The Multimod Application Framework

Marco Viceconti¹, Luca Astolfi¹, Alberto Leardini¹, Silvano Imboden², Marco Petrone², Paolo Quadrani², Fulvia Taddei¹, Debora Testi¹, Cinzia Zannoni²,
¹ Istituti Ortopedici Rizzoli, via di Barbiano 1/10, Bologna 40136, Italy
² C.I.N.E.C.A., via Magnanelli 3/6, Casalecchio di Reno, Bologna, Italy
viceconti@tecnio.ior.it, c.zannoni@cineca.it, leardini@ior.it

Abstract

This paper presents the Multimod Application Framework, a software framework for the rapid development of computer-aided medicine applications. This framework, distributed under an Open Source licence, is being developed as part of the Multimod Project, a multi-national research endeavour partially supported by the European Commission through the Fifth Framework Programme. This application framework provides an effective re-use model for visualisation and data-processing algorithms that may be incorporated into the Framework with moderate overhead and then made available to the biomedical research community as part of a complete set of applications.

1. General Description of the Framework

The Multimod Application Framework (hereinafter MAF) is a very ambitious software development project. The final framework should provide three layered software libraries that different users will exploit to realise the most disparate software systems.

A first possible mode of use is that of a programmer who wants to use some of the algorithms and functions we shall develop during the Multimod project in other applications, or to extend the MAF with his/her own algorithms. This mode of use suggests a software layer independent for the GUI library, and easy to call from high-level programming languages.

The second use is that of an Application Expert who wants to develop a specialised computer-aided medicine application targeting a specific medical diagnostic, treatment or rehabilitation service. This user wants focus on the application and to do as little as possible in terms of implementation. Ideally, there should be a toolbox from which the user can pick only the operations required, specialise such operations by fixing parameters or by combining operations in a specialised macro, and add an application-specific user interface.

The third mode of use is that of the final user. Here, it is necessary to distinguish the needs of the medical technician, who usually prepares the data, from those of the medical professional, who wants to focus on the specific medical service to be performed. The technician usually has some understanding of the technical issues involved with the elaboration of digital data, and wants a powerful and productive user interface for the process of preparing and organising the data. The medical professional, in contrast, requires a user interface that is as familiar as possible, so that he/she can focus on the task rather than on the application.

From these definitions of the scope and uses of the MAF, it is possible to derive a four-layer framework (Fig. 1).

The Multimod Consortium has tried, whenever possible, to avoid “reinventing the wheel”. To fulfil its ambitious objectives, the MAF had to provide very sophisticated visualisation and processing of large scientific datasets. This was realised by basing the MAF on some of the best Open Source software libraries currently available. All visualisation services are obtained from the Visualisation Tool Kit (VTK). This library is becoming a de facto standard in the area of scientific visualisation, thanks to its clever architecture and to the powerful visualisation functions it provides [1]. VTK is the true foundation of the MAF; its architecture of classes has been adopted also for our Multimod Foundation Layer (MFL), the lowest of the three architectural layers forming the MAF. In this sense, the MFL may be seen as an extension of VTK. Because of this, all the classes we added to VTK to realise the MFL are grouped in a library called Surgical-VTK or SVTK for short. Thus, the two main elements forming the MAF foundation layer are VTK and SVTK.
To this core, we are adding specialised libraries, as the need arises. The MAF currently includes V-Collide, a collision-detection package based on the OBB method [2], and the NLM Insight Tool-Kit (ITK) a new library developed by a prestigious consortium of six USA research institutions [3]. ITK implements the current state of the art for segmentation and registration algorithms of diagnostic datasets.

**Fig. 1. General Architecture of the Multimod Application Framework.** For each architectural layer (left side), the type of user who is likely to exploit it is indicated (right side).

In general, these additional libraries are not fully integrated, but rather wrapped into a SVTK class that exposes the sub-set of methods useful to the MAF. Using these foundation libraries, we developed a very powerful representation of biomedical data, called the Virtual Medical Entity (VME). Thanks to this sophisticated data structure, the MAF can integrate into a single program, images, volumes, surfaces, motion data, movies, finite element meshes, etc.; MAF programs can cope with multidimensional time-varying datasets without any particular programming overhead.

The second layer of the architecture is called the Low Abstraction Layer (LAL). It implements all Virtual Medical Services: the framework services such as undo or copy and paste; the Operations, which are high-level procedures that modify an existing VME; and the Views, which provide an interactive visualisation of the VMEs. The user interface is realised by integrating into the MAF a portable GUI library, called WxWindows [4]. Thanks to this, any application based on the MAF can be compiled on a variety of platforms. We are currently maintaining Windows and Linux distributions, but for other projects we have also compiled MAF programs under SGI Irix and other Unix operating systems. Although we have never tried, in principle nothing should prevent a proper compilation also under the Mac OsX operative system.

The third and final layer is called the High Abstraction Layer. The HAL exposes the interface that all programmers should use to create specialised programs starting with the MAF. The general behaviour of MAF applications is coded in a class called Logic (Fig. 2).

Logic communicates with four managers, VMEManager, ViewManager, OpManager, and InterfaceManager, which provide hardwired behaviour to all elements of the same type. To ensure the highest level of modularity, each manager is totally independent from the others and communicates only with its elements, respectively Virtual Medical Entities, Views, Operations and Interface Elements.
The communication takes place following a command-event model that ensures the child modules do not have to know their parent module in the architecture.

![Fig. 2. Logical structure of a MAF Application.]

### 2. How to use the MAF

An application programmer can create a vertical application in two ways. The simplest is to create a new instance of Logic, define which Views and Operations it should make available, and compile it. A new MAF application, fully based on the default MAF logic, is thus created with the following few lines of code:

```cpp
// --- MyApp.h ---
#include wxWindows.h
#include mafServices.h

class MyApp : public wxApp
{
void OnInit();
void OnExit();
mafLogic *m_Logic;
}
```

together with a few other lines of code to define, in the initialisation method, which Views and Operations the application should include:

```cpp
// --- MyApp.cpp ---
#include MyApp.h

void MyApp::OnInit()
{
    m_Logic = new mafLogic();
m_Logic->AddView( new mafV1() );
m_Logic->AddView( new mafV2() );
m_Logic->AddOperation( new mafOP1() );
m_Logic->AddOperation( new mafOP2() );
m_Logic->Show();
}
void MyApp::OnExit()
{
delete m_Logic;
}
```

After compilation, the application programmer obtains a fully featured application, with all MAF services available and with only those views and operations that are relevant for the specific application.

The second approach to the creation of a vertical application involves some customisation. Two levels of customisation are explicitly supported: the addition of new views and new operations developed starting from available templates, and the modification of Logic to obtain an application’s behaviour different from the default. While a default MAF application can be created by anybody able to run a compiler, the creation of new operations and views requires C++ fluency, and the customisation of Logic requires also a deep knowledge of the MAF internals. On the other hand, using the MAF any programmer can produce in the quickest possible way a specialised application, with the level of customisation required.

To provide some examples of use, we can foresee the following scenarios:

- **Biomedical Research user**: a biomedical researcher with no interest in programming, willing to use the MAF technology in research activities. Biomedical researchers will find particularly interesting a MAF application called Data Manager, a default MAF application that exposes all views and all operations currently available. While complex in its use, because of the huge array of options, Data Manager is very powerful and helps the user to understand the potentialities of the MAF [5].

- **Clinical user**: a medical professional with no programming skills, willing to use MAF technology in clinical practice. This user will run vertical applications developed using the MAF. The first to be made available will be Hip-Op 2.0, a CT-based pre-operative planning environment for total hip replacement. Other similar applications are expected to follow.

- **Clinical Bioengineer**: bioengineers active in clinical contexts may develop vertical applications aimed at supporting clinicians in various activities, by simply adding selected views and operations and recompiling the default logic.

- **Medical Technology Researcher**: many researchers are developing complex algorithms to process or to visualise biomedical data. If they code their algorithms as additional Operations or Views, they can test them within a complete application environment, rather than wasting time to reinvent the wheel every time.

- **Biomedical Software Developer**: programmers who develop computer-aided medicine applications will find a great advantage in using the MAF. By adjusting the application logic to their needs, and by adding new specialised views and operations, they can rapidly develop specialised applications focusing on features that give added value to their solutions.

- **Multimod Developer**: the MAF is distributed-ownership library. Thus, the most skilled programmers may join our Open Source program and see their bug fixes, improvements and additions included in the next MAF distribution, while retaining the ownership of the code they develop.
3. Current Capabilities of the MAF

The MAF is a framework still in its infancy, and will be expanded considerably in the next two years. Further, following the formal launch of the MAF Open Source program in late 2003, we hope to attract other programmers willing to contribute to the code base. Nevertheless, the MAF is already a very powerful tool and can be used to solve a variety of problems in clinical practice and biomedical research. We list some of its capabilities and exemplify them using the Data Manager.

3.1 Data Types

An essential step in using the MAF is the ability to import biomedical data into the VME Tree. MAF imports 3D volumes (CT, MRI, PET, SPECT, 3D Ultrasounds) written in DICOM format, the international standard for medical imaging. Polygonal surfaces can be imported from STL or VRML files. STL format, in particular, can be used to exchange geometry with CAD programs, since while MAF does not support the mathematical surfaces such as NURBS that are commonly used in CAD programs, most STL exporters provided in commercial CAD are able to generate a tessellation of the surface with the required accuracy. On its side, MAF can process efficiently very large numbers of polygons, so it is possible to import CAD models with a sufficient level of geometric accuracy. Standard 8 or 24-bit 2D images can be imported from JPEG, TIFF, GIF and BMP files. 16-bit diagnostic images can be imported only in DICOM format. Motion capture data can be imported as PGD and as C3D files. C3D files are translated by the MAF into a moving cluster of models with a sufficient level of geometric accuracy. Standard 8 or 24-bit 2D images can be imported from JPEG, TIFF, GIF and BMP files. 16-bit diagnostic images can be imported only in DICOM format. Motion capture data can be imported as PGD and as C3D files. C3D files are translated into a sub-tree list of moving landmarks, which can be later clustered into a segmental motion using a specific operation. PGD format, less popular but more sophisticated, already provides the segmental trajectory and is translated by the MAF into a moving cluster of rigidly linked landmarks for each segment in the file. Any other dataset, even if it is not supported by the MAF, can be imported and is stored, unchanged, alongside the other data. These unknown data are usually processed with an operation called Open With External Program, where the correct application is launched on the basis of the file extension. Thus, we could, for example, add to our database an Excel file containing all our calculations, which would then be stored together with the other data in a single repository.

3.2 The VME Tree

The efficacy of creating databases of disparate data is considerably extended by the ability to organise VMEs into a hierarchical tree, the VME Tree. The pose matrix of each VME is interpreted as the pose of the VME with respect to the parent VME. In this way, one can create complex spatial chains of datasets, each referring to its parent. With the notable exception of motion data, imported datasets do not contain information on the pose of the dataset with respect to a global reference system. Thus, the relevant VME is created attached to the root of the VME Tree, and an identity pose matrix is assigned to it. Thereafter, one can cut and paste, or re-parent, the VME under another VME. Cut&Paste leaves the pose matrix unchanged, while Reparent leaves the VME pose unchanged. In this way, one can keep the relevant pose of the VME unchanged by attaching it to another parent, or relocating the VME in the VME Tree without changing its absolute pose in space. Specific operations can modify the pose of a VME of known quantities or move it to a new pose in which the VME is aligned with another VME of the same type.

The entire VME Tree can be saved in the Multimod Storage Format. Currently, the directory structure of the MSF is based on the file system. Thus, saving a VME Tree creates a directory, not a file. Inside this directory, there is an XML file that defines the VME Tree structure and all metadata of the VMEs forming it, and various sub-directories containing the raw data of each VME. By copying that directory and its content to another location we move the entire VME tree. We propose that zipped MSF files could become the de facto standard for exchanging biomedical data, bearing in mind that, by use of the Data Manager, the data contained in an MSF file can be not only opened and visualised, but also exported into various standard formats.

3.3 Visualisation

The MAF provides powerful ways to visualise these disparate data. Most views can render data of different types simultaneously, display time-varying data, and support interaction by pan, zoom, rotate of the camera.

Surfaces can be displayed using a perspective surface view, which supports multiple lights, fly-to camera motion and a multi-resolution scheme that ensures interactivity even with huge number of polygons.

Volumes can be visualised with volume rendering views. The latest ray-casting view displays, in the same view, a CT or a MRI volume, together with polygonal surfaces. The volume render transfer function is very sophisticated and can make visible subtle differences, i.e. to visualise muscles in a CT Dataset.

Another volume view is the isosurface View. This classic visualisation method, which uses the Marching Cube algorithm to compute the polygonal surface that lies at a given density value in the volume, has been re-invented by introducing a new contour rendering technique. Contrary to the Marching Cubes algorithm, which required some minutes to run for full-resolution CT datasets, with this method the isosurface view becomes interactive and the user can adjust the density threshold interactively while observing the outcome. The resulting isosurface can also be saved, providing also a first segmentation tool.
A third volume visualisation method re-implements the so-called Digitally Reconstructed Radiograph (DRR) again to achieve interactive refresh rates with full resolution CT volumes. In this view, the user can see the CT volume as a radiographic image projected from a point that can be changed interactively.

The list is completed by simple views to display 2D images or single volume slices, or complex composite views that combine various modalities into the so-called Multimodal Display representations.

3.4 Operations

Operations are tools that can be used to modify the VMEs. There are operations to edit the VME Data, to transform the entire dataset, to register the pose of two VMEs, etc.

The Transform operation applies a generic affine transformation to the selected VME, defined with respect to any existent implicit or auxiliary reference system. After the transformation matrix is computed, a polar decomposition separates the roto-translation and the scaling components. The first pre-multiplies the VME pose matrix, defining the new pose; the second is simply flushed if the transformation was intended to be rigid (in this case the scaling is generated by round-off errors), or applied directly to the VME data if scaling was intended.

The transformation can be defined not only in terms of matrix components, but also expressing rotations in terms of cardanic angles, Euler angles, attitude vector, quaternion, or helical axis.

There are operations to create new VMEs, such as the Virtual Palpation operation, which places new landmark objects on to another VME. Virtual palpation is used for a variety of point-based anatomical registrations. It is possible to register on to a patient-specific skeletal anatomy an atlas of muscle insertions based on Terry’s musculo-skeletal database [6]. Motion data can be used to animate patient-specific bone models, by registering the landmarks palpated in the motion-capture session with those virtually palpated on the bone models. Last, but not least, with virtual palpation the user can define anatomy-based auxiliary reference systems, and use them to define spatial transformations.

4. Future developments

The Multimod Consortium is working full time to continue the development of the MAF. While some future developments will be defined on the basis of the outcomes of current research activities, some of the major additions we plan to make can be anticipated.

A fundamental aspect of biomedical research that has been neglected in the MAF so far is simulation. While the MAF will never become a numerical simulation environment, it could evolve into a specialised environment for pre-processing medical data to form the simulation models, and to post-process the simulation results alongside with diagnostic data, providing anatomically rich and clinically relevant visualisations of the simulation results.

New VME types will be developed to support the finite-element entities (nodes, elements, meshes, materials, and boundary conditions). Importers will be developed for finite-element entities and for the associated simulation results. Specific views will visualise the results of the simulation in unconventional ways. Support for multi-body dynamics solvers is planned in terms of import-export of geometry and of motion data. An essential element that is missing is a good mesh-generation library. While the Multimod consortium does not plan to develop it, we shall consider the opportunity to integrate a third-party library.

On the visualisation front, some of the consortium members will work on the visualisation of deformable anatomical structures, such as muscles, to ensure real-time interactive visualisation of deforming objects, even of their volume data. A possible objective is the creation of a digitally Reconstructed Fluoroscopy view, where a static CT dataset, a bone segmentation, and some motion data may be combined to create a synthetic fluoroscopy to show the joint motion.

A third development front comes from a distinct but related research project, called Multisense. The Multisense consortium has adopted the MAF and plans to extend it to support multisensorial interfaces and virtual reality input-output devices, such as haptic or 3D trackers. This requires an architectural revision that is already in progress.

On the algorithmic front, we plan to develop a 2D-3D-registration algorithm, based on the very efficient DRR, which could be used to provide marker-less RSA solutions. A pre-operative CT scan of the joint would be used to create accurate models of the 3D skeletal anatomy, which would then be registered to post-operative 2D radiographs. The same operation would be repeated with the CAD models of the prosthetic components, which would be registered to their 2D image in the radiograph. Repeating on follow-up radiographs would indicate if there is a change in the bone-implant relative pose, and thus implant migration.

5. Conclusions

These are just examples of the future developments we plan for the MAF. Following the launch of the MAF Open Source program we expect to develop these and other features of the framework in close collaboration with other groups that are willing to integrate their algorithms and add their knowledge to this endeavour.

The idea of developing a software framework to accumulate the collective skill and knowledge of European computational biomechanics and bioengineering is, in our opinion, of great value. The Open Source program, and the collective ownership of the code should assure all
researchers of the opportunity to contribute to the code base.

At the same time, we hope to see the Data Manager and many other applications based on the MAF widely used by our research community; it is a very powerful tool, which can solve a many problems that are confronted on a daily basis. At the same time, we expect to receive from this user community suggestions and indications on how further to improve the Multimod Application Framework, so as to make it a true cornerstone of biomedical technology research.

Acknowledgements

This work was partially supported by the European Commission through the project Multimod, contract IST-2000-28377.

References


