Dynamisch generierte multimodale Anwendungsschnittstellen

Kurzfassung

zur Erlangung des akademischen Grades Doktoringenieur (Dr.-Ing.)/
Doktor rerum naturalium (Dr. rer. nat.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von

Dipl. Inf. Stefan Kost
geboren am 06. April 1974 in Leipzig

Betreuende Hochschullehrer:
Prof. Dr. Wolfgang Wünschman, TU Dresden
Prof. Dr. Klaus Bastian, HTWK Leipzig

Dresden im August 2005

Inhaltsverzeichnis

1 Ausgangspunkt

Software-Schnittstellen (Software-Interfaces) bestimmen die Nutzbarkeit und Barrierefreiheit von Applikationen. Deren Güte läßt sich über die Eigenschaften ihrer Entwicklungs-Werkzeuge (Toolkits) definieren und somit auch bewerten.

Traditionelle graphische Schnittstellen-Toolkits verfügen nicht über Fähigkeiten, die Umsetzung barrierefreier Schnittstellen zu ermöglichen!

Warum ist dies so? Das von Windows-Umgebungen eingeführten Paradigma schlägt sich auch in der Spezialisierung herkömmlicher Toolkits nieder. Diese Spezialisierung überträgt sich nun wiederum automatisch auf die damit entwickelten Anwendungen. Wenngleich diese Ausrichtung in der Vergangenheit zeitgemäß war, behindert sie nunmehr an einer universellen Problemsicht

Die soziale Durchdringung unserer Gesellschaft mit Computertechnologien stellt weitaus größere Anforderungen an Schnittstellen-Toolkits.

Die ausschließlich graphisch orientierten Toolkits sind also nicht mehr ausreichend. Nachfolgende Beispiele sollen dies illustrieren:

1.) ein Computeradministrator:

> Administratoren sind wichtige Arbeitskräfte im Unternehmen. Ein Ausfall der Rechentechnik kann die betrieblichen Abläufe empfindlich stören. Da Administratoren derzeit oftmals nur vor Ort arbeiten können, kann eine Reparatur erheblich verzögert werden. Somit ist es wichtig eine Möglichkeit zu finden, dass Administratoren jederzeit und überall aktiv werden können. Wenn die Benutzungsschnittstellen der von ihnen eingesetzten Anwendungen auf beliebigen vernetzten Endgeräten abrufbar wären, hätte man dieses Ziel erreicht.

2.) ein Autofahrer mit Navigationssystem:

> Navigationsgeräte sind eine hilfreiche Unterstützung bei der Autofahrt. Sie bieten aber auch andere nützliche Zusatzfunktionen, wie Informationen über Hotels, Verkehr und Sehenswürdigkeiten. Diese Geräte müssen multimodal ausgeführt werden. Während der Autofahrt nutzen sie Sprachkommunikation für die Interaktion mit dem Autofahrer. Wenn das Fahrzeug jedoch parkt, kann der Nutzer das Gerät auch über den Touchscreen benutzen und sich Informationen graphisch anzeigen lassen.

Aus diesen beiden Beispielen lassen sich bereits zwei Anforderungen ableiten - Benutzungsschnittstellen sollten

- Endgeräteunabhängig und
- Medienneutral

sein. Des weiteren ist es wünschenswert, wenn sich Schnittstellen an Vorlieben des Nutzers anpassen lassen. Bei genauerer Betrachtung der verschiedenen Adaptionsziele gelangt man zu mehreren Klassen: Adaptivität gegen Technik, Kultur, Individuen und der Nutzungsumgebung.

Für die enorm gestiegenen Anforderungen an die Schnittstellentechnologie gibt es zwei begründente Faktoren: Diversifikation von Mensch und Technik. Zu Beginn des Computerzeitalters gab es das Szenario eines Experten und seiner Großrechenanlage. Heutzutage haben wir stattdessen Nutzer verschiedener Kulturen, mit unterschiedlichen Fähigkeiten, unterschiedlicher Bildung und Ausbildung auf der einen Seite und eine Vielzahl unterschiedlicher technischer Geräte (z.B. Computer, PDAs, Smartphones und Digitalkameras) auf der anderen Seite. [BAMyersEA2000]

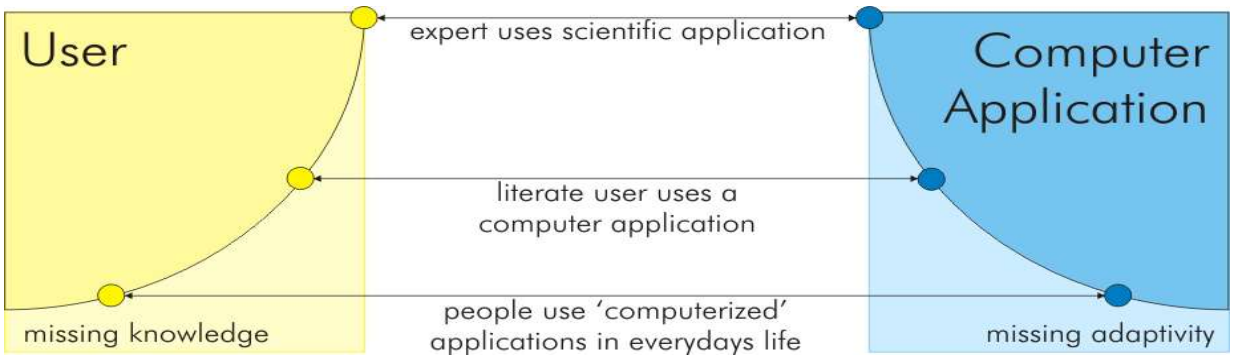

Abbildung 1: Lücke zwischen Nutzern und Anwendung

Durch diese Diversifikationseffekte vergrößert sich die Verständnisslücke zwischen Mensch und Computer. Diese Lücke muß nun aktiv von beiden Seiten überwunden werden. Der Nutzer leistet seinen Beitrag, wenn er sich mit der Technik beschäftigt und deren Benutzung erlernt. Er passt sich damit an Eigenheiten der Technik an.

Der umgekehrte Weg ist auch von der Technik zu gehen. Die Technik sollte sich an Eigenheiten von Nutzern anpassen. Nun gibt es bereits eine Reihe von Ansätzen die sich Teilproblemen der Anpassung von Benutzungsschnittstellen widmen. In dieser Arbeit soll jedoch ein radikaler Ansatz vorgestellt werden.

These 1: **Komplexität vs. Universalität**
Die Begrenzung der Komplexität der Darstellung, ermöglicht eine viel grössere Universalität.

These 2: **Multimodal**
Es gibt keinen Grund für die Beschränkung adaptiver Toolkits auf vorwiegend grafische Darstellung.

These 3: **Trennung von Schnittstellenaspekten**
Es ist durchaus möglich und auch wünschenswert eine klare Trennung von Präsentation und Funktionalität in der Schnittstellendefinition vorzunehmen.

These 4: **Generische Anpassung**
Eine Schnittstelle kann generiert werden, ohne das die Anwendung Adaptionsprofile für die verschiedenen Zielsysteme bereitstellen muss.

These 5: **Adaptionssteuerung**
Die Anpassung von Schnittstellen muss nicht nur unter Berücksichtigung des Nutzers, sondern auch der Nutzungsbedingung (Umgebung) erfolgen.

These 6: **Wiederverwendung von Technologien**
Eine solche Lösung kann auf standardisierten und etablierten Technologien basieren; sie kann sogar als Bindeglied zwischen all diesen spezifischen Lösungen dienen.

These 7: **Generelle Anwendbarkeit**
Nicht nur der Mensch, sondern auch eine Anwendung kann andere Anwendungen nutzen und somit von adaptiven Schnittstellen profitieren.

These 8: **Dynamische Anpassung**
Schnittstellen sollten nicht (statisch) 'erzeugt' werden. Die Anpassung von Schnittstellen ist als kontinuierlicher Prozess zu betrachten.

These 9: **Soziale Auswirkungen**
Adaptive Technologien werden für jedermann benötigt und nicht nur für Minderheiten (wie ältere oder behinderte Menschen).

Das primäre Ziel ist demnach, die Benutzungsschnittstellen von der Anwendung zu entkoppeln. Diese Aufgabe schafft die Grundlage für eine ferneres Ziel: *Interfaces for all* - jeder Nutzer bekommt eine Schnittstelle, welche den aktuellen Anforderungen gerecht wird. Die Arbeitsgruppe um Jean Vanderdonckt hat für diese Eigenschaft eines Schnittstellentoolkits den Begriff *Plasticity* wie folgt geprägt:

A UI is plastic if it is able to adapt to context changes while preserving usability. We define a context of use as the set of values of variables that characterize the computational device(s) used for interacting with the system as well as the physical and social environment where the interaction takes place.

[JVanderrdonckt2002]

Neben den Vorteilen für den Endanwender wird auch dem Softwareentwickler geholfen. Er kann sich auf funktionale Aspekte konzentrieren. Usability-Experten können ihr Spezialwissen bei der Entwicklung der Schnittstellenpräsentationskomponente einbringen.

2 Interaktionsmodelle

Interaktionsmodelle beschreiben das Zusammenspiel von Akteuren und Komponenten eines interaktiven Systemes.

2.1 Etablierte Interaktionsmodelle

Die in der Arbeit näher vorgestellten Modelle wie Seeheim, Arch, PAC, MVC, MMI usw. haben eine Gemeinsamkeit - sie betrachten die Interaktion zwischen Mensch und Computer. Hierbei wird jedoch der Einfluß der Umgebung auf die Interaktion nicht mit einbezogen. Die Arbeitsumgebung wirkt hierbei als Filter auf die potentiellen Interaktionsfähigkeiten der Kommunikationspartner. Ein adaptives Schnittstellen-Toolkit sollte sich also nicht nur an die relativ statischen Anforderungen eines Nutzers anpassen, sondern auch an die dynamisch wechselnden Anforderungen der Umgebung. (These 5) [Arch][MMI]

Einher mit dieser Eigenschaft kommt ein weiterer Aspekt. Eine Anpassung an einen Nutzer ist von relativ statischer Natur. Ein Nutzer ändert sich nicht ständig und mit hoher Häufigkeit. Bei der Nutzungsumgebung kann dies jedoch durchaus der Fall sein. Faktoren wie Helligkeit und Lautstärke können sich situationsbedingt und nicht vorhersehbar ändern. Somit sollte ein adaptives Schnittstellen-Toolkit die Erzeugung der Schnittstelle nicht als einmaligen Prozess auffassen. Die vorgestellten Modelle sehen eine solche dynamische Arbeitsweise jedoch nicht vor. Ausgaben werden lediglich als Reaktion auf Nutzereingaben oder Rechenschritte durchgeführt. (These 8)

Das Problem der Adaption von Benutzungsschnittstellen lässt sich außerdem auch auf die Kommunikation von Rechnersystemen ausdehnen. Ein Rechner bietet einen Service an und ein anderer Rechner möchte ihn nutzen. Dies erfordert ebenfalls die Benutzung einer gemeinsamen Sprache, in diesem Fall z.B. Webservices. (These 7)

Verallgemeinernd kann man also sagen, dass ein Interaktionspartner (ein Rechner) einen Service (Anwendung) anbietet und ein anderer Interaktionspartner (Mensch, Rechner) diesen Service nutzen möchte. Damit es zu einer effektiven Interaktion kommt, passt der Anbieter seine Schnittstelle an die Anforderungen des Nutzers und der Nutzungsumgebung an.

Die in der Arbeit untersuchten Modelle, mit Ausnahme des MMI Modelles, lassen eine Empfehlung für eine konkrete Umsetzung vermissen! Dies untergräbt die Praktikabilität. Insbesondere das Seeheim Modell ist sehr einfach und allgemein gehalten. Ein Interaktionsmodell sollte nicht nur Komponenten nennen, sondern diese auch versuchen zu definieren.[RBastidePAPalanque1999]

Zusammenfassend kann gesagt werden das folgende der Eingangs aufgestellten Thesen durch die besprochenen Modelle noch nicht ausreichend unterlegt werden können.

- These 5 (Adaptionssteuerung): alle Modelle außer MMI
- These 7 (Generelle Anwendbarkeit): alle Modelle
- These 8 (Dynamische Anpassung): Seeheim, ARCH

2.2 Adaptionsinfrastruktur

In der Arbeit führt die Auseinandersetzung mit Begriffen und existierenden Interaktionsmodellen zu einem neuen Modell - genannt *Adaptionsinfrastruktur*. Interaktion ist ein Prozess der stattfindet, wie z.B. der tägliche Straßenverkehr. Analog zu diesem Beispiel wird auch für effektive Interaktion eine leistungsfähige Infrastruktur benötigt. Bei der zwischenmenschlichen Kommunikation ist hierfür eine gute sprachliche und interkulturelle Ausbildung von Vorteil. Dadurch ist eine Anpassung an den Kommunikationspartner möglich. Das Adaptionsinfrastruktur-Modell beschreibt

die Komponenten und deren Wirkung für die Kommunikation mit einem Rechnersystem. Eine Software, die das Zusammenspiel solcher Komponenten steuert, bildet die Grundlage adaptiver Benutzungsschnittstellen.

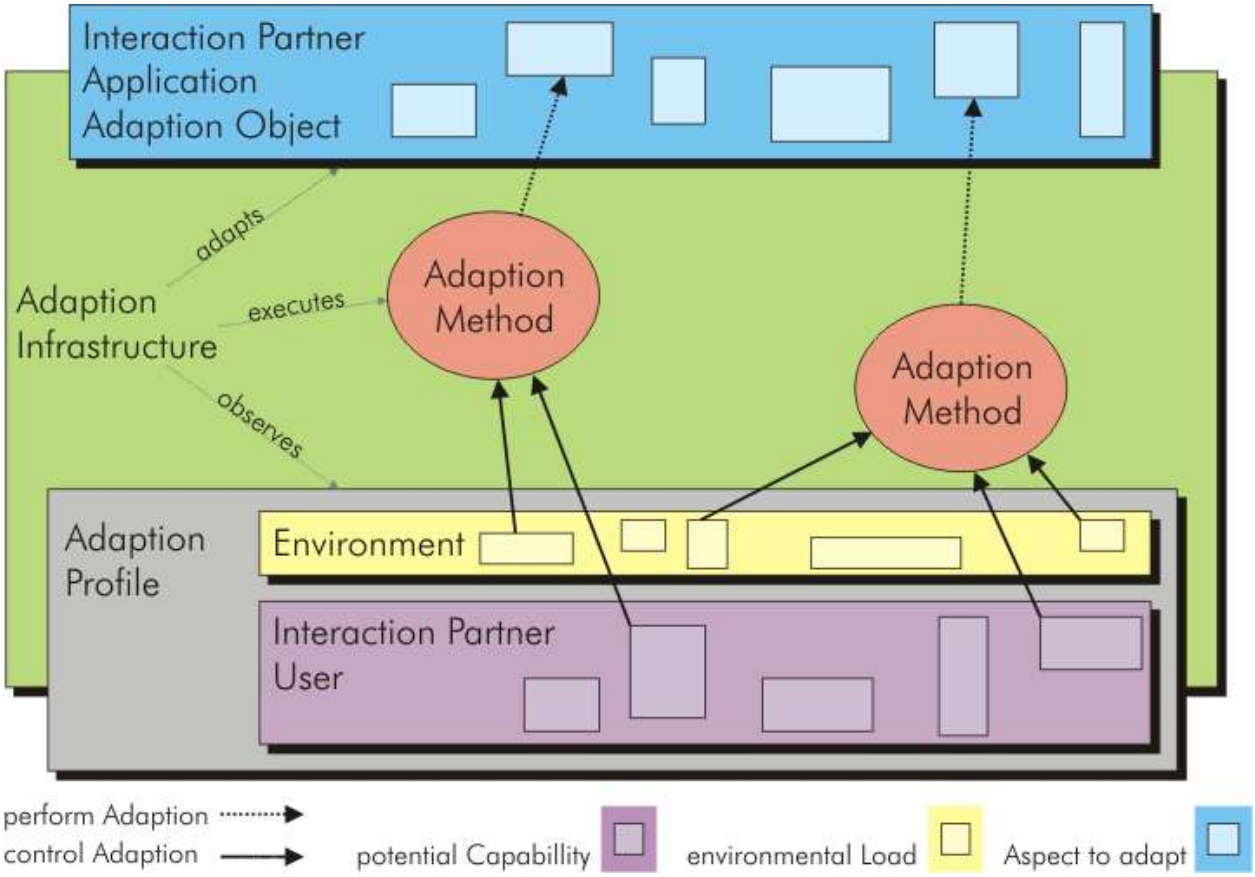

Abbildung 2: Adaptionsinfrastruktur Modell

Zielstellung bei der Modellierung war eine klare Definition der Aufgaben einzelner Komponenten.

- Sowohl der Nutzer als auch der Anbieter (Interaction Partner) haben bestimmte potentielle Möglichkeiten, mit dem Kommunikationspartner Dialog zu führen.
- Die Nutzungsumgebung wirkt filternd auf diese Fähigkeiten ein.
- Dabei gibt es Aspekte die miteinander in Beziehung stehen, z.B. das Sehvermögen des Nutzers und die Art der visuellen Präsentation der Schnittstelle seitens der Anwendung.
- Eine Adaptionsmethode nimmt eine Anpassungen hinsichtlich eines Aspektes der Schnittstelle vor (z.B. Steuerung der Farbdarstellung oder der Schriftgröße).
- Profile liefern Steuerdaten für die Adaption (z.B. das Sehvermögen des Nutzers).
- Sensoren aktualisieren Profile (z.B. die Umgebungshelligkeit).

Die Adaptionsinfrastruktur stellt den Unterbau bereit, der die gerade genannten Komponenten verwaltet, verschaltet und aktiviert. Während *Anwendung* und *Nutzer* auch in den zuvor untersuchten Modell vorkommen, werden die Adaptionsmethoden dort nicht explizit als Komponententyp modelliert.

Adaptionsmethoden kapseln Expertenwissen und stellen dieses nachnutzbar zur Verfügung.

Dadurch wird auch eine weitere Anforderung an die Adaptionsinfrastruktur aufgestellt. Damit Adaptionsmethoden auf Schnittstellenaspekte angewendet werden können, müssen diese Aspekte *adaptierbar* sein. Adaptierbarkeit ist somit Voraussetzung für Adaptivität.

Die Herausforderung bei der technischen Umsetzung einer solchen Infrastruktur besteht genau in diesen Punkten – der abstrakten Schnittstellenbeschreibung und der Kapselung des plattform-spezifischen Know-Hows in wiederverwendbaren Modulen.

3 Architekturmodelle

Softwarearchitekturmodelle beschreiben den Aufbau eines Softwaresystemes und das Zusammenspiel der Softwarekomponenten.

3.1 Existierende Architekturen

Viele der vorgestellten Schnittstellen-Toolkits beschränken sich auf graphische Benutzungsschnittstellen. Somit wird hauptsächlich das am Ende eingesetzte Toolkit abstrahiert und unter Umständen eine verbesserte Geräteunabhängigkeit erreicht (These 2). Bei Projekten wie AUIML, XUL und wxWidgets ist genau dies auch das Ziel. Im Gegensatz dazu streben Projekte wie UIML und XIML einen höheren Abstraktionsgrad an. UIML erreicht diesen durch den Einsatz von Adaptionsprofilen. Ein Entwickler muss für jede zu unterstützende Präsentation der Benutzungsschnittstelle detailierte Umsetzungsrichtlinien angeben (These 4). Damit steht das Adaptions-Know-How aber nicht nachnutzbar zur Verfügung. [Phanouriou2000]

Die im letzten Abschnitt genannten Projekte sind leider nicht einfach an neue Anforderungen anpassbar. Wenn einzelne Eigenschaften der Schnittstellen nicht adaptiv ausgeführt sind, lassen sich diese auch nicht adaptieren. Dazu wäre eine stärkere Modularisierung und vor allem eine konkrete Trennung der Behandlung der einzelnen Schnittstellenaspekte notwendig (These 3).

XML Technologien spielen bei nahezu allen untersuchten Projekten ein Rolle. Die dadurch gegebenen Möglichkeiten werden jedoch nur teilweise ausgeschöpft. Stattdessen werden Eigenentwicklungen eingesetzt. UIML setzt z.B. für der Dokumententransformationen auf Java-Logik statt XSLT (These 6).[W3C XML]

Dadurch, dass bei einigen Sprachen (z.B. XUL) medienspezifische Merkmale bereits in der Syntax der *abstrakten* Schnittstellenbeschreibungssprache vorkommen, ist die Nutzung alternativer Medien ausgeschlossen. Durch diese Spezialisierung auf ein Medium, ist es wesentlich leichter auch komplexe Schnittstellen abzubilden. Eine generische Lösung erfordert einen weit aus größeren Aufwand, um Schnittstellen medienübergreifend abzubilden (These 1) .[GBauer2004]

Wiederum genügen die untersuchten Architekturansätze nicht den Eingangs aufgestellten Thesen:

- These 1 (Komplexität vs. Universalität): alle Architekturen
- These 2 (Multimodal): XUL, UIML, AUIML
- These 3 (Trennung von Schnittstellenaspekten): UIML, XUL
- These 4 (Generische Anpassung): UIML
- These 6 (Wiederverwendung von Technologien): UIML

[Cover2004]

3.2 Beschreibung GITK

Das *Generalized Interface ToolKit* (GITK) ist ein im Rahmen dieser Arbeit entworfenes Architekturmodell für die eingangs beschriebene Adaptionsinfrastruktur. Zu Demonstrationszwecken wurde außerdem eine Referenzimplementation für das GITK erstellt. Diese wird als Open Source Projekt unter [Kost2005] entwickelt.

Kernziele des Systementwurfs und der Umsetzung sind unter anderem:

- Modulare Architektur analog zum Interaktionsmodell
- Spezifikation von Schnittstellen
    - Abstrakte Beschreibungssprache für Schnittstellen: GIML

    - Medienneutrales Namensschema für Interaktionsobjekte
- Verarbeitung der Dialogspezifikationen
    - Flexible Transformationsarchitektur
- Darstellung und Ausführung der Dialoge
    - Kapselung des plattformspezifischen Know-Hows

3.2.1 GITK Architekturmodell

Die Architektur basiert auf einem Mehrschichtenmodell. Ziel war ein Systementwurf mit orthogonalen Komponenten. Ein Mehrschichtenmodell ist gut geeignet um Komponenten voneinander zu entkoppeln.

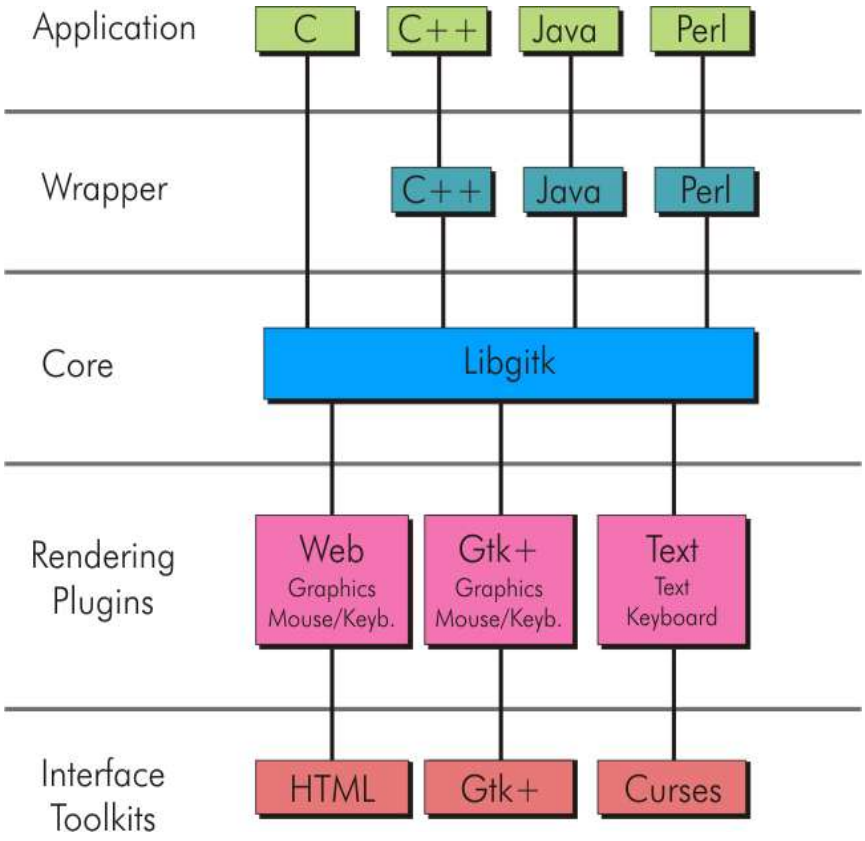

Abbildung 3: Architektur des GITK

Was ist nun die Bedeutung der einzelnen Schichten:

- Application: Benutzeranwendung
- Wrapper: Umsetzung verschiedener Programmiersprachen auf das API der Kernbibliothek
- Core: Kernbibliothek
- Rendering Plugins: austauschbare Komponenten für die Nutzung einer Schnittstelle auf einer bestimmten Plattform
- Toolkits: plattformspezifische Schnittstellentoolkits

Die Schichten *Wrapper* und *Rendering Plugins* entkoppeln den Kern von Programmiersprachen auf der einen Seite und der Schnittstellenplattform auf der anderen Seite.

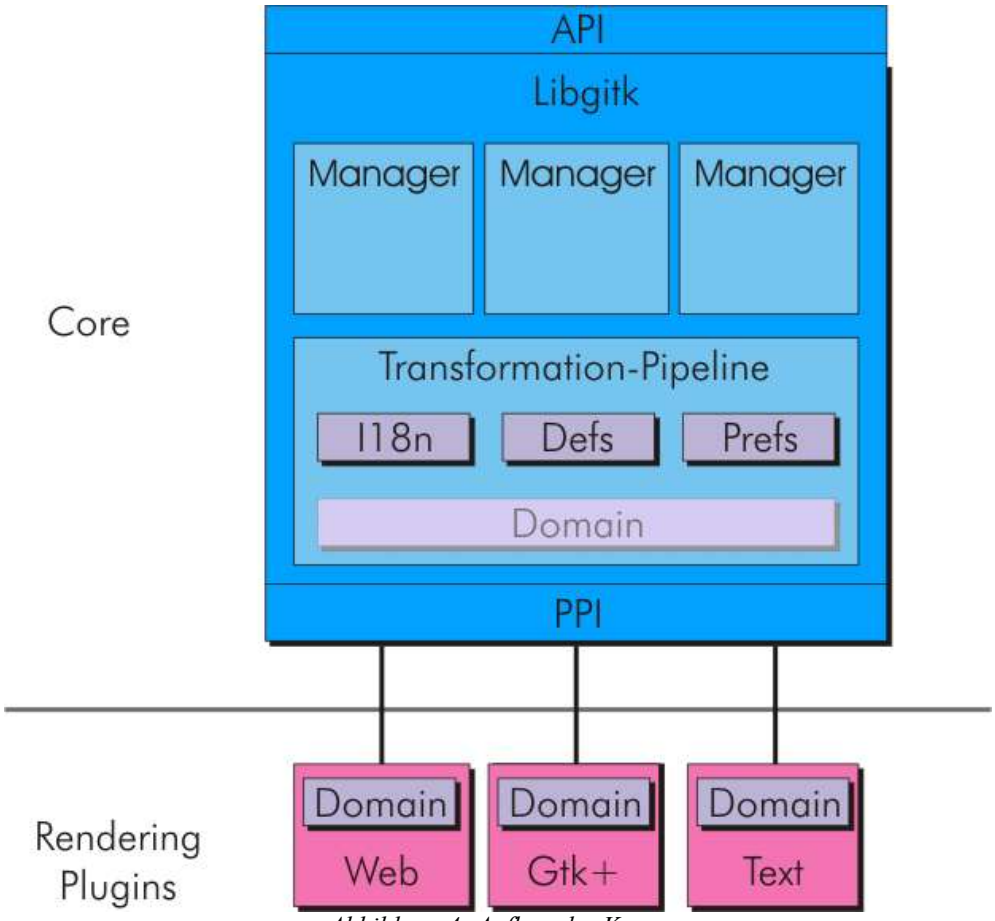

Abbildung 4: Aufbau des Kerns

Die Kernbibliothek wiederum beinhaltet mehrere Komponenten:

- Core API: Angebotene Funktionalität für Anwendungen
- Renderer Pluginverwaltung: Auswahl, Laden/Entladen und Ansteuerung der Darstellungskomponenten
- Transformationskontrolle: Aufbau der Transformationskette, Datensynchronisation und Transformationsausführung
- Profilverwaltung: Verwaltung von Eigenschaftssätzen, Aktualisierung durch Sensoren und Weitergabe an Adaptionsmethoden
- Kontextverwaltung: Verwaltung des Dialogstapels inklusive der Nutzungsmodi
- Ereignisbehandlung: Konsistente Ereignisbehandlung über Toolkits und Medien hinweg
- Fehlerbehandlung: Fehlerbehandlungsroutine in Dialogdokumenten und in der Bibliotheksnutzung
- Protokollierung und Beobachtung: Einblick in das laufende System für den Entwickler und Simulationsplattform
- Core PPI: Angebotene Funktionalität für Plugins

Es ist deutlich zu erkennen, dass diese Architektur die Anwendung von den Rendering Plugins und somit auch von den Schnittstellentoolkits entkoppelt. Die Anwendung hat keine Informationen darüber, welches Rendering Plugin momentan aktiv ist und kann sich somit nicht medienspezifisch verhalten.

3.2.2 GIML: Generalized Interface Markup Language

Die Schnittstellenspezifikationen werden in einer eigenen XML basierenden Sprache verfasst. GIML ist eine im Rahmen der Arbeit neu entwickelte medienneutrale Schnittstellenbeschreibungssprache. Die Medienneutralität wird durch eine funktionsbezogene Namensgebung von Interaktionsobjekten erreicht. XML Namensräume werden für die Trennung der einzelnen Schnittstellen-Aspekte (Structure, Style, Behaviour and Content) genutzt.*[Phanouriou2000]*

Von denen in der Arbeit untersuchten XML Sprachen zur Schnittstellenbeschreibung ist nur der XForms Ansatz ebenfalls medienneutral.

Während des Programmlaufes entsteht aus dem GIML Dokument ein medienspezifisches Dokument. Im GITK System wird eine GIML ähnliche Sprache für die Zieldokumente verwendet. Prinzipiell sind aber auch andere XML Sprachen geeignet, solange sie XML Namensräume unterstützen.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE giml SYSTEM "http://gitk.sourceforge.net/giml.dtd">
3 <giml xmlns="http://gitk.sourceforge.net/"
4       xmlns:dc="http://purl.org/dc/elements/1.1/"
5       xmlns:i18n="http://apache.org/cocoon/i18n/2.0"
6 >
7   <dialog focus="main">
8     <meta>
9       <dc:title><i18n:text>dialogtitle</i18n:text></dc:title>
10    </meta>
11    <dialogwidgets>
12      <dialogwidget id="Okay"/>
13    </dialogwidgets>
14    <widgetgroup>
15      <widget id="widget1" type="label"/>
16    </widgetgroup>
17  </dialog>
18 </giml>
```

3.2.3 Transformationspipeline

Zur Laufzeit wird die in GIML formulierte Eingangsspezifikation in eine Spezifikation für das Zielsystem transformiert. Somit sind zwei Spezifikationen vorhanden, die während der Lebensdauer des Dialoges aktualisiert und synchron gehalten werden müssen:

- medienneutrales Quelldokument
- medienspezifisches Zieldokument

Die Anwendung kennt nur die medienneutrale Sicht auf die Schnittstelle. Der Renderer wiederum versteht nur seine medienspezifische Zieldarstellung.

Ein solches Zieldokument wird in mehreren Transformationsschritten (= Anwendung von Adaptionsmethoden) erzeugt. Implementiert wurden die Adaptionsmethoden mit XSLT. Durch die Synchronisation beinhaltet der DOM-Tree der XML Dokumente jederzeit den aktuellen Stand der Schnittstellenbenutzung. Änderungen von Seiten der Anwendung werden im Quelldokument eingetragen und zum Zieldokument durchgereicht. Entsprechend nimmt das Schnittstellenrenderingplugin Änderungen am Zieldokument vor, welche das Kernsystem nunmehr zurück auf das Quelldokument überträgt. Ereignisbehandlungsroutinen die mit dem geänderten Baumpfad verknüpft sind, werden vom geänderten Knoten her abgearbeitet.

Das XML Dokument ist der Dialog.

Dank dieser Vorgehensweise wird der Zustand der Bedienschnittstelle komplett in den XML Dokumenten gehalten. Wenn nun ein Wechsel der Modalität erforderlich ist, genügt es die

Renderingkomponente auszutauschen, die Transformationpipeline neu zu konfigurieren und das Quelldokument damit zu transformieren.

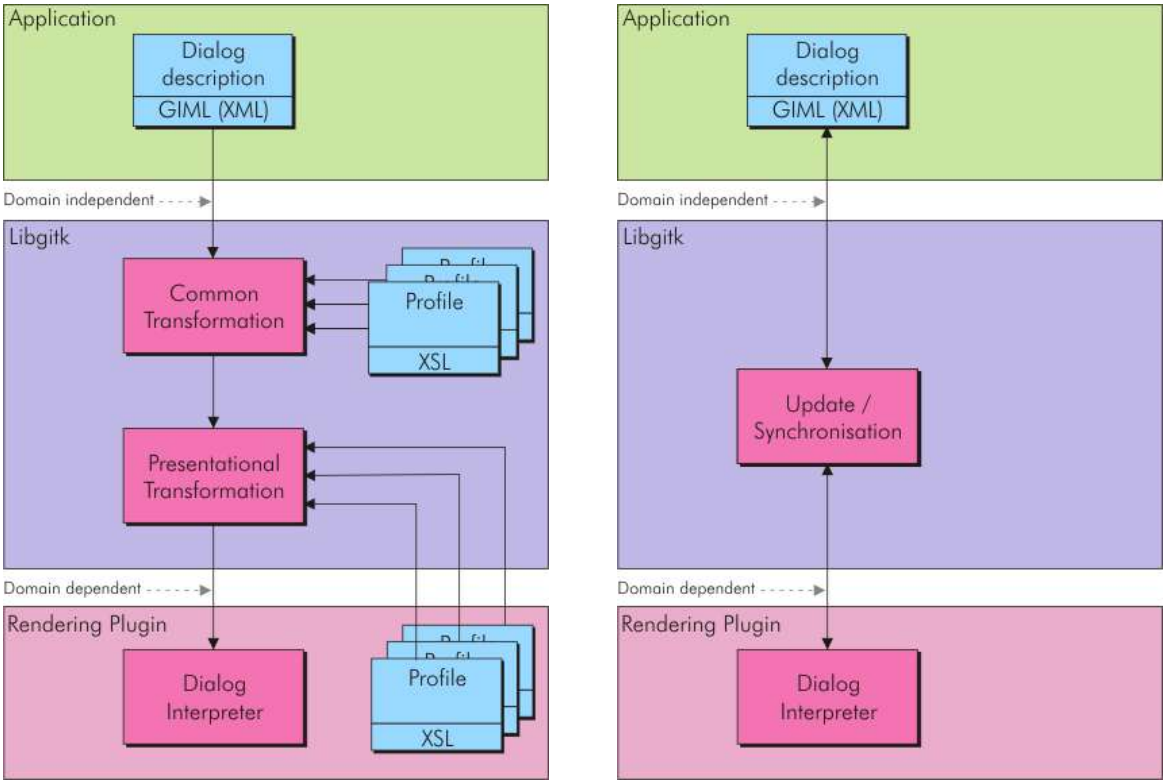

Abbildung 5: Transformationkette (links: Erzeugung, rechts: Änderungsrückführung)

3.2.4 Renderer

Nachdem durch die Transformationskette ein medienspezifisches Zieldokument erzeugt wurde, muss dieses in eine Benutzungsschnittstelle überführt werden. Dies ist die Hauptaufgabe der Renderer. Renderer sind nachladbare Komponenten (Plugins), die notwendiges Wissen für eine möglichst optimale Darstellung auf dem Zielsystem mitbringen. Jeder Renderer besteht aus folgenden Teilkomponenten:

- Domain-Adaptionsmethode: Übertragung des Schnittstellendokumentes in das Zielformat
- Event-Loop: Ereignisbehandlung für die Zielplattform
- Schnittstellenrenderer: Eigentliche Erzeugung und Aktualisierung der Schnittstellendarstellung, Ausführung des Dialoges

Eine Herausforderung des Architekturentwurfes war es, die Renderingkomponenten möglichst schlank zu gestalten. Adaptionsmethoden mussten identifiziert, generalisiert und in die Kernbibliothek ausgelagert werden. Dieser Prozess führt zu einem hohen Grad an Wiederverwendbarkeit.

Mit der Referenzimplementation des GITK werden verschiedene Renderer angeboten:

- Gtk+: herkömmliche graphische Benutzungsschnittstelle
- OpenGL: graphisch aufwendige 3D Darstellung
- Text: text-basierter Zugang mit optionaler Sprachausgabe
- Phone: telefontastaturgesteuerte Schnittstelle mit Sprachausgabe

Weitere Renderer (html, webservices) sind konzeptionell vorbereitet, aber derzeit noch nicht implementiert.

4 Zusammenfassung

Zu Beginn wurden einige Beispiele für die gestiegenen Anforderungen an Benutzungsschnittstellen vorgestellt. Wie sähe die Realisierung dieser Beispiele auf der Basis des GITK aus?

1.) ein Computeradministrator:

> Der Administrator könnte seine Werkzeuge auf jedem System benutzen, welches von GITK Renderern unterstützt wird. Die Hersteller der Administrationsanwendungen müssen dabei keine explizite Unterstützung bestimmter Renderer durchführen.

2.) ein Autofahrer mit Navigationssystem:

> Ein einfacher Sensor kann feststellen, ob das Fahrzeug fährt oder nicht. Im Falle, dass das Auto fährt, benutzt das Navigationssystem die gleichen Adaptionsmethoden, um die Benutzungsschnittstelle anzupassen, welche man auch für die Anpassung an blinde Computernutzer einsetzt. Das System würde z.B. die Menüs serialisieren und partitionieren, sowie Sprachkommunikation verwenden.

Gegenüber der aktuellen Situation gibt es somit eine gravierende Änderung:

Der Anwender wählt (direkt oder indirekt) die Modalität der Bedienschnittstelle.

Bisher traf der Hersteller diese Entscheidung für den Nutzer.

Am Anfang dieses Textes wurden 9 Thesen aufgestellt. Inwiefern können diese durch die Arbeit unterstützt werden?

1. **Komplexität vs. Universalität**:
   Durch den Verzicht medienabhängige Bedienelemente in der Spezifikation einzusetzen, gewinnt man einen Freiheitsgrad bei der Anpassung der Bedienschnittstellen.
2. **Multimodal**:
   Dank medienneutraler Schnittstellenspezifikation sind Abbildungen auf unterschiedlichste Zielsysteme möglich.
3. **Trennung von Schnittstellenaspekten**:
   In der Arbeit wurde gezeigt, dass dies für multimodale Schnittstellen sogar notwendig ist.
4. **Generische Anpassung**:
   Eine vollautomatisch erzeugte Bedienschnittstelle ist möglich. Eine noch bestehende Herausforderung ist, gestalterische Erfahrungen und Ergonomiewissen in Regeln zu fassen und in den Generierungsprozess einzubinden.
5. **Adaptionssteuerung**:
   Die in der Arbeit entwickelte Architektur kann auf verschiedenen Ereignisquellen reagieren.
6. **Wiederverwendung von Technologien**:
   Durch den Einsatz von etablierten Softwaretechnologien, ist eine leichtere Integration in den Entwicklungsprozess möglich. Verbesserungen an diesen Technologien können leichter nachgenutzt werden.
7. **Generelle Anwendbarkeit**:
   Beispielhaft wurde gezeigt, dass auch für Anwendungen ein Bedarf an adaptiven Schnittstellen besteht.
8. **Dynamische Anpassung**:
   Die dynamische Erzeugung einer Benutzungsschnittstelle ermöglicht einen hohen Adaptivitätsgrad, der bis zum Wechsel der Modalität zur Nutzungszeit reicht.

9. **Soziale Auswirkungen**:
   In der Arbeit wird eine 'Computerisierung' unserer Gesellschaft prognostiziert. Die sich daraus ergebenden neuen Nutzungsszenarien stellen deutlich erweitere Anforderungen an Schnittstellentoolkits.

Gerade These 1 bringt jedoch auch Einschränkungen mit sich. Ein Verzicht auf medien-spezifische Interaktionsobjekte stellt ein Problem dar, wenn medienspezifische Daten bearbeitet werden sollen. Dies basiert auf dem allgemeineren Problem der Repräsentation von Daten unter Benutzung alternativer Medien. Für die GITK Architektur sind derzeit Anwendungen am Besten umsetzbar, deren Datenbasis sich gut mit Text darstellen und bearbeiten läßt. Beispiele dafür sind Administrationsanwendungen, PIM (Personal Informationen Management)-Anwendungen und Suchabfragen. Nun ist dies aber auch genau die Gruppe von Anwendungen die einen hohen Nutzen aus einer breiten Verfügbarkeit zieht.

Im Rahmen der Arbeit wurden keine Studien zur Anwenderfreundlichkeit der generierten Dialoge durchgeführt. Dies erscheint erst sinnvoll, wenn die entsprechenden Renderer unter einer solchen Zielsetzung ausgebaut würden. Auch wenn die Beispiele der Referenzimplementation zeigen, dass sehr unterschiedliche Benutzungsschnittstellen aus einer Schnittstellenbeschreibung erzeugt werden können, ist recht offensichtlich, dass Verbesserungen in der Benutzbarkeit ein lohnendes Ziel für weitere Arbeiten wären. Um traditionelle Schnittstellen durch eine Technologie, wie sie in dieser Arbeit vorgestellt wurde, zu ersetzen, müssen die generierten Schnittstellen attraktiver für den Nutzer werden.

Eine Idee dies zu erreichen sind *Interface Patterns*. Typische Gruppierungen von Interaktionsobjekten werden unter einem Namen in einen Katalog aufgenommen. Dieser Musterkatalog wird für die unterschiedlichen Kommunikationsmedien aufbereitet. Die Renderer können dann sofern Umsetzungen vorhanden sind auf diese Muster zugreifen.

Für die zukünftige Entwicklung des GITK Projektes sind zwei weitere Punkte von Bedeutung:

- Renderer: Weitere Renderer können die Universalität des Ansatzes belegen. Vorhandene Renderer können ausgebaut werden, um komplexere Anwendungen umsetzen zu können.
- Umfangreichere Demos: Aktuelle Renderer binden nur eine Untermenge von Interaktionsobjekten ein. Somit sind derzeit auch bei den Demonstrationsanwendungen hinsichtlich der Funktionalität Grenzen gesetzt.

Literaturverzeichnis

BAMyersEA2000: B. A. Myers and S. E. Hudson and R. Pausch, Past, Present, and Future of User Interface Software Tools, 2000

JVanderrdonckt2002: Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Nathalie Souchon, Laurent Bouillon, Murielle Florins, Jean Vanderdonckt, Plasticity of User Interfaces: A Revised Reference Framework, 2002

Arch: verschiedene Authoren, A Metamodel for the Runtime Architecture of an Interactive System, 1992

MMI: James A. Larson and T.V. Raman and Dave Raggett, W3C Multimodal Interaction Framework, 2003, http://www.w3.org/TR/mmi-framework/

RBastidePAPalanque1999: Rémi Bastide and Philippe A. Palanque, A Visual and Formal Glue between Application and Interaction, 1999

W3C XML: W3 Consortium, W3C recommendations for XML technologies, 1999-2005, http://www.w3.org/

GBauer2004: Gerald Bauer, Open XUL Alliance - Creating A Rich Internet For Everyone, 2004, http://xul.sourceforge.net/

Cover2004: Robin Cover, XML Markup Languages for User Interface Definition, 2004, http://xml.coverpages.org/userInterfaceXML.html

Kost2005: Stefan Kost, GITK - Generalized Interface ToolKit, 2000-2005, http://gitk.sf.net

Phanouriou2000: Constantinos Phanouriou, UIML: A Device-Independent User Interface Markup Language, 2000