User-designed web services to support heterogeneous biological data retrieval

Marie-Dominique Devignes[1], Hervé de Palma[1], Laurent Pierron[1], Lionel Domenjoud[2] and Malika Smaïl-Tabbone[1]

[1]LORIA, CNRS-INRIA-Université Henri Poincaré, and [2] UPRES EA3446, Université Henri Poincaré, BP239, 54506 Vandoeuvre-les-Nancy

Abstract

Biological data retrieval from web resources often necessitates multi-step access to several information sources. Web services are being developed to interact in a unified manner with heterogeneous databases. Composition of web services is then required to chain several elementary queries. However, web services are not yet available for any data sources thus limiting the scope of such strategy. The X-collect generic application has been designed to execute user-defined multi-step scenarios. Demos can be scheduled at the workshop. Encapsulation into web services makes such user scenarios re-usable and available for invocation by any application. Work in progress deals with metadata registries to facilitate databases and/or web services discovery.

Keywords

Heterogeneous databases querying, user-defined scenario, web services, XML

1. Introduction

Hundreds of biological databases are made available through web interfaces. Answering a biological question often necessitates querying or browsing through several web resources.

Interesting solutions are grounded on a unified data model that allows integration of partial answers retrieved from distinct databases. Data warehouses (GUS), database federations (DISCOVERYLINK, SEMEDA) or mediation architectures (TAMBIS, TINet) have thus demonstrated their efficiency albeit in relation to a relatively small number of databases. Indeed the selection of appropriate databases for a given query has no obvious solution. The biologists most often have very personal opinion about the information sources to query for their problem (discussed in Lord et al., 2004) and may not appreciate being limited to the small number of databases offered by these integration systems.

Biological data retrieval may become repetitive when it concerns lists of input items and/or requires frequent updating. In such situations, users try to systematize their interactions with the databases in terms of querying scenarios/workflows, in view of automating and speeding up the query process. Many systems available so far in the bioinformatic domain have been designed to facilitate chaining of computing tasks to achieve a bioinformatic analysis (e.g. HUSAR W2H-W3H, PISE, Bionavigator, Helmholtz Network for Bioinformatics, Biopipe, Pegasys, etc.). Recently, attempts to define syntactic components and algebraic operators capable of representing analytical workflow have been reported (Garcia Castro et al., 2005). Analytical workflow design in bioinformatics has been illustrated in the domain of web services composition (Oinn et al., 2004) and has implied the definition of a new language to express workflows (Scufl for “simple conceptual unified flow language”). However these solutions do not yet satisfy all the users’ needs. The first limit is that only the most popular programs are available through these systems. When some new program becomes necessary, users need to wait until it is interfaced with the system or some web service has become available for it. The second limit is that a certain type of tasks is rarely addressed in these

systems, namely the task of retrieving data from heterogeneous web resources (Butler et al., 2002).

We have been dealing with bioinformatic data retrieval scenarios for several years (Devignes et al., 2002, 2004). A single example will be detailed below for sake of clarity to illustrate our proposition. It consists in retrieving heterogeneous annotations to characterize a DNA fragment that has been cloned and sequenced in the frame of a given wet-lab procedure. Retrieving such information is a frequent need in biology and does not imply complex analysis. When performed manually (see section 2), it simply consists in a series of databases requests or programs invocations, mostly performed through web interfaces and accompanied by copying and pasting relevant data to be stored in word processor or spreadsheet files. However, the number of clicks required makes the procedure tedious when it has to be repeated and/or updated. In addition, tracking and managing the stored data is time consuming when no appropriate format is used. The Xcollect application has therefore been designed (section 3) to automate the enactment of user-designed scenarios and store the collected data in a structured (XML) format. To enable unlimited choice of databases, the flexible although fragile screen-scraping techniques have been used. User-designed scenarios are stored as XML documents according to a generic model and can easily be modified to keep in phase with frequent changing of biological databases (web interfaces, query capabilities, creation of a new resource, etc.). To ensure re-usability and possible composition, any instance of the Xcollect application, corresponding to a given scenario, can be encapsulated in a web service (section 4). A UDDI-like Xcollect-WS registry has been developed to store information about the Xcollect web services and enable their discovery. A web-service browser inspired from the Moby-S Web Service Browser[1] has been set up to enable testing of the scenarios (http://bioinfo.loria.fr/Members/devignes/XCOLLECT /WSbrowser). Discussion (section 5) will analyze the advantages and the limits of our approach in the perspective of workflow design.

2. Multi-step biological data retrieval: one example

Biologists involved in isolating and sequencing particular DNA fragments are often interested in characterizing the sequences they obtained on the basis of public resources annotations. The Xprom scenario has been designed to retrieve such annotations for short immuno-selected DNA fragments that have been sequenced to study regulatory elements in human genome (Collet et al., 2004). The annotations to retrieve for each fragment sequence are: (i) how many copies of a 6-base long motif of interest are contained in the fragment, (ii) does the fragment contain known repeated sequences and (iii) where (i.e. in which genomic context) does it map on the human genome. Examples of manually retrieved data are shown in Table 1.

Table 1: Manually retrieved annotations for two immuno-selected DNA fragments

Annotations :	(i) AGGTCA motif	(ii) Repeats	(iii) Genomic context
Source / Program :	PpreFinder[2]	RepeatMasker[3]	Genome Browser[4]
Fragment 1034 (length :150)	2 motifs	No repeat (RepeatMasker,v 1.165 2005/05/04)	chr10:131,063,511-131,063,660 (Assembly : May 2004)
Fragment 1251 (length : 207)	3 motifs	No repeat (RepeatMasker,v 1.165 2005/05/04)	chr20:803,928-804,134 (Assembly : May 2004)

[1] http://mobycentral.icapture.ubc.ca/cgi-bin/gbrowse_moby

[2] Home written java program, finds a specified motif in the entry sequence, with a certain number of tolerated mismatches, here only 1 mismatch.

[3] http://www.repeatmasker.org/

[4] http://genome.ucsc.edu/cgi-bin/hgGateway?org=Human&db=hg17

In this example, user expertise has guided the selection of appropriate sources and program parameters. Some metadata (last update or version) about the queried databases and programs are to be retrieved in order to document the quality of the retrieved data and to propose when needed ranking of homologous answers pertaining from different databases (Devignes et al., 2004).

3. Xcollect application

3.1 Principles

The general philosophy of the Xcollect application is based on two models schematized in Figure 1. The **generic scenario model** (Figure 1A) aims at describing all information required to interact with databases or programs and retrieve selected items. For sake of clarity, sets of elementary steps can be grouped in a common meaningful **step**. The **generic session model** (Figure 1B) has been designed in the absence of any standard model for complex biological data. It simply follows the scenario model and describes the steps of the scenario with their respective input and output data. For each built query, the corresponding link is stored to describe the concerned elementary step and to allow tracking and consultation of the result web page. Both models have been implemented as XML DTDs.

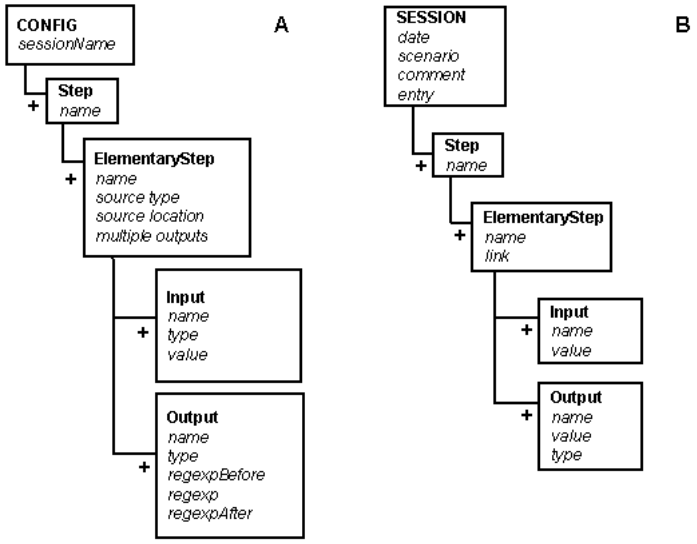
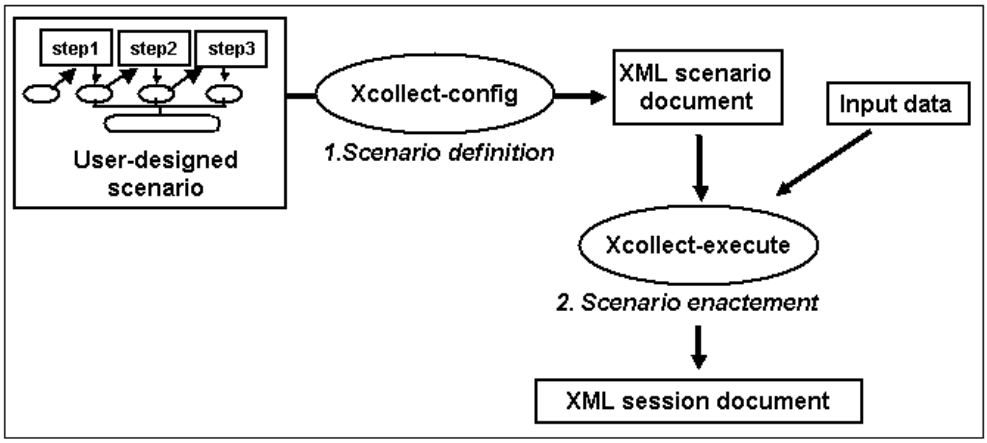


Figure 1 : Models for the Xcollect application.
A: generic scenario model.
B: generic session model.
In the corresponding XML DTDs, bold and italics items will correspond respectively to elements and attributes.

The Xcollect java application is composed of two modules that can be invoked according to the following scheme (Figure 2).

Figure 2 : Schematic architecture of the Xcollect appplication



The configuration module (Xcollect-config) allows users entering interactively through a dedicated

interface all the information specifying their scenarios or modifying existing scenarios. Entered data are stored into an XML scenario document according to the generic scenario_DTD. The enactment module (Xcollect-execute) then takes as inputs the XML scenario document and a set of input data. It executes each step of the scenario and returns an XML session document containing the retrieved data structured according to the generic session_DTD.

3.2. The configuration module

A user-friendly interface has been developed for users to design and enter their scenarios (Figure 3). Linked to the execution module it allows rapid testing of queries and filtering specifications.

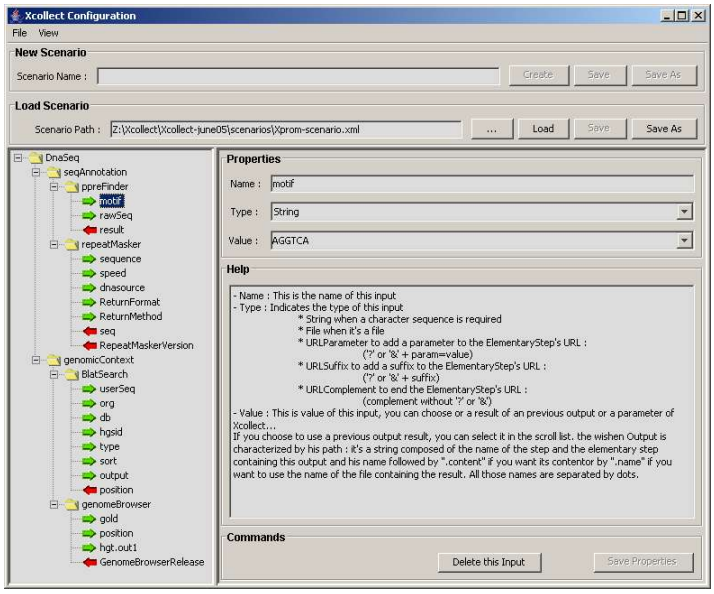


Figure 3: Xcollect configuration interface. (Xprom scenario loaded)

In the left panel are displayed all the elements of the loaded scenario. In the right panel specific windows are proposed to modify in a controlled manner any selected element and its attributes.

Each **elementary step** refers to a single query and is identified by the type of source or program queried and its location. Query construction at this step of the scenario is specified by a list of **input** elements, each of them described by its type (with respect to the Xcollect application), a formal name (user-given) and the value it should take during scenario enactment. Concerning this value, two particular cases have been taken into account. First, the entry data to be provided with the scenario to the Xcollect enactment module necessarily appears among the input elements described in one or more steps of the scenario. Its value in the scenario is thus set to a pre-defined expression (namely *XcollectEntryParameter*). Second, the chaining of the steps means that an input value for a given step can be specified as the output result of a preceding step. This reveals particularly useful to manage hyperlink indirections in the returned HTML pages.

Filtering relevant data from the query result (usually an HTML page) is finally specified through a series of **output** elements, each of them described by its type, a formal name (user-given), and most importantly the patterns necessary for filtering (*e.g.* regular expressions). When the filtering process has to be repeated on a given page to retrieve all possible data fitting to the specified patterns, the boolean attribute "multiple outputs" of the elementary step has to be switched to *yes*.

In many cases, output data requires some processing before being used as input for the next retrieval step. The generic model accounts with this requirement by accepting as a source any available java program capable of executing the processing task. The processing step is simply considered as an additional elementary step in the scenario.

Description of each elementary step of the Xprom scenario mentioned above (section 2) through the Xcollect configuration interface has lead to the automatic production of the Xprom-scenario XML document presented in Figure 4.

Figure 4 : Example of scenario : Xprom-scenario.xml

```
<?xml version="1.0"?>
<!DOCTYPE CONFIG SYSTEM "Xcollect_scenario.dtd">
                <!--Xprom scenario created by MD Devignes-->
                <!--          august 30, 2005           -->
<!--Description : XcollectEntryParameter = personal sequence provided as a FASTA file  -->
<!--        Outputs are (1) PpreFinder results, (2) RepeatMasker results,              -->
<!--           (3) genomic context taken from Genome Browser                           -->

<CONFIG sessionName="DnaSeq">
  <Step name="seqAnnotation">
    <ElementaryStep name="ppreFinder" type="Class" location="PpreFinder.class" multipleOutputs="false">
      <Input name="motif" type="String" value="AGGTCA"/>
      <Input name="rawSeq" type="File" value="XcollectEntryParameter"/>
      <Output name="result" type="String" regexp_before="" regexp="" regexp_after=""/>
    </ElementaryStep>
    <ElementaryStep name="repeatMasker" type="URL"
      location="http://www.repeatmasker.org/cgi-bin/WEBRepeatMasker" multipleOutputs="false">
      <Input name="sequence" type="URLParameter" value="XcollectEntryParameter"/>
      <Input name="speed" type="URLParameter" value="default"/>
      <Input name="dnasource" type="URLParameter" value="human"/>
      <Input name="ReturnFormat" type="URLParameter" value="html"/>
      <Input name="ReturnMethod" type="URLParameter" value="html"/>
      <Output name="seq" type="String"
        regexp_before="&lt;h2&gt;Repeat Annotations:&lt;/h2&gt;&lt;PRE&gt;"
        regexp="" regexp_after="&lt;/PRE&gt;"/>
      <Output name="RepeatMaskerVersion" type="String"
        regexp_before="RepeatMasker version development" regexp="RepeatMasker,v.*" regexp_after="rhubley Exp"/>
    </ElementaryStep>
  </Step>
  <Step name="genomicContext">
    <ElementaryStep name="IndexHgsid" type="URL"
      location="http://genome-test.cse.ucsc.edu/cgi-bin/hgGateway?org=Human&amp;db=hg17"
      multipleOutputs="false">
      <Input name="org" type="URLParameter" value="human"/>
      <Input name="db" type="URLParameter" value="hg17"/>
      <Output name="hgsid" type="String" regexp_before="hgsid=" regexp="[0-9]+"
        regexp_after="&quot; class=&quot;topbar&quot;"/>
    </ElementaryStep>
    <ElementaryStep name="BlatSearch" type="URL"
      location="http://genome.ucsc.edu/cgi-bin/hgBlat" multipleOutputs="false">
      <Input name="userSeq" type="URLParameter" value="XcollectEntryParameter"/>
      <Input name="org" type="URLParameter" value="human"/>
      <Input name="db" type="URLParameter" value="hg17"/>
      <Input name="hgsid" type="URLParameter" value="genomicContext.IndexHgsid.hgsid"/>
      <Input name="type" type="URLParameter" value="DNA"/>
      <Input name="sort" type="URLParameter" value="query,score"/>
      <Input name="output" type="URLParameter" value="hyperlink"/>
      <Output name="position" type="String" regexp_before="hgTracks\?position=" regexp="chr.*"
        regexp_after="&amp;db=hg17"/>
    </ElementaryStep>
    <ElementaryStep name="genomeBrowser" type="URL"
      location="http://genome.ucsc.edu/cgi-bin/hgTracks" multipleOutputs="false">
      <Input name="gold" type="URLParameter" value="full"/>
      <Input name="position" type="URLParameter" value="genomicContext.BlatSearch.position"/>
      <Input name="hgt.out1" type="URLParameter" value="10x"/>
      <Output name="GenomeBrowserRelease" type="String"
        regexp_before="&lt;FONT SIZE=5&gt;&lt;B&gt;" regexp=""
        regexp_after="&lt;/B&gt;"/>
    </ElementaryStep>
  </Step>
</CONFIG>
```

3.3 Enactement of Xcollect scenarios

Three arguments have to be specified : first the scenario to be used, second a comment line to customize the resulting XML session document, third the input data. Input data either correspond to a single value of Xcollect entry parameter or to a file containing a series of such values. When Xcollect entry parameter is itself a file (for example a sequence file), a series of files can be provided as multiple entries in a folder. When multiple entries are provided, the Xcollect enactment module will generate as many session documents as entry values.
The Xprom scenario created for our case-study has been launched for the two sequences taken as examples above. Session documents converted to HTML files via a generic XSL transformation can be viewed on our Xcollect web pages[5,6]. Depending on the desired usage of the data, appropriate XSL transformations should allow easy conversion of the generic XML representation of retrieved data into more biologically expressive models or formats.

4. Web service deployment

Each instantiation of the Xcollect application can give birth to a re-usable scenario. Indeed small linear scenarios can be useful as parts of more complex workflows involving control structures. Such structures have not been taken into account in the Xcollect application in order to keep our generic model very simple. Updating the retrieved data also implies launching the Xcollect enactment module from time to time. To facilitate both machine and user interactions, we decided to encapsulate the Xcollect instantiated applications into web services. An environment has been set up to automate the deployment of any new instance of Xcollect application as a web service[7]. It involves the construction of a client-server architecture including a web services registry and a browser. In practice, a single file has to be defined to deploy a new Xcollect-based web service. Five properties have to be specified:

- the display name,
- the Xcollect scenario to be used,
- the type of output (at present XML file),
- the type of input (for example a sequence file),
- and the description of the service.

Naming of the Xcollect web services uses the scenario names: for example, the Xprom scenario described here has lead to the creation of a web service called Xprom-ws.
The web services registry can be queried by the client to identify relevant web services. Up to now no extension has been added to the native registry[8] to improve this task in contrast to other more advanced systems such as Taverna (Oinn et al., 2004). Invocation of a web service by the client involves on the server side a call to the Xcollect enactment module. The browser is inspired from the Moby-S web service browser and can be used for interactive selection and launching of a given Xcollect web service.

5. Discussion and perspectives

The Xcollect application presented in this paper was developed to answer the biologist needs of very flexible solutions to interact in a sequential manner with several distributed databases and programs. Advantages of the Xcollect solution can be described in terms of openness: many web resources can be included in a scenario, and user-friendliness: thanks to the configuration interface, writing and modifying a scenario do not require advanced computing skills. Limits of our solution are consequences of the model simplicity: only small linear scenarios can be defined, and of the filtering method which involves parsing of changing web pages. This last inconveniency requires users checking their scenario regularly but the user-

[5] http://bioinfo.loria.fr/Members/devignes/XCOLLECT/Xprom-1034.html
[6] http://bioinfo.loria.fr/Members/devignes/XCOLLECT/Xprom-1251.html
[7] java Web Service Development Pack : WSDP
[8] WSDP Registry Server

friendly configuration interface allows them modifying themselves the query parameters or filtering options if necessary.
To circumvent the first limit, Xcollect scenarios have been encapsulated into web services allowing further orchestration. Experimentation is underway to check how the Xcollect web services can be composed with other services in the frame of high order workflow design.
The Xcollect web service architecture allows rapid deployment of dedicated customized web services. Identification of relevant web services, which rely on ontology-aware descriptions of these services, represents a future direction that we have started to address with our BioRegistry project[9] (Messai et al., 2004; Smaïl-Tabbone et al., 2005).

6. References

Buttler D, Coleman M, Critchlow T, Fileto R, Han W, Liu L, Pu C, Rocco D and Xiong L (2005) *Querying Multiple Bioinformatics Data Sources: Can Semantic Web Research Help?* SIGMOD Record, Vol. 31, No. 4.

Collet P, Domenjoud L, Murad H, Devignes MD, Schohn H and Dauça M (2004) *The Human Semaphorin 6B gene is down-regulated by PPARs*. Genomics 83: 1141-50.

Devignes MD, Schaaff A and Smaïl M (2002). *Collecte et intégration de données biologiques hétérogènes sur le Web – Xmap : application dans le domaine de la cartographie du génome humain.* Revue des sciences et technologies de l'information (RSTI) – Série Ingéniérie des systèmes d'information (ISI) 7 : 45-61.

Devignes MD and Smaïl M (2004) *Integration of Biological Data From Web Resources : Management of Multiple Answers Through Metadata Retrieval.* Short paper, ISMB-ECCB, Glasgow, july 31-august 4, 2004.

Garcia Castro A., Thoraval S. Garcia L.J. and Ragan M.A. (2005) *Workflows in bioinformatics: meta-analysis and prototype implementation of a workflow generator.* BMC Bioinformatics 6:87.

Lord P, Bechhofer S, Wilkinson MD, Schiltz G, Gessler D, Hull D, Goble C and Stein L (2004) *Applying semantic web services to bioinformatics: experience gained, lessons learnt.* International Semantic Web Conference ISWC'04, Hiroshima, November 2004.

Messai N, Devignes MD, Napoli A and Smaïl-Tabbone M (2005) *Querying a bioinformatic data sources registry with concept lattices.* 13th International Conference on Conceptual Structures, Kassel, July 18-22, 2005.

Oinn T, Addis M, Ferris J, Marvin D, Senger M, Greenwood M, Carver T, Glover K, Pocock MR, Wipat A, and Li P (2004) *Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics* 20:3045-3054.

Smail-Tabbone M, Osman S, Messai N, Napoli A and Devignes MD (2005*) BioRegistry: a structured metadata repository for bioinformatic databases.* 1st International Symposium on Computational Life Science, Constance, September 25-27, 2005. Accepted paper.

[9]http://bioinfo.loria.fr/Members/devignes/Bioregistry/