Open Web Services-based Middleware for Brokering of Composed eHomeCare Services

Sofie Van Hoecke, Koert Vlaeminck, Filip De Turck, Bart Dhoedt
Department of Information Technology
Gaston Crommenlaan 8 bus 201, B-9050 Gent
Ghent University/IBBT/IMEC, Ghent, Belgium
fax: +32 9 33 14 899 - tel: +32 9 33 14 940
sofie.vanhoecke@intec.ugent.be

Abstract— **Generically integrating heterogeneous services eases the deployment of promising eBusiness applications, such as multimedia content delivery, eHealth and eCommerce, all of them covering multiple service and data providers. Integration and composition of services is a complex task since services and applications are built by different vendors, using different programming languages, data definitions and exchange standards.**
Therefore we describe a middleware architecture that offers dynamic selection and composition of services and automatic load balancing of these services over the available application servers. The architecture also allows the automatic fulfillment of QoS requirements, such as price, availability and response time. The middleware platform translates incoming service requests into an internal composition flow chart. Based on the QoS information from the monitors, the broker selects the best path through the flow chart using advanced broker algorithms. Then the broker starts the chosen services and gathers the result sets, composes them and returns the results to the requester.
New application servers and services can transparently be added to the system. Scalability of the architecture at both broker and provider level are also addressed.
As example case, QoS brokering for eHealth, in particular eHomeCare, is considered.

Keywords: **Web services, broker, QoS, dynamic selection and composition, load balancing, eHomeCare.**

I. Introduction

In view of the broad support for Web services and common XML-based standards such as SOAP, WSDL and UDDI, Web services are a promising concept for the integration of heterogeneous software components. By means of this technology, applications can easily be distributed over the Internet and expose well-defined functionality as a Web service, which consumes and produces XML-messages over HTTP. Based on the exchange of structured text messages, the interaction abstracts the underlying technologies. Consequently, extending existing services with a Web service interface enables integration [1]. Web service composition, resulting in the deployment of multiple promising end-to-end and eBusiness applications, is enabled through the emerging Web service standards on one hand for flow specification, such as BPEL4WS [2] or BPML [3] and WSCI [4], and on the other hand for semantic markup, such as OWL-S [5] and DAML-S [6]. However QoS (Quality of Service) requirements like availability, price and response time can influence and hamper service composition. Therefore we present in this paper a middleware platform for brokering of composed services with QoS guarantees. Implementing this brokering middleware by means of Web service technology creates the required integration of heterogeneous services, taking into account QoS requirements and offers an advanced set of composed services crossing multiple service and data providers. This way the required user interaction is reduced to authenticating (e.g. by means of eID, smart card or login/password) and selecting one of the (composed) services. The presented architecture covers a wide range of application cases. For example multimedia content delivery [9] can benefit from QoS brokering. The broker middleware can dynamically select and compose the needed services (e.g. services for broadcasting, streaming, payment and security) in order to set-up a video-on-demand stream meeting the request (high quality, no delay, limited output device, etc.). Another case can be found in eCommerce. Since eCommerce requires for example payment validation, the call center negotiates with multiple credit checkers. Based on the call center load, the broker middleware can divide the requests over multiple credit checkers in order not to lose or displease clients. QoS brokering can also be applied in B2B (Business to Business) for optimizing the virtual supply chain of delivery companies. QoS brokering can select delivery services with the best QoS (e.g. price, delivery time, quality), resulting in smaller stocks and advanced efficiency. On the other hand, QoS brokering can also be useful in pure software design, planning and measuring the required infrastructure, as well as in offline tuning of load balancing and brokering strategies. Finally eHealth is another case that can benefit from QoS brokering by integrating multiple care providers. The employment of the QoS brokering middleware for eHealth, more in particular for eHomeCare, is described in this paper.
Related work in this area focuses mostly on Web service brokers limited to service lifecycle management. There is however a need for service brokers taking into account QoS (Quality of Service) in order to ensure total response time of composed services, prioritize time-critical services or ensure bandwidth or robustness. In [7] a QoS broker model is described for general distributed systems. Contrary to general distributed systems, Web services have a dynamic nature in

terms of service availability and the clients invoking them. Brokers must support more flexible service selection and be able to adapt to the dynamic server load. In [8] a Web service architecture supporting QoS is presented. However, once the services are selected and the link is established, the client communicates with the server directly without any broker intervention during the actual service process. Due to the dynamic nature of Web services this introduces QoS shortcomings since abruptly failure or unavailability of services needs dynamic selection of another equivalent service. This broker also leaves composition as the client's responsibility.
The remainder of this paper is structured as follows: Section II describes our QoS brokering middleware, while in section III the eHomeCare use case is presented. Finally, in section IV, we will highlight the main conclusions.

II. QoS BROKERING MIDDLEWARE

Figure 1 depicts the open QoS brokering middleware architecture. As can be seen in this figure, the architecture contains three different levels of stakeholders. Authentication can be provided by a third party, while there is both a broker and several provider domains. If the interfaces to the broker domain are followed, other legacy systems in the providers domain are possible, such as Mosix [10] or Unicore [11]. However we believe in the simplicity and openness of our presented solution for the provider domains.
First the main functions of the architecture are described after which a detailed description is given of all the components.

A. Functional description

The architecture is built around a facade portal. This portal contains a UDDI (Universal Description, Discovery and Integration) registry containing all the services offered by the platform, i.e. the simple services offered by the service providers, as well as advanced, composed services only possible by the platform. The platform however transposes all services in a transparent and similar fashion, hiding for the users if it concerns a simple or composed service and thus simplifying required user interactions (see figure 2a). This portal intercepts all service requests and makes sure those requests are redirected to the best provider server, even in the case of composed services. Since the middleware platform has to be able to interpret the service request, the requests are transformed into a composition flow chart (see figure 2b). In order to select the best suited servers for fulfilling the request, the middleware queries for interfaces of all the possible matching services. After filling in this information into the flow charts (see figure 2c), the middleware requests for the appropriate QoS metrics. By using advanced broker algorithms, the best path through the flow chart is chosen. Through this flow processing, service sessions are assigned to the best suited provider server, based on the capabilities of the server (which service(s) does it implement) and its QoS characteristics like load, availability, service price and the running services. This flow processing is described more in detail in section 4. Afterwards, the middleware platform

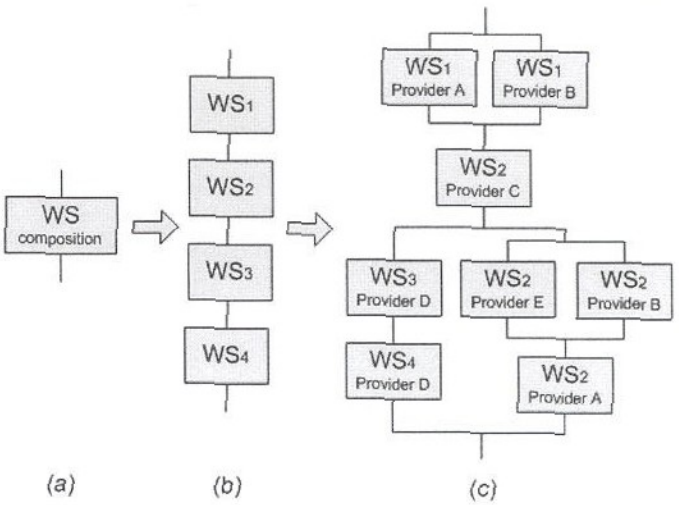

Fig. 2. Service composition flow: (a) Transparent services offered by the Broker Facade (b) Composition flow chart (c) Matching services for composition

aggregates the result sets, translates this flow response into a service response and returns it to the requester.

B. Component description

The architecture consists of the following components:

- **Broker Facade (BF):** The Broker Facade forms the central portal of the brokering middleware. It intercepts incoming service requests from authenticated users, requests needed credentials from the Credential Manager (CM), adds them to the service requests and forwards these requests to the MSF.
- **Credential Manager (CM):** Credential Manager contains user credentials based on the registration input or profile. Incoming service requests from the Broker Facade will be adjusted with these credentials before being forwarded to the MSF.
- **Mapping Service Flow (MSF):** The MSF translates incoming service requests into corresponding flow charts and forwards these flows to the FPS. Incoming flow responses are translated back to service responses before being forwarded to the Broker Facade.
- **Flow Processing Server (FPS):** The Directory Repository returns the Application Servers offering the requested services. Based on the QoS information provided by the Pool Management Server Coordinator (PMSC), the Flow Processing Server can select, by using advanced broker algorithms, the optimal application servers for executing the services, fulfilling the QoS requirements.
- ***Service Directory Repository (SDR):*** The Service Directory Repository is implemented by exploiting the UDDI technology, a framework for the description and discovery of services. The architecture contains one or more linked Service Directory Repositories. Here all services offered by the Application Server are registered and described in detail (location, business properties,

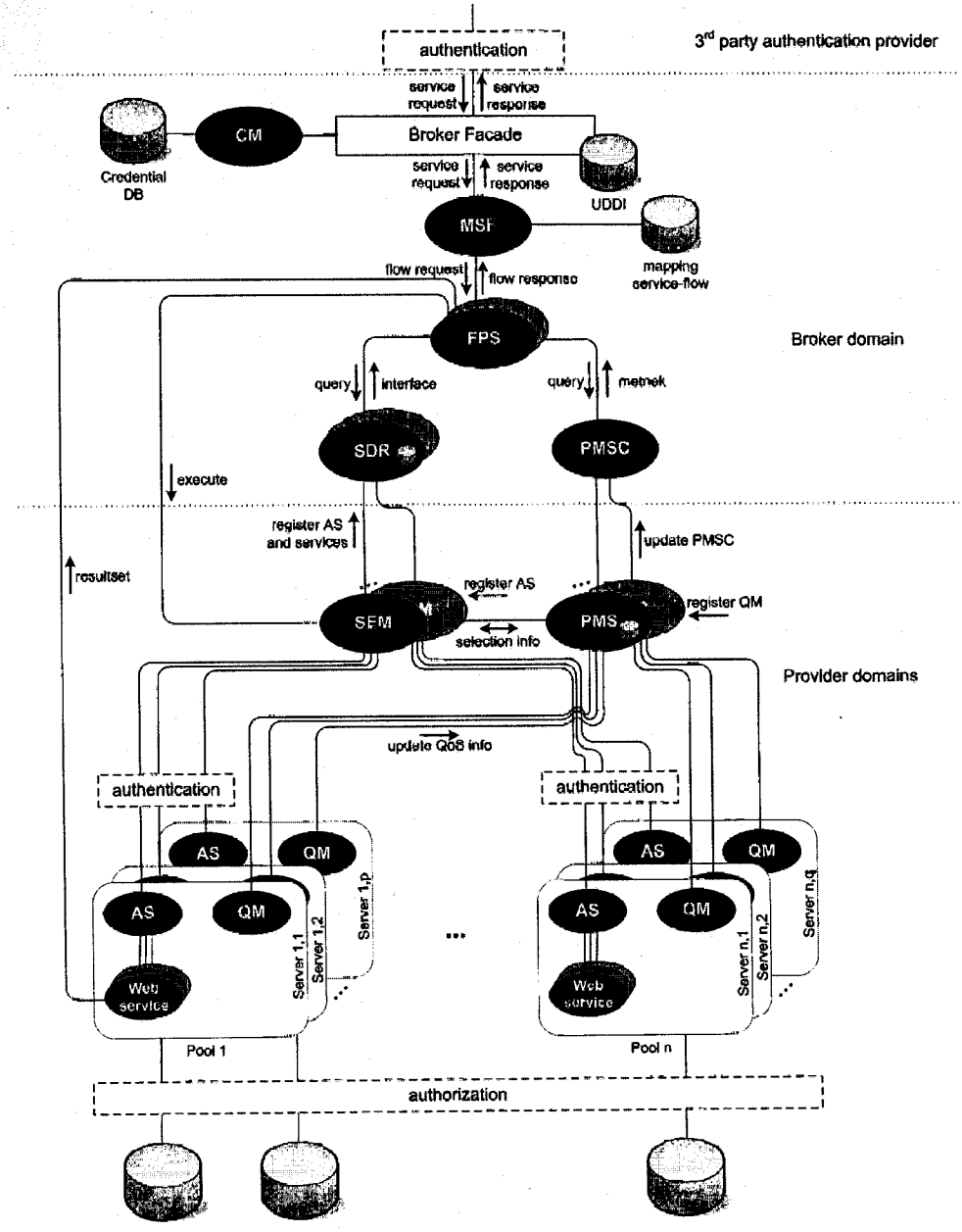

Fig. 1. QoS brokering middleware platform. The following components are shown: Credential Manager (CM), Broker Facade (BF), Mapping Service Flow (MSF), Flow Processing Server (FPS), Service Directory Repository (SDR), Service Execution Manager (SEM), Pool Management Server Coordinator (PMSC), Pool Management Server (PMS), Application Server (AS) and QoS Monitor (QM).

service properties, etc). The Service Directory Repository contains also the Service Level Agreements (SLAs), containing the agreed capability information like service availability, maximum response time, price, etc. When a new Application Server is added to a pool, the Service Execution Manager registers this server and his services to one of the Service Directory Repositories. Since all Service Directory Repositories are referencing each other, it appears like one single SDR. Creation and deletion of pools, as well as registering a new Application Server to the SEM of a certain pool, is done through the Pool Management Interface of the Service Directory Repository. When a new pool is created, both an Service Execution Manager and a Pool Management Server for that pool are started.

- **Service Execution Manager (SEM):** The Service Execution Manager manages a pool of Application Servers. When a new Application Server is started, it has to register itself to the SEM of that pool. The SEM then registrates that Application Server and its Web services to the Service Directory Repository. When new Web services have to be started on specific Application Servers, the

FPS contacts the SEM of the pools these ASs belong to. The SEM then forwards the start commands to the correct AS. Result sets from these Application Servers however are not sent back to the SEM, but directly forwarded to the FPS.

- **Application Server (AS):** Application Servers execute the actual Web services. Each Application Server can implement one or more Web services, can have its own capabilities and can provide services using different programming languages. When a new AS is installed in the system, it must register itself to the Service Execution Manager of the pool it belongs to.
- **Pool Management Server Coordinator (PMSC):** The Pool Management Server Coordinator stores an overview of the QoS information of each server running an AS/QM pair. This information is updated at regular intervals by the Pool Management Server of each pool. At regular time intervals, an overview of the QoS of the platform is pushed from the PMSC to the FPS. When executing the dynamic selection algorithm, the FPS can request more detailed and up-to-date information about each server running an AS/QM pair from the PMSC.
- **Pool Management Server (PMS):** Each pool has its own Pool Management Server. It gathers the QoS information of the monitors in the pool. At regular intervals, the QoS information of the whole pool is reported to the Pool Management Server Coordinator. When a new Application Server is started, the QM that it is paired with has to register itself to the PMS of the pool it belongs to.
- **QoS Monitor (QM):** Each Application Server is paired with a QoS Monitor that monitors Quality of Service (QoS) characteristics of the machine it is running on. Different QoS parameters can be measured: CPU load, memory usage, load average, availability, price, etc. At regular intervals the QM reports the QoS of its machine to the Pool Management Server of the pool it belongs to. When a new AS is started, the QM it is paired with has to register to the PMS of its pool.

C. Flow processing

In order to process the composition flows, the FPS component implements advanced broker algorithms using OWL-S [5] and BPEL4WS [2] for dynamically selecting and composing services, based on the capabilities of the Application Servers and its QoS characteristics like load, availability, service price and the running services.

Supplied with the service interfaces and corresponding service QoS information, the FPS can select the best path through the chart, satisfying the required QoS parameters (see figure 3a). However when one of the components suddenly slows down or becomes unavailable, the FPS choses an alternative path through the flow chart (see figure 3b). This selection and composition process will loop one or more times until the path is completed.

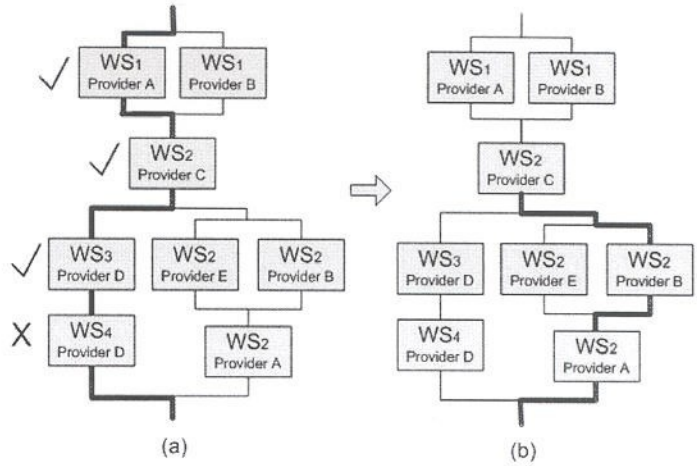

Fig. 3. Handling service failure

D. Component interaction

Figure 4 depicts the sequence diagram, presenting the component interactions within the middleware platform.

An Application Server is paired with an QoS Monitor (QM), that monitors Quality of Service characteristics (like CPU load, memory usage, price and load average) of the machine it is running on. The Application Servers are organized into pools. Pools can be created and removed through the Pool Management Interface of the Service Directory Repository (SDR), implemented by exploiting the UDDI technology.

Each pool has one Service Execution Manager (SEM) component as well as one Pool Management Server (PMS). When a new Application Server is installed into the system, it must register itself to the SEM of the pool it belongs to. The SEM consequently registers the AS and its Web services to the Service Directory Repository. When a Web service has to be started on a specific Application Server, the FPS contacts the SEM of the pool that Application Server belongs to. The SEM then forwards the start service command to the correct AS, taking into account selection information from the Pool Management Server (PMS). At registration time, the Application Server and broker negotiate Service Level Agreements (SLAs) containing capability information like maximum response time, fixed price, minimum availability, etc. These SLAs are stored at the Service Directory Repository. The Pool Management Server (PMS) gathers the QoS characteristics from the monitors in its pool. At regular time intervals, the PMS pushes this information to the Pool Management Server Coordinator (PMSC). This way the PMSC has an overview of the QoS of the total platform. When selecting the services, the FPS can also request the PMSC more detailed QoS information about specific pools or servers.

In order to alleviate the users from privacy and security issues, an additional component is present in the framework, namely the Credential Manager. Users only have to authenticate to the framework (by means of eID, smart card or login/password). Once authenticated, the Broker Facade will investigate the

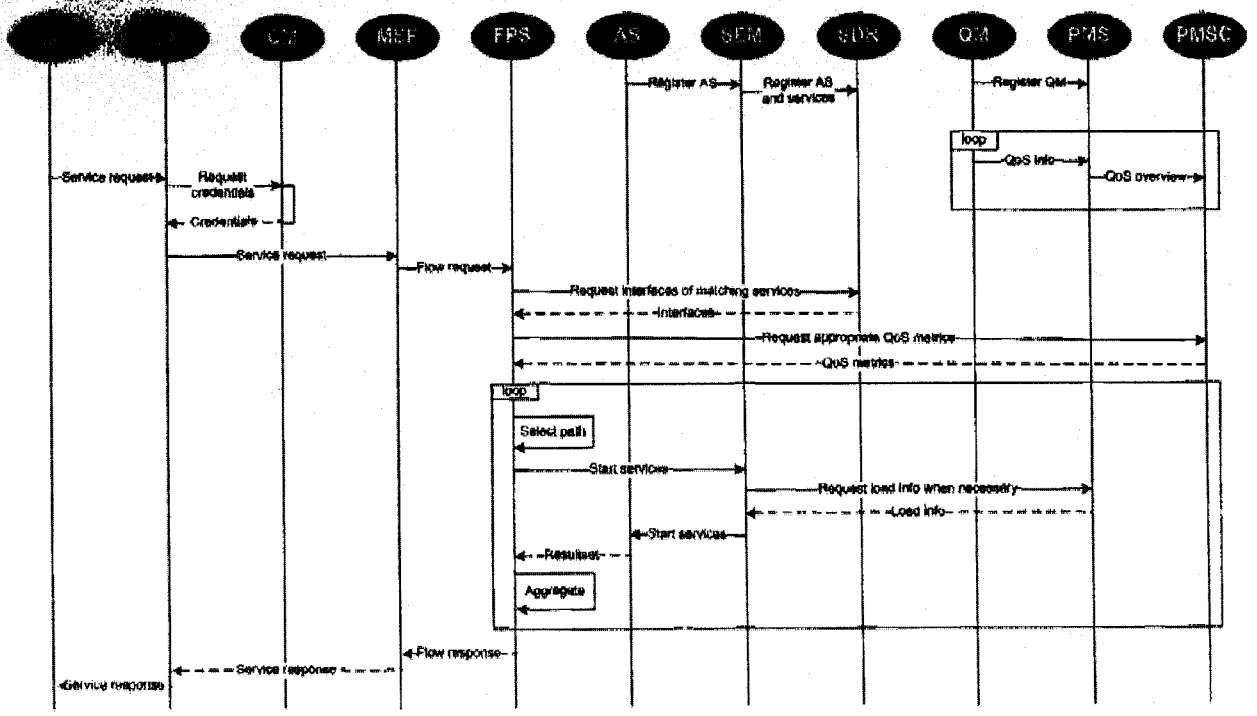

Fig. 4. UML sequence diagram of QoS brokering middleware. The communication between the following components is shown: Broker Facade (BF), Credential Manager (CM), Mapping Service Flow (MSF), Flow Processing Server (FPS), Application Server (AS), Service Execution Manager (SEM), Service Directory Repository (SDR), QoS Monitor (QM), Pool Management Server (PMS) and Pool Management Server Coordinator (PMSC).

service requests, queries the needed credentials from the Credential Manager and forwards the credited requests to the FPS.

E. Scalability considerations

In the middleware platform scalability is ensured both in the broker domain as well as in the provider domains.

Scalability in the broker domain is ensured by avoiding single point-of-failures and bottlenecks by replicating components. The architecture contains one or more linked Service Directory Repositories, containing all services offered by the Application Servers described in detail, as well as the Service Level Agreements (SLAs). Since all Service Directory Repositories are referencing each other, it appears like one single SDR and scalability is ensured. Besides the SDR, the other components can be made up multiple instances, clustered and interacting with the system simultaneously.

This architecture ensures also scalability in the provider domains by organizing the Application Servers into pools. Pools can contain multiple servers, each belonging to one single provider. A provider however can choose to create multiple pools, for example grouped by functionality or performance.

III. Use case: eHomeCare

The ageing population and a shift in the burden of illness from acute (infections and injury) to chronic conditions (asthma, epilepsy, heart disease, cancer, etc.) drive up health costs and create a generation of people living with long-term illness and disability. Furthermore nowadays healthcare, and homecare in special, requires interaction and cooperation between multiple care providers.

Due to the advent of the Internet technologies, the access to information increases, rapid communication facilitates and remote locations can be reached. Therefore information and communication technologies have considerable potential for improving the delivery and quality of care. eHealth brings together information, technology and health and creates opportunities to manage and provide care faster, at lower cost and higher levels of convenience for their patients.

Since the healthcare field evolved to a multi-disciplinary process involving more and more information of a complex nature, there is a need to provide the adequate information to the health provider at the point and time of need. Besides sharing of information, eHealth, and eHomeCare in particular, requires also sharing of responsibility, services (like eScheduling, ePrescription, alarmService) and decision making.

A. Requirements

Despite the proven benefits, eHomeCare is not yet widely used in real-life medical or health situations. In many places it still is in pilot phaze. Consequently integration of eHealth/eHomeCare services and applications is a complex task since these services and applications are built by different vendors, using different data definitions and exchange standards. In the service selection and composition process, availability is one of the most important QoS requirements

for eHomeCare. Besides availability, other QoS requirements like price and response time can influence the choice of service. For example medical data retrieval for emergency services should receive higher priority, resulting in required QoS monitoring. On top of that, service provisioning and user interaction should be as simplified as possible since not all eHomeCare users and providers are technical experts. Moreover eHomeCare actors need to have different profiles (e.g. nurses, general practitioners, social workers, patients, hospital specialists, etc.) resulting in a different set of available services and medical data results. Easy profile management is thus as well a requirement.

B. eHomeCare services

The IBBT project Coplintho [12] aims to explore the possibilities of using ICT in homecare. The domain analysis within the project and multiple interactive discussions between the actors involved in the project (nurses, general practitioners, hospital physicians and specialists) highlight the need for integration and composed, advanced eHomeCare services crossing healthcare providers and medical databases.

Some example end-to-end applications will be described below to illustrate the advantages of service integration and composition.

- **eScheduling:** eScheduling enables care providers and patients to schedule their visit appointments online at any time of the day or night. The eScheduling application can also schedule usage appointments of medical resources and equipment. Both ways users can make appointments very flexible and there is less time spent on routine scheduling tasks.
- **ePrescription:** Eliminating the misreading of handwriting can reduce medical errors. By digitizing presciptions, patients will also no longer be able to receive duplicate prescriptions from different physicians, eliminating present abuse. ePrescription can also save time for care providers and patients by avoiding incorrect or missing prescriptions.
- **teleMonitoring:** A patient with diabetes could input their daily blood glucose level into their interactive television set-top box for analysis. Or rather than relying on the manual entry of data, a range of devices can be connected to the set-top box via infra-red connection. This way, both the patient and a relevant healthcare professional can be alerted to any areas of concern, an appointment can be booked online (eScheduling) if necessary or an ambulatory care provider can be sent automatically.
- **Medical Data Retrieval:** Hospitals and general practitioners are more and more computerizing, resulting in electronical medical patient data. Currently proprietary solutions exist to interchange medical data between care providers, but only between consumers of the same proprietary solution. Since nowadays multiple solutions coexist, an open integration solution is needed to interchange medical data independent of chosen implementations. The platform presented here can coexist with the current (and future) solutions as long as they offer a Web service interface to their services. This way all medical data can be retrieved, independent of the chosen solution, as long as the requester owns the right credentials to access the data.

C. Middleware deployment

The applications presented above are just an illustration of the benefits of integrating and composing multiple eHomeCare services. In this case, example services, offered by care providers or third parties, can be timestamping services, alarming services, health record services, monitoring services or medical decision making services. The middleware platform presented in this paper can achieve this integration and composition, as well as fulfilling all requirements stated in section III-A. Dynamically selecting and composing eHomeCare services, automatically load balanced and with guaranteed QoS, will result in many new and advanced eHealth services offering functionality currently not achievable.

The eHomeCare architecture thereby facilitates and simplifies providing services to the framework. Service providers only need to provide a Web service interface on their applications, register them transparently to the framework and the platform takes care of all the rest.

From the user (human or application) point-of-view, users only interact with the user-friendly eHomeCare Broker Facade, abstracting the technical middleware. Only a simple SOAP request is needed for one of the offered services (see figure 5a) and the Broker Facade adds the required credentials based on their profile and delegates the adjusted requests to the framework where it will be transformed into a composition flow chart (see figure 5b) and processed.

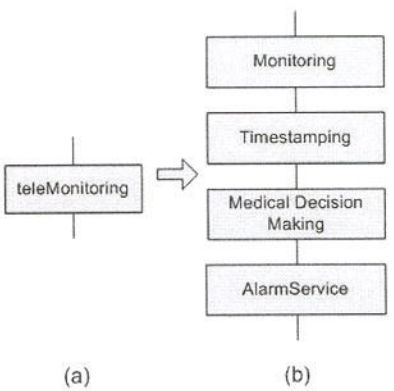

Fig. 5. Flow composition of teleMonitoring service for eHomeCare: (a) Transparent eHomeCare service offered by the Broker Facade (b) teleMonitoring composition flow chart

IV. Conclusions and Future Work

In this paper we presented a Web services-based brokering middleware platform, offering dynamic selection, composition and automatic load balancing of services, guaranteeing QoS requirements. Those Application Servers and Web services may be implemented by different vendors, using different

programming languages and data definitions. Service providers only need to provide a Web service interface on their applications and the platform takes care of all the rest.

The middleware platform is built around an Broker Facade which intercepts all incoming service requests and makes sure those requests are forwarded to the best Application Server, even for composed services. The Broker Facade therefore forwards these service requests to the Mapping Service Flow component which translates the service requests into an internal composition flow chart. This Broker Facade abstracts the underlying technical middleware and simplifies required user interactions.

Besides simplifying required user interactions and service provisioning, the middleware platform is also highly scalable due to the organization of Application Servers into pools.

Due to the generic approach, this broker middleware platform covers a wide range of application cases in B2B, media content delivery and eHealth.

We will continue the design of advanced broker algorithms for selecting and composing the services, fulfilling QoS requirements. Moreover, single point-of-failure components will be splitted into multiple instances that run simultaneously in order to increase the scalability of this platform. A detailed performance evaluation of this architecture will be presented in future publications.

ACKNOWLEDGEMENT

Part of this work is supported by Coplintho and the FWO-project "Intelligent dynamic brokering of Web services based on performance models".

Coplintho is a project of IBBT (Interdisciplinary institute for BroadBand Technology) in cooperation with the companies and organizations: WGK, Televic, MediBRIDGE, Androme, Custodix, UZGent, Sint-Elisabeth Ziekenhuis Zottegem. IBBT is a research institute founded by the Flemish Government in 2004.

Sofie Van Hoecke would like to thank the IWT (Institute for the Promotion of Innovation through Science and Technology in Flanders) for financial support through her Ph.D. grant.

Filip De Turck acknowledges the F.W.O.-V. (Fund for Scientific Research-Flanders) for their support through a postdoctoral fellowship.

REFERENCES

[1] A Darwin Partners and ZapThink Insight, Using Web Services for Integration, http://www.xml.org/xml/wsi.pdf, 2002.

[2] T. Andrews, F. Cubera, H. Dolakia, J. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, S. Weeravarana, Business Process Execution Language for Web Services, 2003.

[3] The Business Process Modeling Language, Business Process Management Initiative, http://www.bpmi.org/.

[4] Web Service Choreography Interface, W3C, http://www.w3.org/TR/wsci/.

[5] The OWL Services Coalition, OWL-S: Semantic Markup for Web Services, Technical White paper (OWL-S version 1.1), 2004.

[6] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, S. A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, H. Zeng, DAML-S: Semantic Markup For Web Services, Semantic Web Working Symposium, California, 2001.

[7] K. Nahrstedt, J.M. Smith, The QoS Broker, IEEE Multimedia Magazine 2(1), 1995.

[8] T. Yu, K. Lin, The Design of QoS Broker Algorithms for QoS-Capable Web Services, Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service, 2004.

[9] S. Van Hoecke, W. Haerick, G. De Jans, F. De Turck, E. Laermans, B. Dhoedt, P. Demeester, Design and Implementation of a Secure Media Content Delivery Broker Architecture, The 2005 International Symposium on Web Services and Applications, Las Vegas, Nevada, 2005.

[10] A. Barak, O. La'adan, The MOSIX Multicomputer Operating System for High Performance Cluster Computing, Journal of Future Generation Computer Systems, 13 (4-5) (1998) 361-372.

[11] Unicore, http://www.unicore.org.

[12] Coplintho, Innovative Communication Platform for Interactive eHome-Care, http://coplintho.ibbt.be.

Proceedings of Middleware for Web Services (MWS) 2005

Edited by: Vladimir Tosic, Aad van Moorsel, Raymond Wong

Middleware for Web Services (MWS) 2005 workshop
held at EDOC 2005 conference,
on September 19, 2005
in Enschede, The Netherlands

TABLE OF CONTENTS