An Application Framework for Rapid Prototyping of Clinically Applicable Software Assistants

Jan Rexilius, Jan-Martin Kuhnigk, Horst K. Hahn, and Heinz-Otto Peitgen

MeVis Research, Universitaetsallee 29, 28359 Bremen, Germany
email: rexilius@mevis.de

Abstract: Computer assistance for clinical diagnosis support and treatment planning are continuously growing fields that have gained importance in several medical disciplines. Although a variety of algorithms are available today, only few are routinely applied for diagnosis support and treatment planning. We propose a hierarchical framework that allows for flexible and efficient development of clinically valuable software prototypes and for systematic evaluation of image processing methods in a research setting. A modular plug-in concept separates algorithmic and framework developments. Several basic components for data-, user-, and workflow-management provide a skeleton that can be customized by both application developer and user. Standardized interfaces allow the communication between both frame and application. Dedicated assistance is provided for an efficient radiological workflow integration. A flexible and simple handling of image processing and visualization algorithms within a modular programming interface is offered through an integration into a visual programming and rapid prototyping platform (MeVisLab). The capabilities of our framework are presented by means of exemplary prototypes, that are currently used in clinical practice.

1 Introduction

The development in a complex application domain such as computerized medical imaging is challenging, and current systems have required significant design and development time and costs. Workflow-oriented software assistants that allow efficient visualization and quantification of image data are required in a clinical environment, for clinical trials, and for algorithm evaluation in a research setting. This is in fact often accompanied with a conflict of interests, since clinical routine requires an easy handling with only few changeable settings, whereas software in a research setting has to support much more elaborate and variable scenarios. Furthermore, several generic issues irrespective of the specific application have to be addressed, such as image import and export support using medical image format standards (e.g. DICOM), user-management, reporting, and documentation functionality. All of these tasks take a substantial amount of development time before any usable application can be completed. Nevertheless, new software systems are often build and maintained from scratch.

Application frameworks are a promising technology that lead out of this dilemma. A framework provides a reusable context that can be specialized to produce custom applications [FS97]. Ideally, components proving common functionality can be easily connected

to create a new product or prototype, while implementation details are encapsulated and thus hidden from the application developer. Therefore, the development process of new applications can be significantly reduced and customer requirements can be considered with low effort. However, the development of new solutions that are not generally applicable is still a common approach for applications in a research setting.

Various tools have been developed in order to assist the design of clinical applications. Software platforms such as Analyze [RH90], SCIRun [JMPW04], VisiQuest [Vis], or the LONI Pipeline Processing Environment [Env] offer a rich set of algorithms for medical and scientific image analysis. Furthermore, image processing and visualization libraries such as ITK [ISea03] or Open Inventor [Wer94] are available. Unfortunately, sophisticated tools for user interface design are often missing within these approaches, especially when using libraries. Even the utilization of PowerPoint® with hyperlinked pages has been proposed for rapid GUI prototyping [SL04], allowing for direct modification on the basis of customer feedback. Besides research applications, several commercial developments are available, such as Easy Vision (Philips, Hamburg, Germany) or syngo (Siemens Medical Solutions, Erlangen, Germany), which provide extensive clinical assistance for different tasks plus general workflow tools. However, the underlying frameworks of these methods are not available for general use. An overview of an architectural design of a medical image processing product family is given in [SL04].

In this paper, we present an extendable framework for the development of software prototypes, focusing on medical applications for image-based diagnosis and therapy as well as for clinical research. Various examples from different clinical disciplines demonstrate the potential of our approach.

2 Methods

Domain specific application frameworks are increasingly becoming important since they can build an infrastructure for applications. Although promising for reuse and easy application creation, common challenges have to be resolved such as high efforts for initial development and maintenance, as well as the often long learning curve for application developers [vGB01]. Furthermore, applications created with a framework are usually tightly coupled with the underlying framework. Particular attention also has to be paid to the usual lack of source code access, which often results in developing components with different or additional functionality from scratch.

Although several free and commercial development platforms and libraries are currently available, a seamlessly integrated and standardized application framework for clinical workflow assistance is still missing. Our proposed framework approach consists of a hierarchical architecture with different process and user interface components that allows for flexible application development. Further flexibility is achieved through integration into a visual programming platform.

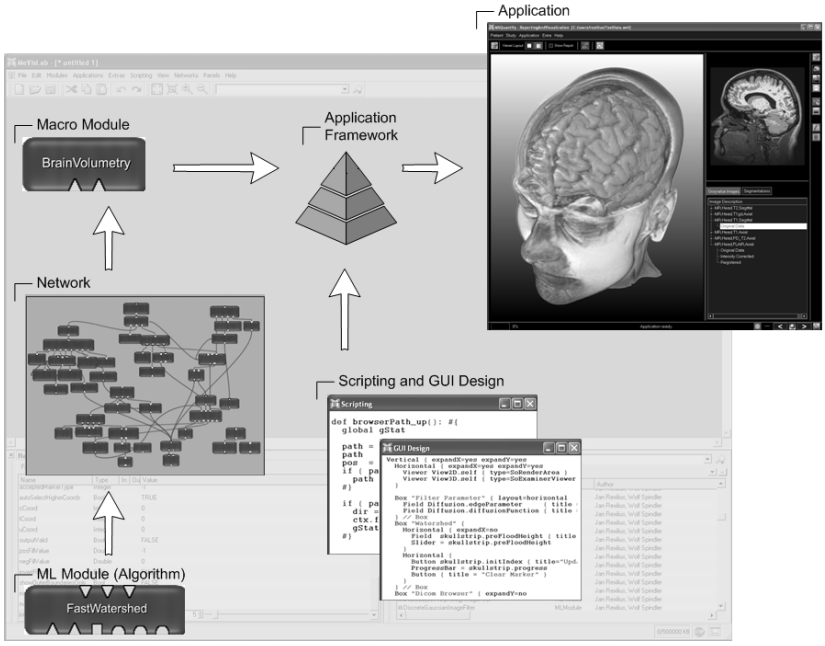

Figure 1: Overview of the various development levels within MeVisLab.

2.1 The underlying development platform

In order to enable rapid prototyping of software assistants, our application framework is conceptualized as an integral part of a modular development platform. MeVisLab [MeV] offers a comprehensive set of image processing and visualization modules for multi dimensional medical images. Detailed module and connection inspectors as well as scripting and debugging consoles offer a flexible development process within a customizable environment with a multiple document interface (MDI).

Image processing algorithms can be created using the MeVis Image Processing Library (ML) that provides sophisticated paging, caching, and multithreading strategies. Various image file formats for import and export are supported including DICOM, Tiff, Analyze, Png, and Raw. For visualization and interactive graphics programming, the open source toolkit Open Inventor [Wer94] as well as own developments such as state-of-the-art volume rendering and customizable 2D and 3D viewers and annotations have been integrated. Furthermore, popular open source libraries such as the Visualization Toolkit (VTK) [SML03] and the Insight Toolkit (ITK) [ISea03] have been wrapped within MeVisLab [KSRea06].

Own algorithmic developments can be easily integrated using a flexible, structured C++ interface or by direct combination of modules through visual programming. An encapsulation of module networks, a so-called macro, is achieved via a user-defined module and user interface. Our multi-layered framework is inserted as an additional level on top of

this conceptual design. An overview of the various development levels within MeVisLab is given in Figure 2.

2.2 The layered architecture concept

A key concept of the proposed application framework is reuse and standardization by relying on component interfaces. This is achieved by a so-called blackbox framework [vGB01]. A basic design decision for building the framework is a multi-layered architecture that encapsulates and abstracts core functionality from the actual application. An event-based information flow permits dynamic interconnections between the different layers. Figure 2 gives an overview of the proposed architecture. Three main layers can be identified:

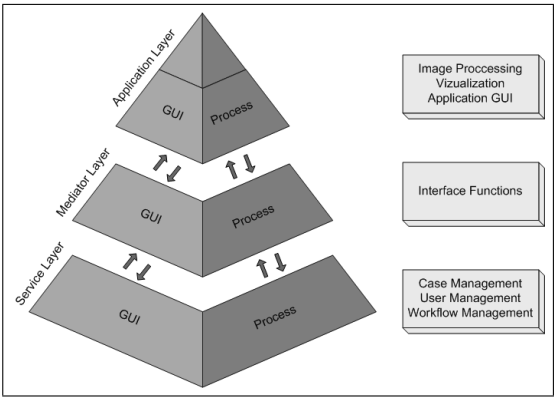

Figure 2: Hierarchical structure of the proposed application framework.

Service Layer. The service layer provides core functionality of the application framework. All relevant information for the general layout, status, and workflow is combined and processed here. To this end, event handling methods are implemented that perform the appropriate state or data changes. Available data formats include multi-dimensional images from various data types, as well as special data objects that can hold, e.g. marker lists, histogram data, or classification parameters. Relevant user and workflow information are accessible via the Extensible Markup Language (XML).

Mediator Layer. In order to allow a decoupling of an application from the internal framework structure, a standard interface has been implemented to handle the flow between these layers. Existing functionality can be reused in each application with interface functions that provide a wide range of potential use cases. Typical user tasks include data load/save, screenshots, and workflow events. In MeVisLab, this event-based handling is accomplished by a list of so-called fields that can be connected to form a demand-driven pipeline.

Application Layer. The application layer implements the external (user) access to the software assistant, which includes the algorithmic development in terms of image process-

ing and visualization, and the associated graphical user interface. Furthermore, application specific handling of the interface functionality is defined here.

Building an application based on the framework now includes the following basic steps:

(1) Implementing algorithms and user interfaces. This first step provides the basis for each software assistant. The usage of MeVisLab allows for an efficient development, independent from the actual framework.

(2) Adding specific input- and output-components (fields) for a direct access to the workflow management of each application.

(3) Configuring the dynamic workflow using the functionality of the mediator layer.

2.3 Features of clinical use cases

As stated in the introduction, several generic issues have to be solved besides the actual application implementation. Our framework provides a wide range of features that allow the developer to concentrate on the application specific features, and offers a clinician general and visually similar functionality in all applications. One of the most important features is an easy and flexible method for accessing and archiving data. Medical image analysis methods are often designed to work with special image sequences from certain modalities. Thus, automatic categorization procedures, e.g. content-based image retrieval systems (CBIR) [LGDea05], are required for convenient data selection. A specialized retrieval system solely based on DICOM information is proposed in [Bre04]. The common problem of false or missing DICOM tags [GKKea02] is resolved by using a set of adaptable rules.

We have developed a case-management system that automatically retrieves the correct images based on their semantic contents. Similar to the method proposed in [Bre04], our approach is based on DICOM information. Furthermore, special structures are available for resulting images (e.g. segmentation masks), seed points, or registration parameters. An XML structure is used to store image and patient data, as well as user and workflow information. This approach allows a support of clinical trials and the retrieval of images with similar diagnostic findings without any time-consuming content-based search on the actual image data. Further features include a user- and workflow-management, batch-processing, and basic customizable layout style definitions. A digital signature can also be attached to each document in order to enhance security and verifiability of the computed quantitative results.

2.4 Features for algorithm development in a research setting

Besides assistance for a clinical environment, our framework also supports algorithm development in a research setting. The utilization of MeVisLab enables modifications of the appearance and behavior of an application at run-time. Thus, crucial spots can already be found early in the development phase. Application templates further increase the usabil-

ity. Dynamic functionality on both the network and the user interface level can be directly added into the framework using scripting languages such as Python.

3 Applications

The application framework has been tested by building several software prototypes for various medical disciplines. Figure 3 (left) shows the user interface of an application for the analysis of diffusion-tensor data (DTI). By means of fiber-tracking, fiber bundles passing trough the vicinity of neurological or neurosurgical risk structures can be reconstructed. A hybrid visualization of DTI and anatomical data enables a dedicated treatment planning. A software assistant for intervention planning with synchronized 2D and 3D visualizations of patient individual intrahepatic anatomy is presented in Fig. 3 (right). All developed applications feature a similar layout and common basic functionality that significantly reduces the learning curve of users, which is invaluable especially for daily routine.

Figure 3: Screenshots of example applications developed with the proposed framework.

4 Discussion and Conclusion

In this work, a generic application framework is presented. A hierarchical architecture allows the flexible development of clinical software assistants. Dynamic connections between different layers are achieved by an event-driven information pipeline. The integration into a visual programming platform adds further flexibility for algorithm development. Convenient DICOM server functionality that can accept data from an external PACS is available. Our approach significantly reduces the required development time for new software assistants, and eases the combination of different but related applications. Currently, the proposed framework is used in more than 10 software prototypes developed at our institute, each consiting of number of applications. Future work will include advanced reporting methods as well as multi-monitor support.

References

[Bre04] J. Breitenborn. *Ein System zur automatischen Erzeugung untersuchungsspezifischer Haengungen und zur Unterstuetzung verteilten kooperativen Arbeitens in der klinischen Radiologie*. PhD thesis, University of Bremen, 2004.

[Env] LONI Pipeline Processing Environment. http://www.loni.ucla.edu/Software/.

[FS97] M. Fayad and D. Schmidt. Object-oriented application frameworks. *Communications of the ACM*, 40(10):32–38, 1997.

[GKKea02] M. Gueld, M. Kohnen, D. Keysers, and et al. Quality of DICOM header information for image categorization. In *SPIE Medical Imaging*, pages 280–287, 2002.

[ISea03] L. Ibanez, W. Schroeder, and et al. *The ITK Software Guide*. Kitware Inc., 2003.

[JMPW04] C. Johnson, R. MacLeod, S. Parker, and D. Weinstein. Biomedical Computing and Visualization Software Environments. *Communications of the ACM*, 47:64–71, 2004.

[KSRea06] M. Koenig, W. Spindler, J. Rexilius, and et al. Embedding VTK and ITK into a visual programming and rapid prototyping platform. In *Proc. SPIE Medical Imaging*, 2006.

[LGDea05] T. Lehmann, M. Gueld, T. Deselaers, and et al. Automatic categorization of medical images for content-based retrieval and data mining. *CMIG*, 29(2):143–155, 2005.

[MeV] MeVisLab 1.3, homepage at. http://www.mevislab.de.

[RH90] R. Robb and D. Hanson. ANALYZE: A software system for biomedical image analysis. In *Conference on Visualization in Biomedical Computing*, pages 507–518, 1990.

[SL04] R. Schwanke and R. Lutz. Experience with the architectural design of a modest product family. *Software Practice and Experience*, 34:1273–1296, 2004.

[SML03] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit*. Kitware Inc. 3rd edition, 2003.

[vGB01] J. van Gurp and J. Bosch. Design, implementation and evolution of object oriented frameworks: concepts and guidelines. *Software - Practice and Experience*, 31(3):277–300, 2001.

[Vis] VisiQuest. http://www.accusoft.com/products/visiquest/.

[Wer94] J. Wernecke. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2*. Addison-Wesley Professional, 1994.