A component-oriented software toolkit for patient-specific finite element model generation

Chee-Kong Chui, Zhenlan Wang, Jing Zhang, Jackson Shin-Kiat Ong, Limeng Bian, Jeremy Choon-Meng Teo, Chye-Hwang Yan, Sim-Heng Ong, Shih-Chang Wang, Hee-Kit Wong, Swee-Hin Teo

Department of Orthopaedic Surgery, Yong Loo Lin School of Medicine, National University of Singapore, Singapore
Department of Diagnostic Radiology, Yong Loo Lin School of Medicine, National University of Singapore, Singapore
Division of Bioengineering, Faculty of Engineering, National University of Singapore, Singapore
Department of Electrical and Computer Engineering, Faculty of Engineering, National University of Singapore, Singapore
Department of Mechanical Engineering, Faculty of Engineering, National University of Singapore, Centre for Biomedical Materials Application and Technology, 9 Engineering Drive 1, Singapore 117576, Singapore

ABSTRACT

A component-oriented software system, i.BioMech (image-based biomechanical modeling) is proposed for generation of patient-specific finite element model. It applies a systematic software engineering approach to patient/subject-specific meshing and assignment of material properties. The prototype program is based on the component object model (COM), which enables ease of combination of existing mesh generation algorithms and material property assignment schemes, and incorporation of new ones. It also facilitates utilization by other programming languages or platforms. Data input comprises a series of medical images captured from the patient. The output is a patient-specific finite element model for computational analysis using commercially available finite element software. The prototype software system provides a framework to compare the different finite element mesh generation methods as well as schemes for material property assignment. Our focus is on patient/subject-specific modeling of the human vertebrae.

1. Introduction

The potential use of computer models to plan and evaluate surgical treatments has been recognized for a long time since the early 1980s when the term “computer aided surgery” was coined. Nevertheless, the use of a patient-specific computer-based anatomical model as a planning tool for surgical treatment in a clinical setting is not widespread. For example, we have yet to see numerical analysis done on a patient with scoliotic spine before a particular loading method is decided for deformity correction. There are many technical issues pertaining to this less-than-satisfactory clinical acceptance. Therefore we have chosen to focus on the issues of engineering software development in this paper.

There are software toolkits readily available for the medical engineering community to use for research, prototyping and application development. Many are conventional software API libraries for experienced C/C++ programmers. Prototyping efficiency could be improved with simpler programming platforms and languages such as MS Visual Basic. Component-oriented software allows easy prototyping and application development for many Visual Basic or Visual programmers. Julius [1] is a popular software framework that has been providing software components to medical application developers in the form of plug-ins for the DICOM 2 loader, 2D slice views and surface visualization. Components such as basic manual segmentation, measurement tools, and drivers for some tracking and haptic devices are in the pipeline. The Strange Engine [2] is a toolkit that allows the display and visual manipulation of analytical phantoms, projection sets and volumetric data through a standard Windows graphical user interface (GUI). In general, there is a lack of software toolkits that can also be used with accurate and high resolution finite element (FE) computation. We have developed a software component-oriented toolkit, i.BioMech (image-based biomechanical modeling) that accepts medical images of a patient as input and generates a finite element model for computational analysis using commercial finite element software such as ABAQUS (ABAQUS Inc., Rhode Island, USA). The software components are based on the component object model (COM) [3]. An important requirement is the resultant FE model should represent the geometry as well as the material of the subject accurately.

Many investigations in biomechanics employ generic FE meshes based on generic patient geometries [4–7]. However, human...
anatomical structures have significant variations in geometrical shape and tissue properties among different individuals. Patient specificity is very important in practice for surgical simulation and planning. ScanFE (http://www.simpleware.co.uk/software/scanfe.php) is a commercially available mesh generation software package that can output various finite element input file formats, including that of ABAQUS, for computational analysis. Together with its sister software product ScanIP (http://www.simpleware.co.uk/software/scanip.php), which provides image segmentation tools, ScanFE can also be employed to generate a patient-specific finite element model from medical images. ScanFE is a voxel-based mesh generation algorithm that can produce a mixed hexahedral and tetrahedral mesh or a pure hexahedral mesh. Much research activity [8–10] has been devoted to automatic patient- or subject-specific FE mesh generation methods. Each method has its own merits, and it is difficult to find the best approach for patient-specific finite element modeling of bone due to its complexity and diversity. i.BioMech with its component-oriented architecture and object-oriented implementation provides the user with a framework which allows one to include a custom mesh generation algorithm for a specific problem.

The development of meshing technologies is an intensive theoretical and practical research area. However, the development of meshing software from the software engineering point of view has been an active research area only recently. Typically, the approach comprises the application of object-oriented design and programming, for example, the design of object-oriented data structures and procedural classes for mesh generation in [11] and the definition of an optimal object-oriented mesh representation that allows the programmer to build efficient algorithms in [12]. Formal methods for improving reliability of mesh generation software have also been proposed [13]. A recent approach towards software development is the software product family, which is a set of products that are built from a collection of reused elements or products in a planned manner [14]. Smith and Chen [15] first reported using this approach to analyze commonalities and variability of meshing tools. Their main focus is the variability of the input and output formats, and the generation of the meshes as a whole. A recent paper [16] describes the design of a family of object-oriented meshing tools for the generation of 2D triangulations based on the Lepp concept. To the best of our knowledge, there is no report of systematic software engineering approaches to patient/subject-specific assignment of material properties.

i.BioMech runs on a PC computer under Microsoft Windows XP, and aims to provide a sample application template within the standard Windows GUI. This toolkit provides a user-friendly environment where finite element meshing algorithms and material properties assignment schemes can be easily fitted for testing. The extendable library of patient-specific mesh generation software components also enables the user to apply the most appropriate software for the problem. We have investigated FE mesh generation algorithms such as direct-voxel conversion [17,18] and its enhancement using the marching cubes algorithm [19] to smooth the boundaries of the finite element model [8], and an improved voxel-based mesh reconstruction approach [20] by smoothing the irregular boundaries at model surfaces [21]. We have also previously reported a grid based mesh generation method [22]. More recently, we have proposed a statistical approach to template based FE modeling [23]. In the statistical template based method, the selected template FE mesh is elastically deformed to map the geometry of organs reconstructed from the patient’s CT or MR data.

This paper presents the i.BioMech toolkit and its component-oriented software architecture, and demonstrates its usage in patient-specific spine modeling and finite element analysis. Section 2 describes the design and requirements of i.BioMech. Section 3 introduces the mesh generation algorithms. Section 4 describes the scheme for material properties assignment. Section 5 presents the applications of i.BioMech, particularly as a framework to study the performance of various mesh generation algorithms and modeling of the entire spinal column. Section 6 describes the results from the applications. The paper is concluded in Section 7 with a further discussion of the usage and technical merits of i.BioMech.

2. The i.BioMech software toolkit

2.1. Toolkit design

The main feature of component-oriented software development is the software component. A software component comprises the interface, an implementation, and a deployment that is the physical executable file. In addition, a component must possess four properties, namely encapsulation, descriptive, replaceable and extensible. A good description of software components can be found in [25].

Fig. 1 is an overview of the architecture of i.BioMech toolkit, including its interface with other software. For realistic biomechanical simulation involving computation of accurate mechanical deformation of an organ due to cutting and/or deformation by surgical instruments, it is important that we incorporate subject-specific finite element modeling, specifically, patient-specific modeling of anatomical structures. Both the geometry and mechanical properties of the model should be accurate. Furthermore, the processing or turnaround time should be acceptable. The turnaround time is the interval between the availability of the CT/MR scan of the patient and generation of the patient-specific model for numerical analysis.
Chui et al. [26] suggested a turnaround time of 4 h for a clinically viable solution. This process flow is denoted with solid arrows in Fig. 1.

Although the components of i.BioMech are implemented in C/C++, the implementation is designed to be object oriented. The development environment has recently been upgraded to IDE Microsoft Visual Studio from Microsoft Visual C++ 6.0. To minimize the restrictions imposed by the framework of Microsoft foundation class (MFC), MFC is only used in the implementation of sample GUI applications. OpenGL is used to provide the graphical display including image rendering. The object model components are implemented using active template library (ATL) [27]. COM (formerly known as ActiveX/OLE extension) is used to implement the software components or plug-in since the software solution has been developed for the Microsoft Windows operating system. COM in-process components reside in different dynamic link libraries (DLL).

Two fundamental objects have been defined. They are Volume for the patient’s image data and FE Mesh Model for the patient-specific finite element model. Each object features its own GUI providing both display and user interaction. There are three standard operators: image data processor for extraction of the model from volume data, FE mesh generation for generation of the finite element mesh model, and FE mesh converter for conversion of the different file formats of FE solvers. Each operator has its own GUI and can be loaded/saved from/to disk in a proprietary ASCII file format.

The image data processor comprises the implementation of medical image processing and segmentation algorithms. Conventional thresholding methods can usually be used in CT images to separate the bone structures from surrounding soft tissue since the former has higher intensities. However, global thresholding may fail due to heterogeneity and imaging artifacts. A local adaptive thresholding scheme [39] may be employed under these conditions. Segmentation of bone structures from MR images is a challenge that we are investigating [40].

The conversion from the 3D volume data of a patient’s vertebrae to finite element meshes is done over two steps. The volume data associated with the object volume is first imported into the image data processor for image processing. The segmented image volume is then fed into the FE mesh generator for FE mesh generation including assignment of material properties. The resultant FE input file can be used directly by the FE software package for computational analysis.

The FE mesh generation operator provides COM IFEMeshGenerator support. The user can write a mesh generation algorithm and plug it into i.BioMech. The plug-in has to support the custom IFEMeshGenerator interface defined in our i.BioMech API. The user can also use the existing mesh generation algorithms and material properties assignment schemes that have already been implemented in the FE mesh generation COM component. Property pages can be displayed within the GUI. This is particularly useful for configuration of the meshing algorithms, for example, the mesh density. The object–interface interaction can be managed through Windows object linking and embedding (OLE) Drag&Drop [28]. In short, the user can select and drop the FE mesh generation COM component onto the interface GUI developed with MS Visual Basic.

Several methods for automatically generating biomechanical models with patient-specific geometry from medical images were implemented. Each method has its pros and cons. Some methods generate models with all-hexahedral or all-tetrahedral finite element mesh elements, respectively, using the direct voxel meshing algorithm. By combining these two types of meshes, it is possible to develop an algorithm for automatically generating marching cube surface smoothed voxel meshes [8]. These models, with high mesh density, are suitable for detailed studies on specific anatomical objects. We also developed mesh-mapping capabilities where template meshes are mapped (or forced) into the geometry of patient radiological images. The mapped models have low mesh density and therefore can be used to analyze a combination of anatomical objects. The other mesh generation algorithms that we have investigated include a grid based method known as enhanced 3D meshing [20] and improved voxel-based meshing [21,22]. Fig. 2 shows a class diagram that includes an illustration of the various automatic mesh generation methods available in the toolkit.

The finite elements can be assigned the material properties inferred from the medical images. The user can assign the material properties in accordance with the voxel values. This can be done using a manually created material lookup table, or in an automated manner for CT images (Section 4). Another option is the assumption of a single material property for all elements. However, the FE model may not represent the complex mechanical properties of the underlying biological organ. The material models include isotropic elastic, isotropic elastic–plastic and orthotropic elastic with mechanical properties including Young’s modulus (E₁, E₂, E₃), Poisson’s ratio (υ₁₂, υ₁₃, υ₂₃), shear modulus (G₁₂, G₁₃, G₂₃), yield stress (σᵧ₁), and yield strain (γᵧ₁).

Since the COM specification is at the binary level, it allows integration of binary components possibly written in different programming languages. For i.BioMech, C/C++ is used in the implementation the COM objects. The COM interfaces are defined in the interface definition language (IDL) [29]. IDL is programming language neutral. Listing 1 is an outline of the definition of the basic FE Mesh Model component (IFEMeshModel).

2.2. Requirements

i.BioMech runs on Intel Pentium 4 computers under Microsoft Windows XP. The computational requirement is dependent on

![Fig. 2. Fundamental component object model diagram. FE mesh generator is the focus of this paper. We have four algorithms developed to date.](image-url)
the implementation of the algorithm for FE mesh generation. When the main memory and virtual memory are used up, the process will simply terminate.

Fig. 3 shows a Microsoft Windows based application “Voxel Mesher” developed for patient-specific FE mesh generation. The simple application allows the user to generate an FE model based on segmented medical images. Material properties of the finite element can also be assigned automatically using the software. The user can view and manipulate the FE model, and select the output format of the FE problem file.

3. Mesh generation algorithms

We have investigated and developed several algorithms that provide versatility in the type of biomechanical model generated. The technique used is dependent on the type of biomechanical analysis to be performed. The algorithms include distortion smoothed or unsmoothed direct voxel-based method [21], marching cube smoothed direct voxel-based method [8], grid based method [20] as well as template mapping [31].

Template-mapped models are based on conventionally developed models that are then mapped via control points to conform to geometries segmented from medical images. The number of elements is optimum. However, they are typically unable to take into account all forms of pathologies and variation in tissue strength. They are excellent for modeling the entire spinal column of patients where the biomechanical response of the column as a whole is of interest. When a specific location is of interest, such as a single vertebral body, voxel-based models are attractive as they can account for defects as well as the variation in tissue properties.

In the voxel-based technique, 3D image voxels are directly converted into finite element cubes. Since accurate delineation of boundaries is an issue, it has been suggested voxel-based models should be used only for an overall structural analysis or internal stress estimation. We have performed 3D extrapolation to smooth the voxel-based FE model to reduce solution errors so as to obtain more accurate surface stress results [21]. Our implemented method is comparable to that of “ScanFE”, a commercially available voxel-based meshing software package.

Wang et al. [8] describes a tetrahedronization scheme that incorporates marching cube surface smoothed together with a smooth distortion factor. In this mesh model, the outer layer is composed of tetrahedral elements while internal elements are hexahedral. In the grid based method of [20,30], a grid plane divides the segmented contours of the anatomical boundaries and its inclusions into discrete meshes. A grid frame connects the grid planes between any two adjacent planes by means of a customized scheme. The resultant meshes comprise brick elements at the interior of the contours and tetrahedral and wedge elements at the boundaries. The nodal points are classified according to their materials, and hence, elements can be assigned different properties.

The inclusion of an FE mesh generation algorithm in the toolkit requires the components to adhere to the COM programming model. Details of the COM specification can be found in [3]. The users of the components are not required to know about the programming.
IFEMeshGeneration *pIFEMeshGeneration = NULL;
HRESULT hr = CoCreateInstance(CLSID_IFEMeshGeneration,
NULL, CLSCTX_INPROC_SERVER,
IID_IFEMeshGeneration,
reinterpret_cast<void**>(&pIFEMeshGeneration));

// to use voxel-based FE meshing algorithm
BSTR bStrMethodName = SysAllocString(L“VOXEL-BASED”);
pIFEMeshGeneration->Execute(bStrMethodName);
// to release the instance
pIFEMeshGeneration->Release();

Listing 2. Executing of IFEMeshGeneration using Microsoft Visual C.

languages that are used to build the components, but are required to adhere to the custom IFEMeshGeneration interface in i.BioMech API.

In the current design, the user can activate a meshing algorithm simply by passing the name of method into the component, and then executing the component. This is illustrated in Listing 2.

4. Assignment of material properties

In addition to the provision for manually assigning the mechanical properties to the finite element model, we incorporated a ‘black box’ for patient-specific biomechanical properties assignment from CT images. This ‘black box’ of mathematical equations allows automatic assignment of isotropic elastic, isotropic elastic–plastic and orthotropic elastic biomechanical properties to the models. Experiments have been carried out to investigate the Hounsfield unit in the three anatomic planes. (The Hounsfield unit the models. Experiments have been carried out to investigate the tic–plastic and orthotropic elastic biomechanical properties to allows automatic assignment of isotropic elastic, isotropic elas-

mental properties to the finite element model, we incorporated a ‘black box’ of mathematical equations 'black box for patient-specific biomechanical properties assign-

rection between the bone density and mechanical properties based on existing literature and experiments:

\[ E_1 = -34.7 + 3230 \rho \]

\[ E_2 = E_3 = 0.33E_1 \]

\[ \nu_{12} = \nu_{13} = 0.104 \]

\[ G_{23} = 0.121E_1 \]

\[ G_{12} = G_{13} = 0.157E_1 \]

\[ \sigma_y = -0.75 + 24.9\rho \]

\[ \sigma_y = 0.73 + 0.43\rho \]

There are other assignment schemes. This can be easily implemented by changing the mathematical relationship between the material parameters and image intensity. The estimated mechanical properties are assigned to the appropriate finite element in accordance with the dominant HU values presented in the region enclosed by the element.

In addition to the above scheme on automated assignment of bone material properties, the current design of our proposed toolkit also allows mechanical properties parameters to pass to the mesh model component assuming a homogeneous model. The parameters are passed by value rather than by reference. This is because marshaling a reference in COM will take longer than passing an array of data when the size of the array is small.

For a transversely isotropic bone, the material properties are represented as a single array of double precision numbers as follows:

\[ [E_1, E_2, E_3, \nu_{12}, \nu_{13}, \nu_{21}, \nu_{23}, \nu_{31}, \nu_{32}, G_{12}, G_{13}, G_{23}] \]

Only \( E_1 \) and \( \nu_{12} \) are used when the material is isotropic.

5. Materials and methods

The automatic generation of patient-specific biomechanical models of the anatomical structure is crucial for the implementation of a surgical simulator [32]. Conventional methods of generating biomechanical models are tedious and require many man-hours. Patient-specific finite element model generation is the most basic usage of i.BioMech. In this paper, the effectiveness of i.BioMech is demonstrated by its ability to model a human organ, which is then used in a clinically relevant numerical analysis. The implementation of several algorithms gives the simulator versatility in the type of biomechanical model generated. The algorithm used should depend on the type of biomechanical analysis to be performed. Each meshing algorithm has its pros and cons. The analysis was carried out using ABAQUS on an Intel P4 3.0 GHz computer with 2 GB of physical memory.

(a) The different meshing methods were compared and evaluated by the modeling of a human vertebra for finite element analysis. The evaluation criteria are the generation time, computational time and memory required, as well as a visual inspection of the stress distributions when the models were subjected to identical loading and boundary conditions.

The input medical CT image dataset is a human L3 vertebra. Direct voxel meshing and marching cube-based methods are used to generate the finite element model. An arbitrary elastic modulus of 12 GPa and Poisson ratio of 0.3 were assigned to all elements. We fixed the inferior end of the vertebral body and displaced the superior end vertically downwards by a distance of 1 mm. This mimicked the experimental studies conducted to determine vertebral compression fractures by other investigators, e.g. [7].

(b) The second test involved the use of marching cube-based and grid based methods in modeling the entire spinal column from the CT images of the VHD dataset. The material properties were assigned based upon the HU values in the CT images in the FE model generated using the marching
cube-based method. A load that mimicked an impact force of 30 N was applied to the front of the spinal column. The two ends of the spinal column are fixed.

6. Results

6.1. Finite element modeling and analysis of vertebrae

All FE meshes have the general shape of the patient’s dataset as shown in Fig. 4. The FE meshes created by the voxel-based algorithm have delineated surfaces and this phenomenon would diminish as the resolution of the dataset increases. There is no visual geometrical difference in the two FE meshes generated by the marching cube-based algorithm.

The voxel-based FE models require a smaller number of finite elements, and hence less physical memory and disk space, compared to the marching cube-based FE models (Table 1). However, the marching cube-based FE models conform better to the geometrical boundaries of the anatomical structure. Voxel-based FE models also have discretized boundaries. The marching cube-based FE models require more iterations compared to the voxel-based FE models (approximately four times more for 256 × 256 and two times more for 128 × 128), resulting in longer computational time.

This is due to the difficulties in mathematical convergence; the distortions of the tetrahedral elements are high in order to conform to the complex geometry of the vertebral body. Therefore, there is a tradeoff between geometrical accuracy versus computational efficiency for the two types of FE models.

The von Mises stress distribution of each FE model is inspected for further comparison. Generally, all models display a similar stress distribution, with stresses running diagonally from the anterior–superior region downwards towards the posterior–inferior region. Regions of high stress are located in the central region and extend towards the posterior–inferior region. As expected, the voxel-based FE models exhibit stress discontinuities due to its delineated boundaries. The marching cube-based FE models do not exhibit any such stress discontinuities, which are deemed undesirable.

6.2. Finite element modeling and analysis of the entire spinal column

There are around 80,000 elements in the human spine model (Fig. 5a) generated using the marching-cube surface smoothed method from VHD images. Fig. 5b shows the model of the same spine generated by grid based meshing. Although the model has a smaller number of elements, the different types of elements pose a challenge to numerical analysis. In the grid based method, each of the vertebrae and intervertebral disc has between 1200 and 6000

![Fig. 4. Comparison of automatically generated finite element meshes, from patient’s CT dataset using our proposed method, voxel- and marching cube-based algorithms, against a volume rendered model.](image)

<table>
<thead>
<tr>
<th>Table 1 Comparison of analyses parameters that govern computational requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE model of human vertebra</td>
</tr>
<tr>
<td>Voxel-based</td>
</tr>
<tr>
<td>256 × 256</td>
</tr>
<tr>
<td>Time required for FE model generation (s)</td>
</tr>
<tr>
<td>No. of elements</td>
</tr>
<tr>
<td>No. of equations</td>
</tr>
<tr>
<td>Physical memory required (MB)</td>
</tr>
<tr>
<td>Disk space required (MB)</td>
</tr>
<tr>
<td>CPU time (s)</td>
</tr>
<tr>
<td>No. of iterations</td>
</tr>
</tbody>
</table>

Please cite this article in press as: Chui C-K et al., A component-oriented software toolkit for patient-specific ... Adv Eng Softw (2008), doi:10.1016/j.advengsoft.2008.03.019
elements, and approximately 1200 elements, respectively. The marching cube surface smoothed method is better than the grid based enhanced 3D meshing method in terms of computational efficiency. Fig. 5c shows a finite element analysis with a load directed at the middle section of the spine model generated using the marching cube-based method.

7. Discussion and concluding remarks

This paper describes a modern object-oriented COM toolkit, i.BioMech, dedicated to automating the finite element modeling of patient anatomy from 3D tomography. A prototype component-oriented software system for structural analysis using the finite element method is presented in [34]. However, there has been not been any reported work on component-oriented software for patient/subject-specific finite element meshing. It is relatively difficult to use COM for the development of a programming system. Nevertheless, the initial steep learning curve and difficulty using COM are clearly outweighed by the advantages of component-oriented software. If component-oriented software systems are executed on the same computer and in the same process, the performance is comparable to similar object-oriented software systems. In the case of distributed environments, where software components are executed on different computers, the overall performance of the system depends on a number of factors, e.g. the request latency. To ensure good performance in either environment, object–interfaces should be designed to reduce the number of method calls. For example, in the FE mesh generation COM component, the user needs to call only the method apply most of the time. This method receives as input the segmented volume data and generates the corresponding finite element meshes using the specified meshing algorithm and material assignment scheme.

Component-oriented programming is language independent; the programming language that is used to implement GUI and to invoke the component does not have to be in C or C++. Application developers can use the FE Mesh Generation COM component from MS Visual Basic. Another way of using the toolkit is via scripting

Fig. 5. Finite element modeling and analysis of whole spine. (a) Volume rendered image of the spinal column; (b) generated FE model using marching cube-based method; (c) generated FE model using grid based method and (c) numerical simulation of FE model under impact from front.
languages like VBScript. VBScript is easy to learn and has good support for using COM objects. Features such as expression, loops and input/output are useful for simulation. Furthermore, the programming language used to develop the component should not be taken into account – existing C codes, or even Fortran codes, can be packaged into a component and integrated into the toolkit. i.BioMech is well placed to be a comprehensive patient-specific mesh generation toolkit by integrating previously existing software components.

The problem of programming software for numerical analysis techniques such as the FE method (FEM) and mesh generation has been treated in the literature [33,34]. Numerical computation such as FEM behaves as a procedural method with a clear flow. The intrinsic sequential characteristic of FEM may be diluted in the objects and their complexity becomes more of an inconveniece than a facility. On the other hand, FEM contains abstract data such as nodes, elements and material properties. Similarly, nodes, elements and material properties are the outcomes of a mesh generation process. Programming an FEM should balance both object-oriented and procedural-oriented concepts. Nevertheless, FE mesh generation is less procedural and more object-oriented relative to FEM. In i.BioMech toolkit, the software components for mesh generation are implemented using C/C++ in accordance with object-oriented programming. Microsoft Visual C++ allows one to develop the sequential programming together with the object organization and, consequently, affords some measure of flexibility to the programmer.

A task of the toolkit is to fulfill the research need to test FE mesh generation algorithms. Due to the complexity and diversity of human anatomy, there is unlikely to be a single best mesh generation strategy for all applications. This paper describes a framework for the developer to easily integrate his mesh generation algorithm into a readily available system that can convert the patient’s medical images to a finite element model.

The toolkit is intended to be used by the authors to develop a surgical simulation system for the spine. Software components possess well-defined boundaries. They can be built on their own. It is possible for one team to develop different components at the same time while another team focuses on the user interface for the application. Since components are encapsulated, it is possible to make changes to their implementation without affecting the system that relies on them. Hence, it is not necessary to recompile the system because an adjustment is made to a single component or a new meshing algorithm is introduced. i.BioMech will increase the productivity of the software engineering team. Fig. 6 shows a snapshot of the GUI for the virtual spine workstation (VSW).

While the i.BioMech toolkit uses COM as the component-oriented technology, the ideas and software development principles that are presented in this paper might also be implemented with other component-oriented technologies such as the common object request broker architecture (CORBA) [35] and Java/RMI [36]. However, the implementation on CORBA and Java/RMI could be difficult due to the difference in architecture. With the emergence of new and better component-oriented development platforms, for example .NET [37] for COM and Jiazi [38] for large scale component programming in Java, difficulties in component programming will be reduced and this will further increase its utilization.

Component-oriented method combines the programming models of object based and distributed computing. This enables engineers to take advantage of the parallel processing capabilities associated with distributing computing, and yet maintain highly reusable codes as in objected oriented software. The ability of COM in supporting multithreading is probably most advantageous compared to other software engineering methods. On a multi-core system, which is now coming into general use, the different threads with the COM components can be executed simultaneously on different cores. For finite element mesh generation which is clearly an extensive software process, exploitation of parallelism is important.

**Acknowledgements**

The authors wish to thank Agency of Science and Technology (A-STAR), Singapore for its financial support of this research.