Concept and Implementation of a Service API for the Remote Access to Life Cycle Data Assessment Databases

Clemens Düpmeier¹, Oliver Kusche¹

Abstract

For implementation of political concepts like the Integrated Product Policy (IPP) of the European Union, which aims at a “sustainable society”, it is inevitable to widely apply in industry methods of Life Cycle Assessment (LCA), which analyze the whole life cycle of products with respect to sustainability topics like resource usage and other environmental impacts. For the efficient and reliable application of LCA, it is necessary that LCA studies are based on consistent, transparent, and quality-assured background data. But LCA modellers often face the problem of needing datasets from different LCA databases or data sources for a study. And these datasets typically have different formats and cannot be easily used together because their nomenclature and references to common elements are not harmonized. This results in tedious manual work which has to be performed by the modellers to merge such datasets into an already existing internal database within their LCA tool.

The open source project SODA4LCA led by the Institute of Applied Computer Science at the Karlsruhe Institute of Technology in cooperation with other partners tries to solve such problems of dataset providers and modellers through concepts and corresponding implementations for building up service-oriented infrastructures for LCA on the Internet. The SODA4LCA LCA database application as one of the main outcomes of the project implements a service-oriented API which allows it to build up networks of LCA database nodes maintained by different data providers. Tools used by modellers can connect to these data networks and query, retrieve and even upload data in an easy and consistent way. The SODA4LCA service API and its applicability will be described in more detail in this paper.

1. Introduction

For implementation of political concepts like the Integrated Product Policy (IPP) of the European Union, which aims at a “sustainable society”, it is inevitable to widely apply in industry methods of Life Cycle Assessment (LCA), which analyze the whole life cycle of products with respect to sustainability topics like resource usage and other environmental impacts. For the efficient and reliable application of LCA, it is necessary that LCA studies are based on consistent, transparent, and quality-assured background data. Because of this, several activities in Europe and worldwide have been established, which are aimed at providing national and international databases and supplies of basic Life Cycle Inventory (LCI) data. Examples in Europe are the Swiss Ecoinvent database (Ecoinvent 2011) and the European Platform on LCA (Europe 2011b) for a Europe-wide data supply.

LCA modellers often face the problem of needing datasets from different LCA databases or data sources for a study. These datasets typically have different formats and cannot be easily used together because their nomenclature and references to common elements are not harmonized. This results in tedious manual work which has to be performed by the modeller to merge such foreign datasets into an already existing internal database within his LCA tool.

¹ Karlsruhe Institute of Technology, Institute for Applied Computer Science, Hermann-von-Helmholtz-Platz 1, 76021 Eggenstein-Leopoldshafen (Germany); email: {Clemens.Duepmeier, Oliver.Kusche}@kit.edu
The service-oriented SODA4LCA (Service-Oriented Database Application for LCA) implemented by the Institute of Applied Computer Science (IAI) at the Karlsruhe Institute of Technology (KIT) for the International Reference Life Cycle Data System (ILCD) Data Network (Europe 2011b) which has been established by the European Commission tries to solve these problems by applying modern Internet concepts. It provides an Internet-accessible service-oriented API (the SODA4LCA service API) which will be described in more detail in the following chapters and is based on a RESTful service approach allowing for deeply linking different physical instances of the database software spread across the Internet to form a harmonized network of LCA databases (see Figure 1) which can be accessed by tools.

Within such a network, the datasets provided by different providers can reference through hyperlinks common background datasets and nomenclature standardized by the corresponding network community and provided by one or more central systems within the network. Such an open linked data approach has many advantages.

Users can find and retrieve datasets using an arbitrary node of the network which can provide access to all the datasets within the network from one place. But also tools can use the service API to provide users access to the whole data network right from within the tool. These usages of the service API will be described in a later chapter of this paper in greater detail.

Because a RESTful service defines a computer language- and operating system-neutral API based on Internet-conform HTTP calls and return documents in language-independent formats, like XML, clients which use the SODA4LCA service API can be written for any programming environment providing HTTP and XML functionality. KIT itself also provides an easy-to-use Java class library for remote access to database systems which implement the SODA4LCA API. This library wraps the RESTful service API and automatically marshals and un-marshals XML documents used by the RESTful API into corresponding Java objects and vice versa. This Java module is already used for several projects as will be described later. Such wrapper modules can also be written easily for other programming environments like .NET using the same approach as the Java module given and providing the same easy interfaces to all tools which use .NET as programming technology.
2. The Soda4LCA Service API

Representational State Transfer (REST) is an architectural style for service-oriented applications on the Internet which put the Web principles back into web services. The term REST was first introduced by Roy Thomas Fielding in Chapter 5 of his PhD thesis “Architectural Styles and the Design of Network-based Software Architectures” (Fielding 2000). In his thesis, Fielding takes a look at the parts of the World Wide Web that work very well and extracts design principles that could make any other distributed hypermedia system – whether related to the Web or not – as efficient. To design a RESTful service, you only need to know the Hypertext Transfer Protocol (HTTP), Uniform Resource Identifiers (URIs), and follow a few RESTful design principles.

In a RESTful distributed service architecture, every service is considered to provide representations of arbitrary abstractions of real-life objects called resources. In the LCA application domain, adequate resources and resource types are LCI data sets, like process, LCIA method, product, waste and elementary flow data sets, physical and chemical flow properties, corresponding units or extended product declarations and eco-labels. In the case of the service-oriented SODA4LCA database application, such LCA resources are actually stored and managed in the database system and can be uploaded, queried and deleted via the SODA4LCA RESTful service interface. But resources need not necessarily be stored as pure information objects in a RESTful service. There, representations can also be dynamically generated at runtime. I.e. this would be the case in a RESTful LCA processing service which calculates some LCA quantities at runtime.

Each resource in a RESTful service is uniquely identified by a URI (Unique Resource Identifier) as main identifier which can be dereferenced in an HTTP call to request some kind of action to be performed by the service on the resource identified by the URI.

Table 1 lists some URI patterns used by the SODA4LCA API and the types of resources addressed by them. <baseUri> should be replaced by an arbitrary URI prefix, like “http://lca-data.org/resource”.

<table>
<thead>
<tr>
<th>URI of resource</th>
<th>Type of resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;baseUri&gt;/processes</td>
<td>list of processes contained in the database</td>
</tr>
<tr>
<td>&lt;baseUri&gt;/processes/{uuid}</td>
<td>unique process resource with identifier uuid</td>
</tr>
<tr>
<td>&lt;baseUri&gt;/flows</td>
<td>list of flow resources in the database</td>
</tr>
<tr>
<td>&lt;baseUri&gt;/flows/{uuid}</td>
<td>unique flow resource with UUID uuid</td>
</tr>
<tr>
<td>...</td>
<td>similar URI patterns for other data set types</td>
</tr>
</tbody>
</table>

Table 1: Different types of URIs used in the SODA4LCA API.

The same pattern can also be used with other data set types like flow properties, unit group, source, contact and method data sets.

The HTTP action verbs GET, PUT, POST or DELETE in a RESTful service call determine the kind of action requested by the caller on the resource. Clearly, a GET request should return some kind of representation of the dereferenced resource to the caller. POST and PUT requests should add or modify and DELETE requests should delete some resources from the server. The following table list the main actions which can be triggered on an LCA database system implementing the SODA4LCA API.

<table>
<thead>
<tr>
<th>Action</th>
<th>URI scheme</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>&lt;baseUri&gt;/processes?params</td>
<td>Retrieves a list of metadata descriptions of processes, which optionally match search criteria specified as URI parameters.</td>
</tr>
<tr>
<td>GET</td>
<td>&lt;baseUri&gt;/processes/{uuid}</td>
<td>Retrieves a representation of a single data set.</td>
</tr>
</tbody>
</table>
set. Can be some form of HTML representation or retrieves the data set in an XML data exchange format. If only the id of the data set is given, the latest available version will be provided. A version number can be specified via an URI parameter in which case the specified version with the given id is returned.

<table>
<thead>
<tr>
<th>Method</th>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>&lt;baseUri&gt;/processes</td>
<td>Creates a new data set or a new version of a data set within the processes resource space (if the client has proper access rights).</td>
</tr>
<tr>
<td>PUT</td>
<td>&lt;baseUri&gt;/process/{uuid}</td>
<td>Updates the data set with the given UUID (if the client has proper access rights).</td>
</tr>
<tr>
<td>DELETE</td>
<td>&lt;baseUri&gt;/process/{uuid}</td>
<td>Deletes a data set from the database (if the client has proper access rights).</td>
</tr>
</tbody>
</table>

Table 2: Different actions of the SODA4LCA API.

The first form of the GET request given in Table 2 represents a query that returns a list of metadata descriptions (called list view) which contain the most important metadata about each data set listed to the caller. If no parameters are given, the list will contain such a list view description for every data set of the specified type. This list can be shortened by specifying query parameters which will filter the data set list according to the restrictions defined by the query parameter values. The API defines some data set type-independent, overall query parameters, like name, type or category of data set, and other data set type-dependent parameters, like location, reference year or the valid until date for process data sets. A complete list of query parameters and their possible values are defined in the SODA4LCA API specification document which can be downloaded from the web site of the SODA4LCA open-source project (SODA4LCA 2011).

The following code shows the typical result of a query for data sets of type processes.

```xml
<dataSetList totalSize="317" startIndex="0" pageSize="100"
   xmlns="http://www.ilcd-network.org/ILCD/ServiceAPI"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:process="http://www.ilcd-network.org/ILCD/ServiceAPI/Process">
   <p:process sourceId="SODA@KIT.EDU"
      xlink:href="http://lca-data.org/resource/process/00043bd2-4563-4d73-8df8-b84b5d8902fc">
     <uuid>00043bd2-4563-4d73-8df8-b84b5d8902fc</uuid>
     <permanentUri>
        http://lca-data.org/resource/process/00043bd2-4563-4d73-8df8-b84b5d8902fc</permanentUri>
     </permanentUri>
     <dataSetVersion>02.01.000</dataSetVersion>
     <name xml:lang="en">Electricity Mix;AC;consumption mix, at consumer;230-240V</name>
     <classification name="ilcd">
       <class level="0">Energy carriers and technologies</class>
     </classification>
     <p:type>LCI result</p:type>
   </p:process>
</dataSetList>
```

Copyright 2011 Shaker Verlag Aachen, ISBN: 978-3-8440-0451-9
A `<dataSetList>` XML tag with information about the total number of search results, the number of returned results, and the index of the first result returned wraps a list of process descriptions contained in `<p:process>...</p:process>` tags. Each list process description contains only the most important metadata about a process to keep the size of such a returned search result list small. The list process description contains information like the id (UUID) of the process, its type and its version number, the name of the process, its classification, location, time information, and if the process is parameterized or contains LCIA results. Some additional information about the method approach used for process modeling and the allocation approaches can be returned in the list overview of a process, too.

One really important design principle of RESTful services is hyperlinking. Resource descriptions like the above metadata description of processes should link to other more detailed descriptions in the same way as HTML pages link to other HTML pages. Hyperlinking is also used in the SODA4LCA RESTful service API. I.e. the process description in the shown metadata process list contains for each process an `xlink:href` attribute in the `<p:process>` tag which defines the process-specific service URL to retrieve a more detailed metadata description of this single process or even to retrieve the whole process in a reasonable data exchange format. These process-specific URLs correspond to the second GET request pattern in Table 2 which uniquely identifies a single process by specifying its data set id i.e. a UUID in our example code. Such single data set service URLs in the SODA4LCA API can have additional parameters “view” and “format” that determine which kind of representation of the process the service should return to the user.

The “view” parameter can have the value “overview” or “full” (this is the default). If the value of this parameter is “overview”, then the service will only return a more detailed metadata overview description of the data set. This more detailed description contains additional XML tags which specify further information about the data set. I.e. for a process, its detailed textual description, the technical purpose or the quantitative reference will be returned. The overview format may also contain information about ownership, license restrictions or review and compliance information for a process. If the view parameter has the value “full” which is the default then instead of only metadata, the full process data set including the exchange and result lists with all flow and result values will be returned in ILCD format. This representation can directly be imported into LCA modeling tools which implement an ILCD data set import.

While GET requests retrieve data set descriptions, POST requests on URLs specifying a certain type of data set will result in the creation of new processes or new versions of processes if data sets are attached to the request as payload. A POST request from the client to the server must contain one or more data sets in
ILCD format as payload. These data sets are then imported into the database. In the case of a single process, the body of the POST request can directly be an XML data set in ILCD format. The service will then analyze the uploaded data set and determine if a new or a new version of an existing data set should be created. The body of the POST request can also be of type “multipart/form-data”. In that case, the service will process all file attachments conforming to the ILCD format in the payload of the POST request to import them into the database. PUT in contrast to POST requests are defined to overwrite existing data sets within a SODA4LCA database service. Thus, a data set with the id of the uploaded data set should already exist in the database. Otherwise, PUT requests work like POST requests. Finally, DELETE requests will delete the data set uniquely identified by the given service URL.

2.1 Authentication and Authorization

It should be clear that an LCA database system should not allow POST, PUT and DELETE requests on data sets without any kind of authorization. The RESTful service design principles do not define any REST-specific authentication and authorization concepts. Instead, REST services delegate authentication and authorization aspects completely to the Web layer. Because of this, the SODA4LCA API does not mandate a specific authentication and authorization mechanism but recommends the usage of standard HTTP authentication mechanisms, like HTTP basic or, more securely, digest authentication. This should be combined with the usage of HTTPS instead of HTTP for sending authentication information over the Internet so that cleartext passwords cannot be read by wiretapping programs.

In the case of the SODA4LCA database, authentication and authorization of the service API calls are done by using the normal login mechanism for web users. I.e. if a service client sends a service request to the server and the server requests authentication from the service client, the client should react by sending username and password via HTTPS to the standard login procedure of the database application (the POST URL which is used as action URL in the login form). The server looks up user and password in its user database and verifies the credentials of the user. If they are valid the server will acknowledge the successful login to the HTTP(s) layer of the service client which will store the acknowledgement information for further requests in its HTTP(s) layer. In further requests, the service client will now identify itself by the acknowledgement information given by the server. The server itself tracks the client within a web session containing the name of the user. The database application can now use normal authorization mechanism, like group- and role-based authorization, to decide if a user has rights to upload, update or delete data sets.

3. Usages of the Service API

In the case of the SODA4LCA database application, the service API decouples the graphical web user interface of the application completely from its data management functionality which can be accessed solely via the service API. This cannot only be used by desktop LCA tools for accessing and managing data sets on the remote database system but also by other applications like web portals. For such portals, small web-based user interfaces (like Java portlets) can be created, which use the service API for remote access to a LCA database in the background. These portlets can then be integrated into the web pages of the portal as interactive user interfaces for accessing data sets from within the portal. The “Fuel Cell and Hydrogen (FC-HyGuide)” EU project (FC-HyGuide 2011), co-funded by the European Commission, and the German BioEnergyDat project (Schebek 2010), funded by the German Federal Ministry for the Environment, use this technique to provide integrated access to an LCA database node specialized on fuel cell and hydrogen or bio energy technology-related datasets, respectively, to their users via a central web portal which contain beside the data access area other content areas where background information like documents and guidelines or report templates are provided. Figure 2 shows a screenshot of the FC-HyGuide
portal of the web page which includes the data set access portlet for process data sets. This integration approach can also be used to integrate LCA database functionality seamlessly into other corporate portals of companies or governmental institutions.

Figure 2: Data set access portlet integrated into the FC-HyGuide portal.

Different LCA databases on the Internet can use the SODA4LCA service API to interconnect each other to build a network of linked database nodes, which can exchange data sets or delegate search queries from one node to other nodes as already described in the introduction. The International Reference Life Cycle Data System (ILCD) Data Network, an initiative of the EU, is aimed to be such a network of interlinked LCA databases providing harmonized LCA data sets in ILCD-compliant format where network nodes will be maintained by different partners within the ILCD network. While a central system within such a network can provide basic background data sets like process data sets for energy, materials, and transport or harmonized lists of flows, physical-property or unit descriptions or LCIA method data sets, other nodes in the network can concentrate on providing only process and accompanying product flow data sets for specific industrial sectors, like plastics, steel production or chemicals.

The SODA4LCA database application, which is another part of the SODA4LCA Open Source project implements the service API and can be used as software infrastructure for installing and maintaining nodes of such a network of LCA database systems. If configured to access other nodes of a data network, a SODA4LCA instance allows users of its graphical web user interface to perform a query for LCA data sets with the local search form of a node which will not only return results from the local database node but also from other connected database nodes.
This distributed search can also be instrumented by other software tools using the service API which can request a distributed search from one database node if wanted. The database node will then perform a distributed search in its network and returns a merged search result list containing results from several nodes to the software tool. Because the SODA4LCA service API and the ILCD format (Europe 2011a) allow the hyperlinking of data sets across node boundaries, a process data set provided by one network node of an e.g. industrial association can reference other product or elementary flows which are provided by a central node of the network that provides common background data sets. The ILCD data set editor provided by the DG JRC (Europe 2011a) and a special version of the openLCA tool (openLCA 2011) already use the service API to provide direct access to data sets from within the tool. The openLCA version mentioned can even automatically resolve references to other data sets and imports referenced data sets from other databases automatically without user intervention. This gives a seamless import experience for the user. Both tools cannot only query database nodes for data sets but can also create and update data sets within the database using POST and PUT service API requests.

3. Conclusion and Outlook

In a nowadays distributed business world where more and more data is provided via cloud services on the Internet, the accessibility of data via simple-to-use services becomes more and more important. This is especially true for LCA data because not only the production processes modeled by LCA are distributed but the persons creating and maintaining data sets and doing LCA studies are also situated at different sites worldwide. LCA database systems and LCA tools implementing and using the SODA4LCA API can work together in a distributed fashion and allow an easy and seamless transfer of data between databases and tools for data creators and modelers.

It is foreseeable that in the near future also other tools like tools performing simplified LCA calculations, engineering applications or environmental data management systems will benefit from LCA databases which make their data available via services.

Bibliography

European Commission - Joint Research Centre (JRC) - Institute for Environment and Sustainability, Karlsruhe Institute of Technology (KIT) - Institute for Applied Computer Science (2011a): International Reference Life Cycle Data System (ILCD) data format and editor. (Accessible via http://lct.jrc.ec.europa.eu)