GITK - Generalized Interface ToolKit

Dynamically generated multi-modal application interfaces

Dipl. Inf. Stefan Kost
Betreuer der Arbeit: Prof. Dr. Wolfgang Wünschmann
TU DresdenFB Informatik, Institut für Angewandte Informatik
, Prof. Dr. Klaus Bastian

16.12.2003

0.2.0

Gliederung

- **Motivation - Warum GITK?**
- **Einordnung - Was ist GITK?**
- **Welche Probleme sind zu lösen?**
- **Wie funktioniert GITK?**
- **Was ist der Stand der Dinge?**

Motivation

Motivation

- <term>UIML</term>
  User Interface Markup Language
- <term>XIML</term>
  eXtensible Interface Language
- <term>XUL</term>
  XML User interface Language
- <term>AUIML</term>
  Abstract User Interface Markup Language
- wxWindows
- Java Swing
- ...

Motivation (Continued)

Warum noch eine Lösung?

 # existierende Ansätze zu spezifisch

Situation

extreme Diversifizierung

2 Bereiche:

- **Technifizierung aller Lebensbereiche**
  # Diversifizierung des Anwenderkreises
- **Computerifizierung von Geräten**
  # Diversifizierung von programmierbaren Geräten

 # Software muß sich anpassen

Adaption

1.) Adaption für unterschiedlichste Technologien

- **Interface-Hardware**
- **Dialogführung / Benutzungsmodi (aktiv, semiaktiv, passiv)**
- **Protokolle & Standards**

2.) Adaption an den Mensch

- **motorische und sensorische Leistungsfähigkeit (potentielle Fähigkeiten)**
- **Einschränkung durch aktuelle Umgebung (potentielle Barrieren)**

 # Beides muß als ganzheitliches Problem verstanden werden

ganzheitliche Adaption

Der Anwendung ist es "egal" warum sie sich an-passen muß

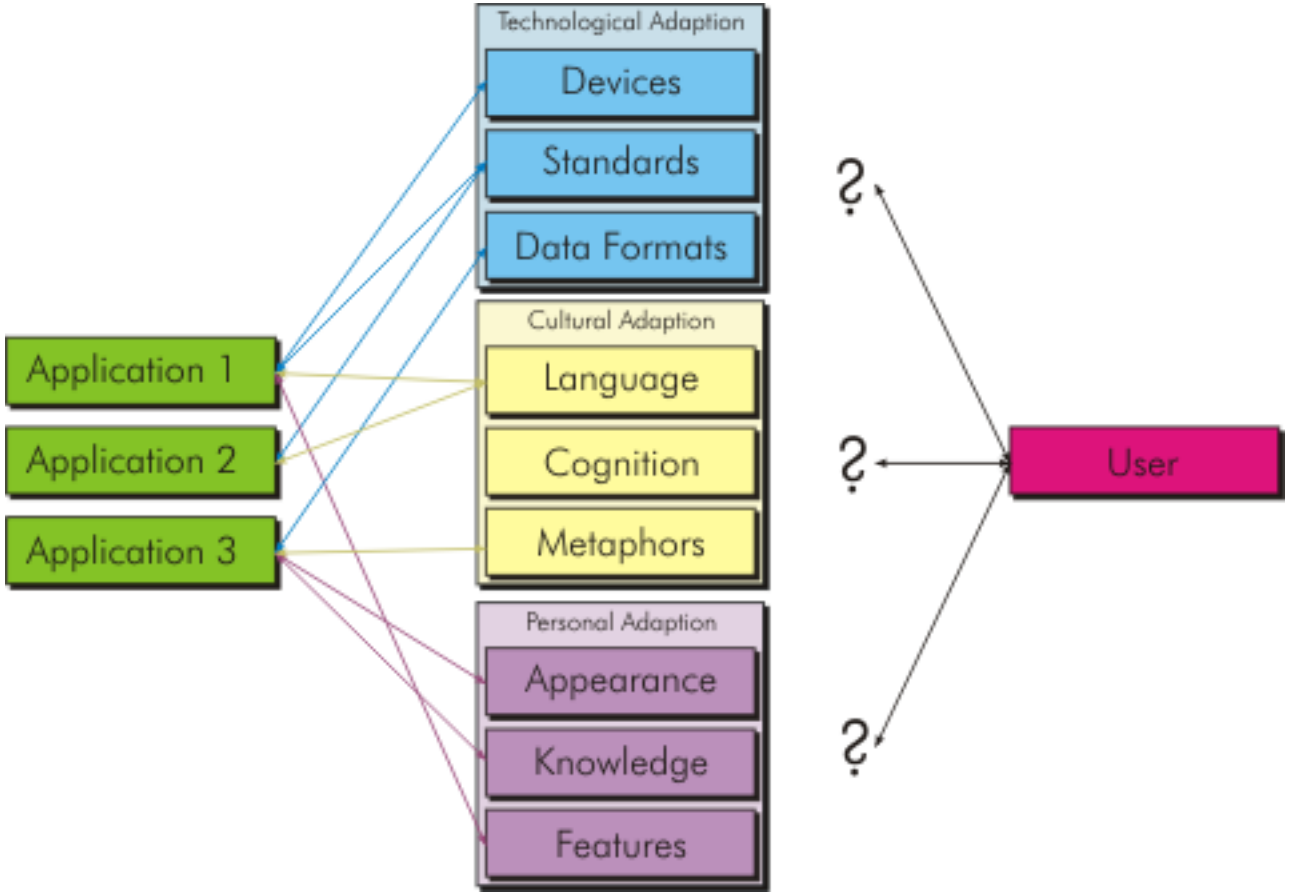
sensorische/motorische Blokade durch Aufgabe

=

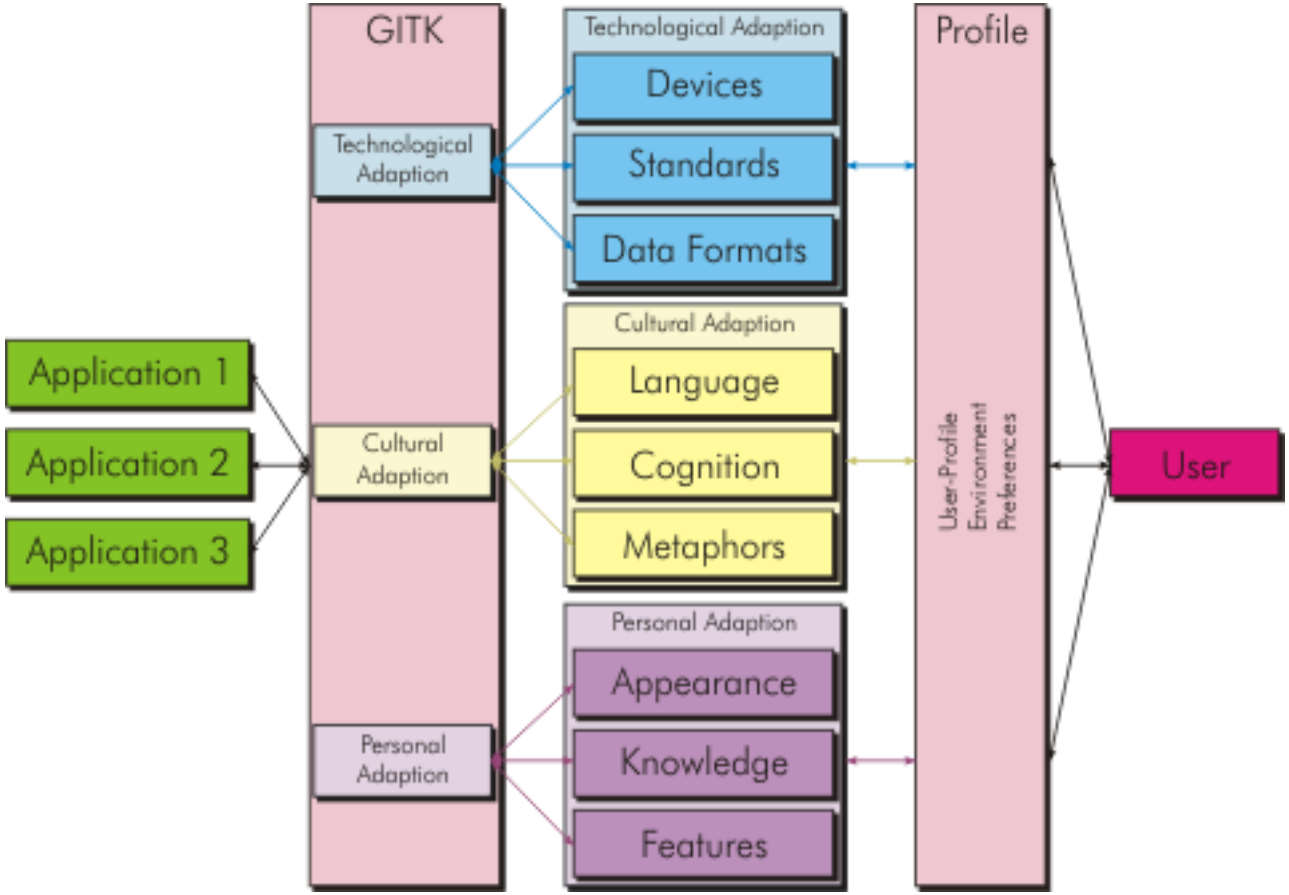
temporäre Behinderung

 # Adaption ist für jederman und überall wichtig

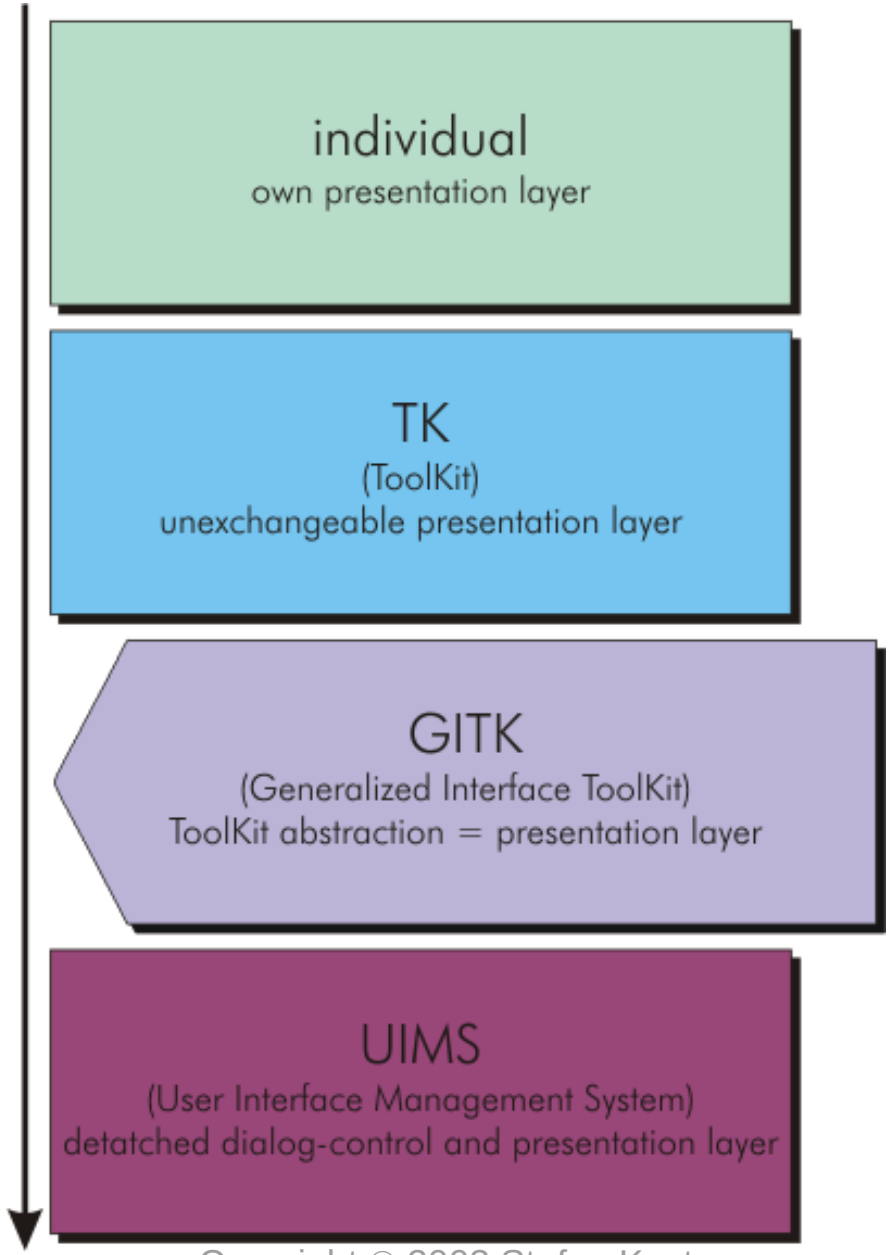
Multidimensionale Adaption: ist



Multidimensionale Adaption: soll



Evolution



Evolution (Continued)

Fazit

Welche Vorteile entstehen für welche Nutzergruppen?

Anwender:

- **er wählt die Präsentation der Schnittstelle**

Entwickler

- **er muss sich keine Gedanken um Präsentation der Schnittstelle machen**

Einordnung

Technologie

- **Software Layer zwischen Applikation und (User)Interface ToolKits**
- **Meta ToolKit**
  - **verhält sich wie ein ToolKit gegenüber einer Anwendung**
  - **verhält sich wie einen Anwendung gegenüber einem ToolKit**

Grenzen

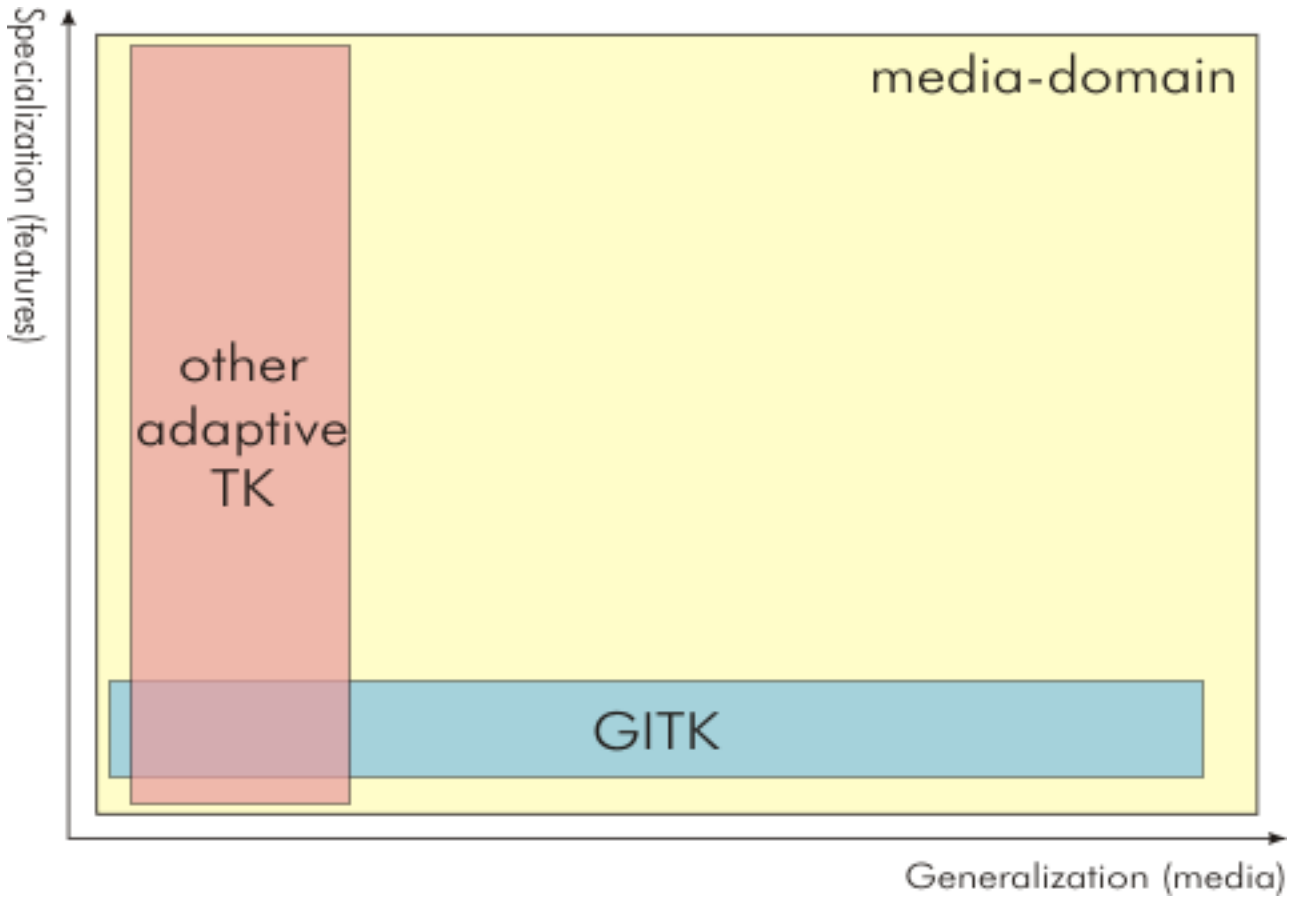
- **GITK ist keine ultimative Lösung!**
- **GITK macht nicht für jede Anwendung Sinn! Focus auf nicht medien-lastigen Anwendungen**

GITK und andere Ansätze

andere Ansätze: UIML, XIML, XUL, AUIML (, wxWindows, Java Swing)

- **meist auf grafische Nutzungsschnittstellen beschränkt**
- **teilweise unflexible Architektur**
- **bestehende Technologien wenig genutzt**
- **teilweise schlecht publiziert**

Ausrichtung von GITK



Welche Probleme sind zu lösen?

konzeptionelle und technische Probleme:

- **1.) Nutzungsmodelle**
- **2.) Toolkit Abstraktion**
- **3.) Transparenz**

1. Nutzungsmodelle

Wie/Wann/Wo können Interfaces genutzt werden?

- **semiaktiv : Sekundärsysteme, Hilfssysteme**
- **passiv : Alarm, Statusmeldungen, Nachrichten**

2. Toolkit Abstraktion

- **unterschiedliche ToolKits**
- **einheitliche Navigation**
- **Navigation in unterschiedlichen Dimensionen**
- **Heraustrennung der Präsentation**
- **Identifikation von logischen und stilistischen Attributen**
- **Reduktion des Interfaces**

3. Transparenz

- **möglichst unsichtbar für den Nutzer (Usabillity)**

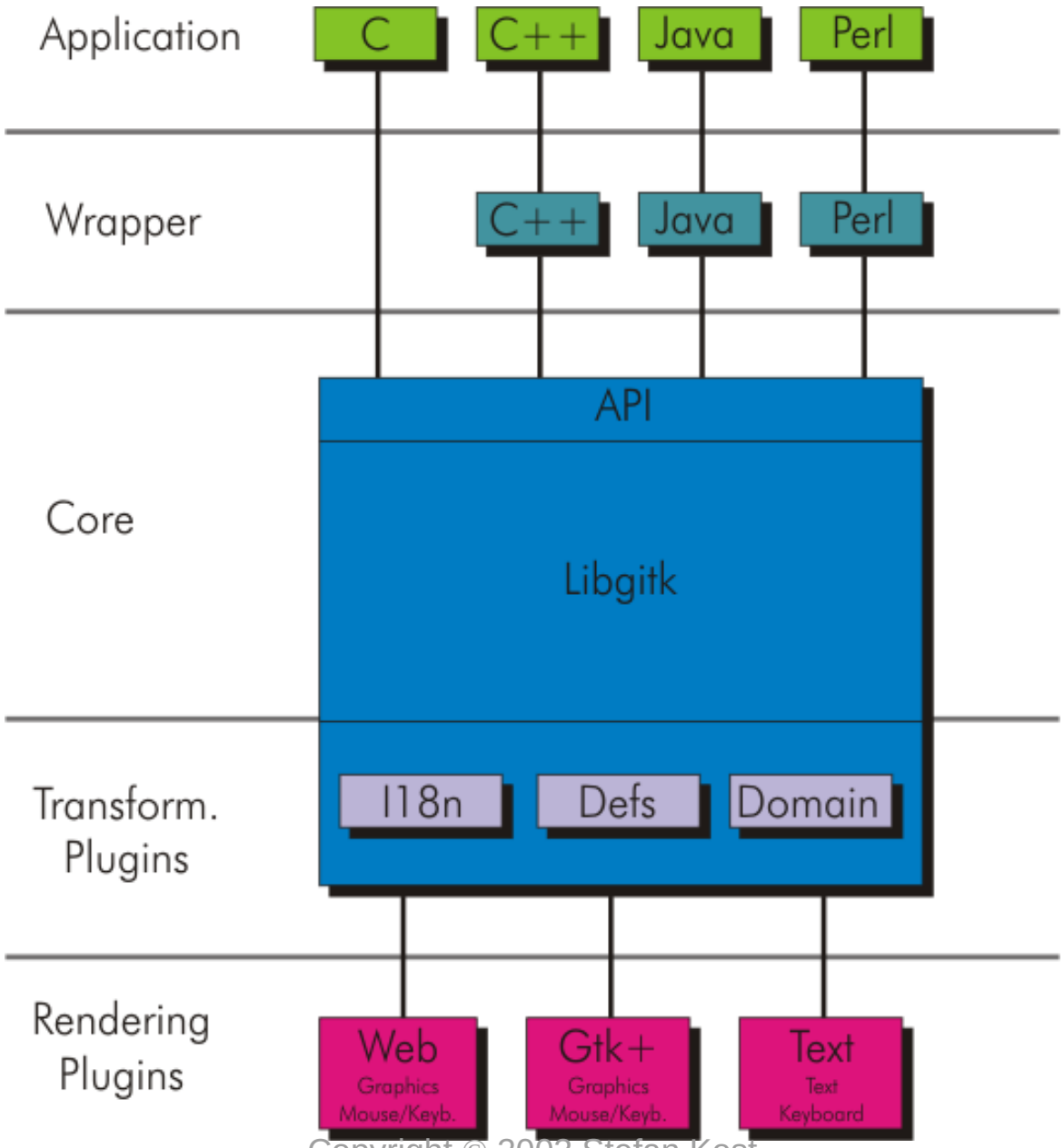
  # automatische Auswahl der Umgebung
- **möglichst unsichtbar für den Entwickler (Akzeptanz)**

  # gängige Paradigmen unterstützen (MVC)

  # Anpassung ist unvermeidbar, sollte sich aber in Grenzen halten

Wie funktioniert GITK?

Architektur



Architektur (Continued)

Wrapper

- **Wrapper = Language Binding**
- **stellen GITK der jeweiligen Programmiersprache zur Verfügung**
- **unterschiedliche Technologien (z.B. Java : JNI, C++ : Klassenbibliothek)**
- **Nutzung von SWIG**

Core

Steuerung der Verarbeitungskette

- **Auswahl der Umgebung (Renderer)**
- **Auswahl der Nutzervorlieben (Styles)**
- **Unterstützung von Application-Hints**
- **Einbindung der Plugins**
- **Steuerung der Navigation**

Transformer Plugins

globale Anreicherung oder Filterung der Dialogbeschreibung

- **<term>I18n</term>**
  : Übersetzung von Texten
- **<term>Defs</term>**
  : Einbeziehung von Standardattributen
- **<term>Domain</term>**
  : Übertragen des Interface-Object Baumes in das Zieldomain
- **<term>Prefs</term>**
  : Einbeziehung der Nutzervorgaben hinsichtlich Presentation und Verhalten

Rendering Plugins

dynamische Erzeugung der Interfaces

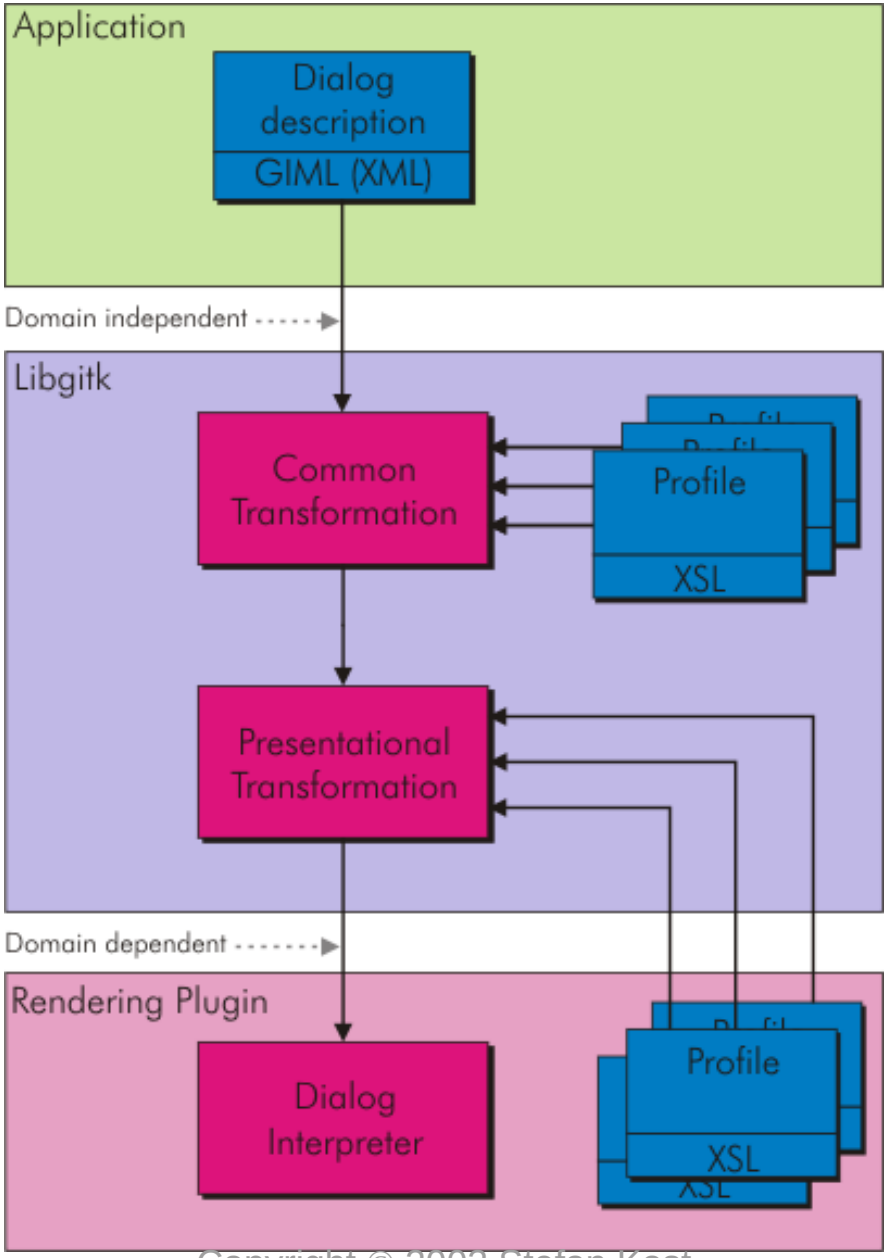
- **<term>Gtk+</term>**
  : graphisches Interface mit Maus- und Tast-tatureingabe
- **<term>Telephony</term>**
  : sprachbasiertes Interface mit Eingabe über die Telefontastatur
- **<term>Text</term>**
  : textbasiertes Interface mit Sprachausgabe und Braillezeilenunterstützung
- **<term>Web</term>**
  : HTML basierendes Interface
- **<term>OpenGL</term>**
  : Interface mit echten 3D Objekten

es sind weitere Renderer denkbar; diese könnten

Rendering Plugins (Continued)

auch *script interfaces* realisieren

Verarbeitungskette



Verarbeitungskette (Continued)

Generalized Interface Markup Language

- eigene Markup Language
- funktionale Beschreibung der Schnittstelle (Inhalte)
- keine domainspezifischen Attribute

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE giml SYSTEM "http://gitk.sourceforge.ne
<giml xmlns="http://gitk.sourceforge.net/"
        xmlns:dc="http://purl.org/dc/elements/1.1
        xmlns:i18n="http://apache.org/cocoon/i18n
  >
  <context type="dialog">
    <meta><dc:title><i18n:text>query user identit
    <dialogwidgets>
      <dialogwidget id="Okay"/>
```

Generalized Interface Markup Language

```
      <dialogwidget id="Cancel"/>
    </dialogwidgets>
    <widgetgroup>
      <label><i18n:text>identity</i18n:text></lab
      <widget id="UserName" type="characterinput_
        <label><i18n:text>user name</i18n:text></
        <disabled>true</disabled>
      </widget>
      <widget id="Sex" type="optionchoice_single_
        <label><i18n:text>sex</i18n:text></label>
        <options>
          <option><i18n:text>male</i18n:text></op
          <option><i18n:text>female</i18n:text></
        </options>
      </widget>
    </widgetgroup>
  </context>
</giml>
```

(Continued)

Interface Object Namensgebung

funktionsbezogene Namensgebung

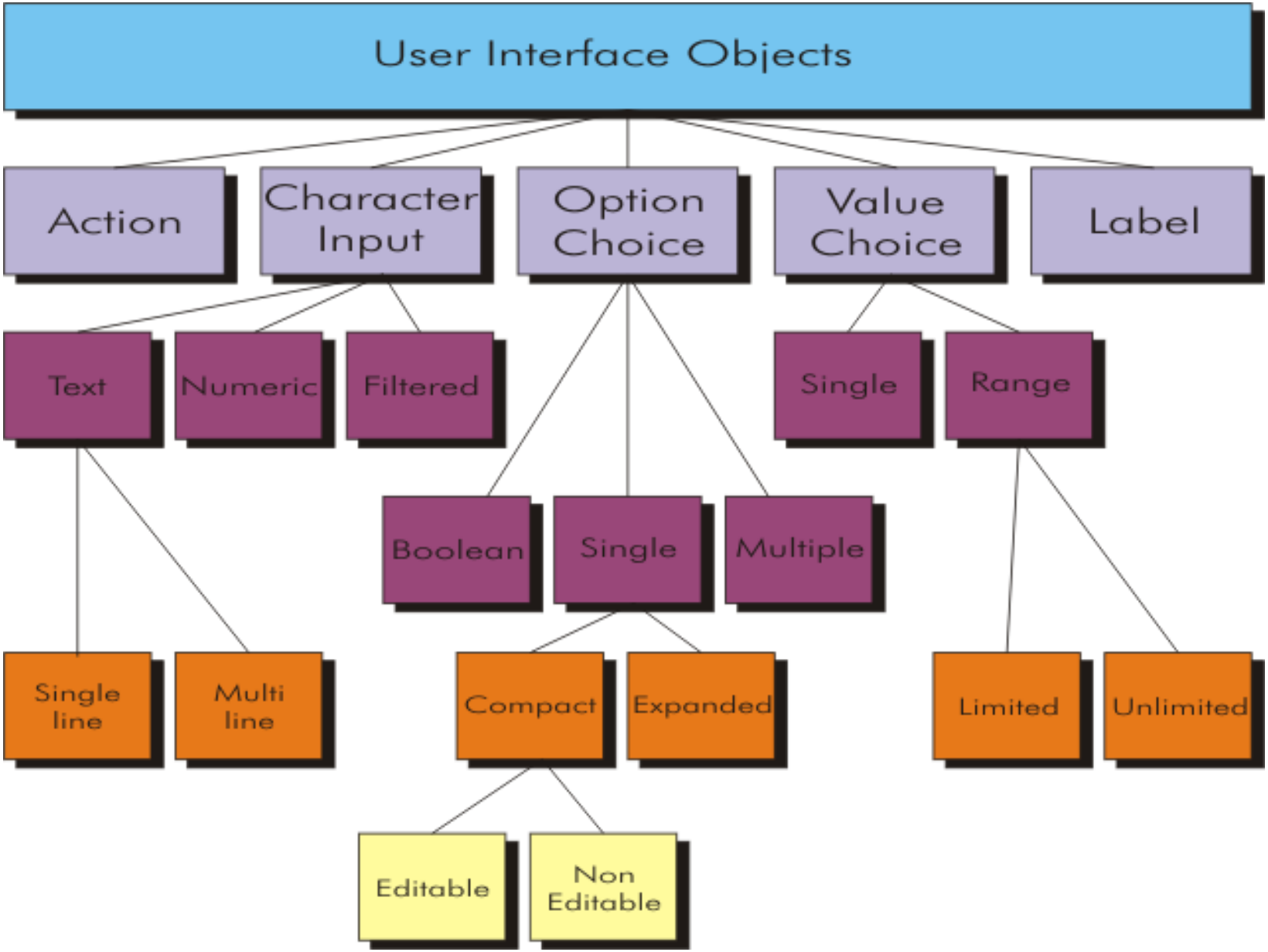
dadurch medien-neutral

- **push-button # action**
- **scroll-bar # value choice/single/limited**
- **radio-button # option choice/single/expanded**

Interface Object Hierachie

gemeinsamer Nenner zwischen den ToolKits

Interface Object Hierachie (Continued)



Interface Object Hierachie (Continued)

Technologie

- **Core-Bibliothek ist in C geschrieben**
- **Renderer sind in C / C++ geschrieben**
- **xml Verarbeitung benuzt LibXML2**
- **xslt Verarbeitung benutzt LibXSLT**
- **für erhöhte Portabilität wird die GLib benutzt**
- **Wrapper können mit SWIG erzeugt werden**
- **Unittests mit Check**
- **Qualitätsreviews mit SPlint und FlawFinder**
- **i18n benutzt GNU gettext**

Was ist der Stand der Dinge?

Was ist der Stand der Dinge?

Konzept : aktueller Stand

- **ganzheitliches Konzept für Adaption**
- **eigene Markup Language**
- **domainunabhängige Interface Objekthierarchie**
- **Transfomationsmodell = Verarbeitungskette**
- **Navigationsmodelle**
- **Systemarchitektur (Schichten, Plugins)**

 # eine offensichtliche Lücke wurde geschlossen

Konzept : offene Probleme

- **User Profile**
- **generische Dialogkontrolle**

Implementation : aktueller Stand

gitk-core

- **Optionsverarbeitung (Startoptionen)**
- **Plugins und Transformer (defs,domain,i18n)**
- **Unicodeunterstzung**

gitk-renderer

- **Gtk+: grundlegende Widgets**
- **Text: grundlegende Widgets, Sprachausgabe, Navigationsmodelle**
- **OpenGL, Telephony: Plugin-Grundstruktur**

gitk-examples

Platformunabhängigkeit

- **Plattformen: Linux, Solaris (Unix generell)**

Implementation : aktueller Stand

- **Programmiersprachen: C, C++**

Beispiel

Textmodus Renderer

Renderer für gtk+

Beispiel (Continued)

Implementation : nächste Schritte

Renderer ausbauen

- **Text: Sprachsteuerung**
- **OpenGL, Telephony: Weiterentwicklung**

Beispiele ausbauen

Platformunabhängigkeit

- **Plattformen: Windows (via CygWin)**
- **Programmiersprachen: Perl, Java (via SWIG)**

Entwicklung

Entwicklung findet auf http://gitk.sourceforge.net/ statt. Dort findet man:

- **Downloads : Archive mit der aktuellen Version**
- **CVS : aktueller Entwicklungsstand**
- **Mailinglisten : Entwickler- und Anwenderforum**
- **IssueTracker : Bug- und Featurerequests**
- **News : Mitteilungen über das Projekt**
- **etc.**