VAVframe: A New Concept in Platform Design

Erhan Durusüt, Uğur Bozkaya, Enver Yağcı, and Burak Acar, *Member, IEEE*

***Abstract* — Existing medical visualization platforms share common features such as user extensibility, a graphical user interface to construct network of computational modules and interaction with the application execution. However, all of these platforms oversee the communication problem between the end-users (MDs – Medical Doctors) and the developers (Engineers). This avoids utilizing the potential of such platforms in full. VAVframe (Volumetric Analysis and Visualization Framework) proposes a new intermediary user group between the above mentioned two user classes to solve this problem to maximize the benefits of such a platform. This new class of users are called *Application Designers (ADs)*. They enable VAVframe, in addition to providing conventional platforms' functionality, to respond to the MDs' needs in a fast and efficient way.**

***Keywords* — Flexible Medical Framework, Software Platform, Volumetric Analysis and Visualization.**

I. Introduction

THE continuing development of technology in medicine, especially in the field of medical imaging has provided doctors with vast amount of data. It is practically impossible to examine that much data using conventional means. Thus, medical volumetric analysis and visualization software have started to play an increasingly important role in medicine. The functionality of such software range from mere visualization of data (such as MRI, X-Ray, etc.) to more advanced systems (such as Computer Aided Detection/Diagnosis, etc.).

Having recognized the common features of medical software developed for very different clinical applications, such as colonoscopy and angiography, flexible, multi-purpose software platforms have gained popularity [1]-[3]. The common features of these platforms are user extensibility, modularity, intuitive graphical user interfaces (GUIs), a set of internal data representations, data/memory/control management mechanism and a library of routines for general visualization. Despite such features exploit the similarities between different medical applications and thus increase efficiency by decreasing redundant code development; they still can not utilize the potential benefits of such platforms in full. The major obstacle is the communication problems between MDs and the engineers. This stems from two facts related to their backgrounds:

- The terminologies they use in their working environments differ significantly
- They have incompatible priorities, such as the design of the GUI screen versus the computational cost of a certain algorithm.

Neither the MDs' nor the engineers' concerns and needs should be ignored for complete utilization of the potential benefits of flexible software platforms.

The software platform presented here, VAVframe (Volumetric Analysis and Visualization Framework), introduces a new and unique class of users, the *Application Designers (ADs)* in addition to the conventional user groups, End Users (MDs) and Software Developers (SDs). VAVframe also provides the common features of flexible software platforms.

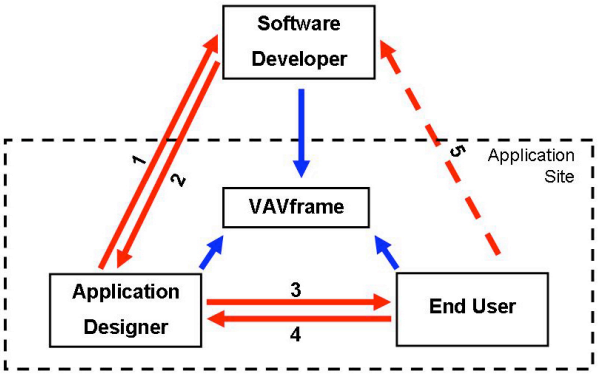

Figure 1 VavFrame system architecture

1- Problems that ADs could not overcome 2- High-level specifications of modules 3- Training 4- All sorts of requests (even the cosmetic ones) and bug reports 5- Observation of usage at application site

II. Triangular User Architecture

VAVframe is based on three types of users: Application Designers (ADs), End Users (MDs) and Software Developers (SDs). Figure 1 shows the relations between these three user groups.

SDs and MDs are conventional user types. They are mainly engineers and medical doctors respectively. As we have explained in Section I, the miscommunication between them degrades the performance of platforms. However, since the SDs are the ones who write the code, they have to be in contact with MDs, i.e. they have to get

This work was in part supported by EU 6th Framework SIMILAR Network of Excellence (www.similar.cc), TÜBİTAK funds under project KARIYER-DRESS (104E035), B.A.P. (03HA202).

E. D. Author is with the Department of Computer Engineering, Boğaziçi University, Turkey (phone: 90-212-3596465; fax: 90 212 2872465; e-mail: durusute@boun.edu.tr).

U. B. Author is with the Instititute of Biomedical Engineering, Boğaziçi University, Turkey (e-mail: bozkaya@gmail.com).

E. Y. Author is with the Program of System & Control Engineering, Boğaziçi University, Turkey (e-mail: enveryagci@yahoo.com).

B. A. Author is with the Department of Electrical & Electronics Engineering, Boğaziçi University, Turkey (phone: 90-212-3596465; fax: 90 212 2872465; e-mail: acarbu@boun.edu.tr).

feedback from them and provide occasional training about the advantages / disadvantages of the algorithms.

The contribution of VAVframe is to introduce ADs. They are computer friendly people who do have certain amount of medical background, they are not necessarily (and preferably) MDs. ADs reside at the application site with the MDs. Their unique position and background provides the following advantages:

- They share the same terminology with MDs.
- They have an insight into the MDs' priorities.
- They have complete knowledge of the capabilities of VAVframe (all of the algorithms implemented) yet they do not know the implementation details.
- They work at the application site and thus can respond to MDs' requests immediately.
- They can even act as a communication link between SDs and MDs when required.

ADs will be responsible for tasks ranging from customizing the GUI screen for individual MDs to designing and developing specific applications. However, the ADs' development does not include algorithm development and/or implementation. They act at a much higher level and work with a special GUI designed for their use. ADs GUI environment is analogous to what similar platforms provide for system design. They design and implement the requested system by defining the interconnections between different modules (implemented algorithms for data processing and visualization available within VAVframe) and setting their parameters. Such a task is too complicated for the majority of MDs and unnecessarily time consuming for SDs in most cases. The delays are due to the communications deficits mentioned above. The ADs' in-depth knowledge of the capabilities of the implemented algorithms and medical jargon provides them a special place within this request/respond loop.

From a component based design approach, the role of AD is similar to that of the analysts and the solution architects. However, VAVframe abstracts the ADs from the framework further and emphasizes its role as a co-worker of the end-user not the software developer. As such, ADs' primary responsibility is to meet the end-users' personalized requests. ADs are envisioned to be technology friendly but non-technical people. Thus, VAVframe aims at surpassing the communication barrier between end-users and developers via ADs. For example, consider two end-users using identical programs, one of them is more confident with it and wishes to set the application parameters, while the second one is inexperienced (say, an intern) and wishes to use a fixed set of parameters without even being able to access the parameters. ADs will be able to customize the same program for these end-users, even setup the user screen.

The interaction of MDs, ADs and SDs will be made clearer in Section V.

III. VAVFRAME ARCHITECTURE

A. Data-Flow

Most of the modular environments are based on the data-flow architecture. The data-flow architecture is described well in [4]. A complete process is composed of a series of operations applied to a data stream. The series consists of a set of interconnected processing modules. The modules are the computational units of the application and their network is the operational unit. So, the application's functionality is determined by:

i) the types of the modules within the network
ii) the interconnections between the modules.

The flexibility, and consequently the dynamicity of the complete system is achieved by designing the applications' architecture as a network of modules. As long as the modules are generic enough, a wide range of problems can be easily solved. The inter-module communication is done by passing messages through unidirectional input and output ports. Depending on the number of ports and their types, modules could be classified as follows:

- Sources: They usually have an interface with an input device and have one or several output ports.
- Sinks: They have an interface with an output device and have one or more input ports.
- Filters: They have both input and output ports. They process the information fed into the input port and write the output to the output port.

Unidirectional input and output ports are not a limitation. They rather increase components autonomy. Provided that there are no feed-back loops, processing is unaffected by the presence or absence of connections at the output port(s). Since all components depend only on the upstream modules, it is possible to change the output connections at runtime.

B. DLL based Modularity

Modules play a key role in data-flow architecture. Applications that follow this pattern manifest an increased degree of modularity, which makes it very easy to distribute the development tasks among different groups, virtually independent groups, such as the visualization group, the image registration group, etc. or the application specific groups such as virtual colonoscopy group, etc. ADs, on the other hand, who are blind to the implementations but have a thorough knowledge about the features of modules, can create new applications by simply connecting modules and setting parameters *at the application site.*

The extensibility of the system is provided by the plug-in based design of the platform. The components which can be added to the system are classified based on their functionality and their number of inputs and outputs. The main components in the system are:

- Data Source: It does not have any input and generates artificial data, such as cubes or spheres, to be used in the system.
- Data Reader: It also generates output without any input, but it reads data, in a preset list of formats, from the file system.
- Task: It can have one or more inputs and one or more outputs. All data processing is part of this group.
- Data Writer: It has only input data and is responsible for storing the given data in a specified file format.
- Display: It is responsible for converting the data fed to its inputs to geometric primitives and rendering them in windows.

All of the components of these types can be added to the system easily by creating a DLL by the SDs. In the DLL, the developer should create a class derived from the base class of the component type he/she chooses. The behavior of the new component can be customized by implementing the virtual functions provided by the base class.

IV. Implementation

VAVframe was written in C++ using Microsoft Visual C++ .NET 2003 development platform on an Intel P4 2.6Ghz Computer running MS Windows XP Professional. Visual programming such as window creation and management is facilitated by the MFC (Microsoft Foundation Classes) library which is available with MS Visual Studio. For the visualization tasks, we have used VTK (Visualization Tool Kit) for our DLLs. VTK is a freely available C++ class library for 3D graphics and visualization merging the power of Object Oriented Programming (OOP) with the luminous world of computer graphics and visualization [5]. We used ITK (Insight Segmentation and Registration Tool kit) for image processing DLLs. ITK is an open source toolkit mainly developed for segmentation and registration [6]. Figure 2 shows the ADs' design screen.

V. Sample Scenarios

In this section, we will elaborate on two possible scenarios to demonstrate the advantages of VAVframe: Developing a Virtual Colonoscopy (VC) and a Virtual Angiography (VA) tool.

A. *Virtual Colonoscopy*

In this scenario, the MD asks for a virtual colonoscopy tool. He wants to navigate through the segmented colon lumen, examine the colon surface visually and further wants the suspicious lesions (tumors, etc.) to be marked. The AD, having received the request, interprets and decomposes them into sub-tasks until the sub-tasks match the plug-ins present in VAVframe. The sub-tasks and the associated modules are;

- Load the colon data in DICOM format: *DICOM Data Reader*
- Scroll through the axial slices of the 3D data and display each slice in 2D: *Orthogonal Slicer*
- Obtain colon surface data through a semi-automatic segmentation operation which uses manually set seed points: *Region Growing*
- Compute the central axis of the colon: *Medial Axis Transform*
- Detect the suspicious lesions: *Polyp Detector*
- Navigate through the segmented colon along its central axis and render the colon wall, mark the detected lesions: *3D Navigator*

The AD, builds the VC system using the drag&drop GUI provided. The data flow diagram and the modules

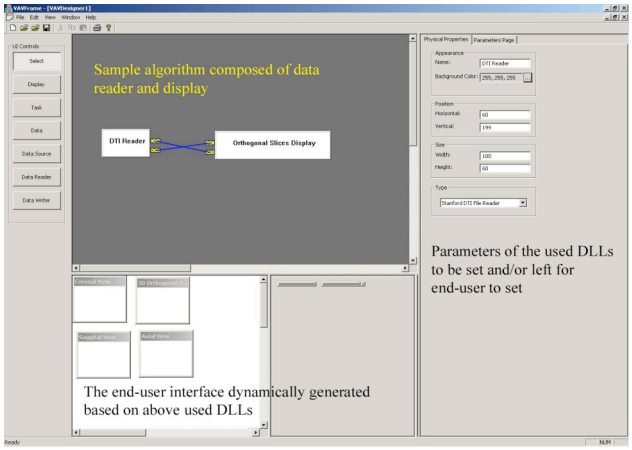

Figure 2 Application Designer GUI is composed of three sections: The data flow design (system design), The MD screen GUI design, The modules' parameters' setting section

used are shown in Figure 3. The MDs user screen GUI is also built / organized by the AD in accordance with the MDs preferences. None of the requests were passed to SDs.

B. *Virtual Angiography*

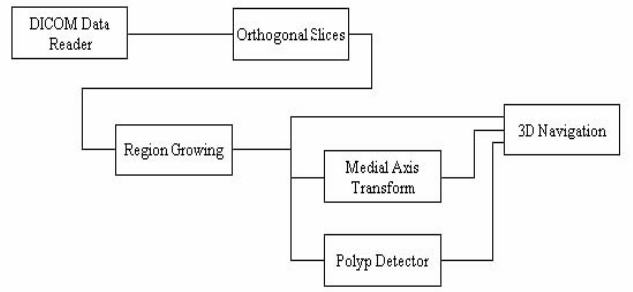

Figure 3 Data flow diagram (Module network) for virtual colonoscopy tool

In this scenario the medical doctor wants a virtual angiography tool. He wants to navigate through the segmented vessels and at the same time he wants to see calcifications on the surface of the vessels. The AD again interprets and decomposes the task into sub-tasks until the sub-tasks match the plug-ins present in VAVframe. The sub-tasks and the associated modules are;

- Load the vessel data in DICOM format: *DICOM Data Reader*

- Scroll through the axial slices of the 3D data and display each slice in 2D: Orthogonal Slicer.
- Obtain vessel surface data through a semi-automatic segmentation operation which uses manually set seed points: Region Growing
- Compute the central axis of the vessel: Medial Axis Transform
- Detect the suspicious lesions: Calcification Detector
- Navigate through the segmented vessel along its central axis and render the vessel wall, mark the detected lesions: 3D Navigator

The operations that the application designer carries out are similar to Scenario 1, they only differ by the way of obtaining regions of interest to be inspected. The data flow diagram and the modules used are shown in Figure 4. The

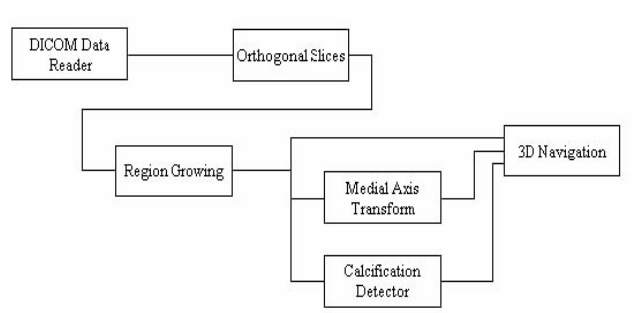

Figure 4 Data flow diagram (Module network) for virtual angiography tool

MDs user screen GUI is also built / organized by the AD in accordance with the MDs preferences.

C. *Discussion*

Although the scenarios are very different from MDs point of view, for application designer both of the requested designs are almost the same. Network modules in VC and VA design both contain data reader, orthogonal slicer, region growing segmentation module, medial axis transformer, 3D navigator and a specific detector appropriate for the requested design.

Normally MDs would use different software packages for the presented scenarios. Here ADs customize the VAVframe for two different MDs and respond to their needs in the same software platform. This will let the computer usage environment more coherent between different specialists while utilizing each others results and different specialists will be familiar to the usage of each others software.

Also another important role of ADs' is to act as a bridge between the MDs and the SDs. A new task which is not available in the current plug-ins can be necessary in the design of a newly requested network module. Here ADs' should decompose the newly requested design into sub-tasks and must find the most basic sub-tasks that are needed and request these sub-tasks as new plug-in modules from the SDs. ADs do not write any code for designing those applications. They only need to be able to decompose the overall task into smaller tasks that are available within VAVframe. As such, VAVframe requires no software programming knowledge for ADs but an abstraction / simplification capability. The above examples were provided to demonstrate the potential use of VAVframe when the required modules are available.

VI. CONCLUSION

This paper introduced a new architecture for flexible software development environments specially designed for medical visualization and post-processing applications. The proposed architecture is based on three user groups: The application designers, the end users and the plug-in developers. This architecture is envisioned to improve the development and customization efficiency. This is achieved via providing;

i) Modularity
ii) Flexible Graphical Algorithm Development
iii) Flexible GUI Design
iv) Easy communication between end-users and SDs while avoiding redundant communication

Taking into account the modularity of the framework it is possible to extend the usage environment easily from medical processes to a large range which requires 3D visualization and processing.

VAVframe is envisioned to act as a repository for volumetric data analysis and visualization tools. All tools can be developed / incorporated into VAVframe easily by the provided wizard. As for all similar platforms, it also avoids unnecessary (repetitive) software development for basic and common tasks and thus it will generate more time for researchers to work on innovative research topics. The unique working environment it provides for inter-disciplinary research (engineers and MDs) will lead to increased productivity by minimizing miscommunication and shortening clinical evaluation processes.

VII. REFERENCES

[1] Brodlie, K. W., J. R. Gallop, A. J. Grant, J. Haswell, W. T. Hewitt, S. Larkin, C. C. Lilley, H. Morphet, A. Townend, J. Wood, H. Wright, "Review of Visualization Systems", Advisory Group on Computer Graphics, Cambridge, England, October, 1995.

[2] Upson, C., Jr. T. Faulhaber, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, A. van Dam, "The application visualization system: A computational environment for scientific visualization", IEEE Computer Graphics and Applications, Vol. 9, No. 4, pp. 30-42, July/August 1989.

[3] Foulser, D., "IRIS Explorer: A Framework for Investigation", Computer Graphics, Vol. 29, No. 2, pp. 13-16, 1995.

[4] Manolescu, D., "A Data Flow Pattern Language", Proceedings of the 4th Pattern Languages of Programming, Monticello, Illinois, September 1997.

[5] Schroeder, W. J., K. M. Martin, and W. E. Lorenson, "The design and implementation of an object-oriented toolkit for 3d graphics and visualization", IEEE Visualization, pp. 93-100, 1996.

[6] Ibánez, L., W. Schroeder, L. Ng, J. Cates and Insight Software Consortium, "The ITK Software Guide", http://www.itk.org, 21 August 2003.