Die Verwendung des Common Toolkits CTK in der computerassistierten Chirurgie – Eine Demo Anwendung

F. Ganglberger[1*], Y. Özbek[1*], W. Freysinger[1]

[1] 4D Visualization Laboratory, Univ. HNO Klinik, Innsbruck, Österreich
[*] haben gleichermaßen beigetragen

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

Abstract:

Für die effiziente Entwicklung von Anwendungen in der computerassistierter Chirurgie (CAS), existieren Frameworks mit wiederverwendbaren Methoden und Programmgerüsten wie IGSTK, MITK oder CISST. Deren Neuimplementierung würde einen nicht zu unterschätzenden Arbeitsaufwand bedeuten. Die Einarbeitung in komplexe Frameworks kann für Entwickler jedoch ein Hindernis bedeuten, auch wenn nur Teile davon benutzt werden.
CTK ist ein schlankes Open-Source Toolkit für den biomedizinischen Bereich, welches ein service-basiertes Plugin-Framework bereitstellt. Die Konzeption des CTK Frameworks erlaubt die Erstellung von modularen CAS-Applikationen. Zur Zeit gibt es jedoch für das CTK Toolkit nur wenige umfassende Tutorials im Netz. Wir stellen eine Demoanwendung vor, die als Template für Standardnavigationen verwendet werden kann.
In dieser Demo wird besonders auf die Verwendung des für CTK zentralen Plugin-Frameworks eingegangen und detailliert erläutert. Dadurch wird die Einarbeitungszeit zur Entwicklung von CAS-Applikationen reduziert und zudem können bereits erstellte Plugins / Module einfach wiederverwendet werden.
Der Quellcode der Demoanwendung kann von den Autoren angefordert werden.

Schlüsselworte: Common Toolkit, Plugin-Framework, Demo

1 Problemstellung

Zunehmende Komplexität und Sicherheitsanforderungen stellen Entwickler bei der Umsetzung von modernen, klinisch verwendbaren Anwendungen vor neue Herausforderungen. Um diesen Problemen zu begegnen werden Frameworks wie IGSTK [1], MITK [2], CISST [3] etc. eingesetzt. Sie bieten wiederverwendbare Methoden und Programmgerüste, welche die Entwicklung vereinfachen. Die aufwändige Neuimplementierung von Standardmethoden wie Visualisierung [4], Registrierung/Segmentierung [5] oder Navigation [1] können somit vermieden werden. Plugin Frameworks [6][7][8] sowie die Unterstützung von State-Maschinen [1][6] helfen zudem den Grad an Komplexität zu bewältigen, und die Ausfallsicherheit zu erhöhen.
Der zunehmende Umfang eines Frameworks kann einen höheren Aufwand für die Einarbeitung bedeuten, auch wenn nur kleine Teile davon verwendet werden. Das schlägt sich vor allem bei der Entwicklung von Prototypen und Tech-Demos zu Buche, für die aufgrund des explorativen Charakters oft nur wenig Entwicklungszeit bleibt. Dafür bieten sich schlanke Frameworks mit kurzer Einarbeitungszeit wie z.B: das Common Toolkit [7] an.
CTK ist ein Open-Source Toolkit aus dem biomedizinischen Bereich, welches ein service-basierendes Plugin Framework bereitstellt [7]. Die Konzeption des CTK Frameworks erlaubt die Erstellung von modularen CAS Applikationen. Zur Zeit gibt es nur wenige umfassende Tutorials im Netz. Obwohl CTK im MITK Framework [2] verwendet wird, gibt es kaum offene Beispielapplikationen, wodurch eine effiziente Entwicklung erschwert wird.
Wir nehmen in unserem Beitrag Bezug auf dieses Problem und bieten eine Einführung in das CTK Plugin-Framework, Minimalbeispiele für dessen Verwendung, sowie eine quelloffene Demo-Applikation, die als Template für Standardnavigation verwendet werden kann. Die dabei erstellten Module/Plugins können dank service-/event-basierter loser Kopplung einfach wiederverwendet werden.

2 Material und Methoden

In diesem Abschnitt wird ein kurzer Überblick des Common Toolkits gegeben, sowie eine Einführung in das Plugin-Framework bzw. dessen Verwendung im Kontext der computerassistierten Chirurgie. Die dadurch gewonnenen Erkenntnisse bilden die Grundlage für die in Abschnitt 3 vorgestellte Demoanwendung.
Das CTK Framework besteht aus einem umfassendem Dicom-Loader [9], einem Dicom Application Hosting System, einer Sammlung von Qt-Widgets mit Bezug auf biomedizinische Anwendungen, einem Commandline Interface und einem Plugin Framework. Die einzelnen Elemente basieren dabei auf C++ unter Verwendung von Qt-Bibliotheken [6]. Im Folgenden wird nur auf das Plugin Framework eingegangen, da es einen modularen Aufbau von CAS Anwendungen ermöglicht und die Demo-Anwendung darauf aufgebaut ist. Die anderen Funktionen des Frameworks können jedoch uneingeschränkt benutzt werden um die Demo-Anwendung zu erweitern.
CTK Plugins sind modulare Einheiten innerhalb einer Applikation, die konzeptionell von einander getrennte Funktionen wie z.B. die Kommunikation zu einem Tracker, die Bereitstellung von GUI Elementen oder das Einlesen von Daten übernehmen. Plugins werden separat kompiliert und vom Plugin Framework zur Laufzeit geladen. Plugins werden über eine PluginActivator Klasse gestartet, für das parallele Ausführen von Plugins muss in dieser Klasse ein GUI- oder Background Thread gestartet werden. Jedes Plugin enthält zudem einen Pointer auf den Plugin Kontext, welcher einen Zugangspunkt zu allen geladenen Plugins und Services bietet. Das Hauptprogramm muss somit nur die Plugins starten und je nach Bedarf ein Gerüst für die grafische Oberfläche initialisieren. Die Kommunikation zwischen Plugins wird über Services bereitgestellt. Als Service werden Objekte bezeichnet, die von einem Plugin in der CTK Service Registry registriert wurden und von der CTK Service Klasse abgeleitet sind. Die CTK Service Registry dient als zentrale Anlaufstelle für Plugins, die entweder einen Service bereitstellen oder auf einen Service zugreifen möchten. Ein Service wird dabei eindeutig über sein Interface (C++ Klasse die üblicherweise nur virtuelle Methoden beinhaltet) und seine Eigenschaften (C++ Hashmap, welche Namen und Zustand als Standarddatentypen von Eigenschaften beinhaltet) definiert. Dabei müssen sowohl das Plugin, das den Service bereitstellt, als auch alle Plugins, die auf den Service zugreifen, das Interface inkludieren. Ein Plugin stellt einen Service bereit, indem er es in der CTK Service Registry registriert. Andere Plugins können der Registry unter Angabe des Interfaces und dessen Eigenschaften ein Tracker Objekt (Instanz einer Klasse, welche die abstrakte Klasse ***ctkServiceTracker*** implementiert) übergeben. Dieses Objekt wird nun benachrichtigt, sobald der Service der CTK Service Registry hinzugefügt oder entfernt wird, oder die Eigenschaften des Services geändert werden. Die Benachrichtigung erfolgt dabei synchron, weshalb Plugins, die eine Änderung am Service Objekt propagieren, keine Locks darauf haben sollten. Alternativ kann ein Plugin oder das Hauptprogramm direkt auf ein Service Objekt aus der Service Registry zugreifen, eine Manipulation des Objektes durch mehrere Plugins sollte aber aufgrund der Speichersicherheit vermieden werden. Abbildung 1a zeigt die service-basierte Kommunikation zwischen den Plugins. Plugin A stellt das Service Objekt zur Verfügung. Plugin B greift über ein Service Tracker Objekt zu, welches bei einer Änderung benachrichtigt wird. Plugin C und das Hauptprogramm greifen direkt auf das Service Objekt zu.

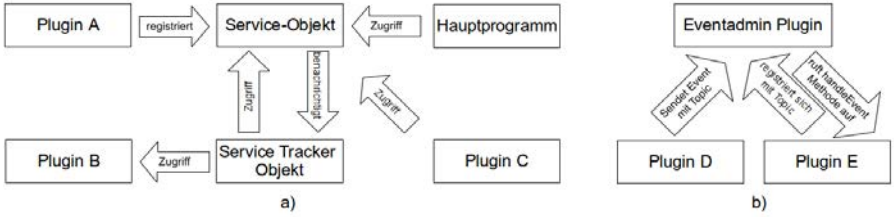

Abbildung 1: Arten der service-basierten (a) und event-basierten Kommunikation (b)

Das Eventmanagement übernimmt dabei das CTK eigene Eventadmin Plugin, welches den Service ***ctkEventAdmin*** für das Senden von Events bereitstellt. Dieser Service ermöglicht Plugins, Events (bestehend aus Eigenschaften analog zu Services) unter einem bestimmten Topic (vom Typ ***QString***) zu veröffentlichen. Um Events zu einem bestimmten Topic zu empfangen, müssen Plugins das Interface ***ctkEventHandler*** implementieren, und sich bei der Service Registry registrieren. Events können dabei synchron oder asynchron gesendet werden. Abbildung 1b verdeutlicht den Zusammenhang zwischen dem Eventadmin Plugin und anderen Plugins. Plugin D sendet ein Objekt unter einem bestimmten Topic, das Eventadmin Plugin ruft die handleEvent Methode von Plugin E auf (nachdem es sich vorher für das Topic registriert hat).
Im Kontext der computerassistierten Chirurgie eignet sich service-basierte Kommunikation besonders für den Austausch von Daten, die nicht durch Standard C++ Datentypen beschrieben werden können, da diese nicht von Events unterstützt werden. Beispiele dafür sind die Übergabe von DICOM [9] Daten zwischen Plugins, oder von Qt GUI Elemen-

ten an das Hauptprogramm. Im Gegensatz dazu können asynchron übertragene Events für kleine, oft zu aktualisierende Daten wie z.B Tracker Koordinaten für Echtzeitnavigation verwendet werden.
Die service/event-basierte Art des Datenaustausches ermöglicht eine lose Kopplung zwischen einzelnen Plugins, und somit den modularen Aufbau einer CAS Anwendung wie sie in Abschnitt 3 vorgestellt wird. Einmal erstellte Plugins können dadurch nicht nur ohne großen Aufwand in anderen Programmen wiederverwendet werden, sondern auch eigenen State-Maschinen beinhalten. Jedes Plugin agiert dabei als eigenständiges Modul, deren Status durch definierte Zugriffspunkte (GUI Eingaben, Service Updates und Events) verändert werden kann. Da CTK auf Qt [6] basiert, bietet sich die Qt eigene State-Maschinen Implementierung an. Da Plugins beliebige Bibliotheken einbinden können, ist auch die Verwendung von anderen Umsetzungen wie etwa IGSTK [1] möglich.

3 Ergebnisse

Um den Einsatz von CTK in der computerassistierten Chirurgie zu validieren, und um den Einstieg in das Framework zu erleichtern, wurde eine Demo-Anwendung entwickelt. Sie ermöglicht das Laden von DICOM [9] Bildern, die Einbindung von Tracking Daten, deren marker-basierte Registrierung mit CT Daten, Navigation und die Anzeige von (Stereo-)Video Streams.
Der Aufbau der Applikation lässt sich in drei Untergruppen Aufteilen:
Hauptprogramm: Das Hauptprogramm übernimmt das Laden von Plugins und bietet ein grundlegendes Qt GUI Gerüst. Diese wird dynamisch von Service Objekten, die von Visualisierungs-Plugins registriert wurde, geladen. Das rechte Frame enthält die Visualisierung des gerade ausgewählten Plugins, das linke die Kontrollelemente für alle Plugins. Mit einem Klick auf den Plugin Titel kann ein aktives Plugin ausgewählt werden.
Visualisierungs-Plugins: Visualisierungs-Plugins stellen dem Hauptprogramm GUI Elemente über einen Service zur Verfügung. Diese bestehen aus einem Visualisierungselement (VE), welches die eigentliche grafische Darstellung übernimmt, und einem Kontrollelement (KE). Für die Demo-Anwendung wurden drei Visualisierungs-Plugins erstellt:
Dicom Loader: Über das KE können CT-DICOM [9] Daten geladen werden. Im VE werden die Daten in einer kranial, sagitalen, longitudinalen sowie in einer kombinierten Ansicht dargestellt. Dafür wird der IGSTK ***DicomReader*** [1] verwendet, die GUI Elemente wurden mit dem Qt Designer [6] erstellt. Ein Pointer auf die DICOM Daten wird in einem Service Objekt gespeichert, bei dessen Aktualisierung wird das Navigations-Plugin benachrichtigt.
Navigation: Das Navigations-Plugin übernimmt die marker-basierte Registrierung der DICOM [9] Daten, deren Visualisierung und die Navigation. Die Visualisierung erfolgt dabei analog zum DICOM Loader Plugin. Die Tracking Daten erhält es über asynchrone Events vom Tracker-Kommunikaions Plugin. Durch die lose Kopplung ist es möglich, das Tracker-Kommunikations Plugin beliebig auszutauschen.
Stereo-Mikroskop-Visualisierung: Als Tech-Demo gedacht, erfüllt das Mikroskop Plugin keinen Nutzen für die Navigation. Das KE enthält keine Steuerelemente, im VE werden links und rechts zwei Videostreams eines Stereo-Kamera Setups angezeigt. Die Streams erhält es als GUI Element über einen Service des Mikroskop-Kommunikations Plugins. Um eine Stereo-Mikroskop-Navigation zu ermöglichen, besteht die Möglichkeit, Events vom Tracker-Kommunikations-Plugin zu empfangen sowie GUI Elemente der Navigation wiederzuverwendenden. Dieses Feature wurde aufgrund der Simplizität nicht implementiert.

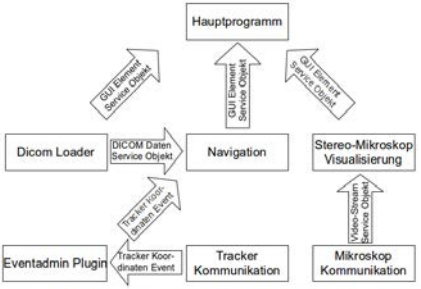

Abbildung 2: schematische Darstellung der Plugin-Kommunikation

Kommunikations-Plugins: Kommunikations-Plugins stellen Visualisierungs-Plugins über Services und Events Informationen zur Verfügung. Sie dienen damit als Schnittstelle zur Hardware. Diese Plugins können jederzeit durch neue ersetzt werden (zB. Um mit anderen Trackern kommunizieren zu können), solange sie die selben Service Interfaces implementieren bzw. die gleichen Events senden.
Tracker-Kommunikation: Das Tracker-Kommunikations-Plugin bildet die Verbindung zu einem spezifischen Tracker. In der Demo Applikation werden mittels IGSTK [1] Marker Koordinaten von einem Optotrak Certus Tracker (NDI, Deutschland) ausgelesen und per Event (über das Eventadmin-Plugin) an das Navigations-Plugin geschickt. Um andere Tracker (beispielsweise über IGSTK [1] oder OpenIGTLink [10]) einzubinden reicht es ein neues Tracker-Kommunikations-Plugin zu erstellen und Events an das EventAdmin-Plugin zu senden.

Mikroskop-Kommunikation: Dieses Plugin implementiert die Verbindung zu einem Stereo-Kamera Setup, welches die Kameras eines Stereo-Mikroskops repräsentiert. Der Video-Stream wird als Qt GUI Element über einen Service dem Stereo-Mikroskop-Visualisierungs-Plugin bereitgestellt. Dieses Plugin stellt alle weiteren Parameter des Mikroskops (Zoom, Fokus, Position, etc.) zur Verfügung.

Abbildung 2 zeigt die Kommunikation zwischen Hauptprogramm, Visualisierungs-Plugins, Kommunikations-Plugins und Eventadmin-Plugin, sowie ihren Services. Das Hauptprogramm startet alle Plugins. Das kann in beliebiger Reihenfolge geschehen, da jedes Plugin über eine eigene State-Maschine verfügt. Solange noch nicht alle benötigten Services initialisiert wurde, wartet jedes Plugin in einem „wait“ Status. Nach dem Starten der Visualisierungs-Plugins lädt das Hauptprogramm deren GUI Elemente in das GUI-Grundgerüst. Wurden mit dem DICOM Loader DICOM Daten geladen, wird das Service Tracker Objekt des Navigations Plugins benachrichtigt. Dadurch holt sich das Navigations-Plugin die DICOM Daten aus dem DICOM Daten Service Objekt. Erhält das Navigations-Plugin Tracker Koordinaten Events (also wenn die Tracker Kommunikation erfolgreich gestartet wurde), kann mit der Registrierung und anschließender Navigation im Navigations-Plugin begonnen werden. Von dem Prozess unabhängig, bekommt das Stereo-Mikroskop Plugin den Video-Stream als GUI Element, sobald die Mikroskop-Kommunikation initialisiert wurde.

4 Diskussion

Im Vergleich zu umfassenden Toolkits im CAS Bereich bietet das Common Toolkit eine kurze Einarbeitungszeit. Obwohl beispielsweise MITK [2] auf CTK basiert, ist seine Verwendung wegen seines extensiven Funktionsumfangs mit hohem Aufwand verbunden. Im Gegensatz dazu erlaubt das CTK Plugin Framework eine effektive Entwicklung von Prototypen und Demos. Plugins können dabei dank einer service-basierten Kommunikation einfach ausgetauscht oder wiederverwendet werden. Etwaige benötigte Funktionen (z.B. Segmentierung/Registrierung, State-Maschinen) können durch Einbindung von bekannten Bibliotheken wie z.B. ITK [5] oder Qt [6] verwendet werden.

Die vorgestellte Demo Applikation repräsentiert ein einfaches Beispiel für Standardnavigation unter Verwendung des Common Toolkits. Durch dessen Implementierung wurde bewiesen, dass CTK im CAS Bereich verwendet werden kann. Dank des öffentlich verfügbaren Quellcodes bietet die Demo eine Hilfestellung bei der Entwicklung mit dem CTK Plugin Framework.

5 Zusammenfassung

In diesem Beitrag wurde Bezug auf diese Probleme genommen, und eine Einführung in das CTK Plugin-Framework gegeben. Dieses ermöglicht die Erstellung von modularen CAS Anwendungen und die Einbindung von externen Bibliotheken um den Funktionsumfang zu erweitern, ohne auf größere Frameworks zurückgreifen zu müssen. In Abschnitt 2 wurde die Erstellung von Plugins beschrieben, sowie deren Kommunikation mittels Services und Plugins. Um den Einstieg in das CTK Framework zu erleichtern, und dessen Einsatz im CAS Bereich zu überprüfen, wurde eine Demo-Applikation für Standardnavigation erstellt, die in Abschnitt 3 vorgestellt wird.

Der Quellcode der Demo-Applikation, Minimalbeispiele für Plugins, und ein Tutorial stehen auf www.voxelmaster.at frei zur Verfügung. Unter Verwendung dieser Hilfestellungen kann die Entwicklungen von CAS Anwendungen einfacher und effizienter gestaltet werden können.

6 Referenzen

[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)