pressing the "Ok" button, the next tab is activated. There, the operator

Figure 6. State transition diagram describing the behavior of the reasoner

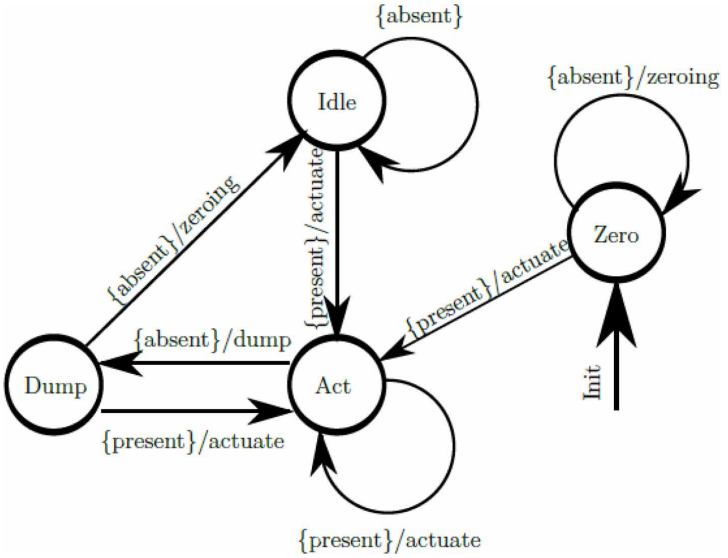
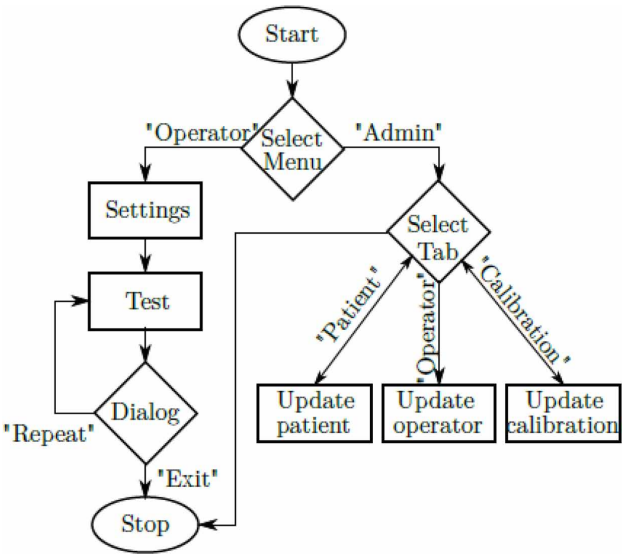


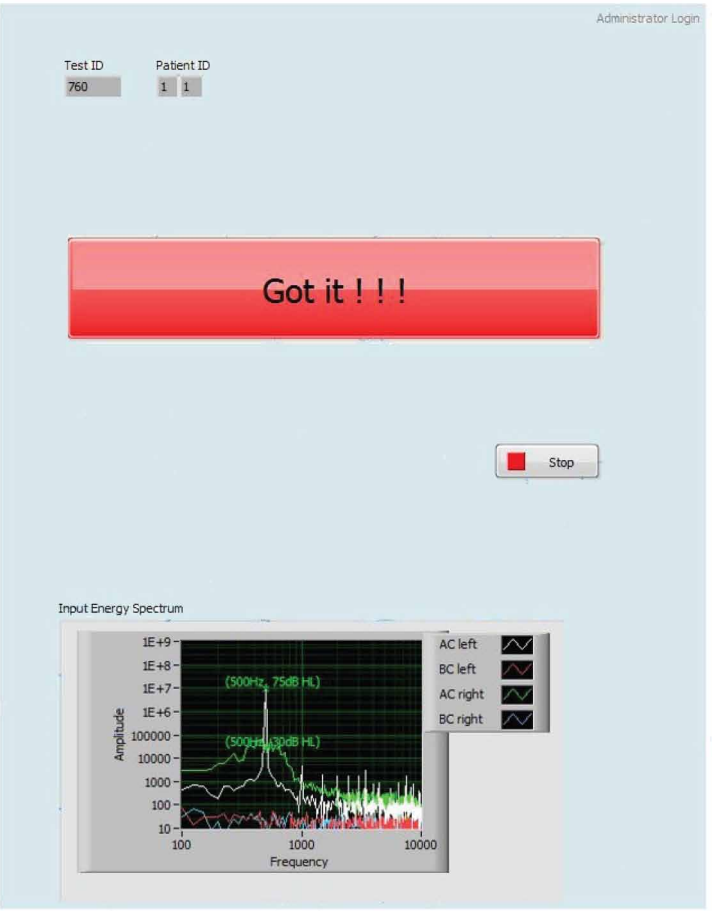
Figure 7. Flowchart for GUI-1, performing monitoring and management



is requested to fill out form fields related to the practitioner, the test site and the transducers used during the session. By pressing the “Ok” button, the operator moves to the last tab, handling the actual audiometric test. As depicted in Figure 8, the layout comprises an indicator bar reflecting the state of the reasoner in Figure 6, a waveform graph showing the input energy spectra at the left/right measurement microphones (AC left/right) and the left/right skull simulators (BC left/right) and a stop button. The data is updated in real-time.

The admin-mode can only be accessed via a password entry dialog box. The password is hashed using the secure hash algorithm SHA-256. This GUI comprises a tab control with four

Figure 8. Snapshot of GUI-1 when the AP is in state Act. Pure-tone at left and masking noise at the right headphone speakers are 75 dB HL and 30 dB HL, respectively.

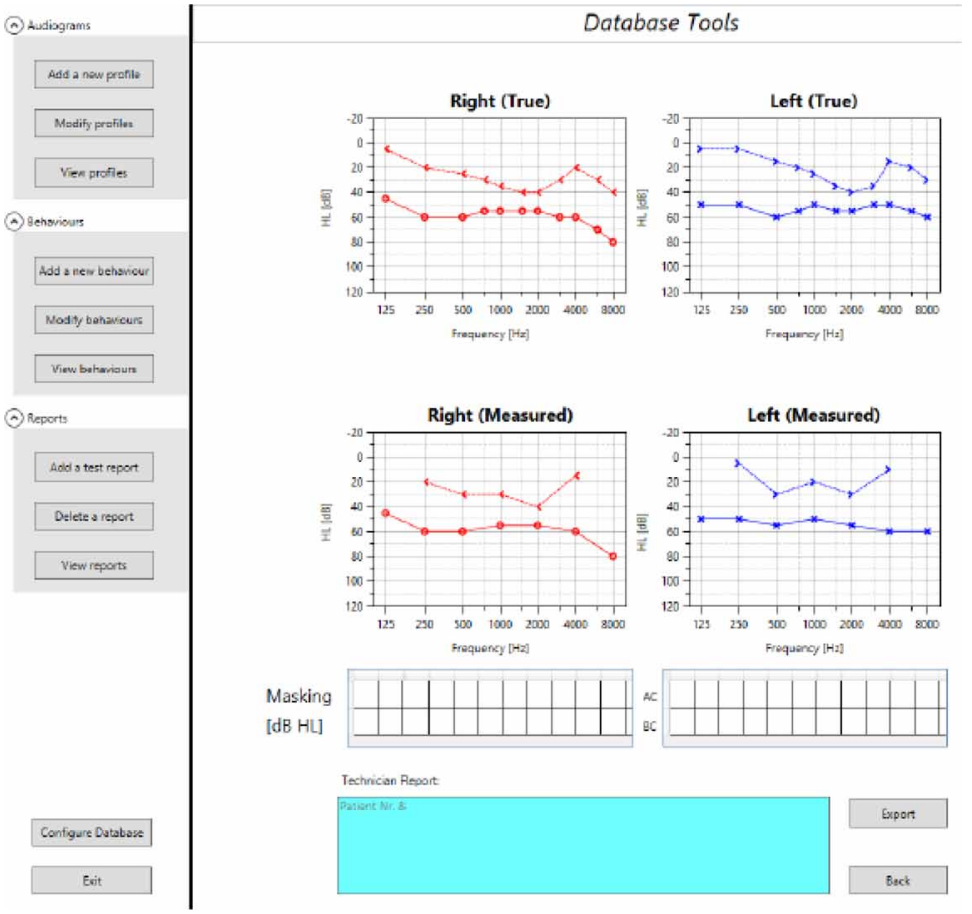


tabs: “Patient”, “Operator”, “Calibration” and “Exit”. The first tab contains buttons to add and delete patient profiles comprising audiograms and behaviors. The second tab allows for adding and deleting operator profiles. In the third tab, the administrator may update calibration points. All data are stored in the local database as described above. The fourth tab leads to a dialog box offering different ways of exit.

The other graphical user interface, GUI-2, is set on top of the diagnostic tool.

Figure 9 shows a snapshot of the C# based graphical user interface GUI-2. Currently, the GUI can be used to add, delete and view reports associated to a particular audiometric tests. By clicking the button “Add a test report”, the application opens a new page with a list of tests. Selecting any element in the list opens a new page with three tabs in a tab control. The first two are for inserting the left and right ear hearing loss levels using combo boxes, respectively. The last one is for adding comments and saving the data in the database. The “Delete a test report” button lets the user select and delete a report from the list of existing reports. The “View reports” button displays a summary. The true audiograms correspond to the emulated version by the AP, while the measured ones are those obtained by the practitioner during the particular test specified in the header of the report.

Figure 9. Snapshot of GUI-2



Stochastic Synchronous Data Flow Analysis

To derive the system behavior of our AP, we rely on data flow models. Data flow programs are directed graphs describing repetitive computation of streaming tasks (Lee and Messerschmitt 1987). The synchronous data flow graph G is a tuple $G = \left(A, E, I, O, D, T\right)$ where:

- A is the set of actors in G;
- $E \subseteq A$ is the set of directed edges in G;
- $I : E \rightarrow \mathbb{N}$ describes the number of tokens consumed from edge $e \in E$;
- $O : E \rightarrow \mathbb{N}$ describes the number of tokens produced on edge $e \in E$;
- $D : E \rightarrow \mathbb{N}$ describes the number of initial tokens on edge $e \in E$;
- $T : A \rightarrow \mathbb{R}^{+}$ describes the (stochastic) response time of actor $a \in A$.

First, we identify the possibly cyclic SDF graph G for our AP, transform the result into the layered DAG $H = \left(A', E'\right)$, and determine a schedule for parallel processors. Then, we develop a theoretical framework for the efficient computation of the joint density function for the response time. Finally, we apply the findings to our particular AP, and analyze its computational complexity.

Schedule for Parallel Processors

Let us start with the high-level SW architecture of the streaming task in Figure 2. The corresponding SDF graph in Figure 10 is composed of $A = 15$ actors such as *Sync* (synchronization), *FFT* (Fast Fourier Transform), $\mathcal{E}_{\nu}^{(n,m)}$ (acoustic energy), *Class* (signal classification), *MAX* (maximum operation), *HLG* (hearing level generator), $\mathcal{E}_{\text{OOK}}$ (equivalent OOK energy computation), *EC* (error control) and *Reasoner*. Database transactions and disk read/write are executed outside the actual hearing test and hence, have no impact on the schedule of our streaming task.

2.2.2. The Homogeneous SDF Graph

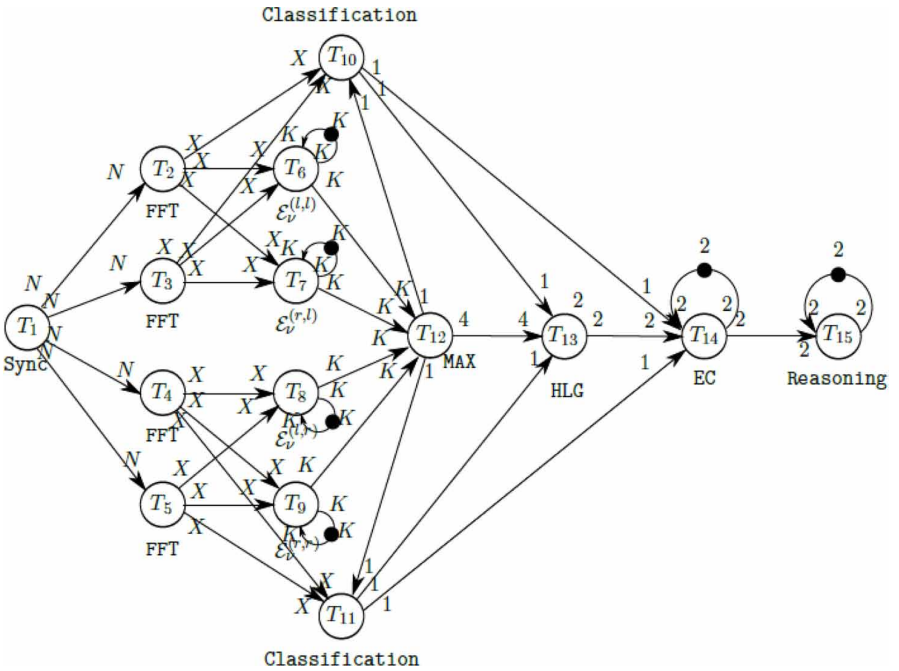
We start with the construction of the homogeneous SDF graph. The topology matrix Γ is a compact representation of the SDF graph that can be used to determine whether the graph is homogeneous. The row entries in Γ are the edges in the graph, and the column entries are the actors in the graph. Self nodes consume what they produce and hence, appear as zero in the matrix (Lee & Messerschmitt, 1987). When the topology matrix has a null-space, the balance equation:

$$\Gamma v = 0 \tag{16}$$

has a non-zero solution. Solving (16) with respect to *v*, the least positive integer vector $v^{\star}$ determines the schedule. In our case:

$$v^{\star} = 1 \in \mathbb{R}^{15} \tag{17}$$

Figure 10. Synchronous data flow graph for the proposed AP with buffer size N and number of audiometric frequencies K. $X \triangleq N / K$.



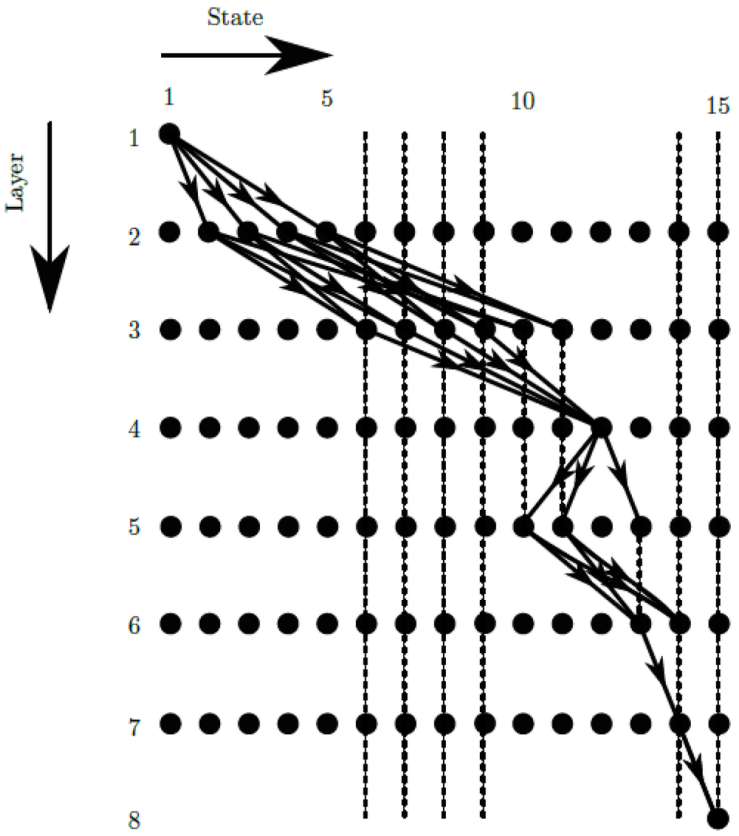
The all-one vector in (17) corresponds to single firing rates, implying the homogeneous SDF and the originial SDF graphs are identical.

2.2.3. The Layered DAG

To transform the directed cyclic homogeneous SDF graph into a directed acyclic counterpart, the actors of the latter are arranged in layers such that the highest and lowest layers consists of the root and the leaf actors a_1 and a_{15}, respectively. Traversing an edge between the actors a_s and $a_{s'}$ in G corresponds to descending from actor a_s to actor $a_{s'}$ in the layered H. Parallel scheduling of DAG's sub-tasks onto a set of processors is a NP-complete problem (Webster, 1997) but good heuristic methods exists. Inspired by the scheduling algorithm of (Hu, 1961), a first worker is assigned to a_1 in layer L_1. When a_1 is finished, then a_2, a_3, a_4, and a_5 in L_2 become starting actors. In the process of assigning workers, a finished actor can be regarded as removed from the graph. This procedure is repeated until the last worker is assigned to the leaf a_{15} in the lowest layer. The resulting layered DAG is illustrated in Figure 11. It can be seen that the maximum number of edges per layer and hence, the number of processors $P = 4$.

Let $\{\Psi_p; p = 1,\ldots,P\}$ be a set of lists where Ψ_p specifies the periodic schedule for processor p. When $P = 4$, corresponding to the maximum number of necessary workers in any layer, above scheduling algorithm yields the blocking-free schedule for parallel processors:

Figure 11. Cycle-free multi-layer representation of the synchronous data flow graph



$$\begin{aligned}\Psi_1 &= \{1,2,6,10\}\\ \Psi_2 &= \{3,7,12,13,14,15\}\\ \Psi_3 &= \{4,8,11\}\\ \Psi_4 &= \{5,9\}.\end{aligned} \tag{18}$$

The resulting path through the DAG is illustrated in Figure 11.

Edges that span two-layers are depicted as solid line while those that span multiple layers are drawn as dashed line. Note that self-nodes in the SDF graph appear as vertical edge in the layered DAG.

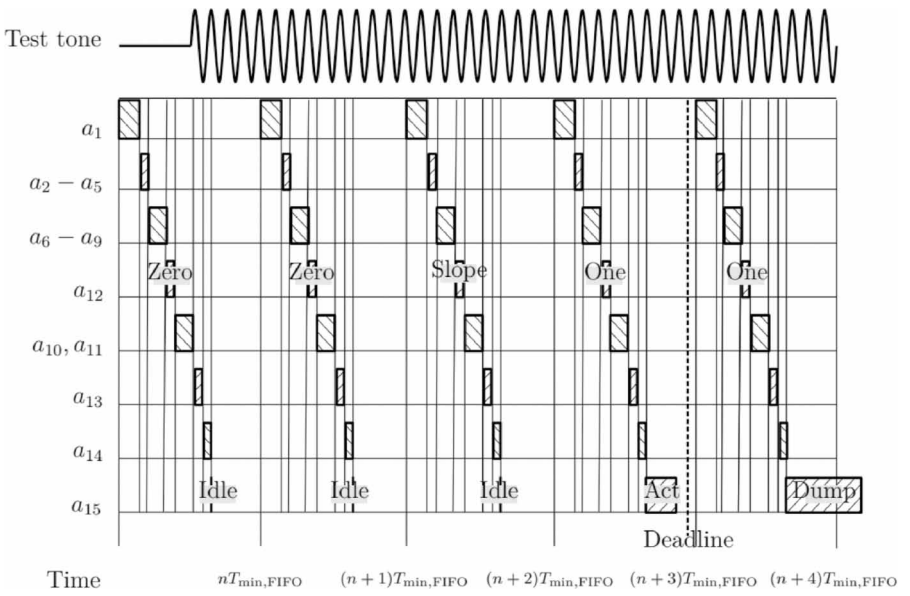
An exemplary timing diagram with boxed state annotations is shown in Figure 12. When the practitioner presents a test tone to our AP, the *MAX* actor a_{12} and the *Reasoning* actor a_{15} change their states as described in Figure 5 and Figure 6, respectively. Eventually after four runs, the *Reasoning* actor is in state *Act* where the USB relay is activated. Note that dump-to-disk is a background process that does not respect deadlines.

To implement the parallel schedule in (18) in LabVIEW-based instrumentation environment, all actors belonging to one schedule Ψ_i are placed in the same while-loop of the code. Note that different while-loops are automatically detected as parallel sections. The built-in Queue Message Handler manages communication among parallel while-loops.

Joint Distribution for the Response Time

To account for the stochastic behavior of the scheduler, let the response time of each actor, T_i, $i = 1,\ldots,A$, be modeled as i.i.d. random variable. Suppose the actor's response time is much larger than the CPU clock sample time so that T_i can be described by a continuous distribution. Without loss of generality, T_i encompasses unavoidable pre-emption delay. When the relative deadline between

Figure 12. Example timing diagram with boxed state annotations for our AP, following the schedule in (18). The USB relay is activated in State Act.



two runs of the streaming task is less than the joint response time $Z = g\left(T_1,\ldots,T_A\right)$, implying zero backlog, the mean response time and jitter can be defined as:

$$\bar{T} \triangleq \mathrm{E}\left\{g\left(Z\right)\right\} = \int_{\mathbb{R}} z\mathrm{f}_Z\left(z\right)dz \tag{19}$$

and:

$$\sigma^2 \triangleq \mathrm{Var}\left\{g\left(z\right)\right\} = \int_{\mathbb{R}} (z-\bar{T})^2\mathrm{f}_Z\left(z\right)dz \tag{20}$$

corresponding to the first and second central moments of the joint pdf $\mathrm{f}_Z\left(z\right)$, respectively. Note that the initial tokens on the edges have no impact on the distribution of Z. The challenge is to find the pdf for the random variable Z.

Sequential Activation of Actors

Consider the graph G in Figure 13, comprising M actors exchanging messages in sequential order according to the balance equation in (16). The total response time Z is the sum of the weighted individual response times i.e.:

$$Z = \sum_{i=1}^{M} v_i^{\star} T_i,\ \ i = 1\ldots,M, \tag{21}$$

where $v_i^{\star}$ is the i-th element in the repetition vector $b^{\star}$. From the linearity of expectations, it follows for the expected response time in (19) that independent of their distribution:

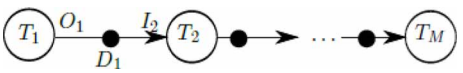
$$\bar{T} = \mathrm{E}\left\{\sum_i v_i^{\star} T_i\right\} = \sum_i v_i^{\star}\mathrm{E}\left\{T_i\right\} \tag{22}$$

If the random variables T_i are independent, their covariances are zero, and the jitter σ^2 in (20) is given by:

$$\sigma^2 = \mathrm{Var}\left\{\sum_i v_i^{\star} T_i\right\} = \sum_i \left(v_i^{\star}\right)^2 \mathrm{Var}\left\{T_i\right\} \tag{23}$$

To find the density of the random variable Z, let us write the sum in (21) as recursion:

Figure 13. Serial activation of the actors. The initial tokens are modeled as FIFO queue.



$$\begin{aligned} Z_i &= Z_{i-1} + v_i^\star T_i \\ Z_0 &= 0 \end{aligned} \tag{24}$$

As Z_{i-1} and T_i are statistically independent, the marginal distribution $\mathrm{f}_{Z_i}\left(z\right)$ is given by (Papoulis 1991):

$$\mathrm{f}_{Z_i}\left(z\right) = \int_{-\infty}^{\infty} \mathrm{f}_{Z_{i-1}}\left(z - v_i^\star t\right) \mathrm{f}_{T_i}\left(t\right) dt \tag{25}$$

corresponding to a convolution between the densities $f_{Z_{i-1}}$ and f_{T_i}. With the assumptions made above, the distribution of the total sum Z tends to be Gaussian $\mathcal{N}\left(\overline{T}, \sigma^2\right)$, as M tends to infinity according to the Generalized central limit theorem (Uchaikin & Zolotarev, 2011).

Parallel Activation of the Actors

Consider the graph G in Figure 14 where M actors concurrently and repetitively transmit messages with individual response time T_i, $i = 1, \ldots, M$ and one actor receives messages repetitively with response time T_{M+1} *after* all messages have arrived. Hence, the total response time Z has the form:

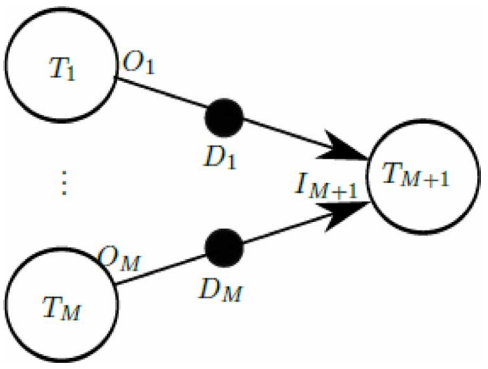
$$Z = \underbrace{\max\left(v_1^\star T_1, \ldots, v_M^\star T_M\right)}_{\triangleq X} + \underbrace{v_{M+1}^\star T_{M+1}}_{\triangleq Y} \tag{26}$$

The response time X corresponds to the maximum of the individual response times T_i, weighted by $v_i^\star$. The cumulative density function (cdf) $\mathrm{F}_{\max(X)}\left(z\right)$ is given by (Papoulis, 1991):

$$\mathrm{F}_{\max(X)}\left(x\right) = \mathrm{f}_{T_1,\ldots,T_M}\left(\left(v_1^\star T_1 \le x\right) \cap \ldots \cap \left(v_M^\star T_M\right) \le x\right) \tag{27}$$

Statistical independence among the parallel actors simplifies (27) to:

Figure 14. Parallel activation of the actors. The initial tokens are modeled as FIFO queue.



$$F_{\max(X)}\left(x\right) = f_{T_1}\left(T_1 \le \frac{x}{v_1^\star}\right) \times \ldots \times f_{T_M}\left(T_M \le \frac{x}{v_M^\star}\right) = \prod_{i=1}^{M} F_{T_i}\left(\frac{x}{v_i^\star}\right) \tag{28}$$

The sum response time $Z = X + Y$ corresponds to convolution between the densities f_X and f_Y. Using the relation $\mathrm{f}_X\left(x\right) = d\mathrm{F}_X\left(x\right) / dx$, it follows from (25) that:

$$\mathrm{f}_Z\left(z\right) = \frac{1}{v_{M+1}^\star} \int_{x=-\infty}^{\infty} \frac{d\mathrm{F}_{\max(X)}\left(x\right)}{dx} \mathrm{f}_{T_{M+1}}\left(\frac{z-x}{v_{M+1}^\star}\right) dx \tag{29}$$

Mean response time $\overline{T}$ and jitter σ^2 have the general form given by (19) and (20), respectively.

When the scheduler iterates through the graph, our algorithm updates the joint probability for the response time according to (25) and (29) until the last actor is reached. The first and second moments of the final result are measures for, respectively, the expected response time and the jitter of the streaming task. For the sake of convenience, above algorithm is dubbed *Max-Sum algorithm*.

Complexity

When the actors are activated in parallel, the computation of the joint probability in (29) dominates the computational complexity. To get basic insight into the behavior of our Max-sum algorithm, consider a scenario with $M = 2$ independent parallel actors. Let all random variables $T_1,\ldots,T_3$ be Normal distributed with mean $\mu_1,\ldots,\mu_3$ and variance $\sigma_1^2,\ldots,\sigma_3^2$. From (Nadarajah & Kotz, 2008), it follows that $X = \max\left(T_1, T_2\right)$ is Skew Normal distributed with mean:

$$\overline{T}_X = \mu_1 \Phi\left(\frac{\mu_1 - \mu_2}{\theta}\right) + \mu_2 \Phi\left(\frac{\mu_2 - \mu_1}{\theta}\right) + \theta\phi\left(\frac{\mu_1 - \mu_2}{\theta}\right) \tag{30}$$

and variance:

$$\sigma_X^2 = \left(\sigma_1^2 + \mu_1^2\right)\Phi\left(\frac{\mu_1 - \mu_2}{\theta}\right) + \left(\sigma_2^2 + \mu_2^2\right)\Phi\left(\frac{\mu_2 - \mu_1}{\theta}\right) + \left(\mu_1 + \mu_2\right)\theta\phi\left(\frac{\mu_1 - \mu_2}{\theta}\right) - \overline{T}^2 \tag{31}$$

where $\theta \triangleq \sqrt{\sigma_1^2 + \sigma_2^2}$. Moreover, $\phi\left(\cdot\right)$ and $\Phi\left(\cdot\right)$ are the pdf and the cdf of the standard normal distribution, respectively. From the sum-rule in (22) and (23), we know that the joint response time $Z = \max\left(T_1, T_2\right) + T_3$ has mean $\overline{T} = \overline{T}_X + T_3$ and variance $\sigma^2 = \sigma_X^2 + \sigma_3^2$. For $M = 4$, we can write $\max\left(\mathrm{T}_1,\ldots,\mathrm{T}_4\right) = \max(\max\left(\mathrm{T}_1, \mathrm{T}_2\right), \max\left(\mathrm{T}_3, \mathrm{T}_4\right)$ requiring $2^{\mathrm{ld}4} - 1 = 3$ recursions of (30) and (31). For $\mathrm{ld}M = k$ where k is integer, then $M - 1$ recursions are needed so that the computational complexity is bounded by $\mathcal{O}\left(M\right)$ operations. Otherwise, dummy actors are added to the parallel branch, so that the condition $\mathrm{ld}M = k$ is met.

EXPERIMENTAL RESULTS AND DISCUSSION

To assess the performance of our AP, we deployed the software architecture developed in Section 2.1 along with the hardware configuration outlined in (Kocian, Cattani, Chessa, & Grolman, 2018), comprising a self-made head, audio source transducers, a fan-less computer and connecting hardware. The practitioner's audiometer generated the audiometric test signals. A prototype of our AP is shown in Figure 15. The system parameters are chosen as follows: sampling rate R = 22050 S/s, $P = 2$ (stereo) audio streams sharing $L = 4$ ring-buffers. The choice of R is determined by the highest frequency used in pure-tone audiometry. The choice of L is a compromise among i) the minimum buffering needed to absorb the stochastic nature of the scheduler, ii) the FIFO latency in (*1*) and iii) the CPU load caused by the interrupt service routine when the buffer is full. Our hardware supports a buffer timing of $N \,/\, R = 40$ ms, corresponding to a buffer length of $N = 882$. To get basic insight into the behavior of the AP, the patient's reaction time τ is set equal zero in the Error Control actor. If not stated otherwise, the test signal had a hearing level of 50 dB HL located at 1 kHz.

In a first experiment, we specify the timing statistics of each actor for the timing diagram in Figure 12 in a semi analytical fashion. We start with Actor a_1 , the L -size FIFO queue that is filled by DMA. From (*1*), the input samples produce bursts of data with idle periods in between. The length of burst and idle periods are geometrically distributed, but *truncated* by the ring size $L-1$ of the FIFO buffer i.e.:

$$\mathrm{f}_1\left(k\right) = \frac{(1-p)^{k-1}\,p}{1-(1-p)^{L-1}} \tag{32}$$

with average (response time):

$$\mathrm{E}\left\{T_1\right\} = \frac{p}{1-(1-p)^{L-1}} \sum_{k=1}^{L-1} k(1-p)^{k-1} \overset{L\to\infty}{\to} 1 \,/\, p \tag{33}$$

Figure 15. Photography of the AP in action

and jitter:

$$\mathrm{Var}\left\{T_1\right\} = \frac{p}{1-(1-p)^{L-1}} \sum_{k=1}^{L-1} k^2 (1-p)^{k-1} - \mathrm{E}\left\{T_1\right\}^2 \overset{L\to\infty}{\to} \frac{1-p}{p^2} \tag{34}$$

Assuming $p = 0.9$, it follows from (33) and (34) that $\mathrm{E}\left\{T_1\right\} = 1.11 T_{\min,\mathrm{FIFO}} = 44$ ms, corresponding to a delay of a little more than a buffer length, and $\mathrm{Var}\left\{T_1\right\} = 0.11 T_{\min,\mathrm{FIFO}}^2 = 183$ (ms)^2, respectively. The response time of the other actors, $a_2, \ldots, a_{15}$, depends on the efficiency of the code besides the latency of the operating system. Their probability distributions are derived from measurements on the target. We proceed as follows: the respective G code is embedded in a flat sequence structure, which is preceded and followed by a tick counter. The time difference between the two tick counters is a measure for the response time of the respective actor. Repeating the measurements, a large number of times, we obtain average and variance listed in Table 2.

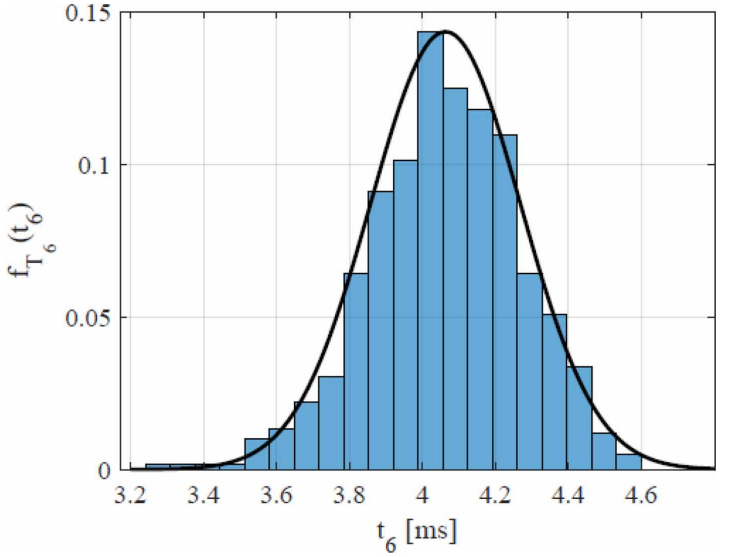
As a representative example for all, Actor's T_6 probability mass function for the response time is shown in Figure 16. It can be seen that the response time can be fit by a Gaussian distribution with high fidelity.

Following this approach, our Max-sum algorithm yields a system execution time Z that has first and second central moments respectively, corresponding to 167 ± 22 ms with 99.7% confidence. The high variance is mainly caused by two sources: i) mostly, Actor T_1 synchronizes on $T_{\min,\mathrm{FIFO}}$ but sometimes it snaps in at multiples of $T_{\min,\mathrm{FIFO}}$; ii) the VISA standard uses buffering to control the USB relay connected to the serial board. In any case, our AP is able to respond almost surely within the required $\tau_{\max} = 190$ ms:

Table 2. Average execution time and variance of the individual actors in Figure 10

T_i	Rate	$E\{T_i\}$[ms]	$Var\{T_i\}$[ms]
1	N	44	183
2	X	0.08	<0.001
3	X	0.08	<0.001
4	X	0.08	<0.001
5	X	0.08	<0.001
6	K	4.1	0.04
7	K	4.1	0.04
8	K	4.1	0.04
9	K	4.1	0.04
10	1	4.1	0.005
11	1	4.1	0.005
12	4	0.02	<0.001
13	2	0.01	<0.001
14	2	0.5	0.02
15	2	0.6 (State I), 10.0 (State A)	0.036 (State I), 12.3 (State A)

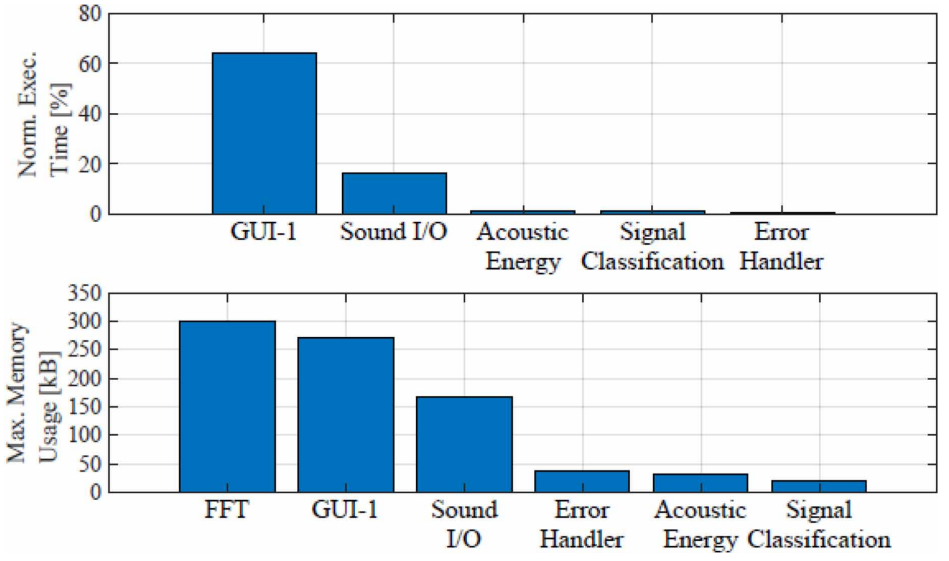
Figure 16. Empirical density function and Gaussian fit for the measured response time of Actor a_6



$$\bar{T} = E\{Z\} = 167ms$$
$$\sigma^2 = Var\{Z\} = 52(ms)^2 \quad (35)$$

We continue with software profiling. LabVIEW's built-in tool VI Profiler is used to record execution time and memory usage of the tasks, and their sub-tasks. Figure 17 illustrates the results sorted in decreasing order. The execution time of the respective task normalized to the execution time

Figure 17. Execution time and memory profile per run



of all tasks per run in percent is documented in the upper plot of Figure 17. The lower plot shows the maximum memory usage per run in kilobytes. For the normalized execution time, it can be seen that the GUI-1 and I/O operations consume 65% and 16% of the total time, respectively. Note that both VIs run in separate threads so that the performance of the audiometric test is not affected. For the VIs related to the audiometric test, the computation of the acoustic energy and signal classification are only active less than 2% of the total time. This high performance is mainly caused by the fact that all critical sub-tasks are imported as C++ DLLs in the respective task. Finally, we want to point out that the normalized execution time for Sound I/O (Actor 1), Acoustic energy (Actors 6-9) and Signal classification (Actors 10-11) are inline with the tick-based version in Table 2. For the memory consumption per run, it can be seen that that FFT causes 299 kB, followed by 271 kB for sound read/write operations, 167 kB for calculating the acoustic energy, and 38 kB for signal classification. The remaining VIs consume even less memory and hence, are omitted from the plot. Note that sampling the input audio stream at a rate of 22050 S/s and 16-bit depth results in a stream of 88 kB/s at the input of the FFT.

CONCLUSION

In this article, we developed the system architecture for a dynamically reacting autonomous patient in pure-tone audiometry. To mimic real patients, our AP listens to real signals in real environments and controls false positive and false negative decision error probabilities as a function of signal level, time and age of the emulated patient from the database. Key properties of our AP are autonomy, consistency, extensibility, reactivity and robustness. The resulting high-level concept is modular, comprising synchronization, (soft) real-time signal processing on a multi-core microprocessor, reasoning, two GUIs, I/O and database management. To bypass the constraints of a real-time operating system and yet achieve accurate reactivity of the AP, the instrumentation environment is LabVIEW™ executed on General Purpose Operating System. Synchronous data flow graph was used to identify a periodic admissible parallel schedule on a multi-core multiprocessor. To accurately predict reaction time and jitter of our AP, we proposed a stochastic approach to synchronous data flow.

The performance has been validated by measurements on the target. Specifically, the AP is able to react within to 167±22 ms and 99.7% confidence coinciding with the reaction time of 190 ms for the normal hearing listener in (Marshall & Brandt, 1980). Task profiling shows that all time-critical tasks are only active 2% of the total time, as they are implemented as linked run-time library. Finally, we observed that all the tasks consume less than 300KB memory per run. Here, the FFT block is the biggest memory consumer. It can be seen that our AP seamlessly emulates reaction time and elements of psychoacoustics such as false response and reaction time in an adaptive fashion. The limitations of work are the dynamic range (currently ca. 80 dB), the minimum reaction time (currently ca. 190 ms), and a missing psychometric function, describing the probability of a positive response as a function of the pure-tone level. Nonetheless, the proposed artificial patient operates in real-time and real-environments, outplaying existing solutions. More features will be added to the artificial patient in future. The proposed approach sets a new quality standard for evaluating practitioners in pure-tone audiometry.

ACKNOWLEDGMENT

The authors thank Leonardo Festa, now with BioBeats, for contributing to the software.

REFERENCES

Andel, C., Davidow, S., Hollander, M., & Moreno, D. (2012). The economics of health care quality and medical errors. *Journal of Health Care Finance*, *39*(1), 39–50. PMID:23155743

Bekooij, M., Hoes, R., Moreira, O., Poplavko, P., Pastrnak, M., Mesman, B., & van Meerbergen, J. (2005). Dataflow analysis for real-time embedded multiprocessor system design. In *Philips Research Book Series* (Vol. 3, pp. 81–108). Springer Dordrecht.

Cooley, J. W., & Tukey, J. (1965). An Algorithm for the machine calculation of complex Fourier Series. *Mathematics of Computation*, *19*(90), 297–301. doi:10.1090/S0025-5718-1965-0178586-1

Daemen, J., & Rijmen, V. (2002). *The Design of Rijndael*. Heidelberg: Springer-Verlag Berlin. doi:10.1007/978-3-662-04722-4

Heitz, A. (2013). *Improving clinical education through the use of virtual patient-based computer simulations* [Dissertation]. Canterbury, New Zealand: University of Canterbury.

Hong, S., Yheulon, C., Wirtz, E., & Sniezek, J. (2014). Otolaryngology and medical malpractice: A review of the past decade, 2001-2011. *Laryngoscope*, *124*(4), 896–901. doi:10.1002/lary.24377 PMID:24105798

Hu, T. C. (1961). Parallel sequencing and assembly line problems. *Operations Research*, *9*(6), 841–848. doi:10.1287/opre.9.6.841

Innoforce Est. (2013). Otis - the virtual patient. Liechtenstein. Retrieved from http://www.innoforce.com

ISO 389-1. (2000, February). Acoustics - Reference zero for the calibration of audiometric equipment. Brussels, Belgium.

Keenan, T. (2008). Developing a real-time full-flight simulator using LabVIEW and Compact FieldPoint. *National Instruments*. Retrieved from http://sine.ni.com/cs/app/doc/p/id/cs-14276

Kocian, A., Cattani, G., Chessa, S., & Grolman, W. (2018). An artificial patient for pure-tone audiometry. *EURASIP Journal on Audio, Speech, and Music Processing*, *2018*(1), 8. doi:10.1186/s13636-018-0131-y

Lee, E., & Messerschmitt, D. (1987). Synchronous data flow. *Proceedings of the IEEE*, *75*(9), 1235–1245. doi:10.1109/PROC.1987.13876

Marshall, L., & Brandt, J. (1980). The relationship between loudness and reaction time. *Acta Oto-Laryngologica*, *90*(1-6), 244–249. doi:10.3109/00016488009131721 PMID:7468185

Martin, T. (2013). *The designer's guide to the Cortex-M processor family*. Elsevier.

Maxim, D. (2013). *Probabilistic analysis of real-time systems* [Dissertation]. Université de Lorraine, Lorraine, France.

Nadarajah, S., & Kotz, S. (2008). Exact distribution of the max/min of two Gaussian random variables. *IEEE Transactions on very large scale integration (VLSI) systems, 16*(2), 210-212.

Nova Southeastern University. (2015). AudSim Flex. Retrieved from http://audsim.com

Papoulis, A. (1991). *Probability, Random Variables, and Stochastic Processes*. Mc. Graw Hill Inc.

Parrot Software. (2009). Audiology Clinic. Retrieved from https://www.parrotsoftware.com/shop/audiology.htm

Peddigari, V., Kehtarnavaz, N., & Loizou, P. (2007). Real-time LabVIEW implementation of cochlear implant signal processing on PDA platforms. In *Proceedings of the Int. Conf on Acoustics, Speech and Signal rocessing (ICASSP),* Honolulu, HI (Vol. 2, pp. 357-360). IEEE. doi:10.1109/ICASSP.2007.366246

Pino, J. L., & Lee, E. (1995). Hierarchical static scheduling of dataflow graphs onto multiple processors. In *Proceedings of the Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* (Vol. 4, pp. 2643-2646). Academic Press. doi:10.1109/ICASSP.1995.480104

Raval, J. A., Sakinala, V., Jadhav, N., & Karia, D. (2017). LabVIEW based real time bio-telemetry system for heathcare. In *Proceedings of the Int. Conf. on Communications and Signal Processing (ICCSP),* Tamilnadu, India (p. 2153-2156). IEEE.

Riek, L. D. (2017). Healthcare robotics. *Communications of the ACM*, *60*(11), 68–78. doi:10.1145/3127874

Round, J., Conradi, E., & Poulton, T. (2009). Training staff to create simple interactive virtual patients: The impact on a medical and healthcare institution. *Medical Teacher*, *31*(8), 764–769. doi:10.1080/01421590903127677 PMID:19811216

Sriram, S., & Bhattacharyya, S. (2000). *Embedded multiprocessors: Scheduling and Synchronization*. NY, USA: Marcel Dekker, Inc.

Uchaikin, V. V., & Zolotarev, V. (2011). *Chance and Stabiity: Stable Distributions and their Applications*. Walter de Gruyter.

van Behren, R., Condit, J., & Brewer, E. (2003). Why events are a bad idea (for high-concurrency server. In *Proceedings of the 9th Workshop on hot topics in operating systems (HOTOS)*. Academic Press.

Vose, G. M., & Williams, G. (1986). LabVIEW: Laboratory virtual instrument engineering workbench. *Byte*, 84–92.

Webster, S. T. (1997). The complexity of scheduling job families about a common due date. *Operations Research*, 65–74.

Zeitnitz, C. (2018, September 20). *WaveIO*. Retrieved from https://www.zeitnitz.eu/waveio

Alexander Kocian received the Dipl. Ing. Degree in Electrical Engineering with a major in telecommunications from the Vienna University of Technology, Austria, in 1997. He finished his Ph.D. degree in Electrical and Electronic Engineering from Aalborg University, Denmark, in 2003. Since 2014, he has been a Research Fellow with the Department of Computer Science, University of Pisa, Italy. He was a Research Fellow (2012-2014) with the Institute of Informatics and Telematics at the Italian National Research Council in Pisa, and (2008-2012) with the Department of Electronics Engineering at the University of Rome "Tor Vergata", Rome, both in Italy. He was a Reader (2005-2008) with the Department of Electronics and Communications Engineering at Birla Institute of Technology (BIT), offshore campus Muscat, Oman. Before that, he was working (1999-2005) in the Digital Communication Laboratory at Aalborg University, Denmark. During his stay at Aalborg University, he joined the Wireless Systems Laboratory at Georgia Institute of Technology, Atlanta, GA, USA as visiting research scholar (2001). He worked on numerous projects funded by Industry, the European Space agency, the Italian National Research Council, and the university. He has co-authored dozens peer-reviewed journals and conference proceedings, and he has been member of numerous program committees of international conferences. He is co-founder of the Master program in Mobile Internet Communications at Aalborg University. His current research areas are in foundation of probability and statistics, machine learning, signal processing, and ambient intelligence.

Stefano Chessa received the PhD degree in Computer Science from the University of Pisa in 1999, and currently he is Associate Professor at the Department of Computer Science of the University of Pisa, vice-chair of the BSc and MSc curricula in "Computer Science" of the University of Pisa and also Research Associate of the Information Science and Technology Institute of the CNR. He has worked several EU projects, in particular he had been site leader of the EU FP7 projects RUBICON and DOREMI. He has co-authored more than 150 papers published on international journals and conference proceedings, and he has been member of several program committees of international conferences. In the period 2011-2013 he was a chair of the Steering Board of the EvAAL Competition (Evaluating AAL systems through competitive benchmarking). His current research interests are in the areas of internet of things, smart environments, ambient assisted living, activity recognition and indoor localization.

Wilko Grolman started his academic career by starting to study economics at the University of Amsterdam in 1984. Dissatisfied with the study and set-up of the education he switched to medicine the next year. Overly seriously he finished the first 5 year within the set date and started with research at the department of ENT at the Academic Medical Center. He started his PhD study on "Voice rehabilitation after total laryngectomy" and started his specialist training in 1995 under the guidance of Prof Paul Schouwenburg. After achieving his medical specialist degree and in the meantime his PhD he decided to stay at the same department he trained. Soon he became the head of the polyclinic. In 2009, he was appointed prof and chair at the department of ENT at the University Medical Center in Utrecht, the Netherlands. Since 2010, he has been involved in the management of the surgical division at the same university hospital. Wilko was head of the division until January 2016. In January of 2016, he stepped back from the challenging managerial arena. He now focuses on research and patient care. Interests: Otology, cochlear implantation, stapes surgery, video conferencing, research, EBM, health economics, telemedicine and live surgery. As of September 2018, he left the University of Utrecht and now works in France at the renowned Causse Ear Clinic. He works at a medical liaison for the Telemedicine Studios (St. Oorzaken) which together with the LION-foundations perform internet based live surgical educational broadcast. Wilko is currently the president of the LION foundation which promotes medical education (LION-Web.org).