# AN OPEN PLATFORM FOR DYNAMIC MULTIMEDIA INDEXING

*Bassem HAIDAR, Philippe JOLY, Jean-Paul BAHSOUN*

IRIT, Institut de Recherche en Informatique de Toulouse
118, route de Narbonne - 31062 Toulouse Cedex 04
{haidar, joly, bahsoun}@irit.fr

## ABSTRACT

*In this paper, we present a multimedia indexing platform that allows the dynamic composition of multimedia indexing tools. The goal is to reduce the amount of human interaction needed to produce some complex multimedia description and to provide more flexibility to the collaboration between multimedia indexing tools.*

*The platform allows the communication and the data transfer between tools based on a central server. For all multimedia tools we define a wrapper, which is a package of code that can dynamically link to the central server. This wrapper will be able to reply to the set of standard messages used by the platform. We propose a chaining algorithm based on the Input/Output data type. The dynamic indexing algorithm will aim at automatically finding a subset of tools and their interactions to produce an index required by a user.*

## 1. INTRODUCTION

The need for collaborative indexing and more complexes indexing tools is rapidly increasing in multimedia indexing domain. Many indexes can not be generated by a simple tool and need to be produced by a sequence of more or less sophisticated tools in a given order. These tools are developed and distributed by different research teams with in general really specialized skills on a single media. This makes the exploitation of these tools for cross-media analysis a really difficult challenge and a hard lock for potential research activities on multimedia index mining.

Taking for example the indexing of audio documents; in order to make the transcription of the speech in an automatic way, we may first analyze the audio content type (Speech/Music/Noise), then we can identify the language of speaker in case of speech content, and then we can try to transcribe the speech into text with the corresponding indexer. The integration of these tools into cooperative applications requires the development of an intelligent platform in which the interoperability of different indexing tools is defined.

We propose a dynamic multimedia indexing platform that allows the dynamic tools composition. These tools are modeled as processes with some different potential customized executions according to the client needs. A composite tool is modeled by a tree. The latter defines the order of execution among the nodes or different processes. The goal of our platform is to generate an index in a completely dynamic way as it is explained in the following scenario: given an index the chaining algorithm must find a subset of tools and their interactions to realize the desired index on any type of source (audio, video and textual contents). Input data is not defined by the user's request. All the documents available on the platform are potential candidate to be indexed and will be selected automatically by the chaining algorithm.

The dynamic tool composition provides two important benefits. The first one is more flexibility in the chain composition which can depend on user's requirement, and can be changed dynamically due to the integration of new tools in the platform. The second one is the reduction of human interaction needed to chain indexers.

## 2. STATE OF THE ART

Most of the work in collaborative multimedia indexing has focused on using a static, manually built, chain of multimedia tools to generate a desired index. Among all the projects developed during this last decade and dealing with that kind of problem, let us mention the KLIMT project [1] which proposes an open framework that allows the integration of multimedia indexing tools. The framework implements the communication and the data transfer mechanisms. The KLIMT platform proposes some scenarios for chaining multimedia indexing tools. The chaining process is limited to some well-defined scenarios, which are written as a script.

The AGIR project [2] aimed at developing technologies and software tools allowing efficient multimedia document search systems based on the automated analysis of media content and on a normalized content description language. The collaboration between tools was here still static and manual.

The Acoi project [3] presents a system that combines independent feature detector programs with a multimedia database technology, to provide a semantic rich index to multimedia data items on the Web. A grammatical

framework called 'feature grammars' is used as a description of the execution sequence of the indexing program.
The chaining process is used to compose web services. The order of messages exchanged between services must be described separately in a flow specification. There are many Web services flow specification languages like BPEL4WS [4] and WSCI [5]. The exact control and data flow that determines when an operation can be executed is provided by a flow composition language. Thus, the composition of the flow is still manually obtained [6].

An automated semantic translation of web service is proposed in [7]. It is based on a recursive back-chaining algorithm to determine a sequence of services invocation. In the case of composing web services, the required result can be produced by several input data. In our case (the chaining of multimedia tools), the chaining process is more complicate; in fact, a description corresponds to only one multimedia document. By consequence, constraints must be added to guaranty the convergence to the same input document.

In eFlow [8], a composite service is described as a process schema that composes other basic or composite services. A schema is modeled by a graph (the flow structure), which defines the order of execution among the nodes in the process. The graph may include service, decision, and event nodes. Service nodes represent the invocation of a basic or composite service; decision nodes specify the alternatives and rules controlling the execution flow, while event nodes enable service processes to send and receive several types of events.

An open framework is proposed in the DCS [9] (Distributed Software Component), it allows the integration of news component into a distributed framework. The integration is based on a specification file containing properties about component and on a wrapper that allows the component to be linked to the platform.

## 3. GLOBAL ARCHITECTURE

The platform is composed of the distributed indexing tools and a central server that implements the control and the dynamic chaining services:

### 3.1. Components

An indexing tool does not provide in general the features required for easy integration on a distributed platform. Our idea is to define a wrapper able to reply to the set of standard messages used for communication between those kinds of tools. This wrapper is a package of code that can be dynamically linked to the central server. It is composed of the following two generic interfaces.

#### *3.1.1. Configurator interface*

Each tool will be defined by a specification file written by the tool developer. This describes some properties such as ComponentRole, Description, ExecutablePath, InputType, OutputType, StorageInputPath, StorageOutputPath. This file will be written in XML.

The configurator interface allows the platform to access these specifications needed for the formulation of messages transmitted between the platform and the distant tool.

#### *3.1.2. Control interface*

This interface defines the method used by the plat-form to communicate with the tool. It provides functionalities for:

- Launching the distant component,
- Getting component status, and
- Data transferring from and to the component.

### 3.2. Platform services

#### *3.2.1. Tool discovery service*

Tool discovery is the process of locating tools available to take part in the indexing chain composition.
An architecture supporting dynamic tools composition must have a repository to store specifications about composable tools. This repository must allow a tool, defined by a set of attributes, to be retrieved by a searching service. A new tool will be integrated in the platform by sending towards the storage service the information contained in the specification file. This information will be available for the other components through the platform.

Some searching methods based on particular criteria are implemented in this service. These methods play a principal role in the dynamic composition of tools by allowing the selection of the component able to generate such type of data.

#### *3.2.2. Data transfer and control service*

This service communicates with the control interface of each tool. It allows a robust data transfer and the control of tools: launching, stopping and getting the tool status at runtime. The FTP protocol will be used to transfer the multimedia contents between tools. A CORBA (and later a SOAP) framework has been implemented to transmit the control messages between the central server and the distributed tools.

#### *3.2.3. Dynamic indexing service*

The dynamic composition problem stands in defining and assuming the interoperability between available tools declared in the tool repository to build a given index. Our idea is to describe the Input/output data type of each indexing tool with a semantic description. For instance, the goal of an audio analyzer [10] we intend to plug on the platform is to localize segments of speech, music, noise and silence in an audio file. The integration of this tool in a dynamic chain is based only on the Input/Output data type, i.e. a .wav file as input and an *AudioSegmentContent* type as output (figure 1). Semantic types are from the ones defined by the MPEG7 standard when it is possible. Other semantic types have to be added to provide pieces of description which are not available in MPEG7.

The composition process is based on a chaining algorithm (Section 4). The indexing service uses the discovery service to find the set of indexers needed in the composition process and the control service to execute the created chain.

```
<Component ComponentRole="Index">
<Description>To localize segments of Speech, Music,
Noises and Silences in an audio file</Description>
        <Authors>J.Pinquier</Authors>
        <Version>v123.2b</Version>
        <Input>
                <InputData>
                        <FileFormat>wav</FileFormat>
                        <StorageInputPath>
                        ftp://aljolson.irit.fr/Pidot/Input
                        </StorageInputPath>
                </InputData>
        </Input>
        <Output>
                <OutputData>
                        <FileFormat>xml</FileFormat>
                        <ContentType>AudioSegmentContent
                        </ContentType>
                <StorageOutputPath>
                ftp://aljolson.irit.fr/Pidot/output
                </StorageOutputPath>
                </OutputData>
        </Output>
</Component>
```

**Figure 1: semantic specification of the Input/Output data type for the audio analysis tool**

## 4. CHAINING ALGORITHM

We propose a recursive algorithm that allows the construction of a composite tool. Based on a given requested index (which is actually a descriptor asked by the user) the chaining algorithm must find a set of tools and their interactions to detect or recognize each occurrence related to this descriptor. The algorithm is recursive and it stops when a primitive multimedia content (audio, videos files) can be used as an input of the last integrated indexer.

We start with a given "*Index*" and we try to find the unitary indexer able to generate this descriptor type. Once such a tool is found, if the input type of this indexer is a multimedia content, we stop and the given index can be generated by application to all documents corresponding to the input data type. In the other case, if the input type of this indexer is another descriptor type corresponding to data generated by another tool, we repeat the same process recursively, integrating each time a new indexer at the top of the chain, until a multimedia content type is found. The following program is a pseudocode representation of the recursive chaining algorithm

```
Construct_composite_tool (in index_type, in out composite_tool)
{
  Search (in index_type, out indexer_tool_ref, out in_data_type, out
  in_data_ref)
  composite_tool.add_tool_ref(in indexer_tool_ref)
  composite_tool.add_data_source_ref(in data_source_ref)
  if        data_source_type == multimedia_contents_type
                    Return;
  Construct_composite_tool(in out data_source_type, in out
  composite_tool)
}
```

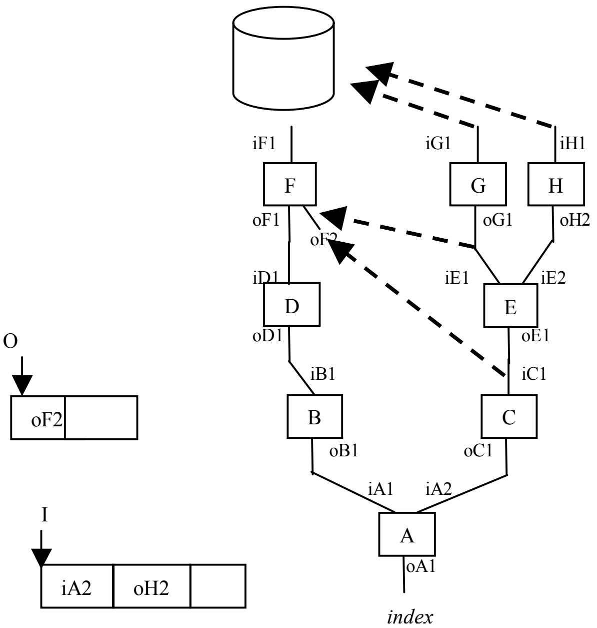


**Figure 2: Execution tree constructed by the chaining algorithm**

The process is simple in the case of linear chains, i.e. there is only one input and one output for each included indexer, but it will be much more complicated if the involved tools require several data inputs. In this case, the chains will take the aspect of a tree and then we have to apply the recursive algorithm to each branch of this tree with the assumption that the tree is devoted to index only one source (i.e. one document at a time). In other words, the tree must have one and only one node at the top.

In the following example, we explain the construction of the tree (Figure 2) by applying the chaining algorithm to generate a given index. We use two FIFO queues: *I* is a list of input requirements that are not matched by the chaining processes; *O* is a list of the output types generated by indexers and which are still not used in the tree.

Given an index, the dynamic indexing service asks the discovery service for a tool able to generate the desired *index*, the discovery service finds the tool *A* which requires *iA1* and *iA2* as Input type, we try to apply the recursive algorithm to the first type *iA1* and we store the second type *iA2* in the *I* queue.
*iA1* is generated by *B* which takes *iB1* as input, *iB1* will be generated by *D* and *iD1* by *F*, but *F* generates another type of description *oF2* that we store it in queue of the available output types O, so we can use it as input for another branch of the tree. We stop with the tool *F* because *iF1* is a multimedia content type.
As far as the *I* queue is not empty, we will apply the same process to *iA2* while trying to converge towards one of the element stored in the *O* queue, or towards the multimedia contents i.e. *iF1*.

## 5. CONCLUSION AND FUTURE WORKS

In this paper, we have introduced the main concepts of a dynamic multimedia indexing platform, which supports a dynamic policy of multimedia indexation; we have presented a global architecture of the platform, and the services that provides the communication mechanisms. We have proposed a recursive algorithm for dynamically chaining indexing tools, based on the compatibility between input and output data types of available tools. We intend to apply this technology for the implementation of the Pidot project, which is a transmedia open distributed indexing platform. Among expected results, we want to identify the best chain between different concurrent solutions to produce one given descriptor or, with the integration of generic aggregation tools, to automatically mine chains which can produce new descriptors.
More research efforts have to be achieved in the future to solve problems linked to the dynamic indexing approach. The first problem is the consistence of the generated chain. For example, in the case of automatic transcription of French spoken contents, the algorithm will allows to build the chain able to generate a textual description starting from an audio file (wav), even when some of these audio files contain only spoken English. In this case, time to time, considering the language used in the document, the chain will not generate any result. So we have to define mechanisms able to inform of this inconsistency regarding to the contents and to stop the execution of the chain in that case.
The second problem is the choice between different tools that have the same output data type. For the moment the only criteria to select an indexing tool is its output data type. But we can imagine that more than one tool may be able to produce the same output type. So we should define some other criteria to make the best choice.
The third problem consists in several possible chains which can be determined to produce a same given index. Here again, some criteria have to be designed to make the best choice.
Finally we have to define the chain updating mechanism which allows to dynamically change the chain after the integration or the removal of an indexer.

## 6. REFERENCES


[1] V. Conan, I. Ferrané, P. Joly, C. Vasserot. KLIMT: Intermediations Technologies and Multimedia Indexing. In Third International Workshop on Content-Based Multimedia Indexing (CBMI'03), Rennes, France, 22 septembre 24 septembre 2003. INRIA, p. 11-18.

[2] AGIR : Architecture Globale pour l'Indexation et la Recherche par le contenu de données multimédia. http://www.ina.fr/recherche/projets/finis/agir.fr.html.

[3] M.Windhouwer, A.Schmidt, M.Kersten. Acoi: A System for Indexing Multimedia Objects. In International Workshop on Information Integration and Web-based Applications & Services, Yogyakarta, Indonesia, November 1999.

[4] Business Process Execution Languages for Web Services, Version 1.0. IBM Report. http://www.ibm.com/developerworks/library/ws-bpel/.

[5] WSCI: A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawagushi, D. Orchard, S. Poligliani, K. Riemer, S. Struble, P. Takacsi-Nagy, I. Trickovic, and S. Zimeck. Web Service Choreography Interface 1.0.

[6] J. Koehler, B. Srivastava: Web Service Composition: Current Solutions and Open Problems, ICAPS 2003 Workshop on Planning for Web Services, pages 28 - 35.

[7] Mandell, D. and McIlraith, S. A Bottom-Up Approach to Automating Web Service Discovery, Customization, and Semantic Translation. In the Proceedings of the Twelfth International World Wide Web Conference Workshop on E-Services and the Semantic Web (ESSW'03). Budapest, 2003.

[8] Fabio Casati, Ski Ilnicki, Li-jie Jin, Ming-Chien Shan: An Open, Flexible, and Configurable System for Service Composition. WECWIS 2000: 125-132

[9] H.J. Batteram and H.P. Idzenga, A Generic Software Component Framework for Distributed Communication Architectures, ECSCW 97 Workshop on Object Oriented Groupware Platforms, 1997, Lancaster UK, pp. 68-73

[10] J. Pinquier, J. L. Rouas, R. André-Obrecht. Robust speech/music classification in audio documents. In International Conference on Spoken Language Processing (ICSLP'2002), Denver, Etats-Unis, 16 septembre 20 septembre2002 . Causal Productions Pty Ltd, www.causal.on.net, p.2005 -2008, Vol.3.