The use of Common Toolkits CTK in the computer-assisted surgery - A demo application

F. Ganglberger¹, Y. Özbek¹*, W. Freysinger¹*

¹ 4D Visualization Laboratory, Univ. ENT Clinic, Innsbruck, Austria
** Contributed equally*

Contact: florian.ganglberger@student.i-med.ac.at

Abstract:

For efficient application development existing in computer-assisted surgery (CAS) frameworks with reusable methods and program frameworks such as IGSTK, MITK or CISST. Whose reimplementation would mean a not to be underestimated workload. The incorporation into complex frameworks may mean an obstacle, even if only parts of them are used.
CTK is a lightweight open-source toolkit for the biomedical area, which is a service-based Plug-in-Framework provides. The design of the CTK framework allows the creation of modular CAS applications. Currently there are only a few comprehensive tutorials of the CTK toolkit on the net. We present a demo application that can be used as a template for standard navigation. In this demo is specifically addressed the use of the central CTK Plug-in-Framework and explained in detail. Thus, the incorporation time is reduced to the development of CAS applications and also created plug-ins / modules can already be easily reused. The source code of the demo application can be requested from the authors.

Key words: Common Toolkit, Plug-in-Framework, Demo

1 Problem Statement

Increasing complexity and security requirements for developers in the implementation of modern, clinically useful applications with new challenges. In order to address with these problems Frameworks such as IGSTK [1], MITK [2], CISST [3] etc. are used. They offer reusable methods and program frameworks that simplify development. Hence the complex re-implementation of standard methods such as visualization [4], registration/segmentation [5] and navigation [1] can be avoided. Plug-in-Framework [6] [7] [8] and the support of state machines [1] [6] also help to cope with the level of complexity, and increasing reliability.

The increasing scope of a framework may mean a higher effort of incorporation, even though only small parts may be used. In return, offer lightweight frameworks with a short incorporation time, e.g.: The Common Toolkit [7].

CTK is an open-source toolkit from the biomedical field, which provides a service-based Plug-in-Framework [7]. The design of the CTK framework allows the creation of modular CAS Applications. Currently, there are few comprehensive tutorials on the net. Although CTK is used in MITK framework [2], there are no open sample applications, whereby enable efficient development more difficult.

We take up in our article regarding this problem and provide an introduction to the CTK Plug-in-Framework, minimal examples of its use, as well as an open source demo application that can be used as a template for standard navigation. The here created modules/plug-ins can be easily reused, by service-/event-based loose coupling.

2 Materials and Methods

In this section is a brief overview of Common Toolkits is given, as well as an introduction to the Plug-in-Framework and its use in the context of computer-assisted surgery. The thereby obtained findings, build the basis of demo application that presented in Section 3.

The CTK framework consists of a comprehensive DICOM loader [9], a DICOM Application Hosting System, a collection of QT Widgets with relating to biomedical applications, a command line interface and a Plug-in-Framework. The individual elements are based on C++ using QT libraries [6]. The following sections discuss only the Plug-in-Framework, since it allows a modular design of CAS applications and the demo application is built upon. The other functions of the framework can be used without restriction to extend the demo application.

CTK plug-ins are modular units within an application that conceptually take separating functions from each other such as communication to a tracker, the preparation of GUI elements or reading data. Plug-ins are compiled separately and loaded by the Plug-in-Framework at runtime. Plug-ins are started by a plug-in activator class, for the parallel execution of a plug-in a GUI or background thread must be started in this class. Each plug-in contains a pointer to the plug-in context, which provides an access point to all loaded plug-in and services. The main program must therefore start only the plug-ins and initialize a framework for the graphical user interface as needed. The communication between plug-ins is provided through services. As a Service Objects are called, have been registered by a plug-in in the CTK Service Registry and are derived from the CTK service class. The CTK Service Registry serves as a central point for plug-in that either provide a service or to access a service. A Service, thereby uniquely defined by its interface (C++ class which usually only contains virtual methods) and its properties (C++ hash map which contains names and status as the standard data types of properties). Thereby it must both the plug-in that provides the service, as well as all the plug-ins that access the service, include the interface. A plug-in provides a service by registering itself in the CTK Service Registry. Other plug-ins can assign the registry, by specifying the interfaces and its properties as a tracker object (instance of a class that implements the abstract class *ctkServiceTracker*). This object will now be notified when the service of the CTK Service Registry is added or removed, or the properties of the service to be changed. The notification takes place synchronously wherefore, plug-ins that propagates a change to the Service Object, it should have no locks. Alternatively, a plug-in or the main program access to a Service Object directly from the Service Registry, a manipulation of the object by several plug-ins should be avoided because of memory safety. Figure 1a shows the service-based communication between the plug-ins. Plug-in A provides the Service Object. Plug-in B accesses a Service Tracker Object, which is notified at a change. Plug-in C and the main program access directly the Service Object.

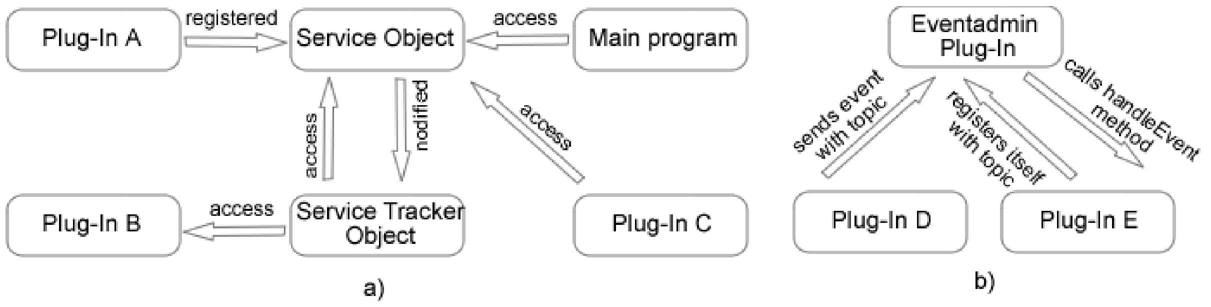


Figure 1: Types of service-based (a) and event-based (b) communication

Another way to communicate between Plug-ins is events. The event management is handled by the CTK own Eventadmin plug-in which provides the service ctkEventAdmin for sending events. This service allows plug-ins (consisting of properties analogous to services) to publish events to a specific topic (of type QString). To receive events for a specific topic, *ctkEventHandler* plug-ins must implement the interface and register itself with the Service Registry. Events may be sent synchronously or asynchronously. Figure 1b illustrates the relationship between the Eventadmin Plug-in and other plug-ins. Plug-in D sends an object under a particular topic, the Eventadmin plug-in calls the handleEvent method of plug-in E (after it has been previously registered on the topic).

In context of computer-assisted surgery service-based communication is especially suitable for the exchange of data that cannot be described by standard C++ data types, because these are not supported by events. Examples for this are the transfer of DICOM [9] data between plug-ins or from QT GUI elements to the main program. In contrast, asynchronous transmitted events; for small, often to update data such as tracker coordinates can be used for real-time navigation.

The service/event-based kind of the data exchange provides loose coupling between individual plug-ins, and thus the modular structure of a CAS application as presented in Section 3. Once created Plug-ins can be not only reused easily in other programs, but also include their own state machines. Each plug-in acts as a standalone module whose status by defined Access points (GUI inputs, service updates, and events) can be changed. Since CTK based on QT [6], the QT offers own state machines implementation. Since plug-ins can include any libraries, the use of other implementations, such as IGSTK [1] is possible.

3 Results

To validate the use of CTK in computer-assisted surgery, and to facilitate the entry into the framework, a demo application is developed. It enables the loading of DICOM [9] images, the integration of tracking data, the marker-based registration of CT data, navigation and the display of (Stereo-) video streams.

The structure of the application can be split into three subgroups:

Main program: The main program handles the loading of plug-ins and provides a basic QT GUI framework (Figure 2a). This is from Service Objects loaded dynamically, that are registered by visualization plug-ins. The right frame contains the visualization of the currently selected plug-ins, left the controls for all plug-ins. By clicking on the plug-in title an active plug-in can be selected.

Visualization-Plug-ins: Visualization plug-ins provide the GUI elements to main program by a service. These consist of a visualization element (VE, Figure 2a, red), which handles the actual graphical representation, and a control element (KE, Figure 2a, blue). For the demo applications are three visualization-plug-ins created:

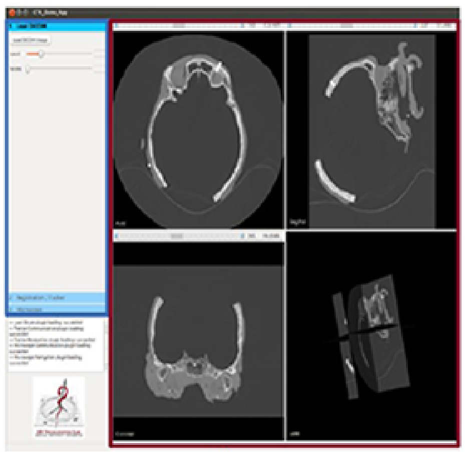
> **Dicom Loader:** By the KE can [9] CT DICOM data is loaded. In VE, the data in a coronal sagittal, axial as well as in a combined view are presented. For this, the IGSTK *DicomReader* [1] is used, the GUI elements are created with the QT Designer [6]. A pointer to the DICOM data is stored in a Service Object, when it is updated the navigation plug-in is notified
>
> **Navigation:** The navigation plug-in takes over the marker-based registration of DICOM [9] data, their visualization and navigation. The visualization is carried out analogously to the DICOM Loader plug-in (Figure2a). The tracking data receives it via asynchronous events from the tracker-communications plug-in. Due to the loose coupling it is possible to replace arbitrary tracker-communication plug-in.
>
> **Stereo-Microscope-Visualization:** Intended as a tech demo, the microscope plug-in serves no benefit to the navigation. The KE contains no control elements In VE the left and right two video streams are displayed on stereo-camera setups. The stream receives it as a GUI element via a service of the microscope communication plug-in. In order to enable a stereo-microscope-navigation, it is possible to receive events from the tracker communications plug-in, and reuse the navigation GUI elements. This feature has not been implemented because of the simplicity.

Communication-Plug-ins: Communication-plug-ins provide visualization-plug-ins by services and events information. They serve as an interface to the hardware. These plug-ins can be replaced any time new (e.g. order to communicate with other trackers) as long as they implement the same service interface or send the same events.

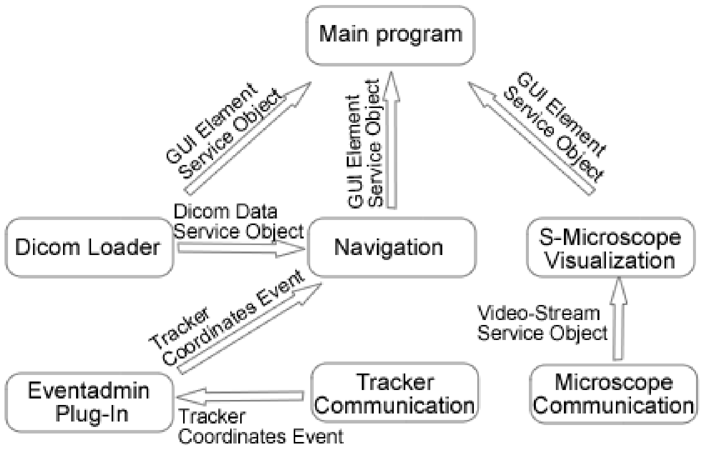
Tracker-Communication: The Tracker-communication Plug-in establishes the connection to a specific Tracker. In the demo application using IGSTK [1] marker coordinates from an Optotrak Certus Tracker (NDI, Germany) are read and by event (via the Eventadmin-Plug-in) to the Navigation-plug-in sent. To integrate other trackers (e.g. by IGSTK [1] or OpenIGTLink [10]) it is enough to create a new Tracker-Communication-Plug-in and send events to the Eventadmin-Plug-in.

Microscope-Communication: This plug-in implements the connection to a stereo-camera setup, which represents the cameras of a stereo-microscope. The video-stream is provided as a QT GUI element by a service to the stereo-microscope-visualization-plug-in. This plug-in makes available all the other parameters of the microscope (zoom, focus, position, etc.).

Figure 2b shows the communication between the main program, visualization-plug-ins, communication-plug-ins and Eventadmin-plug-in, as well as their services. The main program starts all plug-ins. This can be done in any order, because each plug-in has its own state machine. As long as not all required services are initialized, each plug-in waits in a "wait" status. After the starting of visualization-plug-ins, the main program loads the GUI elements in the GUI-framework. Are the DICOM data using the DICOM loader loaded, the Service Object tracker is notified of the navigation-plug-ins. Thereby, the navigation plug-in gets the DICOM data from the DICOM data Service Object. Gets the navigation-plug-in tracker coordinate events (i.e. when the tracker communication has been started successfully), it can begin with the registration and subsequent navigation in the navigation plug-in. From the process independent, receives the stereo-microscope-plug-in the video stream as a GUI element, once the microscope-communication has been initialized.



a)


Main program
GUI Element Service Object
GUI Element Service Object
GUI Element Service Object
Dicom Loader
Dicom Data Service Object
Navigation
S-Microscope Visualization
Tracker Coordinates Event
Video-Stream Service Object
Eventadmin Plug-In
Tracker Coordinates Event
Tracker Communication
Microscope Communication

b)

Figure 2: Screenshot of demo application with marked control element (blue) and visualization element (red), as well as a schematic representation of the plug-in communication

4 Discussion

Compared to comprehensive toolkit in the CAS area, the Common Toolkit provides a short incorporation time. Although, for example MITK [2] is based on CTK, its use is associated with high effort for its extensive range of functions. In contrast, the CTK Plug-in-Framework allows for effective development of prototypes and demos. Plug-ins may be easily replaced or reused by a service-based communication. Any required functions (e.g. segmentation/registration, state machines) can by integration of known libraries such as ITK [5] or QT [6] are used.

The presented demo application represents a simple example of standard navigation using Common Toolkits. By implementing it was demonstrated that CTK can be used in the field of CAS. Using the publicly available source code, the demo offers a help in the development of the CTK Plug-in-Framework.

5 Summary

In the implementation of modern, clinically useful CAS applications, problems such as increasing complexity and reliability demands more and more into the foreground. The use of frameworks suppose these problems, and facilitates the development. With increasing extent, the incorporation time can mean an increased workload, particularly when it goes to fast prototyping or tech-demos. The lightweight Common Toolkit therefore particularly suited for this, but only few tutorials and examples can be found currently in net.

In this article, is taken reference to these problems, and an introduction to the CTK Plug-in-Framework. This allows the creation of modular CAS applications and the integration of external libraries to extend the functionality without having to resort to larger frameworks In Section 2, the creation of plug-ins has been described, as well as their communication by use services and plug-ins. To facilitate introduction into the CTK framework and to review its use in the CAS area, a demo application for default navigation is created, which is presented in Section 3.

The source code of the demo application, minimal examples of plug-ins and a tutorial are freely available on www.voxelmaster.at. Using this assistance, the development of CAS applications can be made simpler and more efficient.

6 References

[1] Cleary K, Cheng P, IGSTK: The Book, Insight (2007)

[2] Nolden M, Zelzer S, Seitel A, Wald D, Müller M, Franz AM, Maleike D, Fangerau M, Baumhauer M, Maier-Hein L, Maier-Hein KH, Meinzer HP and Wolf I, The Medical Imaging Interaction Toolkit: challenges and advances, International Journal of Computer Assisted Radiology and Surgery (2013)

[3] Deguet A, Kumar R, Taylor R, Kazanzides P, The CISST libraries for computer assisted intervention systems,Insight 1-8 (2008)

[4] Schroeder WJ, Martin K, Lorensen WE, The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, Third edition, ISBN 1-930934-07-6, Kitware, Inc. (formerly Prentice-Hall) (2003)

[5] L. Ibanez and W. Schroeder. The ITK Software Guide. Kitware, Inc. ISBN 1-930934-10-6, http://www.itk.org/ItkSoftwareGuide.pdf (2003)

[6] Dalheimer M, Programming with QT, Second edition, ISBN 978-0-596-00064-6, O'Reilly Media (2002)

[7] http://www.commontk.org

[8] Pieper S, Halle M, Kikinis R, 3D SLICER. Proceedings of the 1st IEEE International Symposium on Biomedical Imaging: From Nano to Macro 632-635 (2004)

[9] Bidgood WD, Horii, SC Introduction to the ACR-NEMA DICOM standard. RadioGraphics 12, 345-355 (1992)

[10] Tokuda J, Fischer GS, Papademetris X, Yaniv Z, Ibanez L, Cheng P, Liu H, et al., OpenIGTLink: an open network protocol for image-guided therapy environment. The international journal of medical robotics computer assisted surgery MRCAS 5, 423-434 (2009)