Semantic description of services: issues and examples.

Aurélie Hurault, Frédéric Camillo, Michel Daydé, Ronan Guivarch, Marc Pantel, Chiara Puglisi
University of Toulouse-IRIT, France
e-mail:
hurault,camillo,dayde,pantel,puglisi@enseeiht.fr

Hrachya Astsatryan
Institute for Informatics and Automation Problems, Armenia
e-mail: hrach@sci.am

ABSTRACT
In this paper, we concentrate on the semantic description of services. We explain why it is a crucial problem. Different solutions are described and we will see that the selection a type of description depends of the precision needed.

We illustrate this with two examples that requires different levels of description: the GRID-TLSE project and a trader based on an advanced semantic description of services.

Keywords
Service, Semantic description, Grid

1. INTRODUCTION
Object programming language, component-based software engineering, web-services architecture,... are all techniques that have been introduced to achieve re-usability of software. It is clear that this goal is not fully reached. One of the limitation of these approaches lies in the difficulty to find the "right" object / component / service /.... In the following "objects / components / services /..." will be called services.

This is still an active area of research because of the availability of large scale distributed infrastructure that provide access to a wide range of services and resources (web services over the internet, numerical codes over large clusters, ...) on various architectural platforms. In order to take advantage of this huge number of services, we still need a way to identify the "right" service (or composition of services).

We face two problems. First, finding a description precise enough to be able to say that it is the "right" service. Secondly, developing tools that help us to automatically find the service or the combination of services that solve our problems.
Both problems are obviously correlated, since the matching algorithm depends on the service description used.

In the following, we study the current available solutions, and give two examples in which the description of services is a key point.

2. CURRENT SOLUTIONS
The simplest description which is used in most SOA (Service Oriented Architecture) [5] such as RPC, CORBA[1], COM, DCOM, RMI only makes use of the service signatures (input and output types of the service). This information has the advantage to be easily available. However, it is not sufficient for a sophisticated matching, even if subtyping or type isomorphisms are used to remove the problems of parameter position. Indeed, with such an approach there is no way to distinguish addition from multiplication as they both share the same signature. But, the signature is used in almost all other approaches.

We can add keywords or meta-data to the service signature (this is currently the case in the GRID-TLSE project[2] [4, 8], with some added constraints, see next section). A disadvantage is the difficulty to describe a complex service. For example, it is very difficult to describe without ambiguity and using keywords some of the Level 3 BLAS procedures such as $SGEMM$ that provides the matrix-matrix operation: $\alpha * \mathbf{A} * \mathbf{B} + \beta * \mathbf{C}$ (simple precision). However, such a formalism allows an easy search of the services that answer a user request. It requires a preliminary agreement to define the keywords and their meaning, with all the ambiguities inherent to natural language.

In some other SOA that are web services oriented, a service broker (or service registry) is used to make accessible the service interface and implementation. The UDDI[3] (Universal Description Discovery and Integration) is a way to manage this registry while WSDL[4] is used to describe services interfaces. UDDI is often compared to a telephone book's white, yellow, and green pages. Used as white pages, knowing the name of the service is necessary to find it. Then, all details for using it are provided. The yellow pages are slightly simpler since the services are stored by domain and allow to find the one that fit the user need by using keywords or classification. However, this approach does not provide an easy way to combine services, even if it helps in solving some problems like the distinction of multiplication and addition, since more than the signature is given.

Another approach which extends keywords and metadata is based on ontologies such as OWL [7, 1]. The advantage of ontologies is the possibility to have a formal description. It also provides the logic associated to reason about these descriptions. The disadvantage is that there is no control on this logic that may appen to be undecidable. Moreover the definition of an ontology is not trivial and hard to achieve for a non specialist.

In the Monet[5] and HELM[6] projects, the description of the

[1] http://www.corba.org/

[2] http://gridtlse.org/
[3] http://uddi.xml.org/uddi-org
[4] http://www.w3.org/TR/wsdl
[5] http://monet.nag.co.uk/cocoon/monet/index.html
[6] http://helm.cs.unibo.it/

computational services is based on MathML[7] and OpenMath[8] which provide an accurate description. But the computation of the set of suitable services is based on RDF[9] (Resource Description Framework) and ontologies which are not easily adaptable to other domains. However for the mathematical domain, it is very interesting descriptions. The solution proposed in the second example can be seen as a generalization of those descriptions.

In the NASA Amphion project [9] and more particularly the theorem prover SNARK use a interesting method: independence of the application domain and reasoning based on a description of the domain. But, the Amphion project relies on *"term rewriting and the paramodulation rule for reasoning about equality"*. This supposes that *"a recursive path ordering is supplied when the application domain theory is formulated"*. This last constraint requires that the user be familiar with complex rewriting techniques. The second used case described in the paper is really close to these approaches, but without asking the user to be familiar with the underlying technologies.

3. APPLICATIVE EXAMPLES

We illustrate our approach with two different applications that require a description of services at a different level. The first one is the GRID-TLSE project. All the services have a similar functionality but different algorithms are available. The parameters and interfaces of the services are also not uniform. As a consequence, the description should be at the algorithmic level. The second application is a trader that aims at finding the service or the combination of services that fulfil the user request. In that case, we are not interested by all the algorithmic features of the services that are then seen as black boxes.

3.1 The Grid-TLSE project

The Grid-TLSE[10] is an expert site for sparse linear algebra that provide tools and software for sparse matrices.

This work has been partially supported by the French Ministery of Research throught the GRID-TLSE Project from ACI "Globalisation des Ressources Informatiques et des Données" and by the ANR (Agence Nationale de la Recherche) through the LEGO and SOLSTICE Projects referenced respectively ANR-05-CIGC-11 and ANR-06-CIS6-010.

It allows the comparative analysis of a number of direct solvers (free or commercially distributed) on user-submitted problems, as well as on matrices from collections available on the site.

The site provides user assistance in choosing the right solver for its problems and appropriate values for the control parameters of the selected solver. The computations are carried over a computational grid.

3.1.1 Main Software Issues

The particular point of the TLSE problem is that the same interface allow the users to access the several expertise scenarios and solvers and their parameters (using the DIET middleware to access the GRID (see `http://graal.ens-lyon.fr/DIET`). The solvers have the same functionality: solving a linear system, but with different algorithms / implementation and different kind of parameters.

To answer the user request (e.g. what is the minimum amount of memory required for factorizing my sparse matrix or what is the fastest solver on my problem), experts provide scenarios which reduce the combinatorial nature and produce useful synthetic comparison. It should be easy to add new solvers which can be used by old scenarios, easy to add new scenarios which use old solvers, and use the characteristics of new solvers in new scenarios.

We see that most of the tools deployed within TLSE aim at solving a linear system, but with many possible algorithms, many possible control parameters, many values for each parameter, many metrics to evaluate/compute numerical results, and many metrics to evaluate/compute software runs.

How to provide a common API for all these packages ?

3.1.2 Semantic-based description

First of all, clients and providers of solvers must adapt dynamically to each other. As said in the first part, the description is based on meta-data and they should agree on terms.

Once it is done, meta-data are used to describe: functional decomposition of a sofwtare (it usually involves several functionalities), control parameters and their values, metrics and their values, qualitative and quantitative dependencies between the values of metrics and control parameters.

But to be able to use such an approach on other application areas, or to be adapted, addition of new meta-data and corresponding values should be easy.

3.1.3 Using Abstract Parameters

From the Web interface (to define the objective and parameters of the scenarios) up to the service description, it is critical to use a common set of abstract parameters.

A service has two kind of properties: its functionalities (assembled / elemental entries, type of factorization (LU, LDL^T, QR), multiprocessor, multiple right-hand sides) and the algorithmic properties (unsymmetric / symmetric solver, multifrontal, left / right looking, pivoting strategy).

To describe a scenario in addition to service parameters, the expert user should add the metrics (memory, numerical precision, time, ...) and the control (type of graphs for post-processing, level of user).

Abstract parameters are used to express constraints and/or relations. For example, it should be possible to :

- Select only a symmetric solver if the matrix A symmetric.
- Indicate that time and memory depend mostly on method and permutations but also on scaling and pivoting.
- Indicate that numerical accuracy depends mostly on pivoting but also on scaling and permutations.
- Advise orderings for QR based on A^TA.

[7] `http://www.w3.org/Math/`
[8] `http://www.openmath.org/cocoon/openmath/index.html`
[9] `http://www.w3.org/RDF/`
[10] `http://gridtlse.org/`

- Indicate that multiple right-hand sides option, although not available, can still be performed (simulated within the service dameon for example).

- Select a threshold for partial pivoting in $[0 \dots 1]$.

3.2 A trader based on advanced semantic description

The previous example was particular because all services have the same functionality which implies that the description should go further to distinguish them at the algorithmic level.

In the case of more general functionalities, we propose to use a simpler description for the services for two main reasons:
First, it should be possible for a specialist of the application domain to describe his own libraries without the help of a specialist of the description formalism.
Second, the computation of the combination of services should be performed in a reasonable amount of time.

We also want to have a precise description in order to obtain only relevant solutions. To combine both requirements (a fast algorithm and a precise description), we have decided to work within a specific application domains. Our approach is generic, parameterized by the description of the application domain.

3.2.1 Description of the application domain

The application domain is represented by an order-sorted signature $(S, \leq, \Sigma)$ and a set of equations E. These equations allow to specify the domain operator properties (commutativity, distributivity, zero element, ...).

Overloading of operators is allowed to facilitate the user job. When writing "$a + b$", we do not want to use a different symbol for $+$ depending on the types of a and b : for example $a +_{Int \times Int \to Int} b$ and $a +_{Matrix \times Matrix \to Matrix} b$. When an equation is true for all the possible types of the variables (for example $a+b = b+a$ is true for all the types T_a, T_b such that $+$ is defined on $T_a \times T_b$ and $T_b \times T_a$) the user does not have to specify the types of the variables. As a consequence, in the algorithm, type verification is not enough and a mechanism of type inference is required. Type inference is more complex than type verification and overloading of operators, with subtyping, make this task even more complex.

As a conclusion, the type system must be adapted to subtyping and overloading. In [3], Castagna proposes the $\lambda\&$-calculus for formalizing overloaded functions with sub-typing. Our algorithm relies on his proposal.

Example : Simplified description of dense linear algebra without subtyping and overloading.

$S = \{Matrix\}$

$$\Sigma = \{ \begin{array}{l} 0 : \to Matrix \\ I : \to Matrix \\ +: Matrix \times Matrix \to Matrix \\ *: Matrix \times Matrix \to Matrix \} \end{array}$$

$$\text{E}= \{ \begin{array}{ll} x : Matrix & x * I = x \\ x : Matrix, & I * x = x \\ x : Matrix, & x * 0 = 0 \\ x : Matrix, & 0 * x = 0 \\ x, y : Matrix, & x + y = y + x \\ x : Matrix, & x + 0 = x \\ x, y, z : Matrix, & x + (y + z) = (x + y) + z \\ x, y, z : Matrix, & x * (y * z) = (x * y) * z \\ x, y, z : Matrix, & x * (y + z) = (x * y) + (x * z) \\ x, y, z : Matrix, & (x + y) * z = (x * z) + (y * z) \} \end{array}$$

It is a very simplified example since we only take into account one sort: the matrices. We only define two constants (zero matrix and identity matrix) and two operations (addition and multiplication of matrices). The set of equations allow to define the main properties of the operators: null element, identity element, commutativity, associativity, factorization, ...

3.2.2 Managing services and requests

Once the application domain is described, the services and requests are terms on the signature. Services can be defined in separate libraries.

Example : Services and requests examples:

We define two services which manipulate some matrices:
$s_1(x : Matrix, y : Matrix) = x * y$
$s_2(x : Matrix, y : Matrix, z : Matrix, u : Matrix, w : Matrix) = x * (y * w) + z * u$

We are looking for a service, or combination of services, which allow to add two matrices:
$a, b : Matrix,\ a + b$

3.2.3 Matching algorithm

Based on equational unification, and thanks to the work of Gallier and Synder [6], a matching algorithm has been developed. It allow to find the services and combination of services that fulfill the user requirements.

Example : The user wants to solve the linear system with multiple right-hand side members $Ax = B$ (where no property is known about A). We assume that the BLAS and LAPACK libraries are available. One answer computed by the trader is:

```
p1=A;
p2=(ipiv);
sgetrf(nbRow(p1),nbCol(p1),p,nbRow(p1),p2,(info));
//p2<-LU factorisation (A= P*L*U)?
p3=B;
slaswp(nbCol(p3),p3,nbRow(p3),(k1),(k2),p2,1);
//p3<- line swap B
p4= p3;
strsm('l', 'l', 'n', 'n',nbRow(p4),nbCol(p4),1.0,
      p1,nbRow(p1),p4,nbRow(p4));
//solve L*x=p3; p4<-x;
p5= p4;
strsm('l','u','n','n',nbRow(p5),nbCol(p5),1.0,
      p1,nbRow(p1),p5,nbRow(p5));
//solve U*x=p4; p5<-x;
p5;
```

This is nothing else that what is provided by the expert routine SGESV from LAPACK which validates our approach.

3.2.4 A Grid-Aware Web Interface

We have also develop a Grid-aware Web interface for linear algebra tasks combined with this advanced service trading [2]. Developing efficient and portable codes, requires users to face parallel computing and programming and to make use of different standard libraries, such as the BLAS, LAPACK and ScaLAPACK in order to solve computational tasks related to linear algebra. For this purpose, a scientific computing environment based on a Web interface has been developed. It allows users to perform their linear algebra tasks without explicitly calling the above mentioned libraries and software tools, as well as without installing any piece of software on local computers: users enter algebraic formula (such as in Matlab or Scilab) that are evaluated for determining the combinations of services answering the user request. Services are then executed locally or over the Grid using the Distributed Interactive Engineering Toolbox (DIET) middleware depending on the problem size.

4. CONCLUSION

We have seen in this paper that the description of service is an important issue in some applications. There is not ONE solution that can suite all the cases. We should adapt to the level description needed. We should also adapt depending on how precise should be the description, how fast should be the matching algorithm, should we manage combination or not...

In the first example, combination of services is not needed, then a description based on meta-data is well adapted.

In the second one, we are looking for the service or the combination of services satisfying a user resquest, thus the description used should be precise enough to find the value of the parameter and more formal (to be able to reason on).

REFERENCES

[1] Grigoris Antoniou and Frank van Harmelen. Web ontology language: Owl. In S. Staab and R. Studer, editors, *Handbook on Ontologies in Information Systems*. Springer-Verlag, 2003.

[2] Hrachya Astsatryan, Vladimir Sahakyan, Youri Shoukouryan, Michel Daydé, Aurélie Hurault, Marc Pantel, and Eddy Caron. A Grid-Aware Web Interface with Advanced Service Trading for Linear Algebra Calculations. In *International Meeting High Performance Computing for Computational Science (VECPAR), Toulouse, 24/06/2008-27/06/2008*, 2008.

[3] Giuseppe Castagna, Giorgio Ghelli, and Giuseppe Longo. A calculus for overloaded functions with subtyping. In *Proceedings of the ACM Conference on Lisp and Functional Programming*, volume 5, pages 182–192, 1992.

[4] Michel Daydé, Frédéric Desprez, Aurélie Hurault, and Marc Pantel. On deploying scientific software within the GRID-TLSE project . *Computing Letters*, 1(3):85–92, juillet 2005.

[5] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[6] Jean Gallier and Wayne Snyder. Complete Sets of Transformations for General E-Unification. *Theor. Comput. Sci.*, 67(2-3):203–260, 1989.

[7] Deborah McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview, W3C Recommendation, 2004.

[8] M. Pantel. Test of Large Systems of Equations on the Grid: Meta-Data for Matrices, Computers, and Solvers. In *PMAA'04*, 2004.

[9] Mark Stickel, Richard Waldinger, Michael Lowry, Thomas Pressburger, and Ian Underwood. Deductive composition of astronomical software from subroutine libraries. In *Conference on Automated Deduction*, pages 341–355, 1994.