CASE STUDY: APPLYING THE COSMIC ISO 19761 MEASUREMENT METHOD ON AN MRI MESH GENERATION MEDICAL APPLICATION

F. AbuTalib, D. Giannacopoulos, and A. Abran
Electrical Engineering Department, McGill University, and École de technologie supérieure, Université du Québec, Quebec, Canada

ABSTRACT

This paper addresses the problem of applying the COSMIC ISO 19761 measurement method on a Magnetic Resonance Imaging (MRI) medical software application. The application in our case study uses 3D constrained Delaunay triangulated algorithms to generate volumetric meshes from the segmented 2D MRI images. The various steps of the measurement method are explored and discussed in details after analyzing the requirements and the system architecture of the application. Examples are presented to illustrate how the measurement results obtained can be beneficial to researchers evaluating and comparing such applications.

1. INTRODUCTION

Medical mesh generation software applications can produce volumetric representations and detailed numerical models of human organs. The physicians can use the quantitative data collected from such models to potentially increase the probability of making more accurate diagnoses and better treatments. Accessing such applications is usually limited as many of these applications are commercial and very expensive [1]. Moreover, many of these applications require sophisticated hardware architectures to operate on, which would limit their portability [1]. Although numerous attempts have been made in the public domain to produce software to overcome these problems, the results obtained from such software are rarely evaluated on realistic datasets [2]. Because of all of these limitations and challenges, comparisons performed in the literature are usually limited to a very few mesh generators. As there is no standard method for researchers to use to evaluate and compare such software, there is an obvious need to establish such a method for the medical domain that will be widely accepted.

Today, there are five ISO recognized measurement methods in the software industry [3]: (I) ISO 19761-COSMIC; (II) ISO 20926-IFPUG; (III) ISO 20968-MKII, (IV) ISO 24570-NESMA; and (V) ISO 29881-FISMA. This paper applies the method of the ISO 19761-COSMIC standard to a medical software architecture used to transfer medical images from their native scanned format to volumetric meshes that can be used by the Finite Element Method (FEM) software. The ISO 19761-COSMIC standard was chosen because of its numerous advantages, among them [3]: (I) availability in the public domain; (II) simple design; (III) applicability to a very wide range of software; (IV) ability to capture size from multiple software viewpoints; and (V) underlying concepts that are compatible with modern software engineering concepts.

This paper not only evaluates the current state of the technology of applying software measurement on medical applications, but also suggests a new point of view for designing and developing such applications, with the aim of helping future researchers to better evaluate this type of software.

In section 2, the software system requirements of one of the medical software system applications are listed and the system architecture of the software is outlined. In section 3, the details of applying the COSMIC measurement method to the user layer of the software application are discussed. In section 4, some analysis is presented to illustrate the use of the measurement results obtained in some evaluations and comparisons. Finally, some ideas for future work are suggested in section 5.

2. SOFTWARE REQUIREMENTS AND ARCHITECTURE

In this paper, the strategy of evaluating the application of software measurement methods on medical applications is studied by applying one of these methods on one open-source medical software system. We use the proposed software system application in [4] for this purpose. The main objective of the application in [4] is to transfer 2D medical images into meshes that can be used by FEM software. The meshes are generated by applying 3D constrained Delaunay tetrahedralization algorithms. The requirements of this software system are summarized in the following subsections.

2.1. Functional Requirements

The functional features of the system are: (I) the ability to process MRI images in their native DICOM format; (II) the capability of providing the various intermediate image processing operations; (III) the ability to generate volume adaptive Delaunay meshes; (IV) the capability to visualize the produced meshes; and (V) the ability to report the quality of the generated meshes.

2.2. Non-Functional Requirements

The non-functional features of the system are: (I) Performance: speed is maximized, while memory use is minimized (this is important because of the high computational cost of processing medical data); (II) Usability: minimal knowledge is required to learn it and use it; (III) Expandability: new functionalities, features, and components are easy to develop and integrate; and (IV) Portability: it is readily portable across the various platforms and not limited to specific hardware.

2.3. System architecture

Figure 1 shows an overview of the system's processes and how the various components communicate with each other in order to accomplish the various tasks and provide the necessary functionalities.

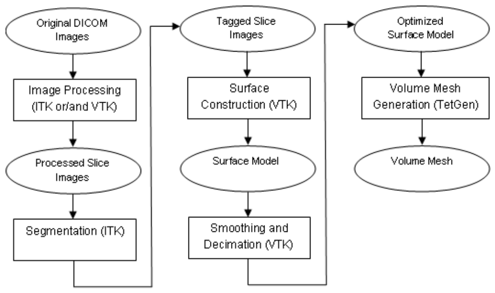

Figure 1. Overview of the system's processes.

2.4. Mesh Quality Evaluation

One of the features of the medical software application in our case study is its ability to produce a report about the quality of the meshes it generates. This is a very essential feature for such applications as it can provide a means for others to compare the results produced and evaluate how successful the application is in achieving the objectives of its functional operations. The quality of the finite element analysis of MRI medical data relies directly on the quality of the mesh representation. The application uses a set of three main criterions in producing these reports. These criterions are [4]: (I) Geometric mesh quality measures; (ii) Mesh surface approximation measures; and (iii) Performance measures.

The first criterion is mainly for the quality of the shape of the small individual tetrahedral elements produced and it is based entirely on the elements' geometric attributes. For this purpose, both α and β_{Baker} measures [16] are considered. The second criteria is based on how well all the elements are in representing the original organ. This is done by computing the Hausdorff distance mean error $d_m(S,S')$ between the generated surface (S') and the original surface (S) [17].

$$d_m(S,S') = \frac{1}{|S|}\iint_{p\in S} d(p,S')dS \qquad (1)$$

|S| represents the area of the original surface and d(p,S') is the distance between the generated surface and a point p in the original surface. The third criterion is to record both the time spent and the memory storage utilized to accomplish the requested task of generating the meshes. Remember that it can be very critical for a surgeon to obtain the needed mesh data within a specific short period of time.

3. COSMIC MEASUREMENT METHOD

The purpose of the measurement is to determine the COSMIC (ISO 19761) size of the main functionality of the medical software application in our case study for transforming DICOM files, produced from MRI or CT scans, into a volumetric mesh and producing a report describing the quality of the generated mesh. It is important to state clearly that it is not the purpose of this case study to measure the size of all the user requirements, functional and non-functional, documented in section 2. To do so, it would be a very lengthy process, and beyond the scope of this paper. The measurement is to be performed using COSMIC – ISO 19761 (2009), Version 3.0.1 and the measurement viewpoint in this case study is that of the researchers who are interested in using and developing such software applications. The measurement scope is the software functional user requirements needed to accomplish the main functionality of the system. Only the functionalities allocated to the software in the requirements are considered.

3.1. Identification of Layers

In the context of the COSMIC measurement method, it is important to identify the layers where the set of software requirements will be operating. One of the most common characteristics of almost all medical software applications is the complexity of their services. This characteristic is, of course, directly related to the complexity of the living objects that are the focus of the MRI analysis. Because of this complexity, a complex software system tends to assign the requirements of its services to various software components [3]. The software system in our case study is no exception. The functional requirements for supplying the main functionality of the product were divided among three main components: ITK, VTK, and TetGen. These individual components, which are actually complete software systems by themselves, were then connected using the Tcl scripting language component. This gluing component was used mainly to establish two interfaces: the ITK-VTK interface and the VTK-TetGen interface. The functional requirements for supplying the main functionality of the software system can be found in three layers. This is best illustrated with a diagram (Figure 2). We refer to these layers as the User layer, the Virtual Environment (Wrapper) layer, and the System layer.

The User layer is the first layer, and consists of the software written/used by the researchers to control and carry out the various intermediate operations to produce the meshes. The second layer is the Virtual Environment (Wrapper) layer, and consists of the software used to transform Tcl commands into their equivalent commands in the ITK, VTK, and TetGen engines. The third layer is the System layer, which consists of the software written to produce the three engines. In this paper, only the User layer is studied. The second and the third system layers will be the subject of future work (see section 5).

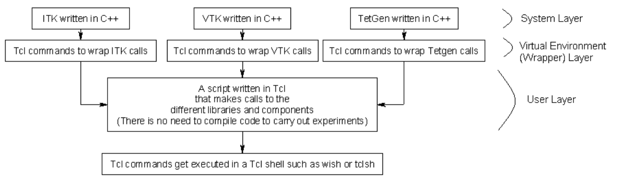
ITK written in C++
VTK written in C++
TetGen written in C++
System Layer
Tcl commands to wrap ITK calls
Tcl commands to wrap VTK calls
Tcl commands to wrap Tetgen calls
Virtual Environment (Wrapper) Layer
A script written in Tcl that makes calls to the different libraries and components (There is no need to compile code to carry out experiments)
User Layer
Tcl commands get executed in a Tcl shell such as wish or tclsh

Figure 2. The three layers of the software system

3.2. Identification of the Functional Users

The functional users who interact with the software in the User layer, sending data to the software, are: (1) the researcher who supplies the DICOM input files; (2) the ITK software wrapped as Tcl commands; (3) the VTK software wrapped as Tcl commands; (4) the TetGen software wrapped as Tcl commands.

The functional users who receive data from the software are: (1) the researcher who receives the report about the quality of the generated meshes; (2) the FEM software to consume the Mesh; (3) the ITK software wrapped as Tcl commands; (4) the VTK software wrapped as Tcl commands; (5) the TetGen software wrapped as Tcl commands.

3.3. Identification of the Boundary

Based on the analysis of the written requirements, the diagram shown in Figure 3 shows the software, along with its functional users, and hence the software boundary.

In other words, for the software in the User layer, there are three software components (ITK, VTK, and TetGen) and one human user, who sends data signals to the software. Also, there are four software components (ITK, VTK, TetGen, and the FEM software) and one human user, who receives data from the software. In addition, the software reads/writes data stored in the persistent storage device (hard disk).

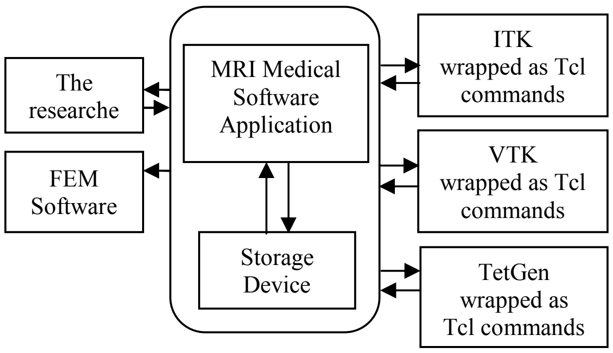
The researche
FEM Software
MRI Medical Software Application
Storage Device
ITK wrapped as Tcl commands
VTK wrapped as Tcl commands
TetGen wrapped as Tcl commands

Figure 3. Diagram of the boundary of the software system.

3.4. Identification of Triggering Events

After analyzing the application in our case study, the following eight triggering events can be identified: (1) Start Event; (2) Segmentation End Event; (3) Extracting a Tissue End Event; (4) Marching Cube End Event; (5) Decimation End Event; (6) Smoothing End Event; (7) Generating the Mesh End Event; and (8) Generating the Report End Event

Note that the triggering event#(1) is created by the human researcher/operator, triggering events#(2) is produced by the ITK engine, triggering events#(3, 4, 5, 6, and 8) are produced by the VTK engine, and triggering events#(8) is produced by the TetGen engine.

Also note that the above list does not include all the events that occur during operation of the software system, but the triggering events only. Note that a triggering event is an event that gets stimulated by a functional user of the measured application and causes the start of one or more functional processes [15].

3.5. Identification of Data Groups

From the documentation of the application, the data groups that the software has to handle are identified. These are categorized into two categories, based on the data movement: sources and destinations. The data groups are summarized in Tables 1 and 2.

Table 1: Source data groups for the software system

Data Sources	**Objects of Interest (Data Group)**
Researcher	-Process Start Signal (ProcessSS) -Location of MRI DICOM files (InputImagesPath)
ITK Software	-Segmentation End Signal (Seg_ES) -Location of segmented files (SegmentedPath)
VTK Software	-Extracting End Signal (ExtractionES) -Memory address of the generated extracted objects (ExtractionAddress) -Marching Cube End Signal (MC_ES) -Memory address of the objects after applying the Marching Cube operation (MC_Address) -Decimation End Signal (DecimationES) -Memory address of objects after applying the Decimation operation (DecimationAddress) -Smoothing End Signal (SmoothES) -Location of the generated smoothed surface object files (SmoothPath) -Generating Report End Signal (ReportES) -Location of the generated report (ReportPath)
TetGen Software	-Generating Mesh End Signal (MeshES) -Location of the generated mesh file (MeshPath)
Storage Device Persistent Data	- ITK generated files that contain the output of the segmentation operation (SegmentedFiles) - TetGen file that contains the output of the mesh generation algorithm (MeshFile)

Table 2: Destination data groups for the software system

Data Destinations	**Objects of Interest (Data Group)**
ITK Software	-Location of MRI DICOM files (InputImagesPath) -Segmentation Start Signal command (Seg_SS)
VTK Software	-Location of converted segmented files (SegmentedPath) -Extracting Start Signal Command (ExtractionSS) -Memory address of the generated extracted objects (ExtractionAddress) -Marching Cube Start Signal Command (MC_SS) -Memory address of the objects after applying the Marching Cube operation (MC_Address) -Decimation Start Signal Command (DecimationSS) -Memory address of objects after applying the Decimation operation (DecimationAddress) -Smoothing Start Signal Command (SmoothSS) -Location of the converted generated mesh file (ConvertedMeshPath) - Generating Report Start Command Signal (ReportSS)
TetGen Software	-Location of the generated smoothed surface object files (SmoothPath) -Generating Mesh Start Signal (MeshSS)
FEM Software	-Location of the generated mesh file (MeshPath) -Process End Signal (ProcessES)
Researcher	-Location of generated report (ReportPath)
Storage Device Persistent Data	-Files representing the conversion result of the SegmentedFiles to a format that can be read by the VTK engine (SegmentedFiles) -A file representing the conversion result of the MeshFile to a format that can be read by the VTK engine(ConvertedMeshFile)

3.6. Identification of Functional Processes

From the requirements, eight candidate functional processes were identified. A functional process (FProcess) is a group of functional user requirements whose data movements can be isolated separately from the other groups. [15]. Note that each candidate process was verified to satisfy the COSMIC conditions [12] to be considered as a COSMIC functional process.

The candidate COSMIC functional processes are: (1) Start FProcess; (2) Extract Tissue FProcess; (3) Marching Cube FProcess; (4) Decimation FProcess;(5) Smoothing FProcess; (6) Mesh Generation FProcess; (7) Report Generation FProcess; and (8) End Process FProcess.

3.7. Identification of Data Movements

For each functional process identified, the COSMIC data movement should be identified. Based on the movement's source and destination, one can classify four data movement types [15]: (I) Entry (E) type where the source is a functional user and the destination is the software application; (II) Exit (E) type where the source is the software application and the destination is a functional user; (III) Write (W) type where the source is the software application and the destination is a persistent storage medium; and (IV) Read (R) type where the source is a persistent storage medium and the destination is the software application.

The details of this step are presented in Table 3. Note that the size is indicated in the right-hand column in the COSMIC unit, that is: 1 COSMIC function point = 1 CFP.

Table 3: Details of COSMIC functional processes, data movements, and the data group

FProcess (Triggering Event)	**Data Movement Description**	**Data Group**	**Type**	**CFP**
Start (Start Event)	-Receiving the triggering event	ProcessSS	E	1
	-Receiving the DICOM file path	InputImagesPath	E	1
	- Sending the segmentation command to ITK	Seg_SS	X	1
	-Sending the file path to ITK	InputImagesPath	X	1
				4
Extract Tissue (Segmentation End Event)	-Receiving the triggering event	Seg_ES	E	1
	-Receiving the segmented file path	Segmented Path	E	1
	- Reading the segmented files from the storage device	Segmented Files	R	1
	-Writing the converted segmented files to the storage device	Segmented Files	W	1
	-Sending the extracting command to VTK	ExtractionSS	X	1
	-Sending the converted segmented file path of to VTK	Segmented Path	X	1
				6
Marching Cube (Extracting a Tissue End Event)	-Receiving the triggering event	ExtractionES	E	1
	-Receiving the memory address of the extracted objects	ExtractionAddress	E	1
	- Sending the marching cube command to VTK	MC_SS	X	1
	-Sending the memory address of the extracted objects to VTK	ExtractionAddress	X	1
				4
Decimation (Marching Cube End Event)	-Receiving the triggering event	MC_ES	E	1
	-Receiving the memory address of the marching cube surface object	MC_Address	E	1
	- Sending the decimation command to VTK	Decimation SS	X	1
	-Sending the memory address of the marching cube surface object to VTK	MC_Address	X	1
				4
Smoothing (Decimation End Event)	-Receiving the triggering event	Decimation ES	E	1
	-Receiving the memory address of the decimated surface object	Decimation Address	E	1
	- Sending the smoothing command to VTK	SmoothSS	X	1
	-Sending the memory address of the decimated surface object to VTK	Decimation Address	X	1
				4
Mesh Generation (Smoothing End Event)	-Receiving the triggering event	SmoothES	E	1
	-Receiving the path of the smoothed file	SmoothPath	E	1
	-Sending the generate mesh command to TetGen	MeshSS	X	1
	-Sending the path of the smoothed file to TetGen	SmoothPath	X	1
				4
Report Generation (Generating the Mesh End Event)	-Receiving the triggering event	MeshES	E	1
	-Receiving the path of the mesh file	MeshPath	E	1
	- Reading the mesh from the storage device	MeshFile	R	1
	-Writing the converted mesh file	Converted MeshFile	W	1
	- Sending the command to generate the report to VTK	ReportSS	X	1
	-Sending the path converted mesh file	Converted MeshPath	X	1
				6
End	-Receiving the	ReportES	E	1

(Generating the Report End Event)	triggering event			
	-Receiving the path of the report file	ReportPath	E	1
	- Sending the path of the report file to the researcher	ReportPath	X	1
	-Sending the path of the mesh file to FEM software	MeshPath	X	1
	- Sending the Process end signal to FEM software	ProcessES	X	1
				5

Total functional size in COSMIC CFP = 37

4. ANALYSIS OF MEASUREMENT RESULTS

For our software system, eight functional processes were identified in the User layer. The COSMIC functional size of the main functionality of the system is 37 CFP. Table 4 summarizes these results for the eight functional processes.

Table 4: Summary of COSMIC measurement results

Functional Process	Number of Data Movements				CFP
	E	X	R	W	
Start	2	2	0	0	4
Extract Tissue	2	2	1	1	6
Marching Cube	2	2	0	0	4
Decimation	2	2	0	0	4
Smoothing	2	2	0	0	4
Mesh Generation	2	2	0	0	4
Report Generation	2	2	1	1	6
End	2	3	0	0	5
Total	16	17	2	2	37

But how can such numbers be useful to researchers who are interested in using or evaluating this medical software system? There are many possible ways to view these numbers. Each would provide researchers with valuable information that could help them to better evaluate and compare the available applications.

For example, a single medical software application usually provides a set of many different services, capabilities, and functionalities. In many cases, the researchers are only interested in some of these services, or even just some of the sub-services that the medical application can provide. Each service or functionality can be presented as a set of COSMIC functional processes. Based on the COSMIC size of each functional process identified, the contribution of each to the overall main functionality of the system can be computed. This ratio would indicate how much each functional process contributes to producing the full service of the system. By adding the ratios of the functional processes that are included in a sub-service, the researcher would end up with the expected utilization percentage from using such software system. To illustrate this point in our example, assume that a researcher is interested only in using the software system to produce the mesh surface, and not the volumetric mesh. The first five functional processes will be needed to create the surface mesh. In Table 5, the contribution ratio of each COSMIC functional process is computed. From this table, the researcher would know that generating the mesh surface service would utilize about 59% of the overall main functionality of the system.

Table 5: Contribution ratio of each COSMIC FProcess

Functional Process	Ratio in Application
Start	11 %
Extract Tissue	16 %
Marching Cube	11 %
Decimation	11 %
Smoothing	11 %
Mesh Generation	11 %
Report Generation	16 %
End	13%

Also, one of the most common characteristics of almost all medical software applications is the complexity of their services. Because of this complexity, a medical software system tends to assign the requirements of its services to various software components, as shown in our system application. Valuable information can also be obtained by computing the functional size associated with each individual functional user (component) of the software system. Table 6 shows the functional size in CFP associated with all the functional users of the software system.

Table 6: The functional size for each functional user

Functional User	CFP
ITK	4
VTK	20
TetGen	4
Researcher	3
FEM Software	2
Persistent Device Storage	4

This kind of information can be very useful for helping researchers who are interested in replacing one software component for another, or even introducing new components to the system to expand it. This information can help them estimate the effort needed and predict the degree of change required to replace/add one or more components. There are many reasons why a component needs to be replaced or added in such systems. For example, it might be needed to apply new algorithms, to handle a new requirement of supporting a certain environment, or to expand the system with new features and capabilities.

From a different point of view, the ratios of the COSMIC data movement types are represented in Table 7.

Table 7: Contribution ratio of each data movement type

Functional Types of Data Movement	Ratio in Application
Entry (E)	43 %
Exit (X)	46 %
Read (R)	5.5 %
Write (W)	5.5 %

From this table, we can see that the largest contributor to the functional size is either the Entry data movements or the Exit data movements (about 90%): this is expected for such functional processes in the User layer. In this layer, the main purpose of functional processes is to transfer data between the various components, and it is not common for software in this layer to manipulate data. This kind of information can help determine the degree of compatibility between the selected components. By choosing components that are compatible with one another, it would be expected that, in the User layer, there would be more Entry and Exit data movements than Read and Write data movements.

The COSMIC measurement method relies on the data movement concept to determine the functional size of the software. The data movement is the fundamental functional component responsible for moving a single data group type [15]. Because of this heavy dependency, the results obtained from such a measurement task are also very good indicators of the performance of the system. Performance is a key requirement of most medical applications (see section 2). For example, our software system has to perform many intermediate operations to generate the mesh surface (18 CFP). If a researcher were to find a component capable of producing the mesh surface directly, instead of applying these multiple intermediate operations, we would expect much better performance in the User layer. Note, however, that this doesn't guarantee better performance of the overall system. Full analysis of all the layers of the system should be carried out and taken into consideration before such statement is made.

Future Work

There are clearly many potential directions for future work. For instance, not all the possible functionalities of the software system were studied in this paper and the system has many other capabilities that were not covered in this study. This includes its ability to visualize the output of many of the intermediate operations, and the ability to refine the generated meshes to reduce errors and improve their quality. Also, the non-functional requirements were not covered in this paper.

Another interesting idea for future work would be to extend the current case study to include the COSMIC measurement of the functional processes in the second layer (the Virtual Environment – Wrapper layer) and the third layer (the System layer), the latter being where all the algorithms are implemented and therefore the performance of such systems should be evaluated there. Even though most of the functional size measurement methods do not take algorithms into account directly [3], it is very possible to apply the COSMIC method on the algorithms themselves and calculate the number of data movements needed by any algorithm. This would make it possible to compare the performance of two algorithms using the COSMIC method. Of course, for this to work, some terms need to be established, and we need to define exactly what the Entry, Exit, Read, and Write data movements mean in the algorithm context. It should also be taken into account that such measurements on the algorithms must be performed on a different level of granularity. The measurement results must then be reported distinctly, as a local extension, for example.

Note that this study covered only one ISO measurement method (ISO/IEC 19761 – COSMIC). There are four other ISO recognized measurement methods in the software industry [3], which can also be applied and studied.

In summary, this paper has reported on an ongoing research project aimed at defining standards that can be used to help researchers better evaluate and compare complex medical and engineering software applications without the need to have access to the implementations, or even to the complete documentation.

REFERENCES

[1] Coronato, A., De Pietro, G., and Marra, I., "An open-source software architecture for immersive medical imaging," in Proceedings of the IEEE International Conference on Virtual Environments, HCI and Measurement Systems, 2006.

[2] A. Fedorov, N. Chrisochoides, R. Kikinis, and S. K. Warfield, "An Evaluation of Three Approaches to Tetrahedral Mesh Generation for Deformable Registration of Brain MR Images," Biomedical Imaging: Nano to Macro, 3rd IEEE International Symposium on Volume, 6-9 April 2006, pp. 658-661.

[3] A. Abran, "Software Metrics and Software Metrology," IEEE-CS Press & John Wiley & Sons – Hoboken, New Jersey, 2010, pp. 328.

[4] F. AbuTalib and D. Giannacopoulos, "A Methodology for Applying Three Dimensional Constrained Delaunay Tetrahedralization Algorithms on MRI Medical Images," CompuMag Conference, Florianopolis, Brazil, November 2009.

[5] Medical Imaging & Technology Alliance, "Digital Imaging and Communications in Medicine (DICOM)," National Electrical Manufacturers Association (NEMA), PS 3.1, 2011.

[6] L. Ibanez, W. Schroeder, L. Ng, and J. Cates, "The ITK Software Guide", NLM Insight Segmentation and Registration Toolkit, Kitware Inc. ISBN 1-930934-10-6, 2003.

[7] W. Schroeder, K. Martin, and Bill Lorensen, "The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics", Upper Saddle River, NJ: Prentice Hall, 1998.

[8] H. Si, "TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator - User's Manual," Version 1.4, 2006.

[9] B. Welch, K. Jones, and J. Hobbs, "Practical Programming in Tcl and Tk," Fourth edition, 2003.

[10] Xenophon Papademtris, "An Introduction to Programming for medical image Analysis with the visualization Toolkit", Yale University, 2006.

[11] M. Sermesant, C. Forest, X. Pennec, H. Delingette, and N. Ayache, "Biomechanical Model Construction from Different Modalities: Application to Cardiac Images," Lecture Notes in Computer Science, Medical Image Computing and Computer-Assisted Intervention - MICCAI 2002: 5th International Conference, Tokyo, Japan, September 25-28, 2002, Proceedings, Part I. pp. 714-721.

[12] COSMIC Group, "Rice Cooker – Cosmic Group Case Study," École de technologie supérieure, Université du Québec, 2003.

[13] ISO/IEC 19761:2003, "Software Engineering- COSMIC-FFP – A functional size measurement method," International Organization for Standardization, Geneva, Switzerland, 2003.

[14] Jonathan Richard Shewchuk, *Lecture Notes on Delaunay Mesh Generation*, Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA, 1999.

[15] A. Abran, J. Desharnais, S. Oligny, D. St-Pierre, and C. Symons, "The COSMIC Functional Size Measurement Method-Measurement Manual," Version 3.0.1, 2009

[16] Mark Dorica, "Novel Mesh Quality Improvement Systems for Enhanced Accuracy and Efficiency of Adaptive Finite Element Electromagnetic with Tetrahedra," Master thesis, McGill University, April, 2004.

[17] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "MESH: Measuring Errors Between Surfaces Using the Hausdorff Distance," IEEE International Conference on Vol. 1, 2002, pp. 705-708.