

---

# Meta-Programming Revisited

## *Web Services and GUI Generation*

Cornelius Schumacher and Tobias König

The KDE Project



# Overview

---

- What's happened so far?
- The "kode" project
- kxml\_compiler
- kwsdl\_compiler
- GUI generation with Kung
- GUI generation with KXForms



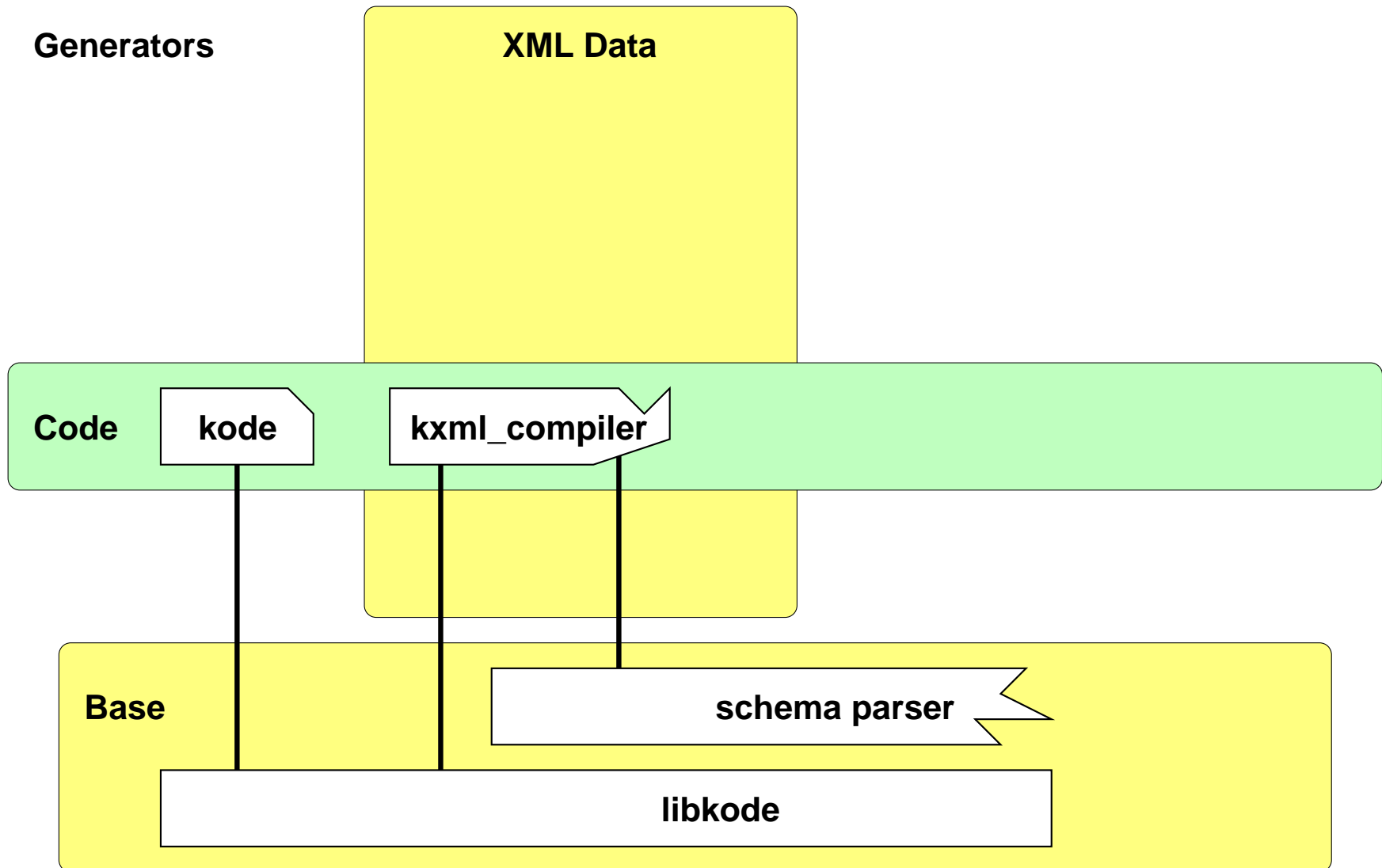
# What's happened so far?

---

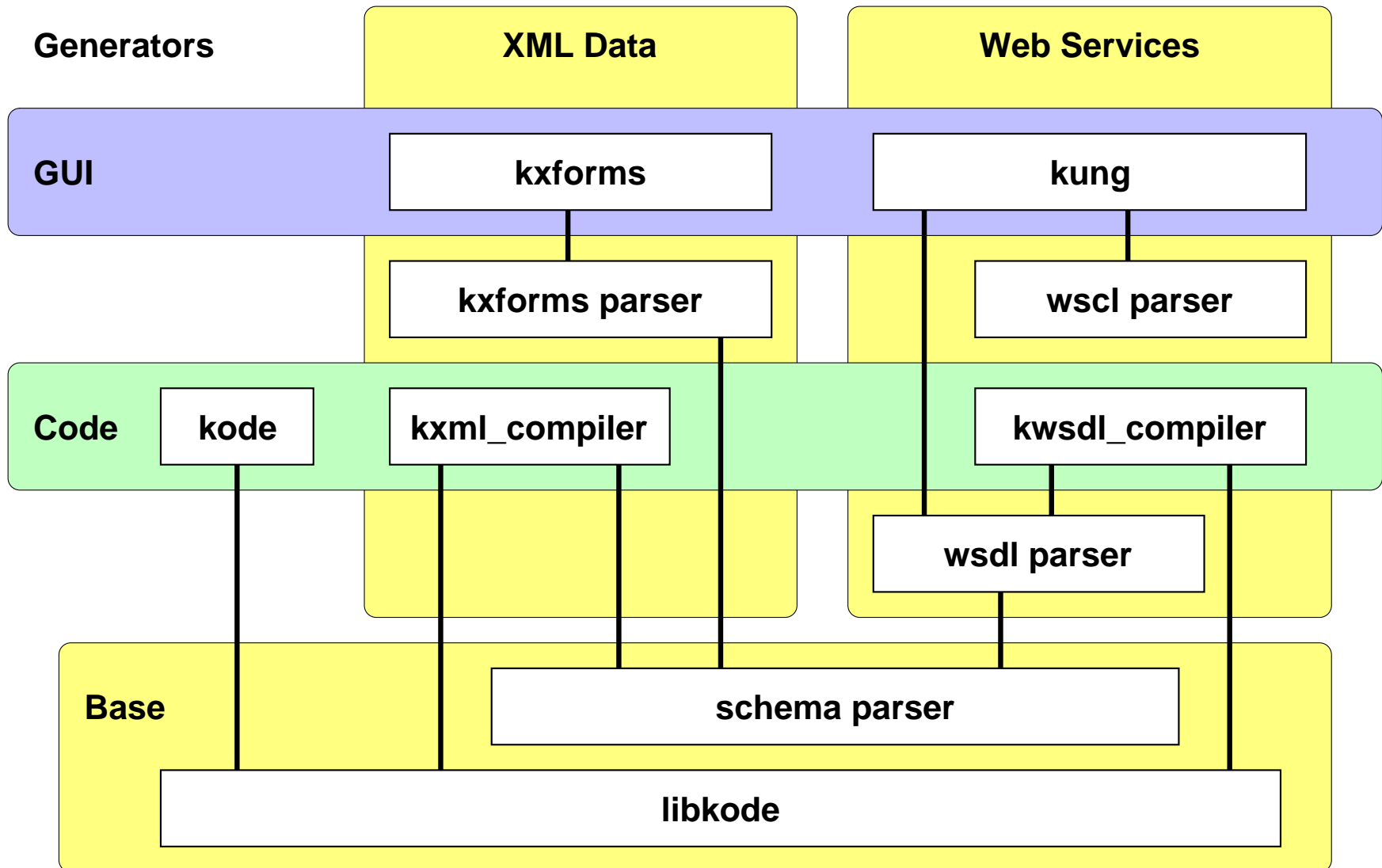
- aKademy 2004: Meta-Programming in KDE - The Technology behind KConfig XT and friends
  - Generating addressbook code in libkabc
  - Generating config code in KConfig XT
  - Generating XML handling code with kxml\_compiler
  - Proof of concept: Read-only feature plan resource for KOrganizer



# Kode Architecture 2004



# Kode Architecture 2005



# libkode

---

- Representing C++ code by C++ objects
- File, Class, Function, Variable, Code
- Special content: StateMachine, License, Automake file
- Classes generating output: Printer, Style
- Pragmatic solution for generating code



# libkode Example

---

```
KODE::Function writer( "writeElement", "QString" );
```

```
KODE::Code code;
```

```
code += "QString xml;";
```

```
QString tag = "<" + element->name;
```

```
QValueList<Attribute *>::ConstIterator it3;  
for( it3 = element->attributes.begin();  
    it3 != element->attributes.end(); ++it3 ) {  
    tag += " " + (*it3)->name + "=\\" +  
          (*it3)->name + "()" + "\\";"  
}
```

```
if ( element->isEmpty ) {  
    tag += "/";  
}
```

```
tag += ">\n";
```

```
code += "xml += indent() + \"" + tag + "\";";
```



# Kode Command Line Tool

---

- `kode` is a utility for code generation tasks

## Templates

- Create class template (license, author information from KDE address book)
- Create dialog template
- Create kioslave template

## Helpers

- Add property to class (inferior implementation, IDEs have much more powerful solutions)
- Codify





# Kode Command Line Options

---

Usage: `kode [Qt-options] [KDE-options] [options] [filename]`

## Options:

<code>-c, --create-class</code>	Create class
<code>-d, --create-dialog</code>	Create dialog
<code>--create-kioslave</code>	Create kioslave
<code>--create-main</code>	Create main function template
<code>-y, --codify</code>	Create generator code for given source
<code>--add-property</code>	Add property to class
<code>--inplace</code>	Change file in place
<code>--author-email &lt;name&gt;</code>	Add author with given email address
<code>--project &lt;name&gt;</code>	Name of project
<code>--gpl</code>	Use GPL as license
<code>--lgpl</code>	Use LGPL as license
<code>--classname &lt;name&gt;</code>	Name of class
<code>--filename &lt;name&gt;</code>	Name of file
<code>--namespace &lt;name&gt;</code>	Namespace
<code>--warning</code>	Create warning about code generation
<code>--qt-exception</code>	Add Qt exception to GPL
<code>--singleton</code>	Create a singleton class
<code>--protocol</code>	kioslave protocol

## Arguments:

<code>filename</code>	Source code file name
-----------------------	-----------------------



# kxml\_compiler

---

- XML writer
- Second parser implementation, schema-optimized parser code.
- Still based on Relax NG, XML Schema is missing, but will come.
- Incomplete implementation, needs more love.
- Useful tool, but it's still not good enough for mainstream adoption
- Writing XML doesn't preserve formatting. (Is it a worthwhile goal to fix that?)



# XML Schema

---

- XML Schema is a W3C recommendation for formally describing XML document classes ([www.w3.org/XML/Schema](http://www.w3.org/XML/Schema)).

## Alternatives:

- DTD (not very expressive, not XML)
- Relax NG (theoretically well-founded, not as commonly used as XML Schema)
- Schematron (based on finding tree patterns, not on grammars)
- Examplotron (lightweight, based on instance documents)



# XML Schema Parser

---

- Parser based on an implementation in C++ from the wsd1pull project ([wsdlpull.sf.net](http://wsdlpull.sf.net)).
- Creates in-memory schema representation suitable for being used by C++ programs using Qt.
- Handles Simple Types, Complex Types, Namespaces, basic XML Schema data types.
- Complete enough for parsing the XML Schema commonly used in WSDL descriptions.
- Namespace handling needs improvement.
- Unions and groups are not supported yet.



# WSDL

---

- WSDL: Web Services Description Language
- [www.w3.org/TR/wsdl](http://www.w3.org/TR/wsdl)
- XML standard format for abstractly describing network services
- End points exchanging messages
- Bindings to concrete network protocols and message formats
- Bindings to SOAP, HTTP GET/POST and MIME



# WSDL Parser

---

- libwsdl for parsing WSDL descriptions.
- Handles messages, ports, bindings, services.
- Makes use of XML Schema parser
- Parses the important publically available web services: Amazon, Google, eBay.
- Is intended to also parse Groupwise WSDL used in the Kontact Groupwise KResource
- Supports SOAP binding.
- Doesn't support HTTP and MIME bindings yet.



# kwsdl\_compiler

---

- Creates code to parse and create SOAP messages from a WSDL description of the corresponding web service.
- Uses libwsdl to parse the WSDL descriptions.
- Uses libkode to create the generated code.



# kwsdl\_compiler Generated Objects

---

- C++ representations of the SOAP messages and their complex arguments.
- Customized Serializer for converting C++ objects to XML representations and back.
- Transport class to do asynchronous SOAP request using the HTTP kioslave (support for SSL, Proxies, etc. for free).
- Transport class could be exchanged with Qt-only transport class.
- Top-Level service access class for conveniently doing web service requests from native code in a type-safe way without having to care for any SOAP or XML details.





# WSCL

---

- WSCL: Web Services Conversation Language
- [www.w3.org/TR/2002/NOTE-wscl10-20020314](http://www.w3.org/TR/2002/NOTE-wscl10-20020314)
- XML standard format for describing business level conversations or public processes of web services
- Specifies conversations of a web service, which documents are exchanged in which order
- Can be associated with a WSDL description
- Parser supports complete specification.



# Kung

---

- Creates a GUI on the fly for interacting with a web service from the WSDL description.
- Can use WSCL to specify flow of messages.
- Specific GUI representations for the types of data provided by the web service description.
- All XML handling, HTTP interaction, loading and saving from the GUI is automatically done in the background.



# The Google Web Service

---

- Google search functionality provided as SOAP based web service
- Licence key required, available for free for personal, non-commercial use, 1000 requests a day



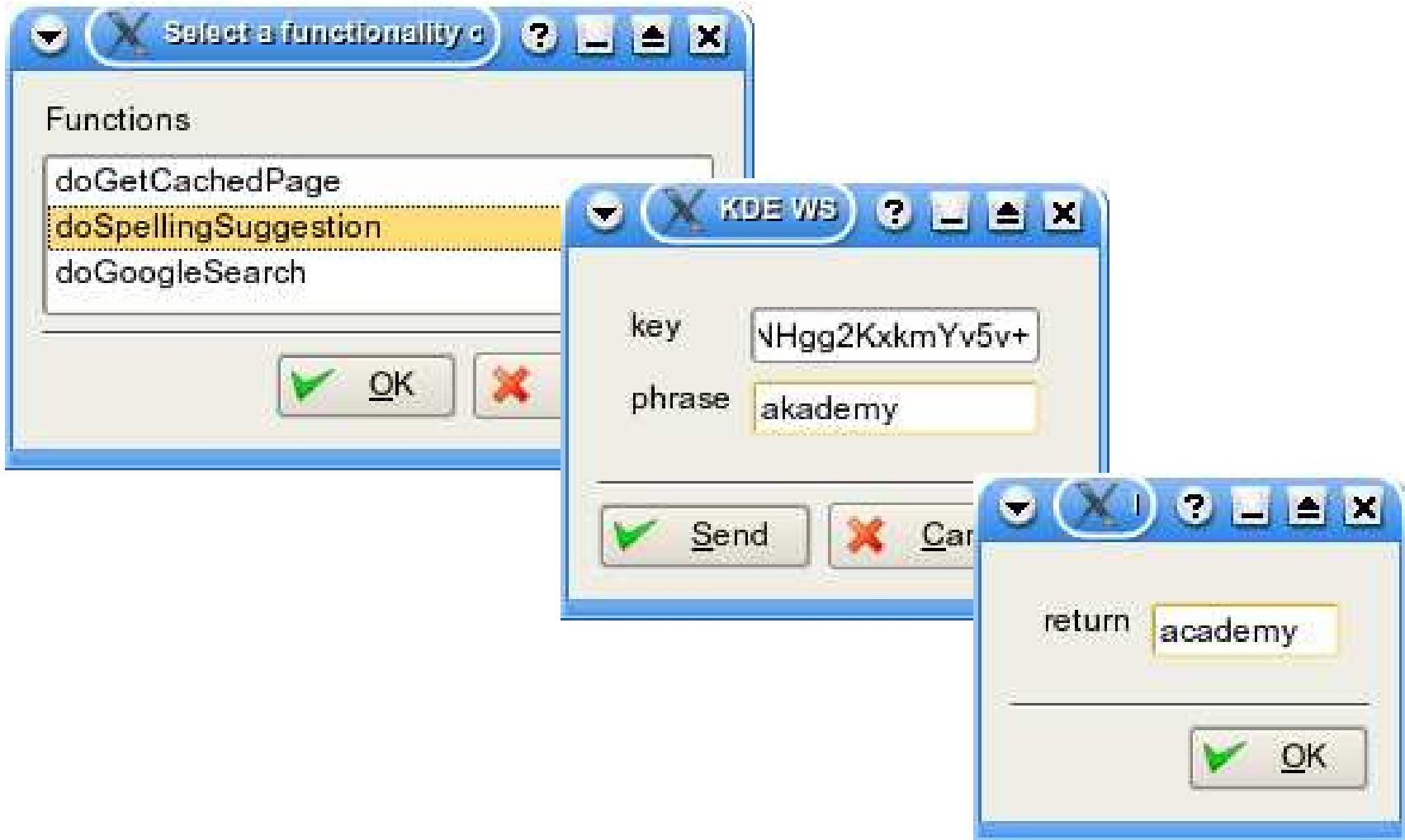
## API:

- `doGetCachedPage`
- `doSpellingSuggestion`
- `doGoogleSearch`
- Objects for representing the search result



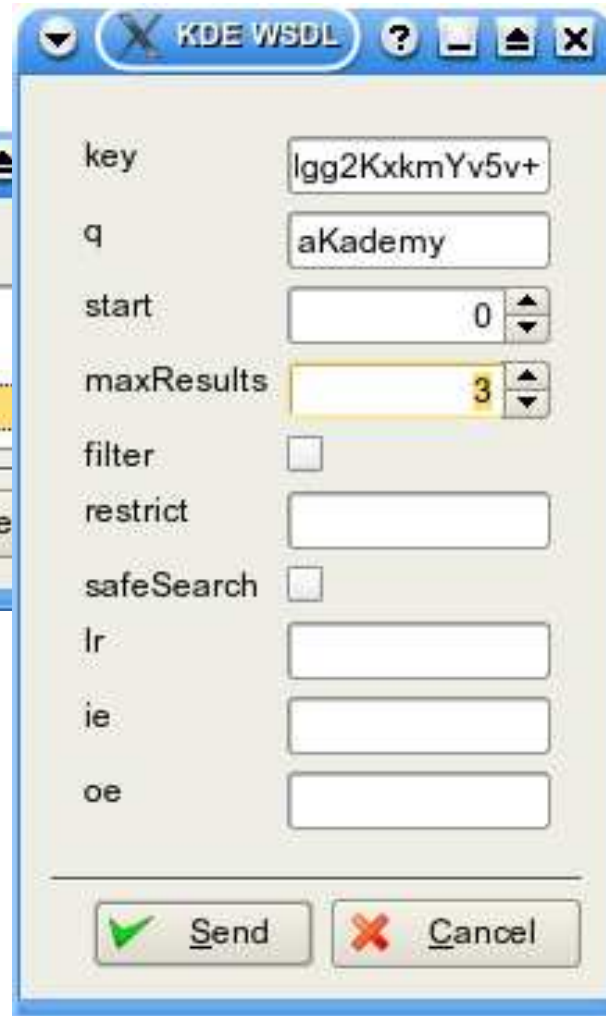
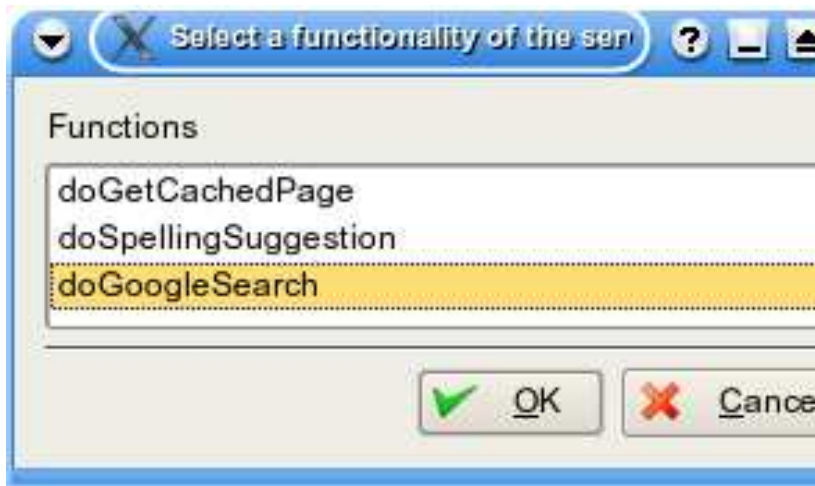
# Kung Demo 1

```
>kung http://api.google.com/GoogleSearch.wsdl
```



# Kung Demo 2 - Request

```
>kung http://api.google.com/GoogleSearch.wsdl
```



# Kung Demo 2 - Response

The image shows two overlapping windows from the KDE WSDL Interpreter. The background window displays search results for the query 'aKademy'. It includes fields for document filtering, search comments, estimated total results count (26300), and search parameters like start index (1) and end index (3). A list of 'resultElements' contains two 'item' entries, with the first one selected. Below this is a 'directoryCategories' section with an empty list and an 'Add' button. The search time is shown as 0.0. An 'OK' button is at the bottom right.

The foreground window shows a detailed view of the selected 'item'. It contains the following fields:

- summary: 9 days with an exciting program.
- URL: http://conference2004.kde.org/
- snippet: 9 days with an exciting program.
- title: 4 &quot;<b>aKademy</b>&quot;
- cachedSize: 15k
- relatedInformationPresent:
- hostName: (empty)
- directoryCategory: (empty)
- fullViewableName: :nts/KDE
- specialEncoding: (empty)
- directoryTitle: 4 &quot;<b>aKademy</b>&quot;

An 'OK' button with a green checkmark is located at the bottom right of this window.



# KXForms

---

- Approach: Use intermediate abstract GUI description to be able to automatically create GUIs
- Applications: Editor for XML data, configuration GUI based on KConfig XT descriptions, more
- Create intermediate description from descriptions of the data to be edited, e.g. XML Schema, KConfig XT



# XForms

---

- XForms is the successor of HTML forms.
- W3C recommendation: [www.w3.org/TR/xforms/](http://www.w3.org/TR/xforms/)
- XML based
- Standard GUI elements: input, secret, textarea, output, upload, range, trigger, submit, select, select1
- Grouping
- Interaction via XML, model, instance data, form element reference XML data elements.
- XPath for referencing data.
- Processing model, event specification
- Embeddable in host languages





# KXForms Format

---

- Host language making use of XForms GUI elements and referencing scheme.
- KXForms as Pragmatic XForms
- Extension element `list` for heterogenous lists (including support for data driven item labels)



# KXForms Example

---

- Description of GUI for editing KDE Feature Plan

```
<kxforms>
```

```
<form ref="category">  
  <xf:input ref="@name">  
    <xf:label>Name</xf:label>  
  </xf:input>  
  <list>  
    <xf:label>Item</xf:label>  
    <itemclass ref="category">  
      <itemlabel>Category <arg ref="@name"/></itemlabel>  
    </itemclass>  
    <itemclass ref="feature">  
      <itemlabel>  
        Feature <arg ref="summary" truncate="20"/>  
      </itemlabel>  
    </itemclass>  
  </list>  
</form>
```

```
(...)
```

```
</kxforms>
```



# KXForms Engine

---

- Form representation
- Referencing XML data
- Common GUI creation and objects
- GuiHandler for handling GUI details, i.e. layout, nesting, widget layering, etc.



# KXForms Demo - XML Data

---

```
<features>

  <category name="KDE PIM (Personal Information Management)" >

    <category name="KMail" >

      <feature status="inprogress" target="3.5" >

        <summary>Client side IMAP filtering.</summary>

        <responsible email="adam@kde.org" name="Till Adam" />
        <responsible email="sanders@kde.org" name="Don Sanders" />

      </feature>

    </category>

  </category>

</features>
```



# KXForms Demo 1

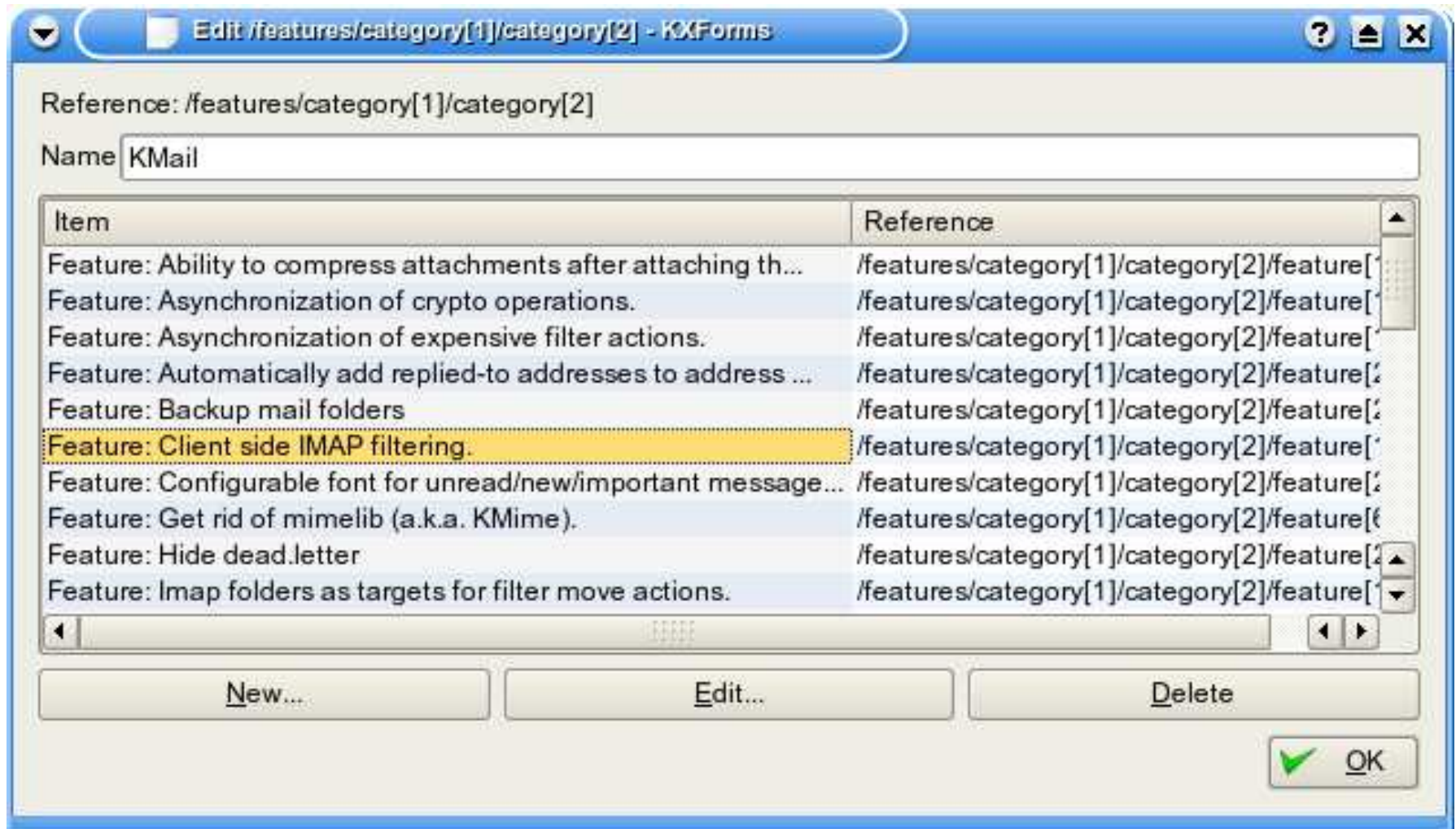
```
> kxforms --kxform gui.kxform features.xml
```

The image shows two windows from the KXForms application. The background window displays a list of categories under the reference path `/features`. The foreground window, titled `Edit /features/category[1] - KXForms`, shows the details for the selected category `KDE PIM (Personal Information Management)`. It includes a table of sub-categories and their references.

Item	Reference
Category: Akregator	/features/category[1]/category[14]
Category: KAddressBook	/features/category[1]/category[10]
Category: KAlarm	/features/category[1]/category[5]
Category: KArm	/features/category[1]/category[6]
Category: KitchenSync	/features/category[1]/category[8]
Category: KMail	/features/category[1]/category[2]
Category: KNode	/features/category[1]/category[9]
Category: KNotes	/features/category[1]/category[7]
Category: KonsoleKalendar	/features/category[1]/category[11]
Category: Kontact	/features/category[1]/category[1]



# KXForms Demo 2



# KXForms Demo 3

The image shows two windows from the KXForms application. The main window, titled "Edit /features/category[1]/category[2]/feature[1] - KXForms", displays the following information:

- Reference: /features/category[1]/category[2]/feature[1]
- Summary: Client side IMAP filtering.
- Status: In Progress
- Target: 3.5
- Responsibles table:

Responsibles	Reference
Don Sanders <sanders@kde.org>	/features/category[1]/category[2]/feature[1]/responsible[2]
Till Adam <adam@kde.org>	/features/category[1]/category[2]/feature[1]/responsible[1]

A "New..." button is visible at the bottom left of the main window. An overlaid dialog window, titled "Edit /features/category[1]/category[2]/feature[1]/resp", is open, showing the details for the selected responsible:

- Reference: /features/category[1]/category[2]/feature[1]/responsible[1]
- Name: Till Adam
- Email: adam@kde.org
- OK button with a green checkmark icon.



# XML Schema and KXForms

---

- Transformation of XML Schema to KXForms
- Schema references in XML can automatically be resolved to create a GUI for editing XML data.
- Data on a server which is accessible by kioslaves can be edited without any more information or tools.
- GUIs needs to be customized.
- Label and hint texts.
- Layout and hierarchy hints.
- Annotating XML Schema, either inline or externally





# Outlook KXForms

---

- Generate KXForms from KConfig XT descriptions to generate configuration dialogs.
- Make use of KXForms in Kung.
- More clever layouting logic.
- Problems can be addressed separately and generic.
- Lots of XML handling: Better integrate with other libraries, make use of existing frameworks, e.g. kdom.



# Conclusion

---

- Kode project has grown
- Playground for some ideas
- Covers code generation, XML technologies, GUI generation
- Current code: `branches/work/kode-x/kode`

