Automatic Web Service Composition: Service-tailored vs. Client-tailored Approaches [1]

Daniela Berardi, Giuseppe De Giacomo, Massimo Mecella [2] and **Diego Calvanese** [3]

1 Introduction

Web services (also called simply services) are self-describing, platform-agnostic computational elements that support rapid, low-cost and easy composition of loosely coupled distributed applications. From a technical standpoint, Web services are modular applications that can be described, published, located, invoked and composed over a variety of networks (including the Internet): any piece of code and any application component deployed on a system can be wrapped and transformed into a network-available service, by using standard (XML-based) languages and protocols (e.g., WSDL, SOAP, etc.) - see e.g., [1]. The promise of Web service is to enable the composition of new distributed applications/solutions: when no available service can satisfy a client request, (parts of) available services can be composed and orchestrated in order to satisfy such a request. Note that service composition involves two different issues [1]: the *synthesis*, in order to synthesize, either manually or automatically, a specification of how coordinating the component services to fulfill the client request, and the *orchestration*, i.e., how executing the previous obtained specification by suitably supervising and monitoring both the control flow and the data flow among the involved services. In this short position paper, we argue (Sec. 2) that most proposals on automatic composition synthesis are service-tailored and only few are client-tailored. We then outline (Sec. 3) the specific approach developed in the years by the authors, which is good example of a client-tailored approach.

2 Automatic Service Composition

In order to discuss *automatic* service composition, and compare different approaches, we introduce here a sort of conceptual framework for "semantic service integration", that is constituted by the following elements [4]:

- the *community ontology*, which represents the common understanding on an agreed upon reference semantics between the services [5], concerning the meaning of the offered operations, the semantics of the data flowing through the service operations, etc;
- the set of *available services*, which are the actual Web services available to the community;
- the *mapping* for the available services to the community ontology, which expresses how services expose their behavior in terms of the community ontology;
- and the *client service request*, to be expressed by using the community ontology.

In general, the community ontology comprises several aspects: on one side, it describes the semantics of the information managed by the services, through appropriate semantic standards and languages (e.g., OWL and OWL-S [6], WSMO [7]); on the other side, it should consider also some specification of the service behaviors, on possible constraints and dependencies between different service operations, not limited solely to pre- and post-conditions, but considering also the process of the service. In building such a "semantic service integration" system, two general approaches can be followed.

- In the **Service-tailored** approach, the community ontology is built mainly taking into account the available services, by suitably reconciling them; indeed the available services are directly mapped as elements of the community ontology, and the service request is composed by directly applying the mappings for accessing concrete computations.
- Conversely in the **Client-tailored** one, the community ontology is built mainly taking into account the client, independently from the services available; they are described (i.e., mapped) by using the community ontology, and the service request is composed by reversing these mappings for accessing concrete computations.

In fact, most of the research on automatic service composition has adopted, up to now, a service-tailored approach. For example, the works based on Classical Planning in AI (e.g., [27], [7]) consider services as atomic actions – only I/O behavior is modeled, and the community ontology is constituted by propositions/formulas (facts that are known to be true) and actions (which change the truth-value of the propositions); available services are mapped into the community ontology as atomic actions with pre- and post-conditions. In order to render a service as an atomic action, the atomic actions, as well as the propositions for pre- and post-conditions, must be carefully chosen by analyzing the available services, thus resulting in a service-tailored approach.

Other works (e.g., Papazoglou's et al. [28], Bouguettaya at al. [20], Sheth et al. [10, 9]) have essentially considered available services as

[1] This work has been supported by the Italian project *MAIS* and the EU projects/NoEs *TONES*, *SemanticGOV* and *INTEROP*.

[2] *Dipartimento di Informatica e Sistemistica*, *Università di Roma "La Sapienza"*, Via Salaria 113, 00198 Roma, Italy. E-mail: {berardi,degiacomo,mecella}@dis.uniroma1.it.

[3] *Libera Università di Bolzano/Bozen*, *Facoltà di Scienze e Tecnologie Informatiche*, Piazza Domenicani 3, 39100 Bolzano, Italy. E-mail: calvanese@inf.unibz.it.

[4] Such a framework is inspired by the research on "semantic data integration" [18, 12, 26]. Obviously that research has dealt with data (i.e., static aspects) and not with computations (i.e., dynamic aspects) that are of interest in composition of services. Still many notions and insights developed in that field may have a deep impact in service composition. An example is the distinction that we make later between "service-tailored" and "client-tailored" service integration systems, which roughly mimic the distinction between Global As View (GAV) and Local As View (LAV) in data integration.

[5] Note that many scenarios of cooperative information systems, e.g., *e*-Government or *e*-Business, consider preliminary agreements on underlying ontologies, yet yielding a high degree of dynamism and flexibility.

[6] cfr. http://www.daml.org/services/owl-s/.

[7] cfr. http://www.wsmo.org/.

atomic actions characterized by the I/O behavior and possibly effects. But differently from those based on planning, instead of concentrating on the automatic composition, they have focused more on on modeling issues and automatic discovery of services described making use of rich ontologies.

Also the work of McIlraith at al. [19] can be classified as service-tailored: services are seen as (possibly infinite) transition systems, the common ontology is a Situation Calculus Theory (therefore is semantically very rich) and service names, and each service name in the common ontology is mapped to a service seen as a procedure in Golog/Congolog Situation Calculus; the client service request is a Golog/Congolog program having service names as atomic actions with the understatement that it specifies acceptable sequences of actions for the client (as in planning) and not a transition system that the client wants to realize.

Finally, the work by Hull et al. [8, 14] describes a setting where services are expressed in terms of atomic actions (communications) that they can perform, and channels that link them with other services. The aim of the composition is to refine the behavior of each service so that the conversations realized by the overall system satisfy a given goal (dynamic property) expressed as a formula in linear time logic. Although possibly more on choreography synthesis than on composition synthesis of the form discussed here, we can still consider it a service-tailored approach, since there is no effort in hiding the service details from the client that specifies the goal formula.

Much less research has been done following a client-tailored approach, but some remarkable exceptions should be mentioned: the work of Knoblock at al. [21] is basically a data integration approach, i.e., the community ontology is the global schema of an integrated data system, the available services which are essentially data sources whose contents is mapped as views over the global schema, and the client request is basically a parameterized query over such a schema; therefore the approach is client-tailored, but neither the ontology nor mappings consider service behavior at all.

The work of Traverso et al. [25, 23, 22] can be classified also as client-tailored: services are seen as (finite) transition systems, the common ontology is a set of atomic actions and propositions, as in Planning; a service is mapped to the community ontology as a transition system using the alphabet of the community and defining how transitions affect the propositions, and the client service request asks for a sequence of actions to achieve GOAL1 (main computation), with guarantees that upon failure GOAL2 is reached (exception handling).

Finally, the line of research taken in [3, 4, 5, 6], but also in [11], has the dynamic behavior of services at the center of its investigation. In order to study the impact of such dynamics on automatic composition, all these works make simplifying assumptions on the community ontology, which essentially becomes an alphabet of actions. Still the notion of community ontology is present, and in fact all these works adopt a client-tailored approach.

A fundamental issue that arises is: if such rich descriptions of the dynamic behavior of the services can be combined with rich (non propositional) descriptions of the information exchanged by the services, while keeping automatic composition feasible. The first results on this issue are reported in [2], where available services that operate on shared world description (in a form of a database) are considered. Such services can either operate on the world through some atomic processes as in OWL-S, or exchange information through messages. While the available services themselves are with finite states, the world description is not. Under suitable assumptions on how the world can be queried and modified, decidability of service composition is shown. Interestingly [2] shows that even if the available services can be modeled as deterministic transition systems, the presence of a world description whose state is not known at composition time, requires dealing with nondeterminism of the same form we have studied here.

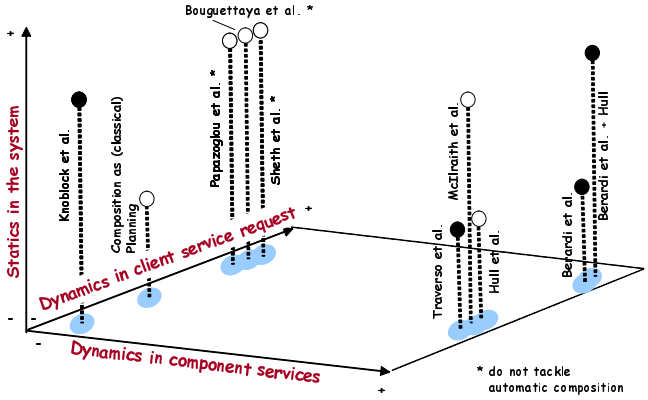

Figure 1. Comparison of the various approaches

Figure 1 summarizes, on the basis of the previous discussion, the considered works. The three axis represent the levels of detail according to which the community ontology and the mappings and the client request can be modeled. Namely, *(i) statics in the system* represents how fine grained is the modeling of the static semantics (i.e., ontologies of data and/or services, inputs and outputs, alphabet of actions, etc.); *(ii) dynamics in component services* represents how fine grained is the modeling of the processes and behavioral features of the services (only atomic actions, transition systems, etc.); and *(iii) dynamics in client service request* represents how fine grained is the modeling of the process required by the client, varying from a single step (as in the case of services consisting essentially of queries over a data integration system) to a (set of) sequential steps, to a (set of) conditional steps, to including loops, up to running under the full control of the client (as in our approach). *Black/white lollipops* represent service-tailored (white) vs. client-tailored (black) approaches.

3 The Roman Approach

The approach to automatic service composition undertaken by the authors, referred to as the *Roman approach* in [15], is an example of a client-tailored approach. Its distinguished features can be summarized as follows.

- The available services are grouped together into a so call *community*.
- Services in the community share a common set of actions Σ, the *actions of the community*. In other words, each available service in the community exports its behavior to the community itself in terms of the actions in Σ.
- Each action in Σ denotes a (possibly complex) interaction between the a service and a client, and as a result of such interaction the client may acquire new information (not necessarily modeled explicitly) that may be of help in choosing the next action to perform.
- The behavior of each available service is described in terms of a *finite transition system* (aka finite state machine) that makes use of the actions in Σ.
- The client request itself is expressed as a finite transition system that makes use of the actions in Σ. Such a transition system, called *target service*, is deterministic, since we assume that there is no uncertainty on the behavior that the client want to realize through composition of the available services.

- The orchestrator has the ability of scheduling services on a *step-by-step* basis. Hence the orchestrator has the ability of *controlling the interleaving* of multiple services executed concurrently.
- The *composition synthesis* consists on synthesizing a program for the orchestrator such that by suitably scheduling the available services it can provide the target service to the client.

To fix the idea, this setting for service composition can be understood in terms of the previously proposed framework as follows:

- the community ontology is simple a set of actions, namely the actions of the community;
- the available services are the actual Web services that have joined the community;
- the mapping from the available services to the community ontology is constituted by the transition systems that represent the available services; note that indeed these are expressed in terms of the actions of the community;
- the client service request is the target service, which again is expressed in terms of the actions of the community.

Therefore this setting adheres to the client-tailored approach.

In [3, 5], we have addressed the simplest case in which available services are modeled as deterministic finite transition systems, then in [4] we have considered the "angelic non/determinism" extension, i.e., the target service is "under-specified" (the client delegates the composer to resolve certain choices) and interactions between services without client involvement can happen. Finally in [6], we have addressed the automatic composition synthesis when the behavior of the available services is nondeterministic, and hence is not fully controllable by the orchestrator ("diabolic non/determinism").

The presence of nondeterministic conversations stems naturally when modeling services in which the result of each interaction with its client on the state of the service can not be foreseen. Let us consider as an example, a service that allows buying items by credit card; after invoking the operation, the service can be in a state `payment_OK`, accepting the payment, or in a different state `payment_refused`, if the credit card is not valid, with not enough credit, etc. Note that the client of a nondeterministic service can invoke the operation but cannot control what is the result of it. In other words, the behavior of the service is partially controllable, and the orchestrator needs to cope with such partial controllability. Note also that if one observes the status in which the service is after an operation, then s/he understand which transition, among those nondeterministically possible in the previous state, has been undertaken by the service. We assume that the orchestrator can indeed observe states of the available services and take advantage of this in choosing how to continue a certain task [8].

Typically reactive process synthesis [24, 17] make use of techniques based on automata on infinite trees. Even if these are perfectly suitable from a theoretical point of view, there are critical steps, such a Safra's construction for complementation, that have resisted efficient implementation for a long time. Only recently, we are starting to understand how to avoid such steps – see [16] for a discussion.

Interestingly the technique proposed by the authors are based on reduction to satisfiability in Propositional Dynamic Logic (PDL) [13] with a limited use of the reflexive-transitive-closure operator. Now, PDL satisfiability shares the same basic algorithms behind the success of the description logics-based reasoning systems used for OWL[9], such as FaCT[10], Racer[11], Pellet[12], and hence its applicability in the context of composition synthesis appears to be promising.

[8] The reader should observe that also the standard proposal WSDL 2.0 has a similar point of view: the same operation can have multiple output messages (the `out message` and various `outfault messages`), and the client observe how the service behaved only after receiving a specific output message.

[9] `http://www.omg.org/uml/`
[10] `http://www.cs.man.ac.uk/~horrocks/FaCT/`
[11] `http://www.sts.tu-harburg.de/~r.f.moeller/racer/`
[12] `http://www.mindswap.org/2003/pellet/`

REFERENCES

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services. Concepts, Architectures and Applications*. Springer, 2004.

[2] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, and M. Mecella. Automatic composition of transition-based semantic web services with messaging. In *Proc. VLDB 2005*.

[3] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic composition of e-Services that export their behavior. In *Proc. of ICSOC 2003*.

[4] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Synthesis of underspecified composite e-Services based on automated reasoning. In *Proc. of ICSOC 2004*.

[5] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic service composition based on behavioural descriptions. *International Journal of Cooperative Information Systems*, 14(4):333–376, 2005.

[6] D. Berardi, D. Calvanese, G. De Giacomo, and M. Mecella. Composition of services with nondeterministic observable behavior. In *Proc. of ICSOC 2005*.

[7] J. Blythe and J. Ambite, editors. *Proc. ICAPS 2004 Workshop on Planning and Scheduling for Web and Grid Services*, 2004.

[8] T. Bultan, X. Fu, R. Hull, and J. Su. Conversation specification: a new approach to design and analysis of e-service composition. In *Proc. of WWW 2003*.

[9] J. Cardose and A. Sheth. Introduction to semantic web services and web process composition. In *Proc. of SWSWPC 2004*.

[10] F. Curbera, A. Sheth, and K. Verma. Services oriented architecture and semantic web processes. In *Proc. ICWS 2004*.

[11] C. Gerede, R. Hull, O. H. Ibarra, and J. Su. Automated composition of e-services: Lookaheads. In *Proc. ICSOC 2004*.

[12] A. Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.

[13] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. The MIT Press, 2000.

[14] R. Hull, M. Benedikt, V. Christophides, and J. Su. E-services: a look behind the curtain. In *Proc. of PODS 2003*.

[15] R. Hull and J. Su. Tools for design of composite web services. In *Tutorial at SIGMOD 2004*.

[16] O. Kupferman and M. Y. Vardi. Safraless decision procedures. In *Proc. of FOCS 2005*.

[17] O. Kupferman and M. Y. Vardi. Synthesis with incomplete information. In *Proc. ICTL 1997*.

[18] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002*.

[19] S. McIlraith and T. Son. Adapting golog for composition of semantic web services. In *Proc. KR 2002*.

[20] B. Medjahed, A. Bouguettaya, and A. Elmagarmid. Composing web services on the semantic web. *Very Large Data Base Journal*, 12(4):333–351, 2003.

[21] M. Michalowski, J. Ambite, S. Thakkar, R. Tuchinda, C. Knoblock, and S. Minton. Retrieving and semantically integrating heterogeneous data from the web. *IEEE Intelligent Systems*, 19(3):72–79, 2004.

[22] M. Pistore, A. Marconi, P. Bertoli, and P. Traverso. Automated composition of web services by planning at the knowledge level. In *Proc. of IJCAI 2005*.

[23] M. Pistore, P. Traverso, P. Bertoli, and A. Marconi. Automated synthesis of composite bpel4ws web services. In *Proc. of ICWS 2005*, 2005.

[24] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. of POPL'89*, pages 179–190, 1989.

[25] P. Traverso and M. Pistore. Automated composition of semantic web services into executable processes. In *Proc. ISWC 2004*.

[26] J. D. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.

[27] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating daml-s web services composition using shop2. In *Proc. ISWC 2003*.

[28] J. Yang and M. Papazoglou. Service components for managing the life-cycle of service compositions. *Information Systems*, 29(2):97–125, 2004.