Aspect-Based Variability Model for Cross-Organizational Features in Service Networks

Stefan Walraven, Bert Lagaisse, Eddy Truyen & Wouter Joosen
DistriNet, Dept. of Computer Science
K.U.Leuven, Belgium
{firstname.lastname}@cs.kuleuven.be

ABSTRACT
Different clients have different needs, therefore adaptability and variability are crucial properties for service compositions to fit those varying requirements. This is hard to achieve in a cross-organizational context where services are implemented and deployed by different organizations (e.g. companies, administrative domains, . . .): a feature, for example security, cannot be condensed into a single module that is applicable to all the different services. This paper proposes an aspect-based variability model for representing cross-organizational features in service networks such as systems of systems or service supply chains. We argue that cross-organizational features should be managed in a multi-layered architecture, distinguishing between policy and mechanism. Such a multi-layered architecture is completely lacking in AOSD currently. Based on this tenet, we first describe a technology-independent feature ontology that is well-defined for a domain or a specific service network and map it to an aspect-based feature implementation.

Categories and Subject Descriptors
C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; D.2.11 [**Software Engineering**]: Software Architectures—*Service-oriented architecture (SOA)*; D.2.13 [**Software Engineering**]: Reusable Software

General Terms
Design, Documentation, Management

Keywords
AOSD, Variability modelling, Service engineering, Feature-oriented

1. INTRODUCTION
Recent trends in service engineering aim to combine the benefits of feature-based and service-based approaches [5, 14, 1, 19]. This combination increases the reusability of services and gives service consumers the opportunity to select different variants of a service. In addition, a service provider can provide fine-grained customization capabilities for a service, without having to create new services for each customization.

A feature is a distinctive mark, quality, or characteristic of a software system or systems in a domain [12]. Features define both common facets of the domain as well as differences between related systems in the domain. They make each system in a domain different from others. Features are also used to define the domain in terms of the mandatory, optional, or alternative characteristics of these related systems. Aspect-oriented software development (AOSD) [9] often has been put forward as a possible solution to enable modularization and composition of features [20, 17, 15].

However, services are mostly used in a service composition consisting of services from different organizations. In such a cross-organizational context, a feature cannot be condensed into a single feature module any more. The reason is that service implementations are black boxes, implemented and deployed by different organizations, and only the interface descriptions are available [1]. But this doesn't exclude the need to share semantically compatible features between those different services. A typical example of a cross-organizational crosscutting feature is security. When implementing an access control concern in an application, for instance, security actions need to be performed for every interaction between application components. However in a cross-organizational application, it is difficult to defend that a single module, for instance an aspect, should encapsulate the implementation of the internal security mechanisms of the organization involved as well as the global security policy governing how security must be addressed in the overall interaction between organizations. The latter security policy belongs to a level of abstraction above the internal security mechanism.

Therefore this paper proposes an aspect-based variability model for representing cross-organizational features in service networks such as systems of systems or service supply chains. We argue that cross-organizational features should be managed in a multi-layered architecture. Such a multi-layered architecture is completely lacking in AOSD currently.

The remainder of this paper is structured as follows. In section 2, we further illustrate and motivate the need for a variability model for cross-organizational features in AOSD. Section 3 elaborates the overall approach and presents the application of the approach to an example. We discuss related work in section 4 and conclude in section 5.

2. MOTIVATION & ILLUSTRATION

In this section we further motivate and illustrate the importance of an aspect-based variability model for cross-organizational features in service networks.

We present an example in the e-finance domain (see Fig. 1). A bank offers a stock trading service to inspect, buy and store stock quotes. To be able to provide this service, it cooperates with the stock market, which in turn cooperates with a settlement company. So the stock trading service is a composition of the services provided by these three companies. Each participant can take up two roles in a composition: service consumer (client) and service provider (server). For example the bank company is a server for the bank customers, but consumes the `QuotesOrderService` of the stock market.

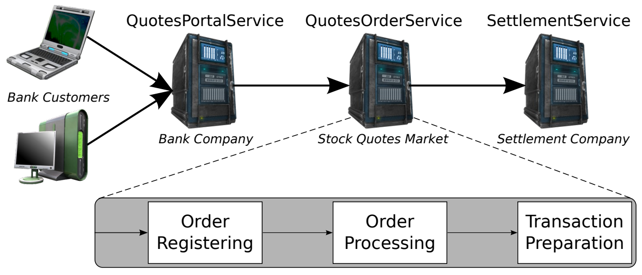

Figure 1: Illustration of the stock trading service composition.

During a typical session, a client inspects stock quote data, inspects the stored stock quotes in his custody account and potentially buys or sells some stocks. Clients can issue a stock order by using the web service portal facility of their bank. The bank service acquires the client's order and forwards it to the stock market. Processing the order request in the stock market consists of three sequential functional steps. Firstly, the client order is registered in the stock market by the `OrderRegistration` unit and then forwarded to the `OrderProcessing` unit. Secondly, at regular time intervals, the `OrderProcessing` unit searches for matches between buying and selling offers. If two orders match, they are forwarded to the `TransactionPreparation` unit, which delegates the actual trade of goods to the settlement company.

Since different clients have different needs, this service composition can be customized with respect to agreed features such as prioritized processing, billing, stepwise feedback, logging, non-repudiation, transaction support, secure communication, authentication, authorization and secure server-side storage. Choosing different features results in differents variants. If selected, the stepwise feedback feature, for instance, informs the client about the progress made in processing its requests, at the level of the individual services as well as the different sequential units within a service. Several alternatives are available for the stepwise feedback feature, such as feedback by email or by mobile text messages. Similarly, a client can select prioritized processing. By having this feature injected, the client's requests are prioritized over other requests. However, the prioritizing feature requires the billing feature: prioritizing requests comes at an expense.

Figure 2 presents the stock trading service composition including the prioritized processing feature. We see that the stepwise feedback feature affects both the `QuotesPortalService`, to retrieve customer account information, and the `QuotesOrderService`, to perform the prioritizing. A more trivial case is the secure communication feature: encryption and decryption operations should be performed at both sides of the connection. This clearly illustrates that a single feature, often consisting of a client and server functionality part, can affect multiple services in a service composition.

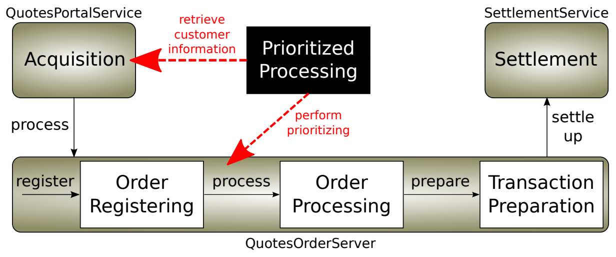

Figure 2: Services affected by the stepwise feedback feature.

However, each company in a cross-organizational service network has its own IT administration and trust domain, and will not allow external parties to add or update feature implementations. The services provided by the different partners are black boxes, loosely-coupled and independently maintained by the company's own administrators. This black-box scenario hinders the feature modularization and composition in a cross-organizational context [1]. Therefore a feature cannot be condensed into a single module any more. Cross-organizational features need to be split up in client-side and server-side aspects, independently implemented with possibly different AO-technologies. However, a uniform high-level representation of those features is crucial to be able to share them in a particular application domain or service network.

3. APPROACH

In this section we present our approach to achieve a multi-layered architecture for the uniform representation of cross-organizational features in AOSD. A multi-layered architecture, distinguishing between policy and mechanism, is a core tenet of the body of research on cross-organizational coordination architectures. We shortly review the state-of-the-art in this field. Subsequently, based on this tenet, we propose that aspect-based variability is first described at the level of a technology-independent feature ontology that is well-defined for a domain or a specific service network. Each organization implements this feature ontology independently using an AOP technology of its choice. The mapping between the independent feature ontology and the aspect-based implementation is then specified as part of the second layer of the model. Finally, we show how this feature ontology is used for cross-organizational service customization.

3.1 Cross-organizational Coordination Architectures

Our approach is inspired by the design principles of cross-organizational design. In the field of cross-organizational coordination architectures, a layered system architecture is a core principle of the reference model [31]. This reference model distinguishes between (i) the type of agreements that are established, (ii) the language for describing the agreements, and (iii) the middleware for establishing and executing the agreements.

The language for describing how the interactions between two or more independent services are to be done is further refined into a conceptual and a computational model [31]:

1. A *conceptual model* provides the modeling concepts to describe the regulations at a sufficient high-level of abstraction that is independent from the organizations internal processes and data.
2. A *computational model* offers behavioral concepts that are mappable to implementable actions in the underlying software system that can be enforced upon contracted services.

The conceptual model of the language should be as independent as possible from the computational model to enable that different organizations can implement the same agreement differently depending on their choice of implementation platform, while adhering to the terms of the agreement.

When multiple independent organizations interact with each other, they have to integrate their business processes in order to be able to operate, gain added-value and survive in a market. To enable this, a certain agreement must be complied by all participating organizations in the business relationship. We think that a common feature ontology can be part of this agreement. Therefore, it is plausible to assume that services in a service network can share a common feature ontology.

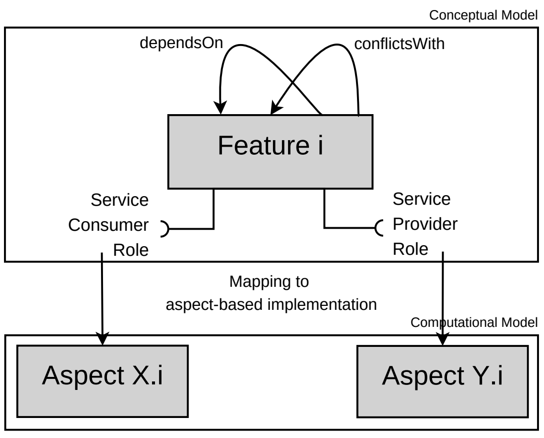

Figure 3: Aspect-based variability model.

3.2 High-level Feature Ontology

The conceptual model in our approach for specifying cross-organizational features consists of a feature ontology. Similarly to the conceptual model of the cross-organizational coordination architectures, this feature ontology should be high-level and independent from the aspect-based implementation to enable organizations in service networks to implement cross-organizational features using an AOP technology of their choice (see Fig. 3). In order to be successful, the feature ontology must have a clear scope on which particular application domain or area it applies, for example, a specific market such as financial services or an individual (long-running) business relationship between multiple organizations.

The specification of such a common feature ontology is divided into a base level and one or more application-specific levels. The base ontology is a framework and vocabulary for specifying application-specific ontologies. An application-specific ontology contains a catalog of features that can be used within a certain cross-organizational service network. Application-specific ontologies are hierarchically structured: the application-specific ontology of a specific service composition imports and extends the ontology of the application domain.

A feature ontology can be seen as a *high-level, technology-independent agreement* between the parties involved (typically a service consumer and service provider). This agreement prescribes the intended behavior of the feature and clearly sets out the roles that different parties involved have to play, as depicted in Fig. 3. These roles are described by a name (e.g. Service Consumer) and a set of responsibilities. These responsibilities specify constraints on behavior (the specification of an algorithm to be used) and interfaces (message types and operations that are required or provided). Further, composition rules can be specified that prescribe which features depend on other features and which features can't be executed during the same request due to feature interference.

Listing 1: Example of high-level features.

```
feature PrioritizedProcessing {
 dependsOn: Billing;
 role ServiceConsumer {
  responsibility retrieveCustomerAccount {
        provides: CustomerAccount;
  }
 }
 role ServiceProvider {
  responsibility performPrioritizing {
        requires: CustomerAccount;
        provides: AccountableItem;
  }
 }
}
```

For example, the `PrioritizedProcessing` feature from Fig. 2 needs two roles: a service consumer who retrieves customer account information, and a service provider, responsible for performing the prioritizing. The service provider role requires a `CustomerAccount` attribute, which will be provided by the service consumer role. After the prioritizing, the service provider role will provide a `AccountableItem` attribute that will be used by the `Billing` feature. The feature description is presented in Listing 1. It also defines a composition rule that prescribes that `PrioritizedProcessing` requires the `Billing` feature.

3.3 Mapping to Aspect-based Implementation

The mapping between the high-level feature ontology and the aspect-based implementations is specified on the level of the service platform, hiding the implementation details for external parties. The use of AOSD [9] enables a clean separation of concerns, in which the core functionality of a service is separated from any feature behavior. Therefore they are implemented separately from each other as composite entities containing a set of aspect-components, providing the behavior of the features (so called advice). This advising behavior can be dynamically composed on all the components of a service – at client-side and at server-side.

By capturing the semantics of the features in a high-level feature ontology, the different features can be implemented independently by each of the service providers using their favorite service platform and AO-composition technology. Hence, the different services in the network may have their own optimized aspect-based implementations of the different features, and the most appropriate feature implementation in each service may depend on environmental circumstances. This decentralized feature management allows a variety of service platforms using different AO-composition technologies to be interconnected. In addition, the implementation of the different features and the software composition strategy are open for adaptation by each of the local administrators. However, the feature implementations have to satisfy certain constraints, enforced by the feature ontology.

Each *feature implementation mapping* within a specific organization is described by means of a declarative specification that specifies: (i) the feature and role that is implemented, (ii) the aspect-component that implements the particular role, and (iii) optionally an AO-composition for weaving the aspect-component into the internal processes and data of the organization (see Listing 2).

Listing 2: Example of a feature implementation mapping.

```
featureImplementationMapping PPImpl {
 implements: PrioritizedProcessing;
 role: ServiceConsumer;
 ao−component: PPAOComponent;
 ao−composition {
   ...
 }
}
```

3.4 Using the Feature Ontology for Cross-Organizational Service Customization

The Web Services Description Language (WSDL) is an XML-based language that provides a model for describing web services. The web service is defined by an interface, describing the operations that can be performed and the message types that are required/provided by these operations. The

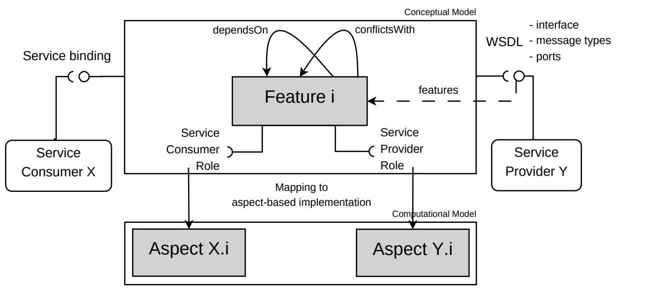

Figure 4: Using the Variability Model for Web Services.

WSDL also defines services as collections of network endpoints, or ports. A port is nothing more than the address or connection point to the web service (typically a http URL). To be able to use our feature ontology, the WSDL should be extended with the set of available features (see Fig. 4).

The variability model is accessible to the clients of the service application and allows them to select a desired set of features. Configuration of features across the service network happens through instantiation of *service bindings.* A service binding is a declarative specification, specifying the web service location, the selected port and which features are desired (see Listing 3).

Listing 3: Example of a service binding.

```
servicebinding {
 URI: http://www.stocktradingexample.be;
 port: StockTradingServiceSoapEndpoint;
 features: PrioritizedProcessing, Billing;
}
```

4. RELATED WORK

We first discuss the work in the context of cross-organizational service provisioning. Next we discuss the related research in the domain of service composition.

Cross-organizational coordination architectures. A multi-layered architecture, distinguishing between policy and mechanism, is a core principle of the body of research on cross-organizational coordination architectures. Firstly, agreements must be represented digitally by means of a language that offers the necessary concepts for describing and enforcing agreements. Second, coordination middleware must be developed in order to establish agreements dynamically, and to enforce the agreements or detect violations against it.

The current state-of-the-art on cross-organizational coordination architectures in the general area of SOA consists of policy-based and contract-based frameworks. Contract frameworks (such as BCL [21], [11], [26], GlueQoS [32], T-BPEL [29] and SLAng [28, 27]) focus mostly on negotiation, enactment and monitoring, while policy-based architectures (e.g. Ponder [7] and LGI [22]) focus exclusively on enforcement. These coordination architectures establish agreements dynamically between two or more organizations, but fail to support the coordination of system-wide customizations of service compositions. Our approach deals with this by providing an aspect-based variability model for cross-organizational features, managed in a multi-layered architecture.

Service composition. Previous research focussed already on automated composition of web services into composite web services [10, 6, 13]. For this purpose, matchmaking algorithms search for matching web services based on their input/output, the interaction protocol and functional behavior, using a forward or backward chaining algorithm and a discovery service. The matchmaking process can be either centralized (i.e. planning a complete composition at once), or decentralized, allowing each web service in the composition to decide individually which web services to select in providing the required services for processing the request. This functional matchmaking process is originally based upon WSDL information in the UDDI directory to select the appropriate services. In more recent work, the matchmaking process is based upon QoS properties of the different web services [32, 33, 34, 16, 4, 8]. Here, non-functional properties such as security, reliability and performance are used by the matchmaking algorithm to select the most appropriate service. For example, in [33], Zheng et al. propose a quality-driven approach to select component services during the execution of a composite service. For this purpose, they define a web service quality model based upon five non-functional properties and a global quality-driven selection algorithm formulating these properties as a linear optimization problem. In this approach, every service is assumed to have one particular QoS profile, described in the quality model. [18] presents an heuristic algorithm for composing services to achieve global QoS requirements in dynamic service environments.

A common denominator in this research domain is the usage of ontologies [3] to store semantic information about web services to automate the matchmaking of services in a web service composition based upon functional and non-functional properties [25, 16, 30]. In our approach, we use ontologies and semantic information to describe features as first class entities rather than describing web services with their properties. In this way, the information about the features is web service independent. Thanks to this ontology, automated reasoning can be done about the customization of the orchestration on a per-request basis, without considering the actual web service composition.

The GlueQoS middleware-based approach of Wohlstadter et al. [32] manages dynamically changing QoS requirements of web services by delaying QoS commitments of the services. Each service describes its QoS preferences, and a middleware-based resolution mechanism searches for a satisfiable set of QoS features to inter-operate for services that encounter each other for the first time. Similar to our approach, GlueQoS uses a fixed ontology for classifying features and describing their interactions and possible interference. However, their selection of features is fully decentralized and on a per-collaboration basis (optimally suited for a highly dynamic web service composition), but lacking support for client-specific customization and consistent processing throughout cross-organizational service compositions, as our approach does.

Finally, our approach does not pretend to replace existing WS-standards such as WS-Coordination [24], WS-Policy [2] and WS-Security [23], but we intend to offer a complementary approach for consistent customization of features in orchestrations. For example, in our approach we use a per-request tagging solution to achieve coordination between the client and the different web services. In case more complex coordination schemes are needed (e.g. if coordination messages don't follow the message flow), our approach can be combined with WS-Coordination. This coordination specification was originally defined for coordinating transaction protocols, but is extensible for all kinds of coordination protocols in a web service environment.

5. CONCLUSION AND FUTURE WORK

In this paper, we proposed an aspect-based variability model for representing cross-organizational features in service networks. Our approach consists of a multi-layered architecture, mapping a technology-independent feature ontology onto an aspect-based implementation.

The approach supports maintaining the compatibility of feature implementations across a service network of independent organizations. The common feature ontology can also be leveraged to support client-specific customization of cross-organizational features across such service networks. As only limited tests have been performed, further validation and evaluation of our approach are necessary.

6. REFERENCES

[1] Apel, S., Kaestner, C., and Lengauer, C. Research challenges in the tension between features and services. In *SDSOA '08: Proceedings of the 2nd international workshop on Systems development in SOA environments* (New York, NY, USA, 2008), ACM, pp. 53–58.

[2] BEA Systems, IBM, Microsoft Corporation, SAP AG, Sonic Software, and VeriSign. Web Services Policy Framework (WS-Policy). `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-polfram/ws-policy-2006-03-01.pdf`, March 2006.

[3] Berners-Lee, T., Hendler, J., and Lassila, O. The Semantic Web. *Scientific American 284*, 5 (2001), 34–43.

[4] Bilgin, A. S. A DAML-based repository for qos-aware semantic web service selection. In *IEEE International Conference on Web Services (ICWS 2004)* (2004), IEEE Computer Society, pp. 368–375.

[5] Cohen, S., and Krut, R., Eds. *Proceedings of the First Workshop on Service-Oriented Architectures and Software Product Lines* (May 2008), Carnegie Mellon University - Software Engineering Institute.

[6] Constantinescu, I., Faltings, B., and Binder, W. Large scale, type-compatible service composition. In *IEEE International Conference on Web Services (ICWS 2004)* (2004), pp. 506–513.

[7] Damianou, N., Dulay, N., Lupu, E., and Sloman, M. The ponder policy specification language. In *Policies for Distributed Systems and Networks* (2001), Springer, pp. 18–38.

[8] Felfernig, A., Friedrich, G., Jannach, D., and Zanker, M. Semantic configuration web services in

the cawicoms project. In *ISWC '02: First International Semantic Web Conference on The Semantic Web* (2002), Springer, pp. 192–205.

[9] Filman, R. E., Elrad, T., Clarke, S., and Akşit, M. *Aspect-Oriented Software Development.* Addison-Wesley, Boston, 2004.

[10] Foster, H., Uchitel, S., Magee, J., and Kramer, J. Compatibility Verification for Web Service Choreography. In *IEEE International Conference on Web Services (ICWS 2004)* (2004), IEEE, pp. 738–741.

[11] Hoffner, Y., Field, S., Grefen, P., and Ludwig, H. Contract-driven creation and operation of virtual enterprises. *Computer Networks 37*, 2 (2001), 111–136. Electronic Business Systems.

[12] Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., and Peterson, A. S. Feature-oriented domain analysis (FODA) feasibility study. Tech. Rep. 21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1990.

[13] Lassila, O., and Dixit, S. Interleaving discovery and composition for simple workflows. In *Semantic Web Services, 2004 AAAI Spring Symposium Series* (2004).

[14] Lee, J., Muthig, D., and Naab, M. An Approach for Developing Service Oriented Product Lines. In *SPLC '08: 12th International Software Product Line Conference* (Sept. 2008), pp. 275–284.

[15] Lee, K., Kang, K. C., Kim, M., and Park, S. Combining feature-oriented analysis and aspect-oriented programming for product line asset development. In *Software Product Line Conference, 2006 10th International* (0-0 2006), pp. 10–112.

[16] Lee, Y., Patel, C., Chun, S. A., and Geller, J. Towards intelligent Web services for automating medical service composition. In *IEEE International Conference on Web Services (ICWS 2004)* (2004), pp. 384–394.

[17] Loughran, N., and Rashid, A. Framed Aspects: Supporting Variability and Configurability for AOP. In *Software Reuse: Methods, Techniques and Tools* (2004), Springer, pp. 127–140.

[18] Mabrouk, N. B., Beauche, S., Kuznetsova, E., Georgantas, N., and Issarny, V. QoS-aware service composition in dynamic service oriented environments. In *Middleware '09: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware* (New York, NY, USA, 2009), Springer-Verlag New York, Inc., pp. 123–142.

[19] Medeiros, F. M., de Almeida, E. S., and de Lemos Meira, S. R. Towards an Approach for Service-Oriented Product Line Architectures. In *Proceedings of the Third Workshop on Service-Oriented Architectures and Software Product Lines (SOAPL)* (2009), S. Cohen and R. Krut, Eds., pp. 151–164.

[20] Mezini, M., and Ostermann, K. Variability management with feature-oriented programming and aspects. In *SIGSOFT '04/FSE-12: Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering* (New York, NY, USA, 2004), ACM, pp. 127–136.

[21] Milosevic, Z., Linington, P. F., Gibson, S., Kulkarni, S., and Cole, J. Inter-Organisational Collaborations Supported by E-Contracts. In *Building the E-Service Society* (2004), Springer, pp. 413–429.

[22] Minsky, N. H., and Ungureanu, V. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *ACM Trans. Softw. Eng. Methodol. 9*, 3 (2000), 273–305.

[23] OASIS Web Services Security (WSS) TC. Web Services Security (WS-Security). `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss`, February 2006.

[24] OASIS Web Services Transaction (WS-TX) TC. Web Services Coordination (WS-Coordination). `http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-os.pdf`, February 2009.

[25] Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. Semantic matching of web services capabilities. In *ISWC '02: First International Semantic Web Conference on The Semantic Web* (2002), Springer, pp. 333–347.

[26] Shrivastava, S. Tapas final report. Tech. rep., Technical Report Project deliverable D20, 2005.

[27] Skene, J., and Emmerich, W. Engineering Runtime Requirements-Monitoring Systems Using MDA Technologies. In *Trustworthy Global Computing (TGC)* (2005), vol. 3705, Springer, pp. 319–333.

[28] Skene, J., Lamanna, D. D., and Emmerich, W. Precise Service Level Agreements. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 179–188.

[29] Tai, S., Mikalsen, T., Wohlstadter, E., Desai, N., and Rouvellou, I. Transaction policies for service-oriented computing. *Data & Knowledge Engineering 51*, 1 (2004), 59–79.

[30] Trastour, D., Bartolini, C., and Gonzalez-castillo, J. A Semantic Web Approach to Service Description for Matchmaking of Services. In *In Proceedings of the International Semantic Web Working Symposium (SWWS)* (2001).

[31] Truyen, E., and Joosen, W. A reference model for cross-organizational coordination architectures. In *International Conference on Enterprise Distributed Object Computing Workshops* (2008), IEEE, pp. 252–263.

[32] Wohlstadter, E., Tai, S., Mikalsen, T., Rouvellou, I., and Devanbu, P. GlueQoS: Middleware to Sweeten Quality-of-Service Policy Interactions. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 189–199.

[33] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., and Sheng, Q. Z. Quality driven web services composition. In *WWW '03: Proceedings of the 12th international conference on World Wide Web* (New York, NY, USA, 2003), ACM, pp. 411–421.

[34] Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., and Chang, H. QoS-aware middleware for Web services composition. *IEEE Transactions on Software Engineering 30* (2004), 311–327.