Real-time integration of models

Background

During model integration, a model can be linked to a number of models in several ways. Each user may have its own integration requirements. Integration scenarios identified and designed by certain group of modelers or developers may not satisfy integration requirements of the whole user community. A particular user may come up with a number of integration requirements even from time to time. Due to this there is a need for a mechanism in which users can select certain models and link them without the need for, example, additional design, coding, debugging.

In computer systems real-time is the actual time in which a process takes place or an event occurs. Real-time integration is an integration method in which users can select and integrate models just during time of usage. To realize real-time integration, models should be made available based on some standards.

Objective

To design a framework for real-time integration of models through Web services.

Method

Models which are available as Web services can be used as basis for establishing real-time integration.
The advantage of using Web services as standard for presenting models is Web service can be published, found, and used on the Web (Erl et al., 2009). The discoverability property of Web services enables users to fetch the available methods and variables of the Web services during run-time. Besides there is no requirement to develop static coupling to use the available functionalities of the Web services.

Architecture of the distributed model integration framework

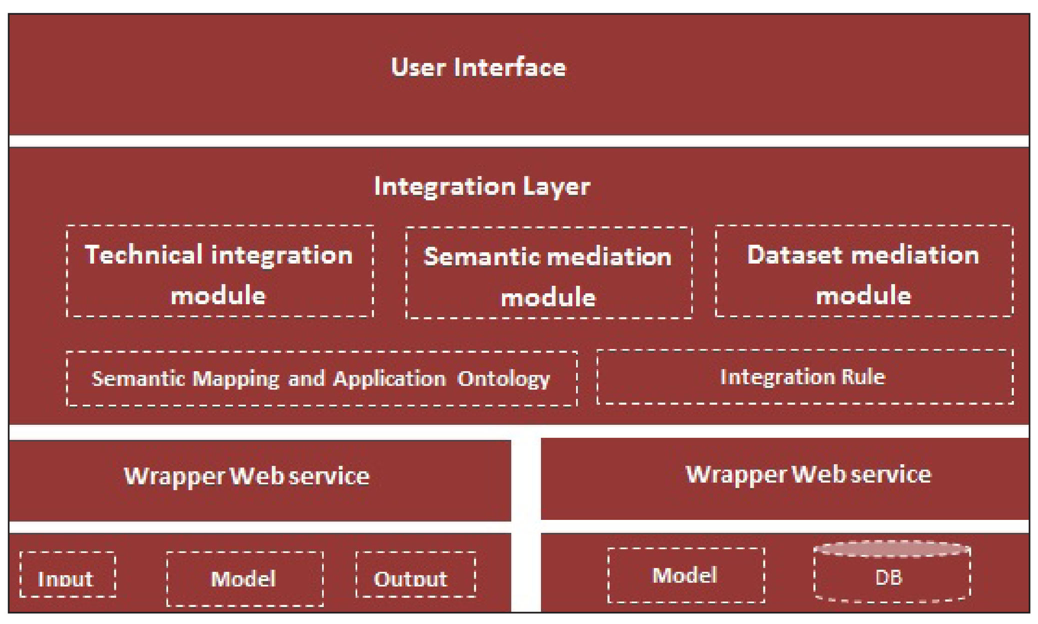

Figure 1. Architecture of distributed model integration framework

The architecture (Figure 1) of the model integration framework is a layered structure developed based on the concept of distributed computing infrastructure. Basically, distributed systems consist a collection of subsystems which could be deployed on different heterogeneous platforms and they communicate each other by exchanging messages (Papazoglou, 2008). The distribution of the subsystems is transparent to the user in which that creates the feeling of single integrated system.

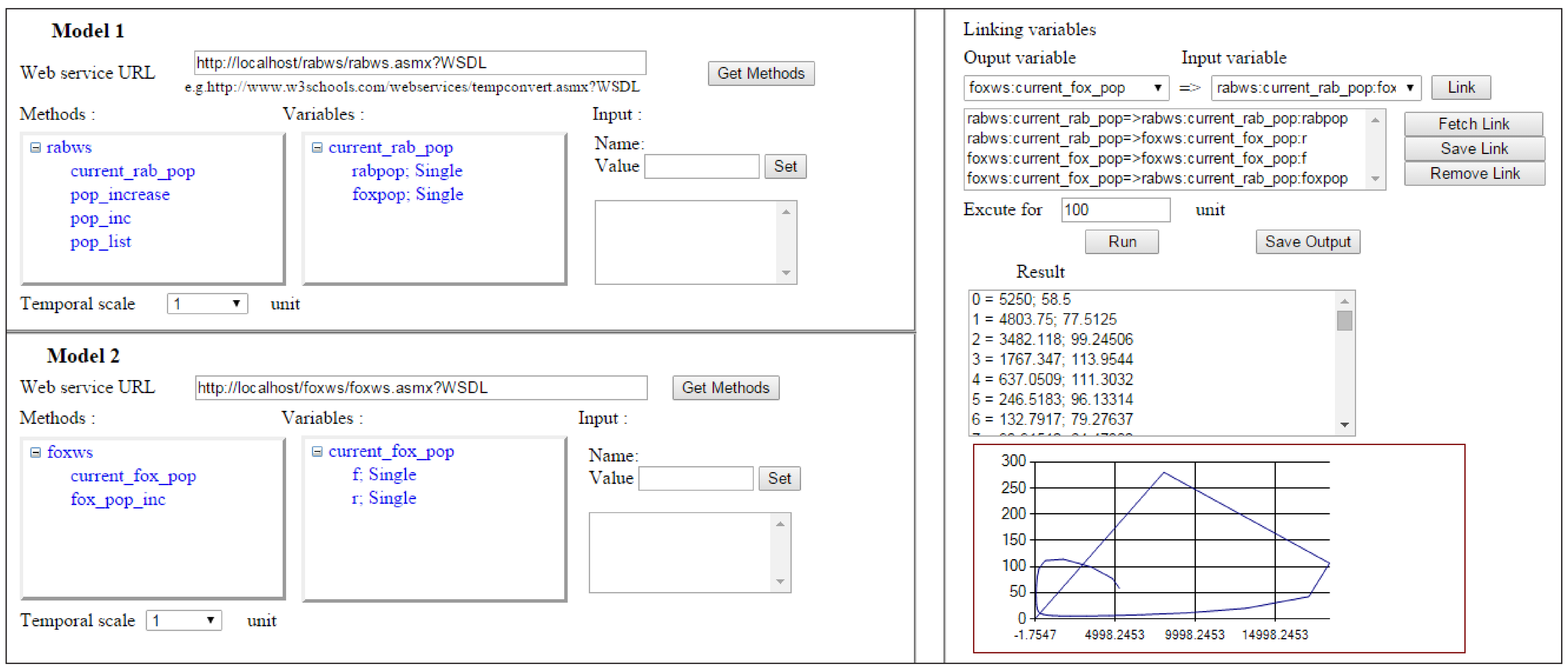

Figure 2. User interface for real-time linking of Web service based models

Following the distributing computing paradigm the proposed architecture provides the benefit of:

- Models can be developed using different programming languages and can be deployed in different operating systems,
- Participating models are autonomous,
- Models can run concurrently on different machine using multithreading technique – which has significant performance gain in linking model with 'longer' execution time,

How it works

As shown in Figure 2, a user should come up with the URLs of Web service based models. Then from the URL and WSDL files of the web service all available methods and the corresponding variables of the model can be discovered during run-time. Input values of the variables can also be set using the graphical user interface. Once the input data is set the user can specify the mapping between variables of model 1 and variables of model 2, i.e. the user can define the semantic mediation pattern manually. However if an already defined semantic mapping exists the user can use 'Fetch Link' button to fetch existing semantic mapping.

For example, assume that we are integrating model M1 with model M2. Suppose M1 has function named function1 with input variables var1 and var2 and after processing, it produces the recent value of var1 as an output. Similarly M2 has input variables var3 and var4 and produce recent value of var3 as an output. From Figure 3, when the integrated models run for more than one time the outputs of M1 will be used as input for M1 and M2. In a similar way the output of M2 will be used as input for M1 and M2 . This semantic relationship is defined as:

M1: function1 => M1: function1:var1
M1: function1 => M2: function2:var4
M2: function2 => M2: function2:var3
M2: function2 => M1: function1:var2

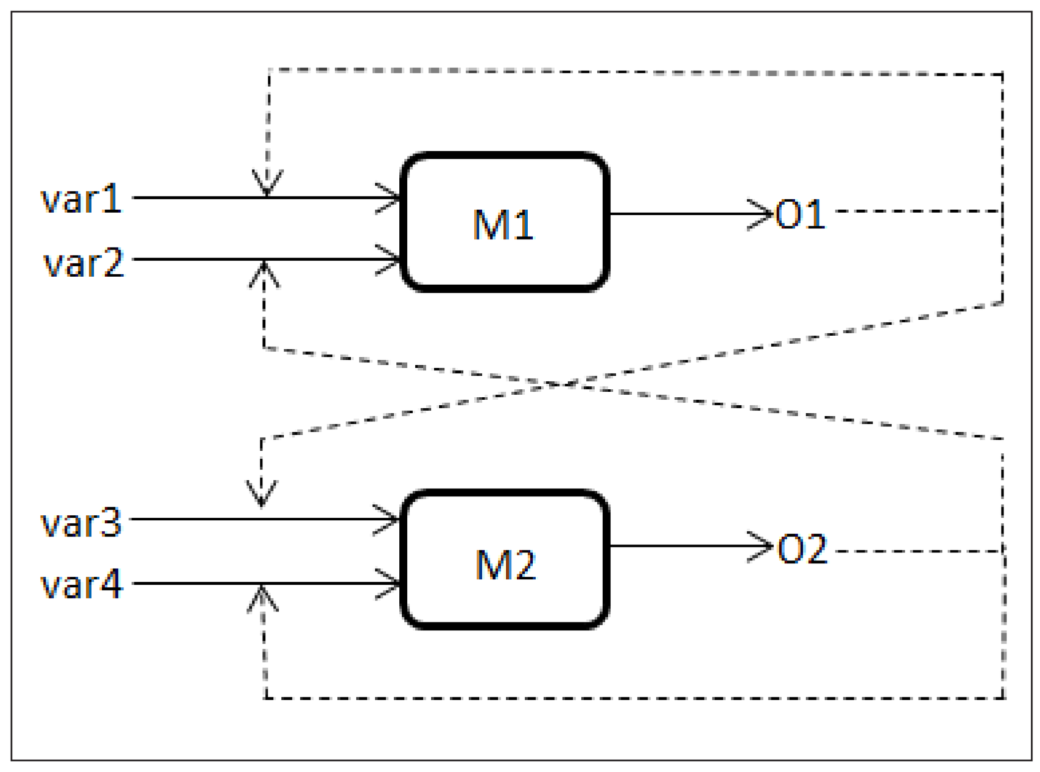

Figure 3. Simple variable mapping in integrating model M1 with model M2. M1 takes inputs var1 and var2 and produces output O1; and M2 takes inputs var3 and var4 and produces output O2

The user can save the semantic mapping for future use. It will be stored in the semantic mapping database (Figure 4) of the model integration framework and it will be accessible by any user when required. When the user chooses the 'Fetch Link' button, the system will query for existing semantic mappings in the database using the selected models and functions as criteria for searching. However, the user can override the semantic mapping fetched from the semantic database.

Conclusion

For models that take and return simple dataset types and that does not require complex semantic and dataset conversion, the proposed approach can be effectively used for real-time integration of models.
For models that take and return complex datasets, e.g. in the form of tables, and for integration process that require complex semantic and dataset conversion works, e.g. upscaling and downscaling, further research and design is required.
The use of standard variable names, e.g. CSDMS standard names, in developing models and wrapper Web services will have great contribution during user defined semantic mediation

mapping	m1_name	m1_method_name	m2_name	m2_method_name	m2_var_name
M1:function1=>M1:function1:var1	M1	function1	M1	function1	var1
M1:function1=>M2:function2:var4	M1	function1	M2	function2	var4
M2:function2=>M2:function2:var3	M2	function2	M2	function2	var3
M2:function2=>M2:function1:var2	M2	function2	M1	function1	var2

Figure 4. Snapshot of semantic-mapping table

References and Acknowledgement

Erl, Thomas, Karmarkar, Anish, Walmsley, Priscilla, Haas, Hugo, Yalcinalp, L Umit, Liu, Kevin, . . . Pasley, James. (2009). Web service contract design and versioning for SOA: Prentice Hall.
Papazoglou, Michael. (2008). Web services: principles and technology: Pearson Education.

Research was supported by COMPLEX – Knowledge Based Climate Mitigation Systems for a Low Carbon Economy Project, EU-FP7-308601.

Authors and Affiliations

Getachew F. Belete, getfeleke@gmail.com

Faculty of Geo-Information Science and Earth Observation (ITC), University of Twente, Enschede, The Netherlands

UNIVERSITY OF TWENTE.