Architectural Models for Client Interaction on Service-Oriented Platforms

Luiz Olavo Bonino da Silva Santos, Luís Ferreira Pires and Marten van Sinderen

University of Twente, Architecture and Services of Network Applications Group
P.O. Box 217, 7500 AE Enschede, the Netherlands

Abstract. Service oriented platforms can provide different levels of functionality to the client applications as well as different interaction models. Depending on the platform's goals and the computing capacity of their expected clients the platform functionality can range from just an interface to support the discovery of services to a full set of intermediation facilities. Each of these options requires an architectural model to be followed in order to allow the support of the corresponding interaction pattern. This paper discusses architectural models for service-oriented platforms and how different choices of interaction models influence the design of such platforms. Service platforms' functionality provisioning can vary from a simple discovery mechanism to a complete set, including discovery, selection, composition and invocation. This paper also discusses two architectural design choices reflecting distinct types of functionality provisioning namely matchmaker and broker. The broker provides a more complete set of functionality to the clients, while the matchmaker leaves part of the functionality and responsibility to the client, demanding a client platform with more computational capabilities.

1 Introduction

Service-Oriented Architecture (SOA) is a paradigm for software architectures that fosters the creation of complex systems by using distributed pieces of functionality (services) accessible through a set of standards. In this architecture, services are provided to clients by services providers. Clients search for services by browsing a list of available service descriptions stored in registries. A service description contains information about a service, such as what the service does, how to access the interface, and which information should be supplied in order to use the service properly, among others. After discovering a suitable service, the client invokes the service interface in accordance with the information contained in the service description.

In this scenario, each client should be able to search in the available registries, decide which service best fits its needs and invoke the service using the published interfaces. To facilitate these tasks, service platforms can play an intermediation role between client applications and service providers. On the provider's side, a service platform can be beneficial by providing a mechanism for rapid creation, deployment and advertisement of services. Examples of facilitators for these activities can be found in

Olavo Bonino da Silva Santos L., Ferreira Pires L. and van Sinderen M. (2007).
Architectural Models for Client Interaction on Service-Oriented Platforms.
In *Proceedings of the 1st International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing*, pages 19-27
DOI: 10.5220/0001349100190027

[9] and [10]. On the client's side, the platform can offer support for discovery, selection, composition and invocation of services, amongst others.

Focusing on the client's side, platform designers should choose the set of activities to be supported by the platform from the activities mentioned above. This choice influences the behavior of the platform and the interaction model between the platform and the clients. Therefore, an investigation of the possible platform behaviors concerning those interactions is necessary to identify platform requirements and architectural components. As a consequence, the platform roles in the interaction with the clients are identified.

In this paper we consider that the roles played by the service platform describe behavioral patterns followed by the platform regarding its interaction with client applications. Platform designers have at their disposal a variety of platform role's choices depending on what they intend the platform to support. A service platform acts similarly to the middle agents described in [5], in which the authors define several possible roles for the middle agents depending on how they solve the intermediation problem. In this paper we focus on two possible roles that illustrate opposite levels of functionality support: the *matchmaker*, which offers a simple discovery mechanism and, the *broker*, which offers more complete set of functionality, including discovery, selection, composition and invocation.

This paper is structured as follows: section 2 gives an overview of platform roles regarding client interaction, section 0 details the matchmaker role while section 4 details the broker role and section 5 concludes and points some future directions for this work.

2 Platform Roles

The architecture of service platforms can be defined in terms of the type of interaction to be offered to the client. Following this approach, the level of support to be offered and the choice of the platform role to be played determine the requirements for the service platform. Among the activities supported by service platforms we can include: discovery, selection, composition and invocation [1]. Here, we define as *level of support* the subset of aforementioned activities offered by the platform.

Among the different possible types of roles for a service platform we present in this paper two alternatives, namely *broker* and *matchmaker*. While the former offers a high level of support the later operates in a simpler way and leaves more responsibilities to the client application.

Fig. 1 shows the different interaction patterns applied in three service architectures. Architecture 1 is the basic architecture for Web Services where the service provider publishes the service descriptions in a registry, the client queries the registry for the descriptions of available services and, after selecting an available service, the client invokes the appropriate service. Architecture 2 places a matchmaker between the client and the registry. In this way the client sends the criteria of the desired service to the matchmaker, which searches the available registries for service descriptions matching these criteria. In the case of a positive match, the matchmaker returns the description of the discovered service to be invoked by the client. Architecture 3 presents an example of the broker role. In this example, the platform not only provides

matchmaking facilities but also invokes the discovered services on behalf of the client. If necessary, the platform also performs transformations on the results received from the invoked services to comply with the data format requested by the client.

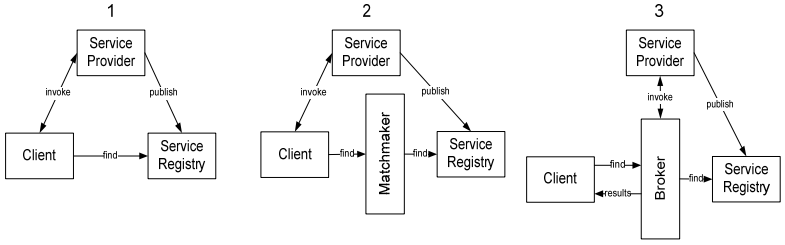

Fig. 1. Platform roles.

The matchmaker and the broker roles are discussed in more detail in the sequel.

3 Matchmaker

In the service matchmaking activity we have three distinct roles: a requester, a matchmaker and a provider [12]. The requester aims at finding services that offer the capabilities dictated to by criteria provided in terms of the desired service interfaces and properties [15]. The matchmaker has access to a set of services descriptions made available by providers and provides facilities to discover services based on the requester's criteria.

Early computational directories offer matchmaking facilities that provide mappings between names and addresses similarly to the white pages in telephone directories. Later on, a more advanced form of matchmaking emerged that supports search based on an entries' attributes allowing matches based on certain desired characteristics. This form of matchmaking resembles the yellow pages in telephone directories. One shortcoming of this approach is that the selection criteria are completely supplied by the requester, providing an asymmetric form of selection [13]. Work such as [14] suggests the introduction of symmetry in the selection process, in which the requester provides a description of the requested service and its capabilities as a client. The provider specifies its demand to the potential clients of its services. This allows the provider to select clients just as clients select services. This symmetrical matchmaking allows dynamic update of service descriptions at matchmaking time rather than at advertisement (publication) time.

A matchmaker acts like as if it were a provider of services of different providers. The requester does not have to interact with several providers or several service registries querying them for the descriptions of their services and then try to match the descriptions with the needed criteria. In a matchmaking environment, the requester sends the criteria to the matchmaker, which searches the services descriptions it has access to for a positive match. Having found services that comply with the given criteria, the matchmaker sends the service descriptions back to the client. The client then analyzes the services descriptions and selects suitable ones. After that, the client

directly interacts with the services by invoking the service operations and receiving results.

Fig. 2 depicts a sequence of interactions between the client, the service provider and the matchmaker platform.

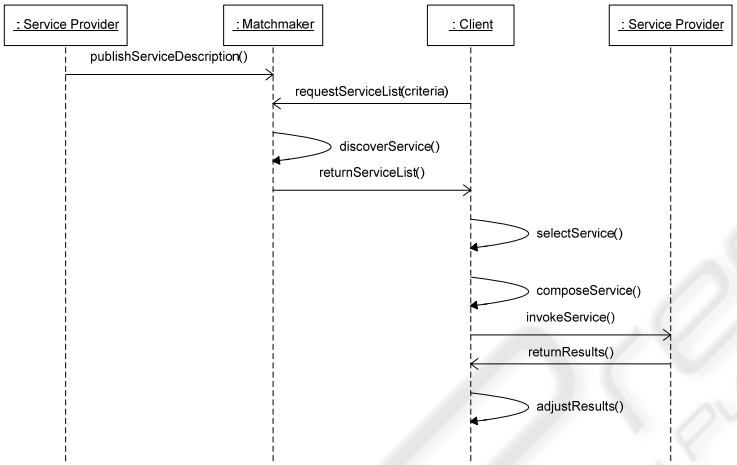

Fig. 2. Interactions in a matchmaking environment.

The level of support of the matchmaker indicates the architectural components of the platform. In case some functionality is not supported by the matchmaker, other applications or the client itself have to cover this functionality. Fig. 3 depicts a possible architecture for both the matchmaker and the client complying with the sequence diagram presented in Fig. 2. The matchmaker supports service publishing by the service provider through the *Service Publisher* component. The *Service Publisher* sends the received service description to the *Content Manager* component, which is responsible for storing it in an available registry (omitted in Fig. 3). The *Content Manager* shields the internal components from interactions with registries. The *Content Manager* can have access to several registries as well as acting as a client to other matchmakers.

A client requests the discovery of a service by calling the *Service Finder* component. The *Service Finder* requests a list of candidate services to the *Content Manager*. After receiving the list, the *Service Finder* tries to match the client's criteria against the candidate services to find the positive matches. If positive matches are found, the service descriptions are sent back to the client.

Since in this example the matchmaker does not support service composition, this task is expected to be performed by the client. Therefore, it is possible that the matches have been obtained by the partial satisfaction of the criteria, i.e., some of the properties given by the client have been satisfied but not completely by a unique service. In this case, the client needs to perform service composition by calling a *Service Composer* component. In Fig. 3 this component is internal to the client. The *Service Composer* tries to find a service composition that fully matches the criteria.

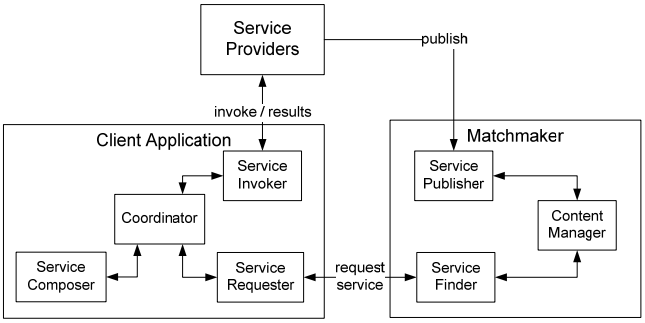

Fig. 3. Example of service composition performed by the client.

We consider now another example of matchmaker with a higher level of support than the one in Fig. 3. In this example, the service composition functionality is provided by the matchmaker. As it can be seen in Fig. 4, the *Service Composer* component has been moved from the client to the matchmaker but its functionality remains the same. The *Service Composer* still tries to compose the services that partially match the criteria into a service composition that fully matches the criteria. In case additional component services are required to complete the composition, the *Service Composer* needs to request the new services by providing other criteria. In the example where the service composition is performed by the client, the *Service Composer* requests the additional services to the *Coordinator* component. The *Coordinator* forwards the request to the *Service Requester* component which calls the *Service Finder* of the matchmaker with the new criteria. In the example shown in Fig. 4 where the composition is performed by the matchmaker, the criteria are passed by the *Service Composer* directly to the *Service Finder*.

The flexibility to assign functionality to the matchmaker or to the client shown in the example of service composition also holds for other functional components, such as the *Content Manager* or the *Service Finder*. Therefore, generic components can be developed to support these functions and the desired level of support for the service platform dictates where those components should be allocated.

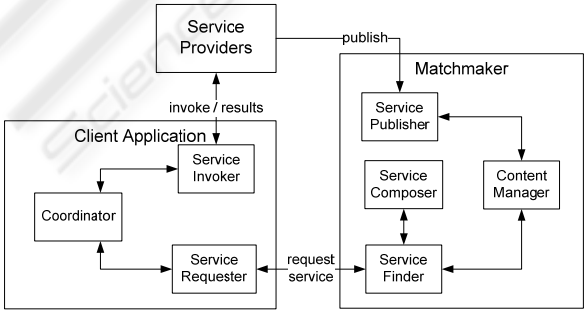

Fig. 4. Example of service composition performed by the matchmaker.

4 Broker

The following definitions of *broker* can be found:

- "An individual who gets buyers and sellers and helps in negotiating contracts for a client", Mortgage Magazine [2];
- "An agent who negotiates contracts of purchase and sale", Merriam Webster [3];
- "A person who buys and sells goods or assets for others", Oxford Dictionary [4].

Moreover, information brokerage can be loosely defined as a set of mediation capabilities and functions aiming to help sellers to broadcast or disseminate information about their products and services, and, at the same time, to assist the end users in order to better retrieve, select and compare the offered information about producers, products and services.

Mapping the definitions above to service-oriented computing we can define a broker service platform as *a platform that acts on behalf of a client application by discovering, selecting, composing and invocating services*. In this role, the platform offers a service selection mechanism, invokes the services on behalf of the client application, monitors the service execution and parses the results, possibly translating the output to client's required format. The broker can also perform service composition based on the client's service requirements and the service descriptions available to the platform.

Additional functionality can be assigned to the broker. For instance, if the service description contains semantic annotations, the platform should be able to perform a set of complex reasoning tasks [7], which includes interpreting service provider capabilities (service descriptions) and client applications' requirements. Moreover, the interpretation of the terms used in message exchanges can be performed by the platform.

Considering the broker role, an example usage scenario is the one in which the platform is available to clients that may request immediate provisioning of a service. The sequence diagram in Fig. 5 shows the interaction pattern between clients, service providers and the broker platform. Similarly to the matchmaker, the broker provides facilities for service publishing by the service providers. A critical difference between the matchmaker and the broker is that the latter acts as a surrogate of the client, i.e., the client requests the provisioning of a given service and the broker performs the necessary steps until the final result of the service, on the client's behalf. Even if the output request by the client is somewhat different from the output received by the broker after the invocation of the necessary services, the broker can perform a transformation to comply with the client's requirements. Examples of transformations are:

1. The client requests a service that returns the current temperature in Celsius for a given city. The broker finds a service that provides current temperatures of cities, but the output is in Fahrenheit instead of Celsius. In this case the broker can perform the output transformation by composing this service with another one that takes a temperature value in Fahrenheit and returns the value in Celsius;
2. The client requests a service that produces a given value in long number format. The broker finds the appropriate service but the output is in integer

number format. The broker can perform the transformation of the output by simply parsing the integer value into long and returning the transformed value to the client.

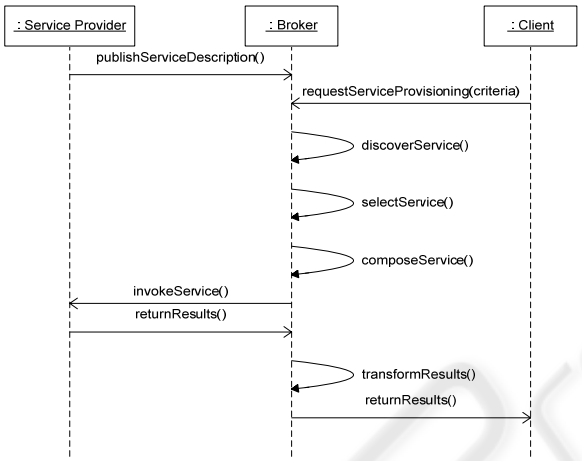

Fig. 5. Interaction pattern for the broker role.

Fig. 6 presents an example architecture of a broker. Here we see the components responsible for the functionality provided by the platform, namely the *Service Publisher* (service publishing), the *Content Manager* (service registry access and semantic repository access), the *Service Finder* (service discovery), the *Service Composer* (service composition) and the *Semantic Mediator* (semantic mediation).

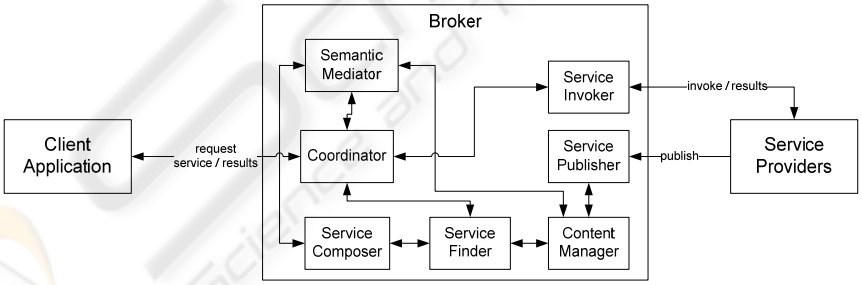

Fig. 6. Example architecture of a broker platform.

Additional characteristics of the broker platform role can be identified, such as:

- Fault tolerance and robustness: if a service becomes unavailable, the platform can try to find another suitable provider;
- Privacy, security and billing: since we assume that clients and service providers have agreed to trust the platform, the platform is the trusting central point for these entities. Therefore, clients do not have to directly interact

with services providers and vice-versa, and the platform can provide anonymization for both parties.

Nonetheless, being a centralized coordinator the platform can become a single point of failure as well. Techniques such as redundancy and clustering, among others, can be used to increase the platform's availability.

Unlike the matchmaker, the broker role has less room for exchanging functionality between the client and the platform. Although some auxiliary functionality, such as results transformations, could be placed on the client side, most of the functionality should remain on the platform side in order to preserve the surrogate characteristic of the broker.

5 Conclusions and Future Directions

In this paper we discuss the roles played by service platforms and the impact on the design of such platforms. To illustrate the discussion we present two choices of platform roles, namely the *matchmaker* and the *broker*. The main characteristics of each platform role are presented together with examples of architecture designs containing an overview of components providing the required functionality.

In this paper we identify the impact of the choice of platform role in the architectural design of the platform. In other words, the designer's choice of interaction pattern and the level of support of this platform implies in different assignments of architectural components to the clients and the platform. This paper addresses this impact for the matchmaker and broker platform roles. A platform design of the broker role for context-aware applications has been defined and proposed in [16].

Future directions of this work include the implementation of the suggested components and the instantiation of scenarios demonstrating the use of the presented platform roles. The exchange of functionality between client and the service platform, as suggested in the matchmaker examples, should be carried out and evaluated. Moreover, the requirements specification of both platforms (matchmaker and broker roles) should be detailed, in order to provide general guidelines for the design of such platforms.

Acknowledgements

The present work is co-funded by the Freeband Communication project A-Muse (http://a-muse.freeband.nl) and the Amigo Project. A-Muse is sponsored by the Dutch government under contract BSIK 03025. The Amigo project is funded by the European Commission as an integrated project (IP) in the Sixth Framework Programme under the contract number IST 004182.

References

1. Preist, C.: A Conceptual Architecture for Semantic Web Services. In Proceedings of the International Semantic Web Conference 2004 (ISWC 2004), pp. 395-409, November 2004.
2. Mortgage Magazine - http://www.mortgages-magazine.com/mortgage-glossary.htm.
3. Merriam Webster Dictionary – http://www.m-w.com.
4. Oxford Dictionary – http://www.askoxford.com.
5. Decker, K., Sycara, K., Williamson, M.: Middle-Agents for the Internet. In Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97), pp. 578-584, Nagoya, Japan, August 1997.
6. Chi Wong, H., Sycara, K.: A Taxonomy of Middle-Agents for the Internet. In Proceedings of the 4th International Conference on MultiAgent Systems (ICMAS 2000), pp. 465-466, Boston, MA, USA, July 2000.
7. Sycara, K., Paolucci, M., Soudry, J., Srinivasan, N.: Dynamic Discovery and Coordination of Agent-Based Semantic Web Services. IEEE Internet Computing, Vol. 8, n. 3, pp. 66-73, May/June 2004.
8. Piedad, F., Hawkings, M.: High Availability: Design, Techniques and Processes, Prentice Hall PTR, 1st Edition, December 2000.
9. Agarwal, V., et al. A Service Creation Environment Based on End to End Composition of Web Services. In Proceedings of the 14th International Conference on World Wide Web (WWW 2005), pp. 128-137, Chiba, Japan, 2005.
10. Srinivasan, N., Paolucci, M., Sycara, K., CODE: A Development Environment for OWL-S Web services. Technical Report CMU-RI-TR-05-48, Robotics Institute, Carnegie Mellon University, October, 2005.
11. Kawamura, T., et al, Web Services Lookup: A Matchmaker Experiment. IEEE IT Professional, vol. 7, n. 2, March/April 2005.
12. Decker, L., Williamson, M., Sycara, K., Matchmaking and Brokering. In Proceedings of the 2nd International Conference in Multi-Agent Systems (ICMAS'96), Kyoto, Japan, December 1996.
13. Facciorusso, C., et al, A Web Services Matchmaking Engine for Web Services. In Proceedings of the 4th International Conference on e-Commerce and Web Technologies, Prague, Czech Republic, September 2-5 2003.
14. Hoffner, Y., Facciorusso, C., Field, S., Schade, A., Distribution Issues in the Design and Implementation of a Virtual Market Place. Computer Networks: The International Journal of Computer and Telecommunications Networking, vol. 32, issue6, pp. 717-730, Elsevier North-Holland, New York, USA, May 2000.
15. Vausdevan, V., Augmenting OMG traders to handle service composition. Object Services and Consulting Inc., September 15 1998.
16. Bonino da Silva Santos, L.O., Semantic Services Support for Context-Aware Platforms, Master Dissertation, Universidade Federal do Espírito Santo, Vitória, Brazil, September 2004.