Automatic Composition of Process-based Web Services: a Challenge

Daniela Berardi, Giuseppe De Giacomo, Massimo Mecella
Università di Roma "La Sapienza"
<lastname>@dis.uniroma1.it

Diego Calvanese
Libera Università di Bolzano/Bozen
calvanese@inf.unibz.it

1. INTRODUCTION

Web Services are software platform-independent applications that export a description of their functionalities and make it available using standard network technologies. Such standards allow developers to access applications deployed over the network based on what they do, rather than on how they do it, or how they have been implemented. Thus, Web Services promise to give rise to new opportunities in developing and deploying distributed software applications. One of the most challenging and hyped research issues posed by Web Services concerns with their composition. It consists of two related problems, *synthesis* and *orchestration*. Currently, basic forms of composition are realized, when the Web Services are "statically" described through *programmatic interface*. Indeed, several tools are available that allow interaction among such heterogenous Web Services, by integrating and wrapping them into newly synthesized applications, usually expressed as simple scripts. The external user is not aware that the application he is using is actually composed of several heterogenous Web Services, since all the integration logic is hidden to him. When these new applications are executed, component Web Services are orchestrated, i.e., they are suitably invoked, by following the script specification. When Web Services are captured by *semantically rich descriptions*, synthesis and orchestration become more compelling, intuitively: (*i*) given a set of component Web Services and a user specification, the synthesis deals with coordinating the component Web Services in order to obtain (the specification of) a process (to be possibly offered in turn as a new Web Service) that exactly matches or that is "as close as possible" to the user specification; (*ii*) the orchestration deals, as before, with executing such a new process.

Several open issues, including foundational ones need to be investigated. Up to now, there does not exist a definition of Web Service composition synthesis that is consolidated and that is common to all approaches, since each approach to composition synthesis has its own definitions and assumptions. This regards, in particular, *automatic* composition synthesis. Therefore, due to the absence of a common vision, it is extremely difficult to compare the various approaches to composition synthesis: as a notable example, results on computational complexity of both the problem of composition synthesis, and the algorithms for composition synthesis are still lacking, and this inhibits practical and commercial developments of tools for composition. Last but not least, a clear and consolidated awareness of the relations between languages (and tools) for describing Web Services and composition synthesis techniques is not present. On the contrary, the problem of orchestration is much better understood, and several mature proposal are available in the literature. Indeed, the research community is well ahead in solving technological issues that are at the basis of Web Service composition.

From the discussion above it stems that, currently, composition synthesis is mainly achieved manually by a human designer. Also, nowadays, the web is populated primarily by applications targeted towards humans. However, Web Services are meant to be used by other applications (and possibly other Web Services), and not only by humans. There is a common understanding (see, e.g., [1, 2]) that to make a leap from Web Services being recognized and used through human intervention, to Web Services being recognized and used also by autonomous software applications, Web Services must export their "semantics". In other words, considering also their semantics will pave the way towards achieving *automatic* Web Service composition.

Several research efforts are leading towards the definition of semantics of Web Services. However, most of such efforts are concentrating on "static" aspects.

We believe that, besides static aspects, the semantics exported by a Web Service must also include descriptions of the process the Web Service carries out. Such "dynamic", or behavioral, aspects of the semantics are crucial in fully understanding what a Web Service does in order to be recognized and used by autonomous software applications.

2. WEB SERVICE COMPOSITION AS PROCESS SYNTHESIS

To understand the kind of problems we want to tackle and the approach to do so we present a explanatory scenario.[1] A researcher wants to arrange for a participation to a conference: she would like to register to a conference, book for a hotel room and arrange for the travel; in particular, the researcher wants to buy either a train ticket or a plane ticket, making the decision during the Web Service execution. Therefore, she connects to a publicly accessible (advanced) repository of Web Services, where she specifies her request and in her turn she receives a Web Service that realizes it (if one exists). Such returned Web Service should interact with the researcher, and allow her to make some choices, which may depend on the results of previously executed interactions. Note that, in general, it is likely to happen that the researcher request cannot be realized by any *single* Web Service, but by a *set* of component Web Services, suitably coordinated: however the researcher is unaware whether the Web Service she is executing is composite or not. Consider the situation when (a software module of the advanced Web Service repository finds out that) the researcher request can be realized by coordinating the following set of compo-

WWW2005, May 10–14, 2005, Chiba, Japan.

[1]The authors would like to thank Rick Hull for useful discussion about this example.

nent Web Services: `Conference Registration` that allows (*i*) first for registering to the conference, (*ii*) then for booking an hotel, and (*iii*) finally for charging the researcher credit card with a suitable amount; `Travel Information` that gives information about a specific location (i.e., weather in specific period, distance from a given location, etc.); `Travel Arrangement` that allows for buying either a plane or a train ticket. Note that a correct execution of the `Conference Registration` Web Service consists of all the three operations *(i)*, *(ii)* and *(iii)*, performed in the specified order. A possible composite Web Service that realizes the researcher request can be as follows: first it allows the researcher for registering to the conference, by executing the first operation of `Conference Registration` Web Service, then it lets she invoking `Travel Information`: on the basis of the information returned, the composite Web Service lets her to choose whether to buy a plane or a train a ticket with `Travel Arrangement` Web Service. Finally, having decided the dates for travel, she is allowed to book the hotel room and to give her credit card number for being charged, by executing the remaining operations of `Conference Registration` Web Service.

The example shown above cannot be fully achieved by exploiting current technology and academic research results since they address description of Web Services by largely focusing on static Web Service composition mechanisms, and therefore making use of semantically poor Web Service representation languages. Current technology such as UDDI Web Service registry supports discovery of (interface based) Web Services, where the search is done only by specifying the name of the Web Service.

We believe that the basic weakness of most of the research efforts proposed so far is that Web Services *do not share a full understanding of their semantics*, expressed in terms of their *exported* behavior, i.e., all the possible sequences of operations that a Web Service performs and quality features of these sequences. Indeed, some approaches to Web Service description consider behavior, but it is not publicly available, therefore, it is not exploited in composition, discovery, etc. Richer models need to be identified in order to characterize Web Services in terms of their exported dynamic behavior. Such richer semantics opens new challenging research directions and makes existing ones more compelling: this regards, for example, new forms of Web Service composition (i.e., based on interleaving component Web Services, as in the example above, and not simply on sequencing them), and more complex forms of dataflow within and among Web Services (e.g., the output of a Web Service goes in input to another one). Also, the discovery phase is largely affected. Indeed, we envision an advanced Web Service registry providing support for semantic discovery, i.e., where the Web Service search is done by considering user specification involving Web Service capabilities and behavior. Indeed, the ultimate goal of Web Services is to enable organizations to seamlessly compose business processes and dynamically integrate them with the partners' processes, by means of lightweight workflow-like technologies. Therefore, the Web Service composition framework will form a conceptual basis to define how internal business processes can be dynamically integrated with those of other organizations as value-added Web Services.

The core functionality of a Web Service based system is to compose available Web Services in order to fulfill clients requests. Once we take into account the behavior of Web Services such a task become essentially a task of program synthesis. It is well known since the work of Manna&Waldiger [9] in the late seventies, that synthesis of programs is actually one of the most difficult tasks tackled in Commuter Science. Results in the years have been generally quite discouraging. There are however specialized fields in which program synthesis has been successful. One prominent example is Planning. Indeed Planning is a form of program synthesis in which, starting from atomic instructions, the so-called called "actions" or "operators", that abstract in a suitable way details of the behavior that we do not want to model, and a client specification of what to be obtained, a program of a simple form (a sequence or a tree) is synthesized. Another, possibly less obvious, example of successful automated synthesis is Query processing especially in Data Integration. Also in this case programs have a special form (they are queries) and starting from constrains of the available programs (the query/views that define the available data) and a client specification of what required (again a query expressing the information needs of the client) we want to synthesize a new query that uses only the ones for which the data are available.

We believe that Web Service composition is another application domain where automated synthesis may become successful. Indeed we believe that the description that a Web Service exports of its behavior must abstract from most realization details, and that is it exactly this abstraction level that can be leveraged to find out suitable automated Web Service synthesis techniques.

Currently, the first results on automatic composition of process based Web Services are those in [10, 4, 3, 11]. The interested reader can refer to [5] for a comprehensive overview.

The last point we want to discuss here regards the distinction between data and process that often shows up in the Web Service literature. Indeed we have two extremes in dealing with data and process. One end of the spectrum is well explored by the literature on data integration that fully takes into account the data, but not the process [7, 8]. Interestingly, there are some proposals that base Web Service composition for data intensive Web Services on such a literature, avoiding to talk about the process as much as possible [6]. The other end of the spectrum is much less studied. Our proposal, together with those in [4, 10, 11], tries to explore such an end of the spectrum. More generally, both ends of this spectrum (only data and only process) deal with problems that are quite difficult. Finding a good way to integrate the two, without multiplying the complexities, is probably going to become one of the key problems in Web Service composition in the future.

3. REFERENCES

[1] A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. DAML-S: Web Service Description for the Semantic Web. In *Proceedings of the 1st International Semantic Web Conference (ISWC 2002)*, volume 2342 of *LNCS*, pages 348–363. Springer, 2002.

[2] D. Berardi. *Automatic Service Composition. Models, Techniques and Tools.* PhD thesis, Università di Roma "La Sapienza", 2005.

[3] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic Composition of *e*-Services that Export their Behavior. In *Proceedings of the 1st International Conference on Service Oriented Computing (ICSOC 2003)*, volume 2910 of *LNCS*, pages 43–58. Springer, 2003.

[4] T. Bultan, X. Fu, R. Hull, and J. Su. Conversation Specification: A New Approach to Design and Analysis of E-Service Composition. In *Proceedings of the 12th International World Wide Web Conference (WWW 2003)*, pages 403–410. ACM, 2003.

[5] G. De Giacomo and M. Mecella. Service Composition. Technologies, Methods and Tools for Synthesis and Orchestration of Composite Services and Processes. Tutorial at 2nd International Conference on Service Oriented Computing (ICSOC 2004), 2004.

[6] S. Ghandeharizadeh, C. A. Knoblock, C. Papadopoulos, C. Shahabi, E. Alwagait, J. L. Ambite, M. Cai, C. Chen, P. Pol, R. Schmidt, S. Song, S. Thakkar, and R. Zhou. Proteus: A System for Dynamically Composing and Intelligently Executing Web Services. In *Proceedings of the 2003 IEEE International Conference on Web Services (ICWS 2003)*. IEEE Computer Society Press, 2003.

[7] A. Halevy. Answering queries using views: A survey. *Very Large Data Base Journal*, 10(4):270–294, 2001.

[8] M. Lenzerini. Data Integration: A Theoretical Perspective. In *Proceedings of the 21nd ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002)*, pages 233–246. ACM, 2002.

[9] Z. Manna and R. Waldinger. A Deductive Approach to Program Synthesis. *ACM Transactions on Programming Languages and Systems*, 2(1):90–121, 1980.

[10] S. McIlraith, T. Son, and H. Zeng. Semantic Web Services. *IEEE Intelligent Systems*, 16(2):46 – 53, 2001.

[11] P. Traverso and M. Pistore. Automated Composition of Semantic Web Services into Executable Processes. In *Proceedings of the Third International Semantic Web Conference*, pages 380–394, 2004.