Automatic Data Reuse in Grid Workflow Composition

Ondrej Habala, Branislav Simo, Emil Gatial, and Ladislav Hluchy

Institute of Informatics, Slovak Academy of Sciences,
Dubravska 9,84507 Bratislava, Slovakia
{Ondrej.Habala,Branislav.Simo,Emil.Gatial,
Ladislav.Hluchy}@savba.sk

Abstract. Many papers, research projects, and software products have tackled the problem of automatic composition of a workflow of computer processes, which computes certain data or performs a specific task. In recent years this has also gained popularity in grid computing, especially in connection with semantic description of resources usable in the workflow. However, most of the works dealing with semantically-aided workflow composition propose solutions only for workflows of processes, without the data necessary to execute them. We describe the design of a system, which will be able to find not only the processes, but also the content for their execution, based solely on the list of available resources and a description of the required target of the workflow. The solution is based on our previous work in the project K-Wf Grid, utilizes semantic description of resources by means of ontologies, and operates on a SOA-based grid composed of web services. It is being developed in the context of a project called SEMCO-WS[1].

Keywords: Semantic grid, SOA, web services, automated workflow composition.

1 Introduction

Many papers, projects, software solutions [1-3] have tackled the problem of automatic composition of a workflow of computer processes. This type of automation is very attractive especially in software engineering applied to scientific research, where complicated simulations and parameter studies often require tens of single steps in order to obtain the solution required by the scientist. Since the inception of grid computing, workflow composition of grid jobs into complex workflows has also gained prominence with its apparent usefulness, long history of previous works not applied specifically to grid, and robust mathematical theory based mainly on direct acyclic graphs. In recent years, advances in semantic web have been applied also in grid computing – creating semantic grid [4] – and specifically in the area of semantically-aided composition of workflows of grid tasks. However, most of the many works on this topic have concerned themselves only with the composition of a

[1] This work is supported by projects SEMCO-WS APVV-0391-06, int.eu.grid EU 6FP RI-031857, VEGA 2/7098/27.

workflow of computer processes – represented by grid jobs, calls to web service interfaces, of other custom tasks – solving the "how", and have omitted the "what" of this problem, it being the data, on which these processes operate. This has been left to the user. While the sought after result is a system, in which the user enters the description of the data he/she requires, and the system composes a workflow able to compute it, most of the existing solutions create only a workflow able to solve a class of problems, and the selection of the one unique member of this class via entering the correct data is left to the user.

We have designed, and begun to implement, a system, which solves also the "what" of automated workflow composition. The proposed system is based on previous work done in the context of the project K-Wf Grid [8], and extends it with tools which are able to determine exactly which data is necessary for which process in the composed workflow in order to get – at the end – the data which the user has described as his/her target. The system is based on semantic description of data and grid services by ontologies. The workflows are modeled as Petri nets, this being a legacy of K-Wf Grid offering very good means to model data (as Petri net tokens). The system interacts with the user only to the extent absolutely necessary to acquire data or services which are required for the solution, but currently not available in the grid.

The rest of this paper first presents the project K-Wf Grid and its results, and establishes a frame of reference for our own work. Then we briefly present the project SEMCO-WS [9], and the main part of the paper describes the proposed solution of automatically composing workflow with not only processes, but also data.

2 Results of K-Wf Grid

The project Knowledge Based Workflow System for Grid Applications – K-Wf Grid – started in September 2004 and ended in February 2007. It was very successful in attaining its goals, and the final review in March 2007, as well as a public showcase at the Cracow Grid Forum '06 was both a success. The consortium was composed of six partners, and the work was very well focused on one goal – automated composition of workflows of grid services using semantic support, and accessible through a comfortable web-based graphical interface.

For the purpose of this paper, we will discuss only the parts of K-Wf Grid middleware, connected with application workflow composition. The middleware has been tested on three pilot applications. Each application went through several stages, beginning with integration, and ending with a successful workflow execution.

K-Wf Grid application is a set of web or grid (WSRF) services. Any application has first to be integrated into the system's knowledge base [10]. The process of integration is mostly automatic [11], the application expert has only to annotate WSDL documents of the application's services with markup denoting input and output structures used in service calls.

Following the integration, the application may be used in the system. User enters a textual description of his/her problem, and a tool [12] connected to the knowledge base finds any available targets – service results – relevant to this problem description. The user selects one or more of the found targets, and thus establishes a context for a workflow.

The workflow is formed and executed in several stages. We have to remember that it is always modeled as a Petri net, so the appropriate terms are activities (for processes), tokens (for data), and data places – for inputs and outputs of the activities. It begins with an abstract description of a problem, having only one activity, and one output (the target of the workflow). This simple workflow is then expanded [13] using descriptions of available services, stored in the knowledge base. The result is a workflow of service classes – descriptions of service interfaces, without actual grounding (concrete service providers). Then in another step [14] the available service providers for each service are found, and the workflow is now composed of a set of activities, each of which presents several possible choices of a concrete web/grid service for its execution (firing, in terms of Petri nets). The final pre-execution step is scheduling, during which the scheduler [5] selects for each activity one service, assumed to be the best one (considering a metric for evaluating different properties of services) for the workflow.

After this workflow construction, the workflow (assuming that the system was able to find all necessary service classes and grounded services for these classes) may be executed. During execution, one or more input data structures may be necessary – the user will be asked to provide data description. Although this step is also comfortable, and the user may use custom web forms developed by the application developer, it is still necessary for the user to know the application and be able to judge which data will be necessary to produce the target he/she wishes to obtain.

Following the workflow execution, the data may be downloaded from grid storage; or, if the application contains also visualization tools and custom services able to cooperate with grid middleware (so-called job packagers), it is transformed into easily readable form and made accessible through the K-Wf Grid portal.

3 Semantic Workflow Composition in SEMCO-WS

The project SEMCO-WS is a small national project with a consortium of four members, all from Slovakia. It started in February 2007 and is scheduled to end in January 2010.

The project is trying to expand and refine several components of K-Wf Grid (adding also other features, not present in K-Wf Grid). While the whole process of workflow construction and execution in K-Wf Grid was observable through a graphical workflow visualization tool, the user could edit only data tokens, not the workflow structure. SEMCO-WS will include an improved version of the visualization tool, supporting also complete workflow editing. The process of knowledge base management will be also supported by comfortable graphical tools, and the knowledge base itself will be decentralized. Most importantly for this paper, the process of data selection, left to the user in K-Wf Grid, will be fully automated. The user will be asked only to provide data, which will not be available and described in the knowledge base. The design of this part of SEMCO-WS middleware is explained in the following chapter.

4 Adding Automatic Data Selection to Workflow Composition

To be able to propose not only the activities of a workflow, but also the content of the initial tokens in it, based only on the content of the final output token (which represents the target data of the workflow), the system has to be able to

1. know the content of any data present in the system,
2. infer the necessary input token parameters, based on the parameters of the output token to be generated by any activity,
3. decompose output structures of web and grid services into separate tokens, and
4. compose input structures for web and grid services from existing tokens.

The solutions of these partial problems are described below.

4.1 Content of Data – Metadata

Any data piece available to the workflow composition system is represented by a token. If it is a file, it can be its content (for smaller files), its URL, LFN, or any other identifier. It can be an URL of a database, and an identifier of an item in the database. The actual content of a token is application dependent, and the system does not need to be able to read it. Any token has first to be either entered by a user, or generated by an application activity, and it is processed only by another application activity or a user, so the application dependence is fully hidden in the application domain. The workflow system needs only to know the metadata of the token, to be able to evaluate its content for possible use in a workflow.

The metadata, represented by OWL and OWL-S constructions is composed of generic, application-independent part, and of other, application dependent part. This layering of ontologies has been also used in K-Wf Grid, where the application ontologies were using a common base ontology layer with basic facilities for description of services, files, resources, computers, clusters, etc. The OWL standard is used mainly because it has been already incorporated into K-Wf Grid middleware, upon which our system is based. Alternatively, also other suitable ontology representation language could be used, or even the WSMO language [15], developed specifically for modeling of web services.

The requirement that the system has to be able to use data computed in the past for current workflows also implies, that all tokens created by any application have to be stored in a database. Since in K-Wf Grid and in SEMCO-WS the workflows (including tokens) are described in an XML dialect called the Grid Workflow Description Language [7], a simple XML database is sufficient for token storage, and later lookup. When the system identifies a data piece based on its metadata, the ontology will contain also the identifier of the token representing the data, and the token can be retrieved from the XML database using this identifier.

Each newly created token has to be described by its metadata. This can be done in two ways – either the metadata is generated by another application component (a simple web service, or other module), or it may be computed using a set of mathematical and logical formulas. As we will discus below, it is necessary to be able to infer the properties of input data from required output parameters, so it may be also possible – at least for a subclass of activities – to describe the inverse transformation and infer properties of output tokens from known properties of input tokens.

4.2 Backtracking from Required Output Token to the Necessary Input Tokens

The process of constructing a workflow in K-Wf Grid has been sufficiently described before [13], [14], [6]. This process did not provide for reusing existing data, and always assumed that the whole workflow chain has to be computed anew, even if some partial results from previous workflow could be reused and could replace parts of the newly created workflow. The workflow construction process used backtracking, from the final activity to the initial activities of the workflow. In SEMCO-WS, the process will also include backtracking from the final token to the initial tokens of the workflow. We can abstract tokens and activities as data providers, with the difference that tokens provide data and require no inputs, and activities also provide data, but require input data – other tokens. So the process of workflow construction can be described using this algorithm:

```
Program construct_workflow (token_metadata_list)
//The input is a list of metadata descriptions of all
//tokens we wish to produce with the constructed
//workflow
Variables:
workflow //list of components of the constructed
         //workflow
token
activity
token_metadata //member of token_metadata_list

1. Foreach token_metadata in token_metadata_list
      a. token ← find in token_db based on
         token_metadata
      b. If token ≠ nil
             workflow = workflow + token
         Else
             Find activity able to produce token
             workflow = workflow + activity
             token_metadata_list = token_metadata_list
             + token_metadata of all input tokens of
             activity
2. If token_metadata_list is not empty, goto 1
3. Output workflow
```

So we see, that we first try to find the data we need, and if it cannot be found (it was not yet computed or entered into the system), we find an activity which can compute such data. Of course, then we have to find the correct input data for this activity too, and the process repeats itself. The K-Wf Grid incarnation of this algorithm was looking only for activities, and essentially omitted step 1a in the algorithm.

In 1b-else clause, we are looking for an activity able to produce a token with certain parameters. To be able to do this, we also need descriptions of the capabilities of activities, i.e. what are the possible parameters of tokens the activity is able to produce. If this description includes also the parameters of the token to be produced, the activity may be used to produce it.

4.3 From Tokens and Activities to Data Structures and Services

Our Petri net model of a workflow operates with activities, places, and tokens. For purposes of management of SOA applications, we need to be able to transform these concepts to web service calls, and input/output structures of web service interfaces (data structures). While the transition from activities to web service calls is straightforward, tokens and data structures do not map directly. Any activity may require more than one piece of data – so on its input are several tokens, coming from several input places, but the underlying service can consume only one input structure. Also, the output data structure of the service may contain several data fragments – values, references to files, etc. – and we want to store them as tokens separately, since they may be later used separately as inputs to other activities.

For the purpose of construction of input data structures for services, and decomposition of their output structures into separate tokens, we have extended the annotation of WSDL documents of application's services with additional elements. These elements (contained inside the definition of data structures) contain XSLT code, which may be used to automatically compose the structure from several XML fragments identical with tokens, and also to automatically split the output structure into the fragments which then become the output tokens of the activity. The composition and decomposition process is quite straightforward; an activity may have several input and output places, but the actual web service hidden behind the activity has only one input, and one output, both represented by an XML structure. Upon activity firing, the input tokens (XML fragments) are concatenated together, and then transformed into the format of the web service input structure using the XSLT provided in WSDL document. Similarly, the output structure is then filtered by XSLT into distinct output tokens – one XSLT document for each output token.

4.4 The Workflow Composition Process

The whole process and problem of automated workflow construction using also existing data has been decomposed into several steps (see Fig. 1):

1. Initial data has to be entered by the user; he/she is a domain expert, so it is easy (using custom web forms which are part of the application) for him/her to create the token, as well as describe it with metadata; both token and metadata are stored.
2. When the GWES looks for components of the workflow, it queries the knowledge base for any existing tokens which can provide the necessary data.
3. If such token is found, it is extracted from the token database; otherwise an activity is used (not shown in Fig. 1 since it is not the focus of this paper, composition of services into workflow has been sufficiently described in other works).

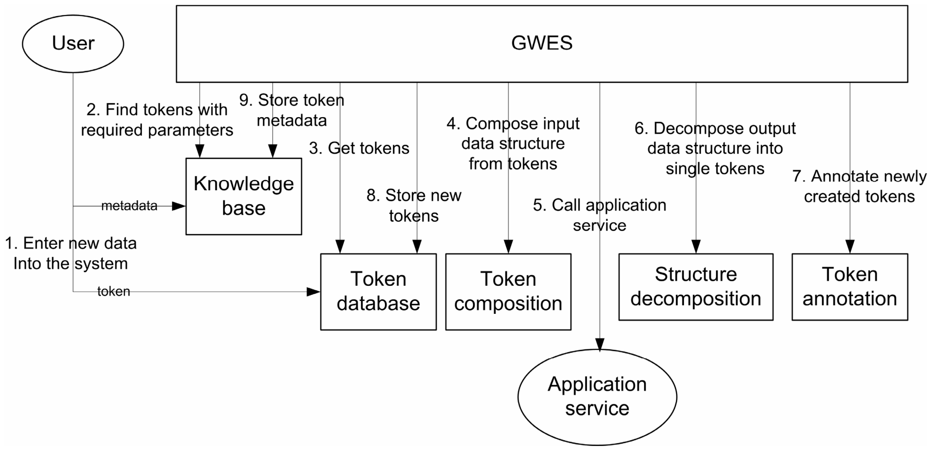
User
GWES
2. Find tokens with required parameters
9. Store token metadata
3. Get tokens
4. Compose input data structure from tokens
6. Decompose output data structure into single tokens
7. Annotate newly created tokens
Knowledge base
metadata
8. Store new tokens
5. Call application service
1. Enter new data Into the system
token
Token database
Token composition
Structure decomposition
Token annotation
Application service

Fig. 1. The data location/creation process

4. During execution of the workflow, GWES has to compose an input structure for the actual application service from the input tokens of the service's activity. This is done automatically, using XSLT transformations present in the WSDL document of the service.
5. The composed input structure is used in a call to the application service; the service replies with an output structure.
6. The obtained output structure is decomposed into single tokens, which represent data items contained in the structure. This is also done automatically using XSLT transformations inside WSDL document (see Chapter 4.3 for details).
7. The created tokens have to be annotated; this is another application-specific step of the process, but (as discussed above), we can avoid helper services by using mathematical and logical formulas which describe the process of transformation of parameters of input tokens (which are known) into parameters of output tokens. Alternatively, if such formulas cannot be used because of the complexity of the transformation, helper service can be used. The formulas or the URL of the helper service can also be found inside the annotated WSDL document of the application service.
8. The created tokens (extracted from output of the application service in step 6) are stored in the token database for later reuse.
9. The metadata of the new tokens (created in step 7) is stored in the system's knowledge base. The cycle may begin with another iteration from step 2, or – if the workflow has no more activities which can be fired – end.

We have not discussed some situation which may arise, when the composition of a workflow is halted by a requirement for data, which cannot be found in the database, nor produced by any known activity. In such case, the user may be asked to provide the data. Alternatively he/she may abort the workflow composition process, enter new application service into the system, and restart the workflow composition. So the situation can be resolved by adding either the data, or a service which can produce it.

5 Conclusions

We have shown, that fully automated construction of workflows of web and grid services can also reuse existing data, provided that the processes is supported by a semantic annotation layer, and the application services are annotated. Three key components of the annotation are formulas for transformation of input metadata into output metadata, formulas for the inverse transformation, and description of capabilities of application services.

The design presented in this paper is currently entering the implementation phase. The project SEMCO-WS is continuing the work of K-Wf Grid in some areas, where the semantic support of grid workflow composition can be further improved, and the inclusion of automatic data reuse is one of them. The prototype of this system is to be ready by 2009, and when SEMCO-WS finishes in 2010, the system will be ready to be used and further improved by other researchers.

References

1. Bubak, M., Gubala, T., Kapalka, M., Malawski, M., Rycerz, K.: Grid Service Registry for Workflow Composition Framework. In: Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2004. LNCS, vol. 3038, pp. 34–41. Springer, Heidelberg (2004)
2. VDS – The GriPhyN Virtual Data System (Accessed January 2008), `http://www.ci.uchicago.edu/wiki/bin/view/VDS/VDSWeb/WebMain`
3. Krishnan, S., Wagstrom, P., von Laszewski, G.: GSFL: A Workflow Framework for Grid Services. In: Preprint ANL/MCS-P980-0802, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, 1L 60439, U.S.A. (2002)
4. Semantic Grid Community Portal (Accessed January 2008), `http://www.semanticgrid.org/`
5. Wieczorek, M., Prodan, R., Fahringer, T.: Comparison of Workflow Scheduling Strategies on the Grid. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Waśniewski, J. (eds.) PPAM 2005. LNCS, vol. 3911, pp. 792–800. Springer, Heidelberg (2006)
6. Hoheisel, A., Der, U.: An XML-based Framework for Loosely Coupled Applications on Grid Environments. In: Sloot, P.M.A., Abramson, D., Bogdanov, A.V., Gorbachev, Y.E., Dongarra, J., Zomaya, A.Y. (eds.) ICCS 2003. LNCS, vol. 2657, pp. 245–254. Springer, Heidelberg (2003)
7. Alt, M., Gorlatch, S., Hoheisel, A., Pohl, H.W.: A Grid Workflow Language Using High-Level Petri Nets. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Waśniewski, J. (eds.) PPAM 2005. LNCS, vol. 3911, pp. 715–722. Springer, Heidelberg (2006)
8. Knowledge-based Workflow System for Grid Applications (K-Wf Grid). EU 6th FP Project, 2004-2007 (Accessed January 2008), `http://www.kwfgrid.eu`
9. Semantic composition of Web and Grid Services (SEMCO-WS). Slovak APVV project, 2007-2009 (Accessed January 2008), http://semco-ws.ui.sav.sk/
10. Kryza, B., Pieczykolan, J., Babik, M., Majewska, M., Slota, R., Hluchy, L., Kitowski, J.: Managing Semantic Metadata in K-Wf Grid with Grid Organizational Memory. In: Bubak, M., Turala, M., Wiatr, K. (eds.) Proceedings of Cracow Grid Workshop – CGW 2005, Krakow, Poland, November 20-23 2005, pp. 66–73 (2005)
11. Habala, O., Babik, M., Hluchy, L., Laclavik, M., Balogh, Z.: Semantic Tools for Workflow Construction. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2006. LNCS, vol. 3993, pp. 980–987. Springer, Heidelberg (2006)

12. Laclavik, M., Seleng, M., Hluchy, L.: User Assistant Agent (EMBET): Towards Collaboration and Knowledge Sharing in Grid Workflow Applications. In: Cracow 2006 Grid Workshop: K-Wf Grid, pp. 122–130 (2007) ISBN 978-83-915141-8-4
13. Gubala, T., Herezlak, D., Bubak, M., Malawski, M.: Semantic Composition of Scientific Workflows Based on the Petri Nets Formalism. In: Proc. of e-Science 2006, Amsterdam (2006) ISBN-0-7695-2734-5
14. Dutka, L., Kitowski, J.: AAB – Automatic Service Selection Empowered by Knowledge. In: Bubak, M., Turała, M., Wiatr, K. (eds.) Proceedings of Cracow Grid Workshop – CGW 2005, ACC-Cyfronet USTs, ACC-Cyfronet UST, November 20-23 2005, p. 58 (2006)
15. Web Service Modeling Ontology (Accessed March 2008), http://www.wsmo.org/