A Framework for personalized and dynamic composition of E-government web services

Hajar Elmaghraoui[1] Laila Benhlima[1] Dalila Chiadmi[1]

[1] Mohammed V[th] University-Agdal, Mohammadia School of Engineers (EMI)
Rabat, Agdal, Morocco

Abstract

E-government has emerged as a government policy to improve the quality and efficiency of administration and make it more proactive and especially more service oriented. To raise these challenges, Moroccan government agencies are providing a wide spectrum of online services to improve government-citizen interactions. However, single service may not be able to serve citizens needs, but a combination of services can serve the purpose. Building composite services can save significant time and cost for developing new applications and enhancing the interoperability and collaboration among Moroccan administrations. This paper proposes a Framework for personalized and dynamic composition of web services.

Keywords: Web service, Semantic, Dynamic composition, E-government.

1. Introduction

E-government is a powerful tool for the development of social and economic life of citizens. Many countries are experiencing its transformative power in revitalizing public administration and moving civil service towards higher efficiency, transparency and accountability. Availability of online public services has been the primary focus of E-government studies and policymaking, but over the past years, the definition of new services that outsources from other E-government services (composition process) and the citizen usage of E-gov services has also become a priority issue. It is for the purpose of facing all these challenges that e-Morocco Strategy for the development of information society and knowledge economy has been launched in January 2005. One of the main issues of this strategy is to insure interoperability between administrations and to provide services that are tailored to each citizen needs and profile. In its midway this strategy has already allowed remarkable progress. However, Morocco has been ranked at position 120 in the United Nations E-Government Survey 2012[1]. This low rank can be explained by the high cost of the implementation and the maintenance of e-services and the low rate of their use. This situation leads to a low return on investment of E-Gov applications. In addition, several reports have also highlighted the dissatisfaction of users of e-services they deem not adapted to their needs and contexts. To support the e-Morocco Strategy, we propose, in this paper, a Framework for personalized and dynamic composition of web services using an approach based on modeling the semantic relationship between all the available web services through a weighted directed graph built before the composition runtime.

The rest of this paper is structured as follows. Section 2 gives an architecture overview of our approach. Section 3 de-

fines the main components of DynaComp Framework. The last two sections present the related work and conclusions.

2. DynaComp Framework

As stated before, E-Gov allows efficient and transparent between citizens and administrations. This interaction may concern purchasing information or documents, filling forms to get some certificate or declare some events; it may also involve several administrations that cooperate to offer a complex service. In this context, our problem can be described as follows: "Given a set of public e-services, and given a citizen goal, our aim is to automatically compose an optimal subset of these services to satisfy the goal, taking into consideration the citizen preferences". We propose a solution based on Graph Theory.

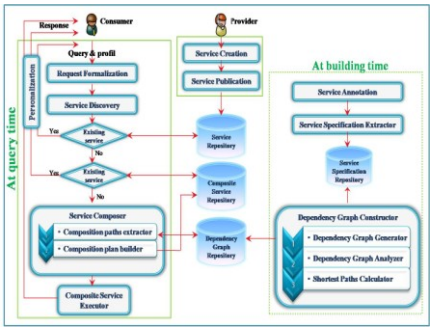

Fig. 1: DynaComp Framework

DynaComp framework (Figure 1) is a global solution that involves both administrations and citizens. In the Moroccan context, each administration has its own formalism to describe services. This leads to interoperability problems when trying to compose services owned by different administrations. To deal with this problem, DynaComp uses an interpreter to extract the needed information from repositories regardless of the used formalisms. The second advantage of our framework is optimization. Dynamic web service composition is known to be an expensive process. To reduce the cost, DynaComp performs composition before runtime execution. By maintaining a repository of pre-computed services compositions, the platform optimizes time execution. Last but not least, DynaComp return more satisfying e-services since it takes into consideration user's profile during the request and execution phase. Interoperability, automation, optimization and personalization are the key issues for a successful E-gov Framework.

3. DynaComp Components

3.1. Service Specification Extractor

This component contains an interpreter for the service description language WSDL that parses the WSDL document and collects the information needed to generate the services dependency graphs. This makes service's representation in our framework language-neutral. In fact services can be described in different languages, and we just need to add an interpreter for each one. The extracted information is then stored in the Service Specification Repository. This operation is done automatically each time new service is published or existing services are changed or removed from repository.

The main information we consider in this work are IOPEs (inputs, outputs, preconditions, and effects), goals and non-functional parameters.

3.2. Dependency Graph Constructor

Dependency Graph Generator: we model services compositions using a weighted directed graph [2]. Our focus is to identify explicit direct dependencies between services and construct Service Dependency Graph (SDG). In this work, we consider that a web service can be described by two sets of properties: functional and non-functional properties. Thereafter we mean by Inputs, service's inputs and preconditions and by Outputs, service's out-

puts and Effects. We start from services specifications stored in the Specification Repository. We use the semantic description of IOPE parameters to build the SDG (For more details see [3]).

Dependency Graph Analyzer: During this analysis phase, we find out all cyclic dependencies if there are any in the SDG and then we regenerate the graph by making each cyclic sub-graph as one compound node. The resulted acyclic dependency graph is stored in the Dependency Graph Repository and sent to the Shortest Graph Calculator.

Shortest Paths Calculator: Discovering a composite service essentially is searching for a solution in the solution space represented by the SDG. Our service composition research aims at reducing the complexity and time needed to generate and execute a composition. We also improve its efficiency by selecting the best services. To achieve these optimization goals, the Shortest Paths Calculator uses the Floyd-Warshall algorithm to precompute, at the building time, all shortest paths between every pair of vertices in the acyclic dependency graph obtained from the previous step. We then store these paths in the Dependency Graph Repository for an eventual use at the query time. Thus, we avoid building the graph and applying the graph search algorithm when receiving a user's request that requires services composition. When new services are published or existing services are changed or removed from the repository, the SDG is updated, the shortest paths are recalculated, and thus the Dependency Graph Repository is updated.

3.3. Request Formalization

A service request consists of a set of parameters (Goals, Inputs, Outputs, Preconditions and Effects). Each parameter of the service request refers to concepts of the Framework ontology. In order to refine the description of the required services, DynaComp takes into account user interests, preferences and context to enrich the service request.

3.4. Service Discovery

In DynaComp, Service Discovery consists of two stages :(i) the process of querying the Registry to find the best services that semantically match the request. If yes, then the service is executed and the personalized results will be returned to Consumer.(ii) If we fail to respond to the user request (or a part of it), we query the Composite Service Repository, which stores previously created composite service plans, and check if any existing plan can fulfill user's requirements. If yes, the identified composite service is confirmed and executed; the personalized results will be returned to Consumer. Otherwise, we use Service Composer.

3.5. Service Composer

Composition Paths Extractor: When a matching service is not found during the Discovery, the Composition Paths Extractor queries the Dependency Graph Repository to check if any existing composition path matches user's request(For more details, see [2]). Thus, we obtain the optimal sub-graph of services that meet the user's needs. Otherwise, we should report the failure to the requestor.

Composition Plan Builder: Once an optimal service composition path is extracted, it is passed on to the Composition Plan Builder, where the preparation and verification of composite service plan is performed. For all the services that are part of the service composition, the information about their inputs, outputs, preconditions and effects is checked automatically to assure that all needed input data are provided; that all operations can be executed; and that all links could be established. The generated composite service plan is stored in the "Composite Service Repository" for future reuse.

3.6. Composite Service Executor

When the planner comes up with a valid plan for the given request, the Composite Service Executor converts the plan into an executable process and executes it. Finally, the response is personalized based on the user profile parameters and then returned to Consumer.

4. Related Works

Many proposals of web services composition methods have been presented in recent years [3]. In this section, we present a brief overview of some techniques that deal with automatic web service composition. We consider only techniques that use service dependency information, graph models, and semantics.

Hashemian et al. [4] store I/O dependencies between services in the DG and then build composite services by applying a graph search algorithm. The authors consider only the matching and dependencies between input and output parameters without considering functional semantics, thus they cannot guarantee that the generated composite services provides the requested functionality correctly. Arpinar et al. [5] present an approach which not only use graphs for web service composition, but also use semantic similarity. They consider edges with weights and deploy a shortest-path dynamic programming algorithm based on Bellman-Ford's algorithm. The authors consider the execution time of each service and input/output similarity but they don't take into consideration the nonfunctional attributes of services. Talantikite et al. [6] propose to pre-compute and store a network of services that are linked by their I/O parameters. The link is built by using semantic similarity functions based on ontology. They represent the service network using a graph structure. Their approach utilizes backward chaining and depth-first search algorithms to find subgraphs of services to accomplish the requested task. They propose a way to select an optimal plan in case of finding more than one plan. However, they also create the graph at the composition time which incurs substantial overhead.

5. Conclusion

In this paper, we propose a Framework to customize and automate composition of E-government web services. We have described its main components and outlined their interactions. The implementation and evaluation of the proposed Framework in the domain of E-gov is the main focus of our ongoing work.

6. References

[1] http://unpan1.un.org/intradoc/groups/public/documents/un-dpadm/unpan048580.pdf

[2] H. Elmaghraoui, I. Zaoui, D. Chiadmi and L. Benhlima, "Graph based E-Government web service composition," IJCSI International Journal of Computer Science Vol. 8, July 2011

[3] A. Alamri, M. Eid and A. Elsaddik, "Classification of the state-of-the-art dynamic web services composition," International Journal of Web and Grid Services, 2006, Vol. 2, pp. 148-166.

[4] S. Hashemian and F. Mavaddat, "A graph-based framework for composition of stateless web services," Proceedings of ECOWS'06, IEEE Computer Society, Washington, DC, 2006

[5] I.B. Arpinar, B.A. Meza, R. Zhang and A. Maduko, "Ontology-driven web services composition platform," Inf. Syst. E-Business Management, 2005, 3(2):175–199.

[6] H.N. Talantikite, D. Aissani and N. Boudjlida, "Semantic annotations for web services discovery and composition," Computer Standards Interfaces, Elsevier, 2009.