# Ein Framework für die Implementierung von Anwendungssystemen zur Verarbeitung und Visualisierung von medizinischen Bildern

C. E. Cárdenas S. [1], V. Braun[1], P. Hassenpflug[1], M. Thorn[1], M. Hastenteufel[1], T. Kunert[1], M. Vetter[1], L. Fischer[2], W. Lámade[2], H.P. Meinzer[1]

[1] Abteilung für Medizinische und Biologische Informatik
Deutsches Krebsforschungszentrum, Heidelberg.
[2] Chirurgische Klinik Heidelberg
Email: C.Cardenas@DKFZ.de

**Zusammenfassung.** Zur Unterstützung von Medizinern bei der Interpretation medizinischen Bildmaterials ist es notwendig, Systemeinheiten zu entwickeln, welche spezialisierte Teilaufgaben in der klinischen Bildverarbeitung übernehmen. Durch den Einsatz eines *objektorientierten Frameworks*, das auf erprobten und bekannten Entwurfsmustern basiert, wird der Entwicklungsprozess von medizinischen Anwendungskomponenten erleichtert, beschleunigt und weniger fehleranfällig gestaltet. In dieser Arbeit wird ein Framework vorgestellt, das die komponentenbasierte Entwicklung von dedizierten Anwendungssystemen für unterschiedliche klinische Fragestellung erlaubt. Dadurch ist es möglich, schneller auf neue Anforderungen zu reagieren. Exemplarisch werden zwei aus dem Framework entwickelte Systeme vorgestellt.

## 1 Einleitung und Motivation

Für die Entwicklung von medizinischen Anwendungssystemen eignen sich objektorientierte Konzepte und Mechanismen, weil sie der menschlichen Denkweise nahe kommen. Der Einsatz objektorientierter Mechanismen reicht aber nicht aus, um wiederverwendbare und flexible Komponenten zu erstellen. Dies kann durch den Einsatz eines übergeordneten Frameworks erreicht werden, das auf erprobten Entwurfsmustern [1] basiert. Ein derartiges Framework reduziert das Risiko, das „Open-Close"-Prinzip (Open for Extension and Closed for Modification) in den bereits entwickelten Modulen zu verletzen. Ziel dieser Arbeit war die Konzipierung eines *objektorientierten Frameworks*, welches den Entwicklungsprozess von medizinischen Anwendungssystemen erleichtert, beschleunigt und weniger fehleranfällig gestaltet.

## 2 Struktur des Frameworks

Das Framework stellt den Anwendungsentwicklern vorgegebene Basisstrukturen und Komponenten zur Verfügung und kann die Verwendung und Ansteuerung einer oder

mehrerer Komponenten realisieren. Somit lässt sich das Framework in verschiedenen Bereichen für unterschiedliche Ziele in der Bildverarbeitung und Visualisierung einsetzen. Dabei bietet das Framework die Möglichkeit, das entwickelte System als *Stand-alone Applikation* oder auch als *PlugIn* eines übergeordneten Systems zu realisieren. Aufgabe der Entwickler ist entweder die Verwendung der von dem Framework zur Verfügung gestellten Komponenten oder die Implementierung der Komponenten, die das Framework an die Anforderung des zu entwickelnden Anwendungssystems anpassen. Der Anwendungsentwickler soll sich auf die spezifischen Details seiner Komponenten konzentrieren können. Die Kommunikation der Komponenten miteinander, die automatische Generierung von graphischen Komponenten, die Verwaltung des Informationsflusses, die Anordnung und Aussehen der Komponenten und die Kommunikation des Anwendungssystems mit anderen Systemen sind einige der Aufgaben, die das Framework übernimmt. Die Integration der neuen Komponenten an das Framework kann der Anwendungsentwickler mit einer einfachen Vererbung realisiert. Darunter verbergen sich die Entwurfsmuster *Factory Method* und *Builder*. Die vorgefertigen und die neuen Komponenten werden von dem Framework in Form von *Bibliotheken* geladen. Diese *Black Box* Einbindung erlaubt die Unabhängigkeit der Komponenten und ermöglicht außerdem die Verwendung der Komponenten getrennt voneinander, so dass beispielsweise die Komponenten allein getestet werden können.

Die Struktur des implementierten Frameworks basiert auf drei unterschiedlichen generischen Modulen. Diese Module sind eine mehrstufige Umsetzung des *Model View Controller*- Modells, die durch die Zusammensetzung verschiedener Klassen im Sinne der objektorientierten Technologie in der Programmiersprache C++ implementiert wurde.

### 2.1 Das Modul *Model*

Dieses Modul verwaltet die Informationen, welche in den Modulen und Anwendungssystemen des Frameworks verwendet werden. Der Anwendungsentwickler kann mit einfachen Aufrufen, die von ihm erzeugten Daten speichern und wieder lesen. Diese Daten werden anderen Anwendungssystemen des Frameworks zur Verfügung gestellt. Der Datenfluss funktioniert auch hier transparent. Wenn das Anwendungssystem als Stand Alone Applikation arbeitet, dann werden die Daten mit Hilfe dieses Moduls im File System gespeichert. Wenn das System auf der Basis eines übergeordneten Systems arbeitet, stellt dieses Modul die Verbindung zu diesem System her. Der Anwendungsentwickler kann bei der Einbindung einer neuen Komponente das Modul *Model* erweitern, so dass er bestimmen kann, welche Daten in dieser Komponente gespeichert werden können. Außerdem kann festgelegt werden, ob die Daten nur temporär im Hauptspeicher gehalten oder gespeichert werden sollen. Der Anwendungsentwickler kann auch beim Einbinden einer Komponente direkte Verbindungen zu anderen schon vorhandenen Komponenten festlegen und kann außerdem, falls erwünscht, den Datenfluss zu bestimmten Komponenten sperren. Standardmäßig wird eine Kommunikation zwischen allen Komponenten hergestellt.

### 2.2 Das Modul *View*

Dieses Modul enthält eine Klassenstruktur, welche das Aussehen und die graphische Anordnung der Benutzungsschnittstellen des Frameworks bestimmt. Anhand dieser Klassen werden die graphischen Komponenten generiert. Diese lassen sich einfach durch andere ersetzen oder erweitern. Man kann hierdurch auch unterschiedliche Plattformen unterstützen. Für die erste Implementierung dieses Moduls wurde *OSF/Motif* als GUI Bibliothek verwendet. Die Austauschbarkeit der graphischen Komponenten, die auf dem Modul *View* basieren, ermöglicht die Portabilität des Frameworks auf andere GUI Systeme. Als Erweiterung wird demnächst eine Implementierung des Moduls *View* mittels der GUI Bibliothek *Qt* realisieren. Damit wird die Verwendung des Frameworks und der daraus entwickelten Anwendungen unter verschiedenen Plattformen gewährleistet.

### 2.3 Das Modul *Control*

Dieses Modul definiert den Kontrollfluss der verschiedenen Komponenten des Frameworks. Es unterteilt sich in drei Untermodule. Das Untermodul *Base-Controller* steuert die Kommunikation der Komponenten untereinander und mit dem Framework. Die Einführung einer neuen Komponente geschieht durch Erweiterung dieses Untermoduls. Dabei wird das Framework nicht verändert, sondern soll durch Anwendung des *Substitutionsprinzips* die Schnittstellen der neuen Komponente implementieren, die wiederum von dem Framework aufgerufen werden. Nach dieser Integration hat die neue Komponente Zugriff auf alle Module des Frameworks. Das Untermodul *Image-Manager* steuert die Verwaltung der Bilddaten (Original Daten, anatomische Landmarks und die Parameter für die Visualisierung). Diese Daten werden an alle Komponenten weiter gereicht. Der Datenfluss ist dabei transparent, d.h. der Anwendungsentwickler kann die Informationen verwenden, ohne zu Wissen, in welcher Komponente sie erzeugt worden sind. Das Untermodul *Algorithm Manager* steuert die Ausführung vorgegebener und neuer Bildverarbeitungsfunktionen. Dieses Untermodul bietet über dreißig kanten- und regionenbasierte Bildverarbeitungsfunktionen. Dieses Untermodul wurde nach den Entwurfmustern *Bridge* und *Facade* Implementiert. Damit ist seine Erweiterung mit neuen Bildverarbeitungsalgorithmen problemlos möglich. Außerdem ist eine automatische Generierung der graphischen Komponenten in dem Framework sowohl für die Steuerung der neu eingefügten Teilsysteme als auch für die Parameter der Bildverarbeitungsfunktionen integriert.

## 3 Einsatz des Frameworks

Die erste Applikation, die auf dem Framework basiert, ist ein Segmentierungstool, das eine semiautomatische Segmentierung ermöglicht. Das Tool unterteilt sich in verschiedene Bereiche, die jeweils für klar definierte Aufgaben zuständig sind.
Der *Systemsteuerungsbereich* enthält Funktionen, die den Zustand des ganzen Systems ändern, wie zum Beispiel das Laden eines neuen Datensatzes. Der *Navigationsbereich* bietet Kontrollelemente für die Selektion einer bestimmten Schicht oder die

Navigation durch die Bilddaten. Die zu bearbeitende Schicht wird im *Arbeitsbereich* angezeigt. Der *Werkzeugbereich* stellt semiautomatische Bildverarbeitungsalgorithmen wie Region Growing, Threshold, Konturverfolgung, interaktiv bedienbare Aktive Konturen, sowie manuelle Werkzeuge zum freien Einzeichen oder Entfernen zur Verfügung. Die Kontrollelemente für das Setzen einer oder mehreren Parameter werden in dem *Parameterbereich* angezeigt. Mit Hilfe dieser Werkzeuge kann der Anwender die Bilddaten schichtweise bearbeiten. Die Segmentierungsergebnisse werden von dem Untermodul *Landmarks Controller* verwaltet und können von dem Endanwender in dem *Landmarkselektionsbereich* selektiert werden. Die Grenzen der Ergebnisse werden als *Overlays* farblich angezeigt. Die nachträgliche Erweiterungsfähigkeit mit neuen Bildverarbeitungsfunktionen wird von dem Untermodul *Algorithm Manager* gewährleistet. Mit Hilfe dieses Untermoduls kann das Framework für die neuen Bildverarbeitungsfunktionen die graphischen Komponenten, sowohl deren statischen, als auch deren dynamischen Teile, generieren [2], mit deren Hilfe der Endanwender die Funktionen selektieren und deren Parameterwerte ändern kann. Das Framework stellt verschiedene Modi zur Verfügung, mit deren Hilfe der Anwendungsentwickler bestimmen kann, welches Interaktionsmuster bei der Verwendung einer bestimmten Funktion angewendet werden muss. Damit ist es möglich festzulegen, wie die Parameterwerte, welche der Algorithmus für sein Funktionalität braucht, gewonnen werden. Diese können z.B. aus dem Eingabebild durch einen Mausklick, durch die Kombination verschiedener Bilder oder durch die vorher erzeugten Segmentierungsergebnisse gewonnen werden. Zur Zeit wird dieses System für die Segmentierung von Bilddaten in der Operationsplanung für die Herzchirurgie [3] und onkologische Leberchirurgie eingesetzt. Außerdem werden zur Zeit verschiedenen texturbasierten Bildverarbeitungsfunktionen getestet und unabhängig von dem Typ und Dimension der Bilddaten eingesetzt [4].

Die zweite Applikation ist ein radiologisches System für die computergestützte Operationsplanung in der onkologischen Leberchirurgie [5]. Dieses System besteht aus mehreren Teilsystemen, die folgenden Aufgaben bewältigen: Segmentierung der anatomischen Landmarks, Segmentierung der Gefäßbäume, Trennung von Gefäßbäumen, Identifizierung der Segmente der Leber sowie anschließende 3D Visualisierung der Leber, der Segmente und der unterschiedlichen Gefäße.

Das Teilsystem für die Segmentierung der Gefäßbaume ist ähnlich wie das Segmentierungstool aufgebaut. Es enthält in dem *Werkzeugbereich* nur eine Funktion, die als Grundlage des Region-Growing-Verfahrens hat. Als Startpunkt für diesen Prozess wird der Pfortaderstamm ausgewählt. Der Algorithmus durchwandert daraufhin den gesamten Datensatz und sucht nach zusammenhängenden Gefäßstrukturen. Die Parameterwerte werden mit einem Mausklick und der Einstellung des Level/Windows vom Anwender gesetzt. Diese Parameterwerte werden in dem *Parameterbereich* eingetragen, in dem der Anwender die Parameter kontrollieren und gegebenfalls ändern kann. Im *Arbeitsbereich* wird sowohl das Originalbild angezeigt, wie in dem Segmentierungstool, als auch der erzeugte Gefäßbaum.

Die dritte Komponente des Systems bietet im *Werkzeugbereich* eine Funktion, die durch Anklicken fälschlicherweise mitsegmentierter Lebervenen den gesamten Pfad bis zum Pfortaderstamm markiert. In einer solchen Darstellung kann die Stelle, an der der Segmentierungsalgorithmus in die Lebervenen übergelaufen ist, leicht erkannt und

entsprechend markiert werden. Dies wird für alle fehlsegmentierten Strukturen durchgeführt, bis am Ende nur noch das Pfortadersystem dargestellt wird.
In der letzten Komponente werden die errechneten Ergebnisse mit Hilfe des Heidelberger Raytracer visualisiert und im *Arbeitsbereich* angezeigt. Der Anwender kann das Volumen zoomen und rotieren und kann bestimmen, welche Daten angezeigt werden sollen. In einer Ansichtsauswahl kann zwischen verschiedenen Rekonstruktionen gewählt werden. Hierzu gehören einfache Visualisierungen, die die Lage des Tumors in der Leber und relativ zu den Gefäßbäumen zeigen. Genauso kann eine Darstellung derjenigen Gefäße, die innerhalb eines gewählten Sicherheitsabstandes liegen, ausgewählt werden. Das System wird bereits von klinischen Partnern eingesetzt und die Ergebnisse werden im Operationssaal verwendet [6].

## 4 Schlussfolgerung

Der Einsatz des Frameworks hat sich als erfolgreich erwiesen, denn die Zeit für das Design und die Implementierung von Anwendungssysteme ist durch seine Verwendung reduziert worden. Das Framework gibt die Infrastruktur vor, auf der die Anwendungssysteme arbeiten können. Die Interoperabilität und Integration von Teilsystemen werden unterstützt. Zusätzlich wird der Wartungsoverhead verringert, weil beispielsweise die Beseitigung von Fehlern oder das Einfügen neuer Elemente automatisch in allen Komponenten wirksam wird.

## 5 Literatur


1. Gamma E, Helm R, Johnson R, Vlissides J: Design Patterns. Addison Wesley- Verlag, New York, 2. Auflage 1994.
2. Cardenas CE, Demiris AM, Makabe M, Meinzer HP: Ein System zur automatischen Generierung graphischer Benutzungsschnittstellen für Bildverarbeitungsalgorithmen. Bildverarbeitung für die Medizin. Grundlagen, Modelle, Methoden, Anwendungen. Springer-Verlag, Berlin, 1. Auflage 1998.
3. Heimann T, Schroeder A, Giess C, et al.: Integrierte visuelle und haptische Darstellung von Blutflüssen an Herzklappen. Bildverarbeitung für die Medizin. Algorithmen, Systeme, Anwendungen. Springer Verlag, Berlin, 1. Auflage 2001.
4. Hastenteufel M, Cardenas C, Hassenpflug P, et al.: Evaluierung von interaktiven, texturanalytischen Segmentierungsverfahren. Bildverarbeitung für die Medizin. Algorithmen, Systeme, Anwendungen. Springer Verlag, Berlin, 1. Auflage 2001.
5. Cardenas C, Glombitza G, Thorn M, Heid H, Vetter M, Hassenpflug P, Fischer L, Lamade W, Richter G, Meinzer HP: A radiological software module for computer aided operation planning in oncological liver surgery. Computer Assisted Radiology and Surgery. Proceedings of the 14th International Congress and Exhibition. Amsterdam: Elsevier Verlag, 1. Auflage 2000.
6. Pressemiteilung des Deutsches Krebsforschungszentrums Nr. 32: Computergestützte Operationsplanung bei Lebertumoren- erstmalig in der Klinik eingesetzt. Heidelberg, 2000. (http://www.dkfz-heidelberg.de/presse/pminhalt.htm)