Ein System zur automatischen Generierung graphischer Benutzungsschnittstellen für Bildverarbeitungsalgorithmen

Carlos E. Cárdenas S., Athanasios M. Demiris,
Manuela H. Makabe, Hans-Peter Meinzer

Abt. für Medizinische und Biologische Informatik
Deutsches Krebsforschungszentrum
INF 280, 69120 Heidelberg
Email: C.Cardenas@dkfz-heidelberg.de

Zusammenfassung. Seit Anfang der achtziger Jahre ist eine große Anzahl von Tools für die Entwicklung graphischer Benutzungsschnittstellen (GUI) auf unterschiedlichen Abstraktionsebenen entstanden. Einige dieser Tools zwingen den Entwickler sich mit unnötig vielen Details auseinanderzusetzen. Andere Tools lassen dem Benutzer keine Flexibilität oder sind so komplex, daß dadurch die Arbeit nicht erleichtert wird. Mit Hilfe von Generierungssystemen erreicht man eine Senkung der Entwicklungszeiten und Kosten. Wir stellen eine generische Architektur vor, die eine weitestgehende Automatisierung des Generierungsprozeßes graphischer Benutzungsschnittstellen für Bildverarbeitungsalgorithmen erlaubt. Dabei werden keine zusätzlichen Anforderungen an den Benutzer gestellt.

Schlüsselwörter: Generierungssystem, Benutzungsschnittstellen, Bildverarbeitungsalgorithmen, Frameworks

1 Einleitung

Zur Unterstützung der Mediziner bei der Interpretation tomographischer Schichtbildserien, wie MR oder CT, ist es notwendig, Bildverarbeitungsalgorithmen zu erstellen. Das effektive Entwickeln eines neuen Bildverarbeitungsalgorithmus, das Testen und dessen Einsatz wird durch die entsprechende Benutzungsschnittstelle unterstützt. Außerdem erlaubt eine Benutzungsschnittstelle die interaktive Bedienung und Parametrisierung der Algorithmen.

Die Entwicklung einer graphischen Benutzungsschnittstelle (GUI), welche einfach zu bedienen ist, und außerdem die Anforderungen des Benutzers erfüllt, ist sehr oft ein langwieriger und komplexer Prozeß. Im Durchschnitt besteht der Code einer Applikation zu 48% aus einem "User Interface"-spezifischen Teil, und über 50% der Zeit bei der Implementierung eines Systems verbringt man mit dem User Interface [1].

2 State-of-the-Art

In den letzten Jahren wurden verschiedene Ansätze eingesetzt, um den Aufwand für die Erzeugung von GUIs zu verringern. Dabei werden Ansätze, wie das "Model-Based Interface” Paradigma verwendet [2], welches die Zentralisierung aller nötigen Informationen ermöglicht, um mit Hilfe bestimmter Regeln den statischen Teil des GUIs ohne konventionelle Programmierung und ohne Interface Entwicklungsumgebung realisieren zu können. Die für die vollständige Generierung des GUIs fehlenden Informationen werden durch punktuelles Eingreifen des Benutzers gewonnen.

Bei der Verwendung dieses Ansatzes kann der dynamische Teil des GUIs nicht automatisch generiert werden, er muß statt dessen getrennt definiert werden. Der Interaktionsfluß (dynamischer Teil) zwischen den graphischen Elementen wird normalerweise mit graphischen Editoren oder auch mit der Einführung von "pre-” und "postconditions” durchgeführt. Die Trennung des Generierungsprozeßes in zwei Teile ist ein großer Nachteil, weil die Konsistenz der Elemente dadurch gefährdet ist. Je höher die Anzahl der graphischen Komponenten ist, desto größer ist die Gefahr der Inkompatibilität zwischen den Teilen. Es entstehen außerdem sehr komplexe Systeme, weil die Anforderungen an den Anwender ständig anwachsen. Die Modelle, welche solchen Systemen zugrunde liegen, sind eher an die Generierungskomponente des jeweiligen Systems angepaßt, und weniger an den Anwender. Innerhalb der Tätigkeiten, welche der Anwender erledigen muß, zählt zwar nicht die Syntax von graphischen Bibliotheken, jedoch muß er sich mit deklarativen Modellen, deren Aufgabe die Erzeugung der verschiedenen Modell-Mechanismen ist, auseinandersetzen.

3 Methode und Vorgehensweise

In der hier vorgestellten Arbeit wurde ein System entworfen und implementiert, welches die Erzeugung graphischer Benutzungsschnittstellen für Bildverarbeitungsalgorithmen so weit wie möglich automatisiert. Alle nötigen Informationen für die Generierung des GUIs werden anhand der Datenstrukturen gewonnen, welche der Bildverarbeiter in seinem Arbeitsfeld verwendet. Neben anderen Eigenschaften sollte das System anpassungsfähig, benutzerfreundlich und erweiterbar sein.

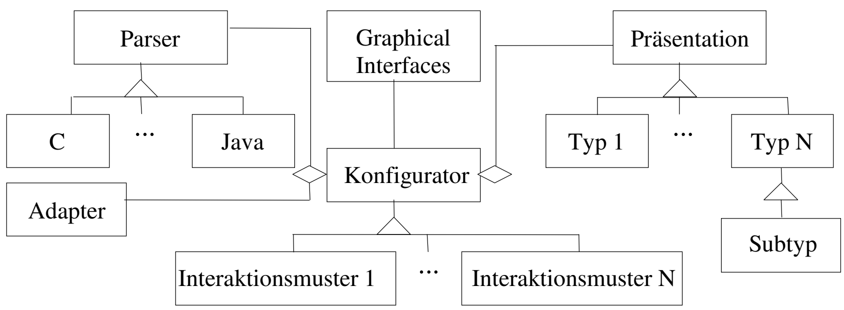

Abb. 1: Klassenstrukturdiagramm des Generierungssystems [3].

Mit dem in dieser Arbeit verwendeten Ansatz ist für die Generierung des GUIs kein Dialogmodell erforderlich, d.h. der Bildverarbeiter braucht, im Gegensatz zu existierenden Ansätzen, weder eine zusätzliche Beschreibung zu erzeugen, noch deklarative Modelle für die Generierung zu erstellen. Das System erzeugt nämlich anhand einer programmatischen Schnittstelle des jeweiligen Algorithmus (Signatur) die entsprechenden Präsentationselemente. Bei der Generierung des GUIs wird sowohl der statische, als auch der dynamische Teil erzeugt. Diese Zusammensetzung der Generierung der statischen und der dynamischen Teile des GUIs wurde durch den Entwurf und die Implementierung der graphischen Komponenten als selbständige Elemente mit lokaler Verantwortung erreicht. Die erstellte graphische Benutzungsschnittstelle bietet verschiedene Interaktionsmuster an. Es ist somit möglich festzulegen, wie die Informationen, welche der Algorithmus für seine Funktionalität braucht, gewonnen werden. Diese können z.B. aus dem Eingabebild durch einen Mausklick gewonnen werden. Das System erlaubt auch die Kombination von mehreren Bildern als Eingabe. Die Ergebnisse, die durch die Ausführung der Algorithmen erzielt werden, können gespeichert und durch andere Algorithmen weiterverarbeitet werden. Die einzelnen Komponenten des Frameworks wurden mit objektorientierten Konzepten nach dem Model View Controller-Paradigma (MVC) implementiert. Der Einsatz objektorientierter Konzepte erfolgte so, daß sowohl ein eigenständiges System zum Testen von Bildverarbeitungsalgorithmen, als auch eine Komponente eines übergeordneten Systems generiert werden können.

4 Das Generierungssystem

Das gesamte System (Generierungssystem) ist eine mehrstufige Umsetzung des MVC-Modells, d.h. anhand verschiedener Modelle (Algorithmen), werden verschiedene Views (statisches GUI), mit ihren ensprechenden Controllern (GUI Interaktion) generiert. Das System ist als ein sog. objektorientiertes Framework implementiert. Es besteht aus vier Modulen. Diese Module wurden durch die Zusammensetzung verschiedener Klassen, im Sinne der objektorientierten Technologie, implementiert (siehe Abb. 1).

1. Das *Konfigurator-Modul* ist das Modul, welches sowohl die Kommunikation der Komponenten des Generierungssystems miteinander erlaubt, als auch ihre Ansteuerung durchführt. In diesem Modul werden die sog. Interaktionsmuster als Untermodule implementiert. Das Generierungssystem bietet insgesamt drei Interaktionsmuster. Das erste Interaktionsmuster wird für die Bildverarbeitungsalgorithmen verwendet, bei denen die Eingabeparameter die Werte sind, welche der Benutzer in die generierten graphischen Komponenten eintragen wird. Das zweite Interaktionsmuster bietet für die Einstellung der Parameterwerte zusätzlich die Möglichkeit, Werte aus dem Eingabebild zu lesen, z.B. das Lesen der Position eines Mausklicks. Das dritte Interaktionsmuster bietet für die Parametrisierung der Algorithmen zusätzlich zu den Parameterwerten die Möglichkeit, mehrere Eingabebilder zu verwenden.

2. Das *Parser-Modul* hat die Aufgabe, die Signatur der Bildverarbeitungsalgorithmen zu identifizieren. Dafür wird einfach der Dateiname des entsprechenden Algorithmus eingegeben.
3. Das *Präsentationsmodul* enthält eine Reihe von Klassen und Subklassen, welche als Präsentationselemente für bestimmte Datentypen verwendet werden. Anhand dieser Klassen werden die graphischen Komponenten generiert. Diese lassen sich einfach durch andere ersetzen oder erweitern. Hierdurch können auch unterschiedliche Plattformen unterstützt werden.
4. Das *Graphical Interfaces-Modul* ist die Schnittstelle zwischen dem Framework und dem Benutzer oder einem anderen System (siehe Abb. 3).

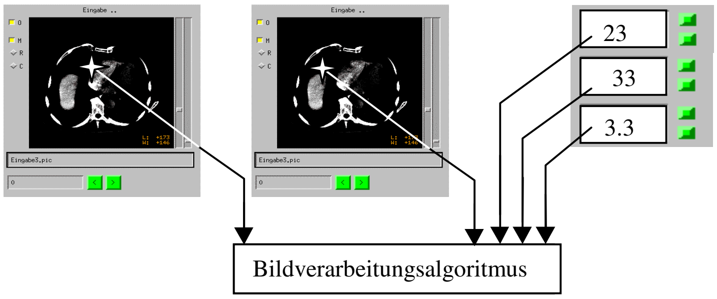

Abb. 2: Im dritten Interaktionsmuster lassen sich Informationen aus mehreren Bildern mit den Einträgen in die entsprechenden Elemente kombinieren und als Parameterliste für die Ansteuerung der Algorithmen einsetzen.

Die Struktur der einzelnen Komponenten wurde nach verschiedenen "Design Patterns" implementiert [4]. Durch den Einsatz der objektorientierten Konzepte bei der Implementierung des Systems, war es möglich ein System zu erstellen, welches ohne Änderung des bestehenden Codes einfach zu erweitern ist. Der Austausch der Komponenten ist auch problemlos realisierbar. Man ist in der Lage, das System als eine "stand-alone" Applikation zu verwenden (Black Box Framework), um kleine prototypische GUIs zu erzeugen oder auch als Teilkomponente von größeren Systemen z.B für die schnellere Erzeugung von "Plug-ins". Es ist außerdem möglich, das System mit einem Algorithmen-Repository zu verbinden. Dadurch kann es zum Bestandteil einer übergeordneten Architektur zur Vereinfachung der Entwicklung klinischer Bildverarbeitungssysteme werden.

Ein wichtiger Bestandteil des Frameworks ist die Dokumentation. Diese wurde in mit einem “Top Down”-Verfahren entlang verschiedener Abtraktionsebenen des Systems durchgeführt, um die Komplexität der Beschreibung des Systems zu bewältigen und die Wiederverwendung auf verschiedenen Ebenen zu ermöglichen. Die Darstellung wurde mit Hilfe der verschiedenen Diagramme der “Unified Modelling Language” Notation realisiert [3].

5 Diskusion und Ausblick

Das Generierungssystem beschränkt sich zur Zeit auf Bildverarbeitungsalgorithmen, welche in den Programmiersprachen C oder C++ geschrieben sind. Durch die Erweiterung des Parser-Moduls, anhand vorhandener Komponenten, ist man in der Lage die Funktionalität des Systems für andere Programmiersprachen zu implementieren.

Das Generierungssystem ist auf UNIX-Plattformen verfügbar. Seine Verfügbarkeit auf anderen Plattformen ist durch die Erweiterung der graphischen Elemente möglich. Dabei wird der GUI-Elemente-Pool des Systems mit neuen Komponenten unter Motif, TeleUSE, MFC, etc. erweitert.

Die generierten GUIs verfügen über 2D-Interaktions-Elemente, um die Parametereinstellungen des Bildverarbeitungsalgorithmus zu ermöglichen. Eine denkbare Erweiterung ist die Erzeugung einer neuen Klasse, welche die 3D-Interaktion ermöglicht.

Das Framework wird zur Zeit in der Abteilung für Medizinische und Biologische Informatik am Deutschen Krebsforschungszentrum auf verschiedenen Bildverarbeitungsfunktionen getestet und unabhängig von der Anzahl der Parameter, dem Typ der Bilddaten, der Dimension und der Art der Algorithmen (kantenorientiert, regionenbasiert) eingesetzt. Als nächster Schritt wird die Verbindung des Systems mit einer modularen Architektur zur Vereinfachung der Entwicklung klinischer Bildverarbeitungssysteme durchgeführt [5]. Für die Realisierung dieser Verbindung ist weder eine Erweiterung noch eine Änderung des Generierungssystems erforderlich.

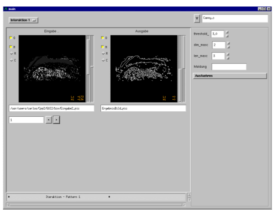

Abb. 3: Die generierte graphische Benutzugsschnittstelle für unsere Implementierung des Canny-Filters anhand der C-Quelldatei.

6 Literatur

1. Myers BA, Roson MB: Survey on the User Interface Programming. ACM CHI’92 Conference Proceedings, New York, 1992.
2. Puerta AR: A Model-Based Interface Development Environment. IEEE Software, 14(4):40-47, 1997.
3. Fowler M, Scott K: UML Distilled - Applying the Standard Object ModelingLanguage. Addison-Wesley, Reading MA, 1997.
4. Gamma E, Helm R, Johnson R, Vlissides J: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, New York, 1994.
5. Demiris AM, Cardenas CES, Makabe MH, Meinzer HP: An Architecture for Implementing Customizable Medical Image Processing Systems. Akzeptiert als Vortrag in MedInfo '98, Korea.