Int. J. Adv. Res. Sci. Technol. Volume 4, Issue 6, 2015, pp.414-420.

IJARST

International Journal of Advanced Research in Science and Technology

journal homepage: www.ijarst.com

ISSN 2319 – 1783 (Print)

ISSN 2320 – 1126 (Online)

Testing the Clinical Programming Software of Cochlear Implant System.

P. Seetha Ramaiah[1], B. Uma Maheswararao[2], M. Sreedevi[2], Chiranjeevi. A[2] and Putta Anushaa[2]*

[1]***Dept. of CS&SE, Andhra University, Visakhapatnam, India.***

[2]***Dept. of CSE, Indo American Institutions Technical campus, Sankaram, Anakapalle, Visakhapatnam, India.***

****Corresponding Author's E-mail: hasini.view@gmail.com***

ARTICLEINFO

Article history:

Received 24 Aug. 2015

Accepted 21 Sep. 2015

Available online 28 Sep. 2015

Keywords:

cochlear implant,
clinical programming software,
bugs,
software safety,
hazard.

ABSTRACT

Computer based bio-electronic systems are used for replacement of damaged human parts such as Bionic-ear for deafness, Bionic-eye for blindness, Deep Brain Stimulator for diseases of the brain, and Bionic-arm for arm prostheses. Algorithms for controlling bionic system are based on the specific bionic devices like bionic ear with sound processing software and bionic eye with image processing software. A lot of research is going on to improve the performance of bionic systems in respect of achieving near 100% functionality, low cost and smaller sizes with higher reliability and safety. The aim of the paper is to design, development and testing for different bugs in clinical programming software (CPS) of a Cochlear Implant system that is also called as Bionic Ear. The focus of this paper is identification of different types of bugs like some of the common programming errors for CPS with suitable examples, Requirements, and Coding bugs. Testing must show that hazards have been eliminated or controlled to an acceptable level of risk. This collection is to create programmer aware of these errors of CPS of cochlear implant, so they avoid them and also develop the VB.NET based Clinical Programming for data monitoring of patient with profoundly deaf. In this some inadequate programming practices that are often the cause of many kinds of bugs in CPS.

PAPER-QR CODE

IJARST

Volume-4, Issue-6

Citation: *Ramaiah. et al. Testing the Clinical Programming Software of Cochlear Implant System. Int. J. Adv. Res. Sci. Technol. Volume 4, Issue 6, 2015, pp.414-420.*

Introduction:

Safety critical software performs functions which could directly cause serious injury to people or/and the environment and cause deaths if they failed to execute properly. Some fields of safety critical software systems are Medicine, Nuclear engineering, Transport, Aviation, Aerospace, and Civil engineering, military. There is a huge demand for low cost and high performance of Cochlear implants in developing countries. Several researchers proposed and attempt to develop a low-cost cochlear implant system but none of these products are available in the market. The development of Cochlear Implant system involves the potpourri of mechanical engineering, physiology, electronics engineering and computer science and engineering. The cochlear implant device (CID) has four hardware modules such as are Speech Digital Speech Processor System (DSPS), Impedance Telemetry Monitoring System (ITMS), Headset Cable (HSC), and Implantable Receiver Stimulator (IRS). ITMS are used for finding the active electrodes of electrode array of 12 Electrodes and their respective Threshold Comfort Level (TCL) and Most Comfortable Level (MCL).The valid Electrode numbers with their TCL/MCL are downloaded from PC and upload into speech processor (DSPS) using CPS) in PC. DSPS is connected to patient with implant using headset cable and set comfortable volume levels in DSPS. The main functions of speech processor are speech signal processing, speech data encoding and transmission of encoded speech data to Implanted Receiver Stimulator

(IRS) via transcutaneous RF inductive link. Implementation of clinical programming software without error plays an important role in the development of different modules such as maintaining patient information, impedance measurement, and determination of stimulation parameters (fitting) and mapping THL/MCL of active electrodes into Speech Processor. A failure is caused in CPS software because of a bug and may alter the external behavior of the program.

A software bug can be defined as that part of the code which would result in an error; fault or malfunctioning of the program [1].Some bugs can be detected easily during development. But some bugs will be found late in the development process. These are low probability errors which are hard to detect and occur on very sparse set of inputs [1]. According to IEEE standards, a 'bug' is an incorrect instruction in a program. A failing is triggered because of a bug and may alter the exterior actions of the system. The major categories are requirement and functionality bugs, structural bugs, data bugs, coding bugs, interface, integration and system bugs, test and test design bugs. These bugs increase the cost and development of CPS of cochlear implant system. The primary task of programmer is minimizing these defects, identify and eliminate existing bugs early in the development process. The defects detected beginning will cause much smaller harm than those which are recognized later in the utilization. A common category of bugs can be done in accordance with the regularity of the incident of that bug and severity of that bug. The effect of bug is depending on the software and the system. However we can generally have following sessions for severity:

Catastrophic: potential of multiple deaths or serious injuries;

Critical: Defects that could cause serious consequences for the system like losing some important data or potential of death.

Marginal: potential of injury.

Negligible: These may not necessarily hamper the system performance, but they may give slightly different interpretation and generally avoidable.

Software-based medical devices are playing an increasingly important part in medical device software [7]. The increased demands on such devices have resulted in increased software complexity and have created powerful growth development challenges for their manufacturers [8]. Programs are vital in guaranteeing traceability from specifications via requirements to execution [9]. In order to market their devices within a nation, a medical device development company must adhere to the regulating specifications of that nation [10]. Although assistance prevails from regulating bodies on what application actions must be performed, no specific method for performing these actions is defined or required [11]. The variation between the documents extends to other areas such that "In already developed guidance documents different software quality issues, software lifecycle phases and consequences of risk evaluation for software malfunction or fraud are addressed to different extents"[12]. A cochlear enhancement (CI) is a real-time included processing system that can provide a sense of audio to people who are hard of hearing or significantly hearing-impaired [4] [5] [6]. Software had been introduced when changes were made to the software after its initial production and distribution [3]. Beginning electric stimulation tests in the auditory system were conducted with individual electrodes placed in the inner and center ear if the hearing nerve [16, 17]. The first electrodes with mechanical properties were designed at university of California [18, 19]. Cochlear implant system with less than 8 electrodes does not produce the good outcomes in digital speech processor system from various researches [20-22]. Thus, an eight channel program was sufficient to assurance a high level of speech understanding [23]. The cochlear implant system has been continuously improved front-ending

The rest of this paper is organized as follows: section 2 describes related work to medical device software. Section 3 addresses the requirements and programming bugs in CPS. Section 4 applies the test and validation of CPS. 5 conclude the discussion.

Related Work to Medical Device Software:

Accidents occurred due to failure of medical device software:

Cedar Sinai Medical Centre in Los Angeles, California happened between the years February 2008-August 2009 because of Software miss-configuration in CT scanner used for brain perfusion scanning and get Very high level of damage (wrong dosage to 206). A Medtronic heart device was found vulnerable to remote attacks in March 2008. The Therac-25 is a medical linear accelerator manufactured by AECL. A linear accelerator is a particle accelerator, a device that increases the energy of electrically charged atomic particles. The charged particle is accelerated by the introduction of an electric field, producing beams of particles which are then focused by magnets. Linacs are used to treat cancer patients. A patient is exposed to beams of particles, or radiation, in doses designed to kill a malignancy. Since malignant tissues are more sensitive than normal tissues to radiation exposure, a treatment plan can be developed that permits the absorption of an amount of radiation that is fatal to tumor cells but causes relatively minor damage to normal tissue. Shallow tissue is treated with electrons, but to reach deeper tissue, X-ray photons are needed (Grolier, 1985).Accidents occurred because of Therac-25 are

Table: 1. Causes of hazards

Date	Country	Cause
June 3, 1985	Marietta, GA	Breast removal, loss of use of arm
July 26, 1985	Ontario, Canada	Total hip replacement needed
January 6, 1986	Yakima, WA	Minor disability and scarring
March 21, 1986	Tyler, TX	Death
April 11, 1986	Tyler, TX	Death
January 17, 1987	Yakima, WA	Death

Radiation therapy software by Multi data Systems International miscalculated the proper dosage, exposing patients to harmful and in some cases fatal levels of radiation. The physicians, who were legally required to double-check the software's calculations, were indicted for murder.

Cause: The software calculated radiation dosage based on the order in which data was entered, sometimes delivering a double dose of radiation.

Software defects in Infusion pumps:

Software is found in medical care in a variety of safety-critical programs, from patient records to radiology devices. Over 56000 medical device reports, 1% were reported as deaths, 34% as serious injuries, and 62% as malfunctions. Many of the issues of infusion pumps that have been revealed are relevant to software *malfunctions*. For example, some pumps do not to activate pre-programmed alarms when problems happen, while others activate an alarm in the absence of a problem. Other software errors can lead to over- or under-infusion. In one case, a program issue known as a 'key bounce' triggered an infusion push to sometimes sign-up one key stroke as several key strokes.

Common Errors in CPS of Cochlear Implant:

Requirements bugs in CPS

Requirements and the specifications developed from errors of CPS can be imperfect, uncertain. They can be difficult to understand. The requirements bugs in CPS are wrong patient's treatment history records retrieved, current treatment profile appear to wrong patient's record, communication failure in ITMS, Incorrect calculation of volume is too high and volume is too low, wrong measurement of impedance and neural responses of active electrodes in ITMS, Incorrect identification of active electrodes in the implanted unit of IRS, Storing the Impedance values of all electrodes in the internal memory is failed by using CPS, new CPS system is implemented and upgrades are not done properly, Wrong maintenance of CPS software, For the new patient, do not follow the all steps to identify the implant and electrode type, Alter files used by the Fitting Software, The display of Connection to hardware failed message, Running of other programs in PC of clinical programming software, Telemetry measurement not possible message, Control program software is corrupted, Adjustable points TCL & MCL for setting the Minimum and maximum comfortable levels of the Electrodes are wrongly stimulated in cochlea, unwanted background noise is generated by DSPS.

Software Fault Tree Analysis (SFTA) of CPS:

SFTA aids in the analysis of events, or combinations of events, that will lead to a hazard. This SFTA demonstrates the ability to integrate hardware, software, and system events. Figure 1 shows the sample fault tree for the top event the display of Hardware not found message [4].

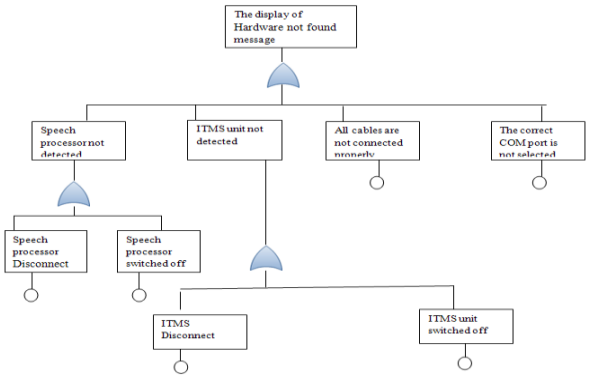

Fig.1: SFTA for Hardware not found message

Programming bugs in CPS:

The common bugs which happen during coding which directly or indirectly manifest themselves to a bigger harm to the running program or patient.

Memory leaks in CPS

A memory leak is a situation where the memory is assigned to the system which is not released consequently. The problem of memory leaks has affected developers in c and c++. In vb.net, an extensive garbage collection package and managed memory can stop memory leaks, but, under some conditions, a memory leak can occur when you use unmanaged code as part of the application.

```
Dim ds1 as New DataSet
ds1 = GetDataSet ("Select * from ActiveChannels where recipientID = '" & pubRecipientID & "'")
    If ds1.Tables (0).Rows.Count > 0 Then
    Label8.Text = ds1.Tables (0).Rows (0).Item (5).ToString
    End If
```

This is not a memory leak, but instead it is a problem in managing the memory like so many programming

errors occurred in CPS such as Inappropriate compression strategy in CPS, not ordering appropriate options, not activating or programming additional memories, not incorporating past history into programming logic, over-reliance on first fit, not entering bone conduction scores when an air bone gap is present,

Run time issues management of CPS:

An operating-system kernel and application programming interface often perform the most important role in a safety-critical system. Exception handling, deadlocks, process and stack management, scheduling and flow control and memory protection all have repercussions on the safety function and can be key elements of meeting safety-integrity requirements. Traditional testing techniques such as unit testing are ad hoc and informal. It is only a partial proof of correctness in that it does not guarantee that the system will operate as expected under untested inputs. In terms of its ability to guarantee software correctness, runtime verification is stronger than testing. Testing can only guarantee the correctness of a limited set of inputs at implementation time. As a result, undiscovered faults may result in failures at runtime and even allowing the system to propagate corrupted output because the failure was not detected. By always monitoring the software for correctness, such failures can be caught when they happen, for any input which causes them to occur. Runtime verification is a lightweight verification technique that complements traditional techniques such as model checking and testing [2]. It checks whether the current execution of a system under scrutiny satisfies or violates a given correctness property. One of the main distinguishing features of runtime verification is that it is performed-as the name suggests-at runtime. This opens up the possibility not only to detect incorrect behavior of a software system but to react whenever incorrect behavior is encountered.

Other common errors in CPS:

Temporary values returned is an error in CPS which cause catastrophic damage to running program. These errors occurred rarely in CPS. If some information needs to be shared, then declare the variable as public and use that to access the common data. Need of Unique addresses is a bug in clinical programming software. These types of errors occur frequently and effect is major. When we expect source and destination addresses are different, but they have same address. If we perform string concatenation function, then we may get runtime error. To prevent this, keep a check on the factors before using them. Race condition is an error in CPS which outcomes when two processes try to access the same resource and the result depends on the order of the execution of the processes. These types of errors also occur frequently and effect is major. Uninitialized storage is rarely happened in CPS with major harm [3]. Safety initialization is solution to such problems.

Race condition:

Race condition is an error in clinical programming software which results when two processes try to access the same resource and the result depends on the transaction of the performance of the processes. These types of errors occur frequently and effect is major. Example of medical device such as cochlear implant device software safety issue of safety that have been responsible for patients' deaths is some of incidents involved the Therac-25 radiotherapy device in the mid 1980's, when a software design error several patients to be killed from radiation dosages purchases of scale greater than what had been designed. The immediate cause was a so-called "race condition" where rapidly-entered individual feedback took the device into an unexpected state. Consider the following example which illustrates this error.

```
Public class form1
    Private y as Integer
    Set (byVal x as integer)
    If x <= y then
    y=y-x
    Return 1
    End If
Return 0
```

If two processes simultaneously call set(y), then even though y value not enough for both of them, they may both be returned ‘1’. It creates a critical section.

Testing & Validation of CPS:

Testing and Validations of CPS tool can be done in four sections these are

1) Measuring Impedance Values of active electrodes,
2) Finding and Fitting the THL/MCL values of active Electrodes,
3) Programming the BWSP unit with found THL/MCL values of active Electrodes, and
4) Performance testing of DSPS unit after programming the required parameters.

These four sections are described in briefly with Laboratory model Experimental test setup in following sections.

To the measuring of impedance values of each active electrode with respect to Reference electrode (ELR) where placed in amount of Ringer Lactate (RL) Saline water which is equal to human body fluids in a small beaker at the same time tap the electrode array points sensitively with the help of electrode array Test

jig, this test jig also connected to the Data Acquisition System (Wave Book) through connector. Place the Coil side of headset cable over the Implant coil of IRS and connect the other end of headset cable to the ITMS unit. Then connect ITMS unit to PC through serial port as shown in figure below.

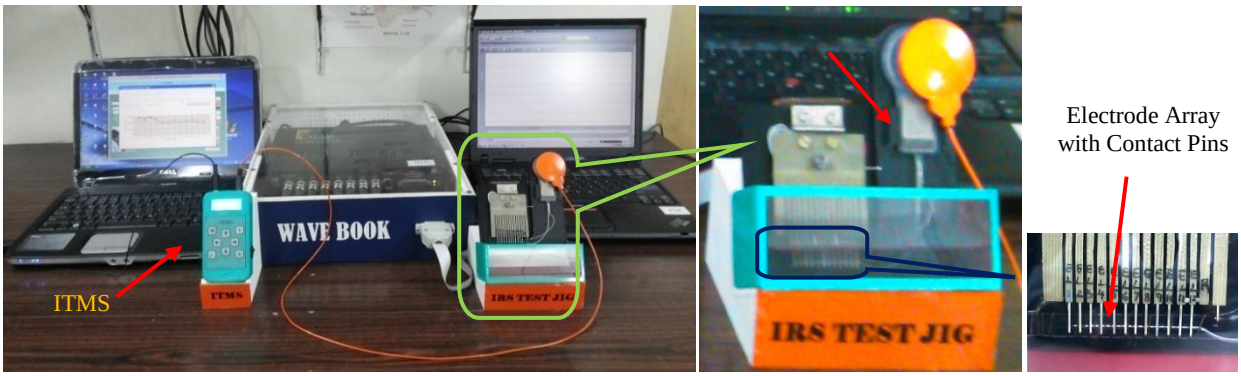

Fig. 2: Image of Experimental Test setup for Reading Impedance values of RL Saline water

After done the connections as per the above test setup then power ON both PC/LAPTOPs and run the CPS software in left side PC and run Data Acquisition Software on right side PC, and power ON Data Acquisition instrument(Wave book).

Power ON ITMS unit and press M key in front panel of ITMS. In CPS software connect ITMS unit by adjusting port settings in home page, then click on "IMPTEL" button for measuring the impedances of RL saline water at electrodes with corresponding to ELR point, after completed measuring then it stores in the data base automatically and it displays the numeric values as well as Graphical Representation as shown in figure below.

Table: 2. Impedance Values Table:

Electrode No. / Impedanc(Ω)	1	2	3	4	5	6	7	8	9	10	11	12
	4950	4671	4759	4026	3315	3459	3459	3459	3459	3459	3747	3459

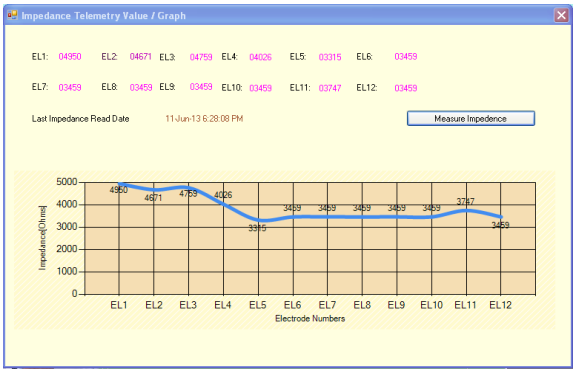

Fig. 3: Output Values Graph for Measured Impedance Values of Active electrodes

After complete the measuring Impedance values close the Impedance Telemetry Value tab then open Fitting THL/MCL Values tab for fitting the THL/MCL values from fitting tab is used to measure optimized values of the THL and MCL values for each active electrode/channel. Whenever this tab is loaded it reads the impedance values from the data base and displays in the corresponding textboxes of each channel and creates a new row in mapping table with default values of THL and MCL as shown in figure below.

Fig. 4: Image for fitting THL/MCL values Tab after

The audiologist may vary THL as well as MCL of particular channel by moving the dynamic range bar upwards or downward and then press Stimulation button. If the determinations of all THL and MCL values based on patient requirements are done, the audiologist may update the THL and MCL values in the Mapping table. The stimulation levels are either THL which is the minimum electrical level that the

patient can perceive and the MCL is the maximum stimulation level that the patient can accept without experiencing uncomfortable sensations. The acoustical dynamic range in each audio band is mapped into the electrical dynamic range defined by the THL and MCL levels as estimated for each channel by the audiologist. This estimation is very important and varies from patient to patient.

After completing the fitting procedure by the audiologist then go for Programming the BWSP unit from the mapping tab by clicking the **generate file** button it generates three files which gives the comprehensive information about active channels and corresponding threshold and comfort levels of the active electrodes. The Body Worn speech Processor (BWSP) must be programmed in order to adapt the cochlear system to patient requirements. It is programmed taking into account the functionality of each electrode and the stimulation levels required for each channel. The Experimental Test Setup for Programming BWSP unit as shown in figure below.

Fig. 5: Experimental Test Setup for Programming DSPS unit

Conclusion:

Clinical programming software was developed for cochlear implant for bionic ear and identified the requirement and programming bugs which are occurred in CPS. In this paper, a basic overview of medical-device software has been given and some techniques used to test CPS have been described for measuring impedance values, programming DSPS. The main idea behind the testing techniques mentioned is to reduce the programming and requirement errors.

References:

1. List of Common Bugs and Programming Practices to avoid them ,Vipindeep V, Pankaj Jalote Indian Institute of Technology, Kanpur
2. Leucker, M., 2008. Checking and enforcing safety: Runtime verification and runtime reflection. http://ercim-news.ercim.org/content/view/459/699/
3. Microsoft Testing Unit. Common coding errors.
4. ANS/IEEE 1061-1998.IEEE Standard for a software quality metrics methodology.
5. Graeme M. Clark,(2006), "The multiple-channel cochlear implant: the interface between sound and the central nervous system for hearing, speech, and language in deaf people—a personal perspective", Phil. Trans. R. Soc. B , Vol. 361,pp 791–810
6. M. McHugh, F.McCaffery, V. Casey, Software process improvement to assist medical device software development organizations to comply with the amendments to the medical device directive, The Institution of Engineering and Technology 6 (2011) 431–437.
7. S.R. Rakitin, Coping with Defective Software in Medical Devices, Computer, 2006, pp. 40–45.
8. A. Bond, A. Hacking, Z. Milosevic, A. Zander, Specifying and building interoperable eHealth systems: ODP benefits and lessons learned, Computer Standards & Interfaces, 35 (2013) 313–328.
9. J. Burton, F. McCaffery, I. Richardson, A risk management capability model for use in medical device companies, Proceedings of the 2006 international workshop on software quality, ACM, Shanghai, China, 2006, pp. 3–8.
10. F.M. Caffery, A. Dorling, V. Casey, Medi SPICE: An update, International Conference on Software Process Improvement and Capability Determinations (SPICE), Pisa, Italy, 2010, pp. 195–198.
11. T. Tasić, U. Grottker, An overview of guidance documents for software in metrological applications, The Institution of Engineering and Technology 28 (2006) 256–269.
12. Guohong Zhou, Sha Liu, Shuqian Luo,(2008), "A Cochlear Implant Database Software Seeks for Wireless Technology", Proceeding of the third International IEEE Conference on Pervasive Computing and Applications, ICPCA 2008, 6-8 Oct. 2008, pp 753-756
13. Medical device software development: towards a single best practice framework, Dr Marion Lepmets, Dr Paul Clarke, Dr Fergal McCaffery and Anita Finnegan, Regulated Software Research Centre, Dundalk Institute of Technology
14. List of Common Bugs and Programming Practices to avoid them ,Vipindeep V, Pankaj Jalote
15. A. Djourno and C. Eyries, "Prosthese auditive par excitation electrique a distance du nerf sensoriel a l'aide d'un bobinage inclus a demeure," *Presse Med.*, vol. 35, pp. 14–17, 1957.
16. W. F. House, "Cochlear implants," *Ann. Otol. Rhinol. Laryngol.*, vol. 85, no. 27, pp. 1–52, 1976.
17. M. M. Merzenich, S. J. Rebscher, G. E. Loeb, C. B. Byers, and R. A. Schindler, "The UCSF cochlear implant project," *Adv. Audiol.*, vol. 2,pp. 199–144, 1984.
18. G. E. Loeb, C. L. Byers, S. J. Rebscher, D. E. Casey, M. M. Fong, R. A. Schindler, R. F. Gray, and M. M. Merzenich, "Design and fabrication of an experimental cochlear prosthesis," *Med. and Biol. Eng. Comput.*, vol. 21, pp. 241–254, 1983.
19. B. S. Wilson, "New directions in implant design," in *Cochlear Implants*, S. B. Waltzman and N. L. Cohen, Eds. New York: Thieme Medical, 2000, pp. 43–56.
20. S. Brill, W. Gstïettner, J. Helms, G. von Ilberg, W. Baumgartner, J. Müller, and J. Kiefer, "Optimization of channel number and stimulation rate for the fast continuous interleaved sampling strategy in the COMBI 40+," *Am. J Otol.*, vol. 18, no. Suppl. 6, pp. S104–S106, 1997.
21. M. Dorman, P. Loizou, and D. Rainey, "Speech intelligibility as a function of the number of channels of stimulation for signal processor using sine-wave and

noise-band outputs," *J. Acoust. Soc. Am.*, vol. 102, pp. 2403–2411, 1997.

22. B. S.Wilson, C. C. Finley, D. T. Lawson, R. D.Wolford, and M. Zerbi, "Design and evaluation of a continuous interleaved sampling (CIS) processing strategy for multichannel cochlear implants," *J Rehabil. Res. Dev.*, vol. 30, no. 1, pp. 110–116, 1993.
23. P. C. Loizou, "Introduction to cochlear implants," *IEEE Eng. Med. Biol.Mag.*, vol. 18, no. 1, pp. 32–42, Jan/Feb. 1999.
24. F. G. Zeng, S. Rebscher, W. V. Harrison, X. Sun, and H. Feng, "Cochlear implants: System design, integration, and evaluation," *IEEE Rev. Biomed. Eng.*, vol. 1, pp. 115–142, Jan. 2008. DOI: 10.1109/RBME.2008.2008250.