Automated Composition of Web Services: the ASTRO Approach

Annapaola Marconi, Marco Pistore, and Paolo Traverso
`FBK-irst - via Sommarive 18, 38100, Trento, Italy`
`[marconi,pistore,traverso]@fbk.eu`

With automated composition we mean generating an executable process that satisfies a given composition requirements by communicating with a set of existing Web services. Several approaches have been proposed to tackle this problem. However, most of them either omit or oversimplify important aspects of the Web service composition problem. The driving idea of the approach we're presenting in this paper is to overcome these limitations, in order to deal with real world composition problems. The ASTRO approach is able to cope with complex control and data flows, i.e., with Web services exposing complex protocols and exchanging structured data, and with composition requirements expressing constraints not only on the service interactions but also on the exchanged data. The ASTRO approach has been implemented and evaluated on real world composition domains.

1 Introduction

The ability to compose services, reducing development time and effort by re-using existing functionalities, is one of the most promising ideas underlying Web services. However, the complexity of service-based applications, the heterogeneity of the components, the dynamic nature of the environment, and the intrinsic distributed structure of the systems, make the manual development of the new composite application a difficult, error-prone and time-consuming task. Given this, techniques and methods allowing to automatically compose and adapt Web services are essential to substantially decrease time and costs in the development, integration, and maintenance of complex service oriented applications.

With automated composition we mean generating an executable process that satisfies a given composition requirements by communicating with a set of existing Web services, and that can be published itself as a Web service providing new higher level functionalities. Several approaches have been proposed to tackle this problem (e.g. [4, 2, 3, 12, 5]). However, most of them either omit or oversimplify several important aspects of the Web service composition problem. The main aim of the ASTRO approach is to overcome these limitations by providing an automated composition framework that is able to tackle real world Web service composition problems.

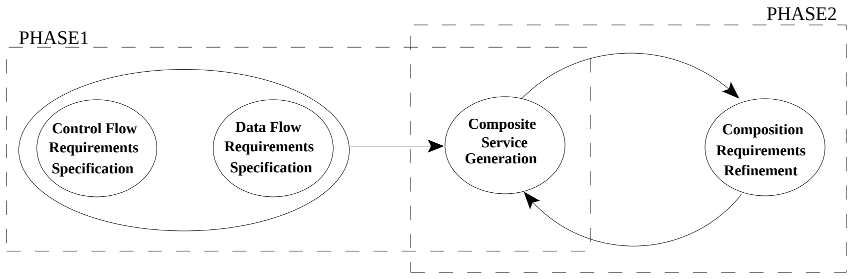
Among the most important characteristics provided by this approach are (i) the ability to consider component services that are complex stateful processes exhibiting an asynchronous and non deterministic behavior, (ii) the possibility to specify composition requirements specifying both data and control constraints on the execution of the new composite service, and (iii) the possibility to gradually refine the composition requirements and to iteratively re-generate a solution in a continuous semi-automated composition process.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

The ASTRO approach is implemented and incorporated into a prototype tool that supports all the phases of Web service automated composition: from the specification of control-flow and data-flow requirements by means of graphical tools for drawing data net diagrams and specifying control-flow requirements, to the automatic synthesis of the desired service, to the deployment, simulation, and execution of the new composite service. Using the prototype implementation, we evaluated our framework on a set of real-world case studies emerged from industrial applications and found in the literature. A significant example is the combination of Amazon on-line shopping services with on-line payment services provided by banks.

2 An Overview of the Approach

The ASTRO approach conceives the automated synthesis of the composite process as a step of a more complex iterative process that covers the different phases of the composition problem. In particular, the ASTRO composition process [8] consists of two phases (see Figure 1). The aim of the **first phase** is to obtain a preliminary version of the composite process starting from initial composition requirements. During this phase the developer analyzes the component service protocols (abstract WS-BPEL and WSDL) and specifies control flow and data flow requirements. Given the description of the component services and the requirements specification we automatically generate the internal executable composite process (executable WS-BPEL) and its user interface (WSDL and abstract WS-BPEL). This preliminary version of the composite service can be iteratively enhanced in the **second phase** of the process. During this phase the developer, on the basis of the automated composition outcomes, can refine both the composition requirements and the customer interface and automatically re-compose.


PHASE1
PHASE2
Control Flow Requirements Specification
Data Flow Requirements Specification
Composite Service Generation
Composition Requirements Refinement
Figure 1: ASTRO Web Service Composition Process

2.1 Specification of Composition Requirements

In order to cope with a wide range of composition problems we need a way to express requirements that define complex conditions, both for what concerns the behavior of the composition (termination conditions, failure recovery, transactional constraints) and for the data exchanged among the component services. Moreover, to make the automated composition an effective and practical task, the requirements specification should be as user-friendly and easy refining as possible.

We propose to separate the specification of data flow requirements from that of control flow requirements, and provide formal notations for their specification. In particular, in the control flow requirement specification step the developer defines termination conditions and transactional issues by exploiting minimal semantic annotations in the component service abstract WS-BPEL. Our approach, described in details in [9, 10], provides the developer with the ability to specify with a simple tabular notation these requirements that are then automatically translated into a formal internal notation that allows for the automation of the composition task. For what concerns requirements on data, we propose a formal language, the data net language [6], that allows to specify complex data flow composition requirements through an intuitive graphical notation. The data flow requirement specification step concerns the specification of how incoming messages must be used by the composite service (from simple forwarding to complex data manipulation) to obtain outgoing messages. During this step the developer also specifies messages received from and sent to the composite service user.

2.2 Automated Composition

Given the description of the component services and the composition requirements, the final step of the first phase is the development of the new composite service. The outcome of this completely automated phase is the executable WS-BPEL implementing the internal behavior of the new process and the description of the interaction protocol that the new service expects its customers to follow (WSDL and abstract WS-BPEL). For this automated synthesis task, the approach exploits sophisticated AI techniques for planning in asynchronous domains, extending them with new methods and algorithms in order to handle the peculiarities of the Web service automated composition problem. In particular, component services define the planning domain, composition requirements are formalized as planning goal, and planning algorithms are used to generate the composite service.

The formal framework, presented in [9], differs from other planning frameworks since it can deal with partial observable, non deterministic domains and asynchronous, message-based interactions between the domain (encoding the component services) and the plan (encoding the composite service). Moreover, the framework can handle complex transactional and termination requirements since it supports a goal language that allows to specify conditions of different strengths and preferences among different (e.g., primary and secondary) requirements. We extended this framework with new techniques and methods to overcome its limitation for what concerns the specification of data flow requirements and the encoding of data knowledge within the composition domain. Clearly, the data flow is as critical for the composition problem as the control flow, since the execution of a service is driven by the received and manipulated data. However, considering data in Web service composition has to deal with several problems: data domains are often infinite, and the semantics of data structures is complex (e.g. service messages are XML documents and service functions are XPath expressions). One of the key contributions is the possibility to handle the complex data flow composition requirements defined thorugh the data net language [6]. Moreover, we extended the framework with the K-level approach [11]: a novel abstraction-based approach for handling data, which ranges over an infinite domain, in a finite, symbolic way.

2.3 WS-Compose

The approach presented in this paper has been implemented as a prototype toolkit, namely WS-Compose, and integrated in the ASTRO Toolset [1], a toolkit providing an integrated environment for the composition of Web services. The ASTRO Toolset covers several aspects of the Web service composition process by providing tools and techniques supporting the analyst in the different phases (e.g. design time verification, run time monitoring, automated composition), and allows for the usage of industrial standards such as WSDL and WS-BPEL in the definition of Web services. For what concerns the automated synthesis of new services, WS-Compose supports all the phases of Web service automated composition: from the specification of control-flow and data-flow requirements by means of graphical tools for drawing data net diagrams and specifying control-flow requirements (WS-Req), to the automatic synthesis of the desired service (WS-Synth), to the deployment, simulation (WS-animator), and execution of the new composite service.

The ASTRO approach has been evaluated on a wide range of experimental domains, including real Web service composition domains. A significant example is the scenario that requires the composition of the Amazon E-Commerce Services and the e-payment service offered by Banks of Monte dei Paschi di Siena Group (MPS) [7]. The goal of the composition is to generate an *e-Bookstore* application that allows to order books and buy them via a secure credit card payment transaction. This composition scenario is particularly challenging since all component services export complex interaction protocols and handle structured data in messages. The following table shows the results of the eBookstore automated composition problem[1].

[1]The composition times have been obtained on a Pentium Centrino 1.6 GHz with 512 Mb RAM of memory running Linux

	Time (sec.)		WS-BPEL
	model construction	composition & emission	complex activities
E-BOOKSTORE	2.7	605.2	177

We distinguish between model construction time (translate the WS-BPEL component services into STS and encode the composition goal) and composition time (synthesize the composition and emit the corresponding executable WS-BPEL). The task of manually encoding and testing the same composition required several hours of work (more or less 20 hours).

The ASTRO approach has thus shown to be applicable also to this real domain providing a first positive answer to the question of the practical applicability of automated composition techniques.

3 Conclusions

Developing composite processes interacting with complex real world web services requires a time consuming analysis of the component services, both for what concerns their interaction protocol and the data structure of their messages. Moreover, it requires a detailed implementation of the new composite service that takes into account all the possible interaction evolutions (faults, exceptions). We propose an automated composition approach that can deal with real world composition problems and that dramatically reduces the effort for the composition by automatically generating both the internal executable composite process (executable WS-BPEL) and its user interface (WSDL and abstract WS-BPEL). Interesting features to be investigated in the future would be to extend the approach in order to handle *peer-to-peer* and *run-time* automated composition problems.

References

[1] ASTRO Project: *Supporting the Composition of Distributed Business Processes* - http://astroproject.org

[2] R. Akkiraju, B. Srivastava, A. Ivan, R. Goodwin, and T. Syeda-Mahmood. *SEMAPLAN: Combining Planning with Semantic Matching to Achieve Web Service Composition.* Proc. of IEEE International Conference on Web Services (ICWS'06), 2006

[3] D. Ardagna and B. Pernici. *Dynamic web service composition with QoS constraints.* International Journal of Business Process Integration and Management, V.1, N.4, 233-243, 2006

[4] D. Berardi, D. Calvanese, G. De Giacomo, and M. Mecella. *Composition of Services with Nondeterministic Observable Behaviour.* Proc. of International Conference on Service Oriented Computing (ICSOC'05), 2005

[5] R. Hull, M. Benedikt, V. Christophides, and J. Su. *E-Services: A Look Behind the Curtain.* Proc. PODS'03, 2003

[6] A. Marconi, M. Pistore, and P. Traverso. *Specifying Data-Flow Requirements for the Automated Composition of Web Services.* Proc. of Fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM06), 2006

[7] A. Marconi, M. Pistore, and P. Traverso. *Automated Web Service Composition at Work: the Amazon/MPS Case Study.* Proc. of IEEE International Conference on Web Services (ICWS07), 2007

[8] A. Marconi, M. Pistore, and P. Traverso. *An Iterative Approach for the Process level Composition of Web Services.* Workshop Proc. of 3rd South-East European Workshop on Formal Methods (SEEFM07), 2007

[9] M. Pistore, P. Traverso, and P. Bertoli. *Automated Composition of Web Services by Planning in Asynchronous Domains.* Proc. of the International Conference on Automated Planning & Scheduling (ICAPS05), 2005

[10] M. Pistore, P. Traverso, and P. Bertoli and A. Marconi. *Automated Synthesis of Composite BPEL4WS Web Services.* Proc. of IEEE International Conference on Web Services (ICWS05), 2005

[11] M. Pistore, A. Marconi, and P. Traverso and P. Bertoli. *Automated Composition of Web Services by Planning at the Knowledge Level.* Proc. of International Joint Conferences on Artificial Intelligence (IJCAI05), 2005

[12] S. McIlraith and S. Son. *Adapting Golog for Composition of Semantic Web Services.* Proc. of the Eighth International Conference on Knowledge Representation and Reasoning (KR'02), 2002