Generierung von platformunabhängigen Benutzerschnittstellen für einen CORBA basierten Bildverarbeitungsserver

Christoph Giess, Carlos Cárdenas, Hans-Peter Meinzer

Abteilung für Medizinische und Biologische Informatik
Deutsches Krebsforschungszentrum
Im Neuenheimer Feld 280, 69120 Heidelberg
E-mail: Ch.Giess@DKFZ-Heidelberg.de

Zusammenfassung. Das Paper beschreibt ein System zur Generierung von graphischen Benutzerschnittstellen für ein verteiltes Bildverarbeitungssystem. Die in einem verteilten System ständig auftretenden Veränderungen werden durch eine dynamisch anpaßbare Software vor dem Anwender verborgen. Durch die Verwendung von CORBA als Middleware und Java zur Implementierung lässt sich das Generierungssystem mittels Repository und Dynamic Invocation Interface realisieren.

Schlüsselwörter: Benutzungsschnittstellen, Generierungssystem, Client/Server, CORBA

1 Einleitung

In den letzten Jahren setzen sich immer leistungsfähigere Bildverarbeitungssysteme in verschiedenen medizinischen Bereichen durch. Für eine einfache Bedienbarkeit dieser Systeme sind graphische Benutzungsschnittstellen (Graphical User Interfaces – GUI) unabdingbar. Neben dem Einsatz in klinischer Routine bieten GUIs auch eine wichtige Unterstützung während des Entwicklungsprozesses neuer Algorithmen und Systeme.

Die Entwicklung einer 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. Bei der Implementierung eines Systems entfallen über 50% der Zeit auf das User Interface [1].

Durch eine kontinuierliche Erweiterung von Bildverarbeitungssystemen mit immer neuen Algorithmen entstanden in den letzten Jahren verschiedene Ansätze, um die aufwendige Erstellung von graphischen Benutzerschnittstellen durch Generierungssysteme zu automatisieren und somit Entwicklungszeit einzusparen.

Die immer weiter wachsende Bedeutung des Internets stellt zudem neue Anforderungen an Bildverarbeitungssysteme. Dazu gehören der Zugriff auf im Netzwerk verteilte Resourcen sowie eine Web-basierte, plattformunabhängige Benutzungsschnittstelle.

2 Stand der Forschung

Zur Generierung eines User Interfaces existieren Ansätze wie das „Model Based Interface“ Paradigma [2]. Diese erzeugen den statischen Teil von GUIs basierend auf deklarativen Modellen. Eine Programmierung der Schnittstelle bzw. Interface-Entwicklungsumgebungen wird nicht benötigt. Der dynamische Teil hingegen kann damit nicht automatisch generiert werden. Der Informationsfluß zwischen den graphischen Elementen muß in einem weiteren Schritt spezifiziert werden.

Von [3] wurde ein Generierungssystem speziell für Bildverarbeitungsalgorithmen entwickelt. Dieses generiert sowohl den statischen als auch den dynamischen Teil einer GUI. Die dafür benötigten Informationen bezieht das System aus dem Sourcecode des jeweiligen Algorithmus. Die Verbindung zwischen Bilddaten und Funktionen wird dabei vom System zur Laufzeit mittels verschiedener Interaktionsmuster hergestellt. Die Nachteile dieser Implementierung lagen in der begrenzten Portierbarkeit (Motif-basiert), ihrer Beschränkung auf die Verarbeitung von C- und C++-Funktionen sowie deren lokaler Ausführbarkeit.

Im Bereich der verteilten Bildverarbeitung entwickelten Ansätze betreffen die Integration existierender Systeme, Parallelisierung, Loadbalancing und Resourcensharing.

Das von [4] vorgestellte DISCUS Projekt präsentiert die Integration verschiedener Komponenten eines Bildverarbeitungssystems. In Khoros existiert die Funktionalität, verschiedene Teile eines Bildverarbeitungssystems auf verschiedenen Rechnern ablaufen zu lassen. Mit „IDL On the Net“ besteht die Möglichkeit, mittels eines Webbrowsers auf existierende IDL-Programme (Interactive Data Language) zugreifen zu können.

Im Rahmen des EVIMED Projektes [5] entstand ein Bildverarbeitungsserver mit einem intelligenten Bilddatenmanagement, welcher es mehreren Clients ermöglicht, leistungsfähige Hardware in Netzen gemeinsam zu nutzen, ohne daß sich die Netzwerkinfrastruktur allzu limitierend auswirkt. Die Anwendung dieses Konzepts in zwei Applikationsprototypen für Operationsplanungssysteme demonstrierte dessen Anwendbarkeit.

3 Material und Methoden

Die im folgenden beschriebene Arbeit stellt eine Weiterentwicklung des Generierungssystems von [3] dar. Sie basiert auf einem Bildverarbeitunsserver, welcher unter Verwendung von CORBA die Ideen von [5] erweitert und vollständig neu implementiert. Die Integration beider Komponenten erfolgt unter Einbeziehung der in [6] vorgestellten Konzepte. Die Erfahrungen aus den beiden vorherigen Arbeiten wurde in dieses Projekt eingebracht, bestehende Nachteile wurden beseitigt und neue Konzepte ausgearbeitet.

Ein zentrales Entwicklungsziel bestand in der Wiederverwendung von getesteten und standardisierten Lösungen.

Das Gesamtsystem besteht aus einem oder mehreren Bildverarbeitungsservern, einem oder mehreren *Interface Repositories* sowie der jeweiligen Applikation. Ein *Naming* oder *Trading Service* dient zur Lokalisation der verteilten Komponenten.

3.1 Bildverarbeitungsserver

Der entwickelete Bildverarbeitungsserver erfüllt drei Funktionen. Er stellt ein Factory-Objekt für Bilddaten dar, ist Container für Bilder und Algorithmen sowie zuständig für den Bilddatenaustausch mit anderen Servern.

Bilder, Algorithmen sowie der Bildverarbeitungsserver selbst sind als CORBA-Objekte implementiert, d.h. ihre Schnittstellen sind in sprachunabängiger Weise mittels CORBA-IDL (Interface Definition Language) beschrieben. Mit diesem Ansatz lassen sich beliebige Bildverarbeitungsbibliotheken kapseln und mittels einheitlicher Schnittstelle verwendbar machen.

Algorithmen können *stateless* und *stateful* implementiert werden. Aus Performance-Gründen entsprechen die meisten Algorithmen genau einer Methode eines Objektes, welches nur eine gruppierende Funktion besitzt (stateless). Algorithmen, welche eine größere Anzahl an Parmetern benötigen wurden als Sessions implementiert. Mittels get- und set-Methoden können die Parameter abgefragt und gesetzt werden. Der aktuelle Status wird in dem jeweiligen Algorithmen-Objekt gespeichert.

Die Implementierung erfolgte in C++ unter Verwendung von objektorientierten Konzepten, um die Wartbarkeit und Erweiterbarkeit zu erhöhen. Zudem wurde konsequent die von CORBA gebotene Funktionalität in der Implementierung verwendet. Die Anzahl der geschriebenen Lines of Code reduzierte sich damit um 45% im Vergleich zu der vorherigen Implementierung, wobei sich gleichzeitig die Anzahl der integrierten Algorithmen verzehnfachte.

3.2 Interface Repository

Das *Interface Repository* (IR) ist ein Bestandteil des ORB und dient als Archiv für IDL-Definitionen. Die in einer IDL-Definition enthaltenen Informationen werden im IR durch CORBA-Objekte repräsentiert, so daß Clients und Server auch von entfernten Rechnern darauf zugreifen können [7]. Im IR verwaltet jeder Bildverarbeitunsserver die Schnittstellen der in ihm implementierten Objekte. Somit sind die Bildverarbeitungsserver selbstbeschreibend.

Mit Hilfe des IR ist es möglich, zur Laufzeit für beliebige Objekt-Referenzen Informationen über den Aufbau der zugehörigen Interfaces einzuholen. Dies ist unabhängig davon, ob diese bei der Entwicklung des Programms bereits bekannt waren.

3.3 Generierungssystem

Das System wurde nach dem *Model View Controller*-Paradigma entwickelt. Dabei wird auf der Basis von CORBA (Controller), die Verbindung zwischen den graphischen Komponenten (View) und den Algorithmen (Model) hergestellt.

Die gewünschte Plattformunabhängigkeit sowie ein mögliches Ausführen des Programmes in einem Webbrowser legten die Verwendung von Java zur Implementierung des Generierungssystems nahe. Dabei ermöglicht der Einsatz von CORBA als standardisierte Middleware einen transparenten Zugriff auf Bilddaten und Algorithmen sowohl lokal als auch über das Internet.

Das System besteht aus drei Modulen. Im Gegensatz zu dem vorherigen System konnte die Implementierung eines eigenen Parsers entfallen. Dessen Funktionalität wird durch das Interface Repository und dessen Werkzeuge abgedeckt.

- Das *Konfigurator Modul* ist die zentrale Komponente des Systems. Sie verbindet die Komponenten des Generierungssystems und steuert diese.
  Mittels *Dynamic Invocation Interface* (DII) erfolgt der Aufruf der jeweiligen Algorithmen.
- Das *Präsentationsmodul* enthält eine Reihen von Klassen und Subklassen, welche als Präsentationselemente für bestimmte Datentypen verwendet werden. Anhand dieser Klassen werden die graphischen Komponenten generiert. Diese lassen sich erweitern bzw. durch andere ersetzen. Hierdurch können unterschiedliche *Look-and-Feels* unterstützt werden.
- Das *Graphical Interfaces Modul* ist die Schnittstelle des Systems zum Benutzer bzw. zu einem anderen System.

4 Diskussion und Ausblick

Die Kombination von Java und CORBA für Client/Server-Systeme eröffnet über die Verwendung der dynamischen Konzepte hinausgehende Möglichkeiten. Das Interface Repository zusammen mit dem Dynamic Invocation Interface beinhaltet bereits einen Großteil der Funktionalität, welcher zur automatischen Generierung von *User Interfaces* notwendig ist.

Eine Limitation des vorgestellten Ansatzes besteht in der beschränkten Anzahl von Metainformationen, welche zu einem Algorithmus existieren. So sind unter anderem seine Anwendbarkeit auf bestimmte Bilddaten sowie Wertebereiche einzelner Parameter nicht beschrieben.

Mit dem *Property Service* und dem *Meta Object Facility* [8] stehen aber bereits CORBA-Komponenten zur Verfügung, mittels derer die fehlenden Funktionen implementiert und in das bestehende Framework integriert werden können.

Literatur

[1] Myers BA, Roson MB: *Survey on the User Interface Programming.* ACM CHI'92 Conference Proeedings, New York, 1992.

[2] Puerta AR: *A Model-Based Interface Development Environment.* IEEE Software, 14(4):40–47, 1997.

[3] Cárdenas CE, Demiris AM, Makabe MH, Meinzer HP: *Ein System zur automatischen Generierung graphischer Benutzungsschnittstellen für Bildverarbeitungsalgorithmen.* In: Lehman T, Metzler V, Spitzer K, Tolxdorff T (Hrsg.). Bildverarbeitung für die Medizin 1998, Springer, Heidelberg, 189–193, 1998.

[4] Zahavi R: *Common Facilities RFI Response – Imagery Facility.* TC Document 94-06-05, Object Management Group, Framingham, MA, 1994.

[5] Mayer A, Meinzer HP: *High performance medical image processing in client/server-environments.* Computer Methods and Programs in Biomedicine, 58(1999), 207–217.

[6] Demiris AM, Cárdenas CE, Meinzer HP: *Customizable Medical Image Processing Systems.* In: Jähne B, Haußecker H, Geißler P (Hrsg.). Handbook of Computer Vision and Applications – Vol. 3 Systems and Applications, Academic Press, San Diego, 53–76, 1999.

[7] Redlich JP: *CORBA 2.0 – Praktische Einführung für C++ und Java.* Addison-Wesley, 1996.

[8] Object Management Group: *Meta ObjectFacility (MOF) – Specification.* TC Document ad/97-08-14, Object Management Group, Framingham, MA, 1997.