TIRA: An MDA Approach to Implement Personal IR Tools

Sven Meyer zu Eissen and Benno Stein

Bauhaus University Weimar
Web-Technology and Information Systems
IR in Distributed Environments

IR use case:

Realization of multi-user IR systems:

Client Computer

Server (Gateway)

e.g. Browser
e.g. Yahoo! search, HyperWave server, Google desktop search
Personal Information Needs (The Client Side)

- Article on genetically modified food?
- Find old version of this document on my hard drive.
- Who plagiarized my work?
- Find documents that contain market analyses; today for the RFID market.
- Extract opinions about mobile phones from blogs.
Operationalization of IR Tasks

List of wishes: An IR system should

- adapt to personal data
- adapt to personal preferences (e.g. result presentation)
- adapt to personal skills (e.g. query formulation)
- adapt to personal knowledge (e.g. about collection)
- adapt to personal IR tasks
Operationalization of IR Tasks

List of wishes: An IR system should

- adapt to personal data
- adapt to personal preferences (e.g. result presentation)
- adapt to personal skills (e.g. query formulation)
- adapt to personal knowledge (e.g. about collection)
- *adapt to personal IR tasks*

Today:
A query is almost always formulated in the form of *keywords*. The IR process is hard-wired *at the server side*.

Tomorrow (with TIRA):
A query can be an *IR process specification* (soft-coded *at the client side*).

What are the building blocks of an IR process?
Example IR Task: Categorizing Search

Introduction

Personal Inform. Needs

Modeling IR Processes

The TIRA Architecture

Σ

TIR-07 Sep. 3rd, 2007 Meyer zu Eissen/Stein
Example IR Task: Categorizing Search

Required are modules for

- importing various formats (HTML, PDF,...),
- language detection,
- stemming, stopword identification,
- clustering ($k$-means, MajorClust,...), cluster labeling,
- classification (discriminant analysis, SVMs,...),
- ...

TIR-07 Sep. 3rd, 2007 Meyer zu Eissen/Stein
Example IR Task (Simplified)

For the sake of simplicity we regard the following IR process:

1. Download an HTML document from a URL.
2. Build document representation according to topic.
3. Build document representation according to genre.
4. Classify according to topic and genre.

Key question: How can such an IR process be specified?
The Modular Nature of IR Processes

Some characteristics:

- An IR process is a sequence of transformations.
- IR processes: composed of autonomous building blocks.
- IR theory: different solutions for the same task (e.g. stemming, categorization; cf. Strategy Pattern).
- One base algorithm for similar tasks (e.g. stemming; cf. Factory Pattern).
- IR processes are subject to frequent change (optimization, new ideas, changing information needs).
- IR process subtasks may be executed in parallel.
- Set of useful standard modules for any application.
Specification of IR Processes

Standard solution (as we call it): the “Library Approach”.

→ Design generic interfaces.
→ Build software libraries.
→ Build special-purpose application.

Specification of example task as code:

```
Input: URL u, dictionary dict, stopword list stl.
Output: genre and topic class for the document at URL u.

Text ht=download(u);
Text plain=removeHTMLTags(ht);
Text filtered=removeStopwords(plain, stl);
Features topicModel=
    buildTopicModel(filtered, dict);
Language lang=detectLanguage(plain);
Features presentF=buildPresentationF(ht);
Features posF=buildPOSF(plain, language);
Features genreModel=union(presentF, posF);
int topicClass=classifyTopic(topicModel);
int genreClass=classifyGenre(genreModel);
return(topicClass, genreClass);
```
Specification of IR Processes

Drawbacks of the library approach:

- Needs expert knowledge in specification language / libraries
- Changing the process is tedious and error-prone.

More flexible, more abstract, more expressive:

Diagrammatic language that specifies the data / control flows.
Specification of IR Processes

Drawbacks of the library approach:

- Needs expert knowledge in specification language / libraries
- Changing the process is tedious and error-prone.

More flexible, more abstract, more expressive:
Diagrammatic language that specifies the data / control flows.

Classification of diagrammatic modeling tools: [Teich 1997]

- Control flow dominant or state oriented (FSM, state charts)
- Data flow dominant or activity oriented
  (Petri nets, UML activity diagrams, marked graphs)
- Structure oriented (UML class diagrams)
- Time oriented (UML time diagrams)
- Data oriented (ER diagrams)
- hybrid
Petri Net Specification of Sample Task

- URL
- Download
- Hypertext
- Plain text extraction
- Plain text
- Stopword list
- Stopword removal
- Filtered text
- Dictionary
- Topic model builder
- Presentation feature builder
- POS feature builder
- Feature union
- Genre model
- Genre classifier
- Genre class
- POS features
- Presentation features
- Language detection
- Language

Introduction
Personal Inform. Needs
Modeling IR Processes
The TIRA Architecture

TIR-07 Sep. 3rd, 2007 Meyer zu Eissen/Stein
Petri Net Specification of Sample Task

- URL
- Download
- Hypertext
- Plain text extraction
- Plain text
  - Stopword list
  - Stopword removal
  - Filtered text
    - Dictionary
    - Topic model builder
- Presentation feature builder
- POS feature builder
  - POS features
  - Language detection
  - Language
  - Copy
    - Plain text
    - Plain text extraction
- Presentation features
- Feature union
  - Genre model
    - Genre classifier
      - Genre class
      - Topic model
        - Topic classifier
          - Topic class
Petri Net Specification of Sample Task
Petri Net Specification

Discussion

+ Petri nets are well researched.
+ Modeling concurrency is possible.

- Data types cannot be modeled.
- Modeling iterations is complicated.
- Iterations based on the “content” of tokens are impossible.
- Scheduling policy for the places cannot be specified.
UML Activity Diagram Specification of Sample Task

1. **URL**
   - Download
     - hyperText:Text
     - u:URL

2. **PlainText Extraction**
   - plainText:Text
   - stl:Text

3. **Language Detection**
   - lang:Language
   - hyperText:Text
   - presentation:FeatureVector

4. **Presentation Feature Builder**
   - presentation:FeatureVector

5. **POS Feature Builder**
   - posFeatures:FeatureVector

6. **Stopword Removal**
   - filteredText:Text
   - dict:Dictionary

7. **Feature Union**
   - genreModel:FeatureVector

8. **Genre Classifier**
   - genre class

9. **Topic Model Builder**
   - topicModel:FeatureVector

10. **Topic Classifier**
    - topic class
UML Activity Diagram Specification

Discussion

+ Intuitive and widely accepted.
+ Modeling of iterations, concurrency, and data types.
+ Advanced concepts like exception handling, streams,...
+ Diagrams are updated frequently.

- “Simulation” unclear.
Operationalizing IR Processes with TIRA

Activity diagrams are *independent from*

- programming languages,
- operating systems,
- middleware platforms,
- system architectures.

⇒ An activity diagram is a platform independent model (PIM).
Operationalizing IR Processes with **TIRA**

Activity diagrams are *independent from*

- programming languages,
- operating systems,
- middleware platforms,
- system architectures.

→ An activity diagram is a platform independent model (PIM).

Required for execution (in terms of MDA):
A sequence of transformations to a platform specific model (PSM).
Operationalizing IR Processes with **TIRA**

**TIRA solution:**

- Encapsulate library functions as Web services.
- Specify data types with XML schema. Serialize data as XML. Visualize data with XSLT.
- For global access: data are published under a certain URL.
- Simulate the activity diagram:
  
  Execute Web services with the data URLs as parameters.

---

**Client code**

*(Platform specific):*

```java
WebService ws=new WebService();
ws.setParameter(plain, stopwords);
URL filteredText=ws.call(service);
```

**Flowchart:**

- **plaintext**: Text
- **stl**: Text

**Stopword removal**

- **filteredText**: Text

---

TIR-07 Sep. 3rd, 2007 Meyer zu Eissen/Stein
Operationalizing IR Processes with \textit{TIRA}

Example: Operationalizing action nodes.

\begin{itemize}
\item \texttt{Node}
\item \texttt{ForkNode}
\item \texttt{JoinNode}
\item \texttt{DataService}
\item \texttt{ActionNodeFactory}
\item \texttt{ServiceRegistry}
\item \texttt{ActionNode}
\end{itemize}

\begin{itemize}
\item \texttt{ForkNode}
\item \texttt{JoinNode}
\item \texttt{DataService}
\item \texttt{ActionNodeFactory}
\item \texttt{ServiceRegistry}
\item \texttt{ActionNode}
\end{itemize}
Conclusion / Outlook

- TIRA lets a user specify and execute personal IR processes.
- TIRA is an open MDA-based architecture for personal IR. Open means that everybody can contribute own services.
- TIRA is flexible, modular, scalable.
- Development of PSM-transformations to other platforms: clusters, P2P (BSP), grids,...
- Research question: Estimation of IR module execution times, scheduling, binding.

Demo
http://webis1.medien.uni-weimar.de/tira/
http://webis1.medien.uni-weimar.de/aisearch/aisearch-demo.html
Thank You!

Questions?
Introduction

Personal Inform. Needs

Modeling IR Processes

The TIRA Architecture

Σ

TIRA Screenshot

TIRA Editor

Module Explorer

Module Library

Processes

Clusterer

Group Average Link

iMeans

Major Clust

Single Link

Converter

Datasources

Indexer

Labeler

Viewgenerator

Documents

Cluster Labels

Cluster View

Compressed Word Vectors

Converted Files

Clustering

Uploaded Files

Search Results

Java Applet Window

Search Results 1

Search Results 2

Search Results 3

Stop Words Extraction 1

Porter Stemmer 1

Compressed Word Vectors 1

Indexer 1

Major Clust 1

Clustering 1

Cluster Labels 1

Cluster View Generator 1

Cluster View 1

Labeler 1
**TIRA Architecture**

- **PIM specification**
- **PSM generation**
- **Middleware platform**
- **Computing platform**
- **Operating system**

**Middleware platform**
- **XML object serialization**
- **XML object visualization**
- **Web service abstraction**

**PSM generation**
- **UML activity diagrams: GUI, modeling, management**
- **UML activity diagrams: compilation, deployment, processing**

**PIM specification**
- **TIRA**

**Computing platform**
- **Java Development Kit**

**Operating system**
- **IR module library**

**Introduction**

**Personal Inform. Needs**

**Modeling IR Processes**

**The TIRA Architecture**

Σ

TIR-07 Sep. 3rd, 2007 Meyer zu Eissen/Stein