Service Orientation in Middleware Components for Scalable Service Marketplaces

Josef Spillner
Technische Universität Dresden
Faculty of Computer Science, Chair for Computer Networks
01062 Dresden, Germany
E-mail: josef.spillner@tu-dresden.de

Abstract – The purpose of marketplaces in an economic sense is the offering of a venue for selling and buying goods. On the internet, virtual marketplaces for trading goods have also been known since the 1990s. More recently, the marketplace idea has been extended towards the exchange of highly-specialised, custom services which are technically implemented as redistributable web services. As opposed to tradable goods which are sold at a discrete point in time, the service marketplace retains its functionality beyond the purchase through continuous contracting and management of services. This additional activity increases the amount of required resources. For a scalable exchange between thousands of users, the underlying system and middleware thus needs to adhere to scalability principles known from service-oriented architectures. This paper confirms the idea of representing the middleware as a set of distributed platform services which can themselves be offered on service marketplaces.

I. Principles of Service Orientation

The definition of the principles of service orientation is subject to contextual and chronological variations [1,2]. Often-mentioned core principles include:

- Loose coupling between services and their users. This allows users to exchange one service for another based on functional equality and non-functional preferences.
- Reusability. This software-technological aspect is important for the idea of service orientation on the middleware level, as it mandates the offering of parts of the middleware as actual services.
- Discoverability and composability. These two properties lead to the requirement of a user-centric approach which can be realised by a service marketplace.

The application of these principles to service marketplaces will yield highly scalable platforms with many users and parallel service execution. Both the interactive marketplace and the execution need sophisticated middleware in order to enable rapid development of higher-level applications.

II. Overview on Middleware functionality

A middleware's task is to mediate between services and systems. On service marketplaces, its functionality must support the provider and the consumer of services. Hence, it must offer interfaces for installing or referencing new services, for searching for services and for using them. This includes the pre-execution (configuration, contracting, allocation), execution (invocation, monitoring, adaptation) and post-execution (billing, rating, optimisation) steps.

Now that the functional requirement on middleware has been explained, the architectural requirement of dividing the functionality into services needs to be fulfilled. The principles of service orientation act as a guideline for the segmentation of common middleware functionality into separate, loosely-coupled, self-described, reusable and composable platform services.

III. Emergence of Platform Services

As shown in figure 1, platform services are aligned with other services. Both the system and users can search for them in the service registry, inspect their formal interface definition and invoke them through any protocol, with any authentication and contracting scheme already available for the other services offered on the marketplace. If for example the monitoring service provides different monitoring quality levels, each one having a unique price, a reseller can bundle distributable services with as much monitoring as is necessary to control a guaranteed quality level. Grid solution providers can buy as many BPEL execution and deployment services as necessary to offer a high availability service execution on top of the middleware, using the same or a custom marketplace.

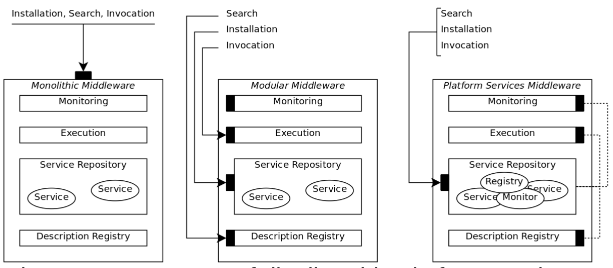

Figure 1: Emergence of distributable platform services

Figure 2 shows a concrete example for services making use of monitoring information. A chain of value-adding platform services, each with a number of configurable and hence payable non-functional properties, is offered to service users. Each user can decide to purchase basic statistical information or aggregated, analysed charts and tables related to the performance indicators of the service execution.

The realisation imposes a number of challenges, most importantly the description of feature-based pricing rules and the restriction of access rights to trusted platform services. Only when these challenges are solved, the full potential of service users offering higher-level composed services can be utilised.

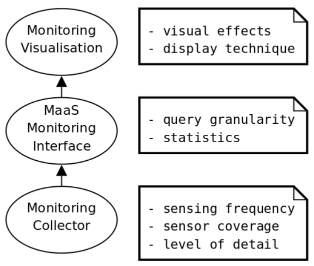

Figure 2: Value-adding chain of configurable platform services

This concept reaches beyond the traditional perception of platform services or Platform-as-a-Service (PaaS) where service and web interfaces to middleware components are offered, while the functionality is not encapsulated and made available as a redistributable service itself. A definite advantage can be seen in the inherent distributed computation through service orientation. At any time, parts of the platform can be externalised and run at different computers or data centres. In the example of the monitoring service, the collection of monitoring data can be decoupled from the aggregation of higher-level metrics and statistical computation.

IV. Analysis of Existing Middleware

Middleware systems can be analysed with comparison frameworks based on key requirements, architectural style and fundamental services [3]. The requirements include scalability, openness and distribution transparency. These properties can easily be covered by platform services. The architectural style calls for modularity, encapsulation and inheritance (or composition), which can also be offered by service orientation.

Fundamental services seem to differ according to the middleware's purpose and type such as transactional, message-oriented or object-oriented middleware [4]. The ones required for the operation of service marketplaces have been mentioned in section II already and will be used for the selection of middleware components for the comparison. We therefore introduce exposition as a service as an additional explicitly derived requirement.

Currently, only few web service hosting and trading platforms expose service interfaces to the controlling middleware. Even fewer offer platform services as redistributable, self-described, composable entities. The lack of formal service descriptions and instant-on usage as web service are the main limiting factors. The following table compares a selection of existing platform services. It includes middleware components from our own research as well as popular components with a high user base.

Platform/ Service	**Functionality**	**Service exposure**	**WSDL/ WADL description**
Apache ODE	BPEL execution	SOAP management API: service listing, process details, property manipulation	none
Dynvoker	Web service invocation	HTTP API for parameter control; XMLRPC API for service inspection and invocation	WSDL
jBilling	Billing	SOAP API for bill management, invoices and payment	none
Grand SLAM	SLA monitor	SOAP API for service and SLA registration	XSD
Contract Wizard	Interactive SLA management	HTTP API for parameter control	none
ConQoMon	Service discovery	SOAP APIs for clients, providers and monitors	WSDL
Puq	Unified hosting environment	SOAP API for service deployment	WSDL

Other execution platforms and middleware systems like WSMX and WSO_2 and monitoring tools like WSMonitor are not exposing their functionality as services and were therefore not included in the table [5].

V. Discussion

Dividing monolithic middlewares into modular platform services and offering their functionality on virtual marketplaces along with other redistributable services leverages the advantages of service orientation. In particular, it allows for a greater variety of custom location-independent service offerings which is essential in service markets as opposed to tradable goods markets [6].

These concepts are being validated by us with a number of newly developed platform services, including contracting services with a HTTP/REST interface and Monitoring-as-a-Service (MaaS) with a SOAP interface. We expect that in the future, distributed service-oriented middleware will be composed out of individual, specialised and redistributable platform services. This migration will have to be followed by intra-middleware and inter-middleware standardisation efforts on data and metadata exchange, including a common security concept, as well as a global approach to finding services. While UDDI's Universal Business Registry was for a long time seen as the solution to the last issue, service search engines like Service Finder [7] and marketplaces like TEXO [8] appear to become suitable means for runtime integration and composition. Eventually, they will allow the composition of arbitrary value-added service and customised middleware systems based on offerings from specialised providers.

VI. Acknowledgment

The project was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference ``01MQ07012". The authors take the responsibility for the contents.

VII. References

[1] *C. Pautasso, E. Wilde:* Why is the Web Loosely Coupled? A Multi-Faceted Metric for Service Design. WWW, April 2009.
[2] *T. Erl:* An Introduction to the Service-Orientation Paradigm. Online: http://soaprinciples.com, May 2009
[3] *A. Zarras:* A Comparison Framework for Middleware Infrastructures. Journal of Object Technology, Vol. 3, No. 5, May-June 2004
[4] *H. Pinus:* Middleware: Past and Present - A Comparison. June 2004
[5] *E. Cimpian, M. Zaremba (eds.):* Web Service Execution Environment (WSMX). W3C Member Submission, June 2005
[6] Globalisation and Structural Adjustment: Summary Report of the Study on Globalisation and Innovation in the Business Services Sector. OECD, 2007
[7] Service Finder research project. Online: http://www.service-finder.eu, May 2009
[8] *C. Janiesch, K. Fleischmann, A. Dreiling:* Extending Service Delivery with Lightweight Composition. Web Information Systems Engineering (WISE) workshops, September 2008