Evaluation of Quality of Design for Document-Centric Software Services

George Feuerlicht[1,2]

[1] Department of Information Technology,
University of Economics, Prague, W. Churchill Sq. 4, Prague, Czech Republic
[2] Faculty of Engineering and Information Technology,
University of Technology, Sydney,
P.O. Box 123 Broadway, Sydney, NSW 2007, Australia
george.feuerlicht.uts.edu.au

Abstract. As the size and complexity of service oriented applications increases ensuring the quality of design of services that constitute these applications is becoming critical. Poor design of services results in unnecessarily complex and inflexible applications that are difficult to maintain and evolve. Service design has been the subject of intense research interest for almost a decade and there is a wide agreement about the key service design principles that promote maintainability of software services. Recent research efforts include attempts to develop reliable metrics for assessing design quality of service-oriented applications. Most of these metrics were adapted from metrics for object-oriented software and focus on measuring intra-service cohesion and inter-service coupling. In this paper we argue that such metrics are of limited use in assessing the quality of coarse-grained document-centric services used in majority of SOA applications and propose a Message Data Coupling Index (MDCI) - a metric that evaluates orthogonality of a family of XML schemas based on the level of data coupling. We describe the implementation of a prototype tool that computes several variants of the MDCI metric.

Keywords: service design metrics, XML schema evolution, data coupling.

1 Introduction

Service oriented computing has emerged as an active research area more than a decade ago and has now reached high level of maturity with extensive range of technologies available for the construction of complex application systems based on the SOA (Service Oriented Architecture) paradigm. Today, it is not uncommon for such applications to consist of hundreds, and in some cases thousands of individual services that support complex business processes and involve multiple business partners. As the complexity of SOA applications grows it is becoming imperative that software services can be maintained and evolved without costly and time-consuming software modifications often exacerbated by poor design. It is widely accepted that software maintenance accounts for a significant part overall costs and time spent on

A. Ghose et al. (Eds.): ICSOC 2012, LNCS 7759, pp. 356–367, 2013.

software projects [1]. Predictions of software maintainability during the design stage of SDLC (Software Development Life Cycle) and early rectification of design defects can lead to a significant reduction of maintenance costs [1]. Maintainability is closely related to software quality as measured by structural properties of software (size, complexity, cohesion and coupling) [2-4]. Design metrics based on measurements of cohesion and coupling have been used extensively in object-oriented software development and were recently adapted for service-oriented software [5]. However, while there are many similarities between object-oriented software and software services, there are also significant differences that make it difficult to apply similar metrics to both approaches. Service-oriented applications typically use coarse-grained document-centric services that lack important characteristics normally associated with software components limiting the potential for reuse and making metrics that measure inter-service coupling inapplicable. Another significant difference is that service interfaces are often based on pre-defined XML (message) schemas developed by various consortia and standards organizations, or internally as organization-wide standards. For example, travel domain web services are based on the OTA (Open Travel Alliance) specification [6] that defines the structure of message payloads that are used to implement travel applications. Changing requirements result in the need to modify message schemas with a corresponding impact on existing services, and related applications and databases. It follows that the problem of maintenance and evolution of SOA applications is closely related to the problem of schema evolution, and that the quality of design of message schemas determines the impact of requirements changes on services. This situation is illustrated in Figure 1 that shows the various layers involved in the implementation of service-oriented applications. The top layer is constituted by the Domain Data Model that represents information requirements for a particular domain of interest and can be expressed as UML class diagram. In practice, the Domain Data Model may need to be retrofitted from the underlying XML schemas and this presents a number of technical challenges as the schemas may overlap and contain inconsistencies [7]. A family of standard XML schemas that define the message structures used for business interactions forms the Message Schemas layer. Services layer represents the services that implement business transactions such as travel booking. For example, OTA airline ticket booking request/response dialog uses the message pair OTA_AirBookRQ/RS to implement flight bookings. This message interchange constitutes a service where the request and response messages form the service interface:

```
air_booking(OTA_AirBookRQ, OTA_AirBookRS)
```

The Applications layer consists of SOA applications that are compositions of individual services and implement high-level business functions (e.g. booking a multiple-segment flight). Finally, the Database layer represents databases that store data records generated by business transactions.

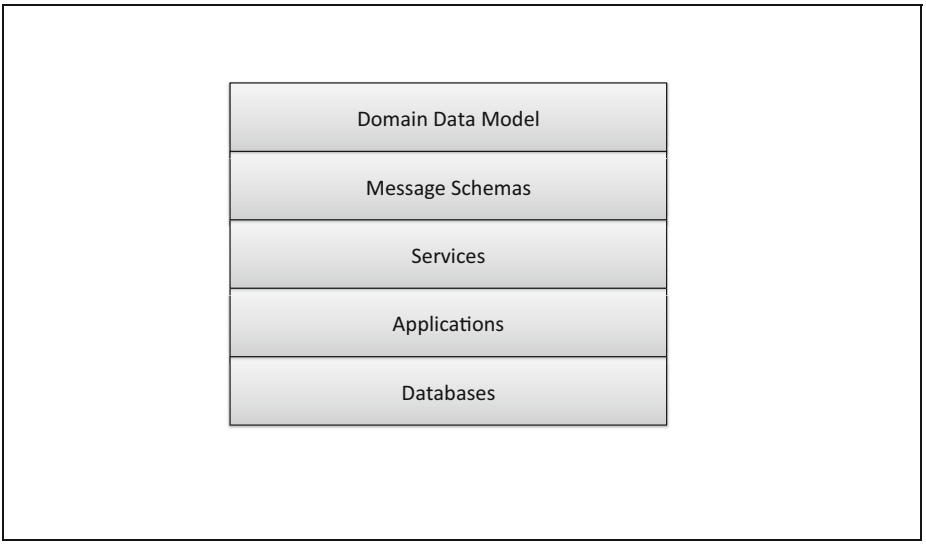

Fig. 1. Layered model of service-oriented applications

As services are implemented on top of pre-existing XML message schemas the design of the schemas determines the structural properties of services and hence their maintainability. Design of XML schemas usually follows document engineering [8-10] or similar methodology that produces XML documents by identifying and aggregating common data elements [11]. For example, OTA message level schemas that represent business transactions are hierarchical collection of XML schemas constructed by aggregation of simple (OpenTravel Simple Types) and complex (OpenTravel Common Types, and Industry Common Types) schema elements [6]. This design approach while ensuring uniform structure and semantics of data elements results in overlapping message schemas and high levels of data coupling reducing maintainability of services [12], [3].

Predicting maintainability of SOA applications has been the subject of recent research interest [1, 5, 13], however the proposed metrics assess structural properties of services at the Service layer and are of limited use in assessing the quality of coarse-grained document-centric services that are used extensively in SOA applications. In this paper we propose a service design metric - MDCI (Message Data Coupling Index) that evaluates orthogonality of a family of XML schemas by estimating the level of data coupling. We describe the implementation of prototype tool that computes the MDCI metric by matching complex schema elements across a family of XML schemas. We first review related literature dealing with service design metrics (section 2.1) and XML schema metrics and change management techniques (section 2.2). In section 3 we describe our proposal for the MDCI metric (section 3.1) and the implementation of a MDCI prototype tool (section 3.2). Section 4 presents our conclusions and outlines further work.

2 Related Work

Metrics for evaluating the quality of service design are essential to allow the comparison of different design strategies and evaluation of their impact on service maintainability. Maintainability of software systems has been the subject of extensive research in the area of object-oriented software and more recently in the context of SOA. Maintainability is closely related to software quality, and a key determinant of software quality is orthogonality of software components. Orthogonality allows individual components to be maintained independently without undesirable side effects (i.e. without impacting on other components) and is achieved by maximizing cohesion and minimizing coupling during the design phase of SDLC [14, 15]. Both coupling and cohesions have been used in traditional software design as indicators of software quality [16], [17, 18]. Maximizing intra-service cohesion and minimizing inter-service coupling improves stability of service-oriented applications by reducing the impact of changes, and at the same time increases potential for reuse [19, 20]. In the following section we review efforts to develop service design metrics that attempt to measure quality of service design (section 2.1), and then discuss XML schema metrics and change management techniques (section 2.2). In the context of the layered model in Figure 1, service design metrics operate at the Services layer and XML metrics operate at the Message Schema layer.

2.1 Service Design Metrics

While the underlying principles of service design are extensively documented in the literature, relatively little work has been done so far on how the adherence to these principles may be quantitatively measured [5]. The original work by Chidamber et al. who proposed the Lack of Cohesion in Methods (LCOM) metric for object-oriented software has been used as the basis for developing metrics for software services. LCOM evaluates similarity of methods for a given class by counting the number of method pairs whose similarity is zero minus the number of method pairs whose similarity is not zero, where similarity of methods is defined as the intersection of the sets of instance variables used by the methods [21]. Several authors have proposed service design metrics based on LCOM. For example, the Service Interface Data Cohesion metric (SIDC) proposed by Perepletchikov et al. [13] measures cohesion by comparing the messages of service operations based on data types. Sindhgatta et al. developed a comprehensive set of metrics to measure service cohesion, coupling, and composability and applied these metrics to case studies in order to evaluate their applicability to practical SOA scenarios [5]. Two variants of the LCOM metric ($LCOS_1$, $LCOS_2$) for use with services were adapted, and several additional metrics have been proposed by the authors in order to evaluate service and message coupling. These metrics include: Service Operational Coupling Index (SOCI) - a measure of dependence of a service on the operations of other services, Inter-Service Coupling Index (ISCI) – based on the number of services invoked by a given service, and Service Message Coupling Index (SMCI) that measures the dependence of a service on the messages derived from the information model of the domain (i.e. messages that

service operations receive as inputs, and produce as output via the declared interface). Finally, Sindhgatta et al. propose several metrics dealing directly with service reuse and composability, including Service Reuse Index (SRI), based on the number of existing consumers of a service, Operation Reuse Index (ORI) that counts the number of consumers of a given operation, and Service Composability Index (SCOMP) defined on the basis of the number of compositions in which the service is a (composition) participant and the number of distinct composition participants which succeed or precede the service. Service granularity is closely related to reuse and composability and is evaluated using Service Capability Granularity (SCG) and Service Data Granularity (SDG) metrics, where higher values indicate coarser granularity (i.e. larger functional scope).

As the above design metrics are based on metrics for object-oriented software, the underlying assumption is that the service model consists of a set of services S= [s_1, s_2…s_S] and that each service has a set of operations O(s) = [o_1, o_2,….o_O] with interfaces formed by input and output messages M(o) [5]. However, most SOA applications use coarse-grained (document-centric) services that implement the request/response message exchange pattern and do not involve service operations, making such metrics not applicable.

2.2 XML Schema Metrics and Change Management Techniques

Another direction of research of relevance to assessing the quality of service design is work that focuses on evaluating the design of XML schemas [1, 22, 23]. As discussed in section 1 above, pre-existing XML message schemas define the structure of service message payloads, and consequently the structural properties of services. As the message schemas constitute an *interface contract*, poor schema design can affect the overall quality of the software system. Numerous XML schema quality metrics have been proposed primarily with the objective to measure various aspects of schema complexity. McDowell et al. proposed metrics based on counts of complex type declarations, derived complex types, number of global type declarations, etc. [22]. Basci et al. proposed and validated XML schema complexity metric that evaluates the internal structure of XML documents taking into account various sources of complexity that include recursion and complexity arising from importing external schema elements.

An alternative perspective on the problem of maintainability of SOA applications is to focus on change management; i.e. rather than attempting to predict maintainability by assessing the quality of XML schema design, such techniques alleviate the impact of changes in requirements by providing tools for automating change management. The problem of XML schema change management (i.e. schema evolution) has been investigated, but not adequately solved [23]. XML schema evolution in the context of SOA presents a particularly difficult problem as the schemas are often developed in the absence of a Domain Data Model, and are characterized by complex and overlapping data structures [22]. Current work in this area focuses on identifying the impact of changes on XML schemas and developing methods and tools for automating the propagation of these changes. Necasky, et al. proposed a five-level XML evolution architecture with the top level Platform-Independent Model (PIM) that represents the data requirements for a particular

domain of interest. PIM model is mapped into a Platform-Specific Model (PSM) that describes how parts of the PIM schema are represented in XML. PSM then maps into Schema, Operational and Extensional level models. Atomic operations (create, update, and remove) for editing schemas are defined on classes, attributes, and associations, and a mechanism for propagating these operations from PIM to PSM schema proposed. Composite operations are constructed from atomic operations to implement complex schema changes [7, 24, 25].

3 Proposed Message Data Coupling Index Metric

In our earlier work we proposed a service design metric that evaluates the quality of service design by estimating the level of data coupling between services (i.e. at the Services layer) [26]. Given a family of XML schemas consisting of pairs of request (RQ) and response (RS) messages [m_{rq}, m_{rs}] that constitute interfaces [si_1, si_2, si_3, ….si_n] for services [s_1, s_2…s_n], DCI (Data Coupling Index) is defined as the average number of (complex) schema elements that are shared between the service interfaces.

In this paper we describe a MDCI (Message Data Coupling Index) metric that evaluates orthogonality of a family of XML schemas that represent a domain of interest (e.g. travel) directly at the Message Schema layer. The rationale for the metric is that maintenance effort required to accommodate new requirements can be estimated based on the level of overlap (lack of orthogonality) of message schemas; i.e. schemas that share a large number of complex data elements are more likely to be impacted as requirements evolve. Conversely, impact of change in requirements on a family of orthogonal schemas is likely to be limited. We rely on data coupling, estimated by counting the number of shared complex elements, as an indirect measure of orthogonality.

Complex elements (composite XML structures with multiple levels of nesting) are used extensively in XML schema specifications. For example, the OTA TravelPreferences complex element type consists of eleven simple elements with two levels of nesting and is embedded in two parent complex element types (AirSearch and OriginDestinationInformation) and three top-level OTA messages (OTA_AirAvailRQ, OTA_AirFareDisplayRQ, and OTA_AirLowFareSearchRQ). Changes in the TravelPreferences element type as the specification evolves will affect all parent data types and messages, with corresponding impact on existing applications and underlying databases.

Coupling through sharing complex schema elements is known in software engineering literature as *stamp coupling* and is regarded as undesirable as it inhibits evolution [17, 27]. Avoiding stamp coupling increases reusability by creating services with simpler interfaces and greater reuse potential [28-30].

3.1 Message Data Coupling Index (MDCI)

Given a family of XML message schemas that represent a domain of interest [m_1, m_2, m_3, .. m_N] MDCI (Message Data Coupling Index) is defined as the sum of the

number of shared schema elements for each message pair combination (i.e. cardinality of the intersection of schema elements) divided by the number of message pair combinations r:

$$\text{MDCI} = \frac{1}{2r}\sum_{j,k=1}^{N} |M_j \cap M_k| \text{ ; where } j \neq k, \text{ and } r = \sum_{i=1}^{N-1} i$$

The calculation of MDCI is based on matching complex schema elements only (i.e. it is a measure of stamp coupling) and gives the average number of shared complex elements evaluated across all message combinations. This is consistent with the approach adopted by McDowell et al. and Visser et al. [22, 23].

3.2 Prototype Tool for Evaluating MDCI

In this section we describe the implementation of a prototype tool for the evaluation of the MDCI metric developed using the Rich Ajax Platform (http://www.eclipse.org/rap/). Given a set of XML messages the MDCI tool reads the message level elements types and the types included in the corresponding libraries. For each message pair combination the tool identifies all complex element types and attempts to match these types producing an average count of matches across all message combinations.

Evaluation of the MDCI metric involves a number of important decisions. Firstly, a decision needs to be made about what constitutes a match between data elements. As noted above we restrict matching to complex element types, i.e. match occurs if two messages that are being compared contain the same complex element type. It is common practice to use extension of complex element types, i.e. complex types derived from other complex types by adding additional elements. For example, as shown in Figure 2 OTA SpecificFlightInfo extends SpecificFlightInfoType by adding elements FlightNo, Airline, and BookingClassPref.

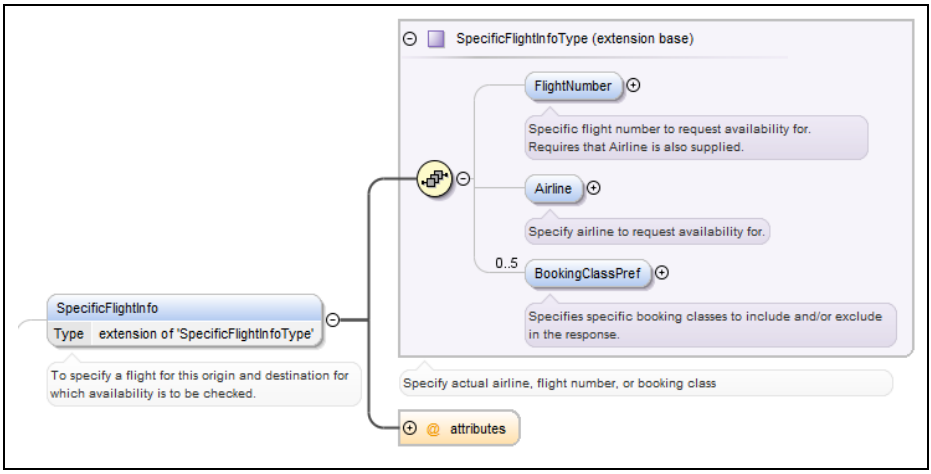

Fig. 2. Extended complex element types

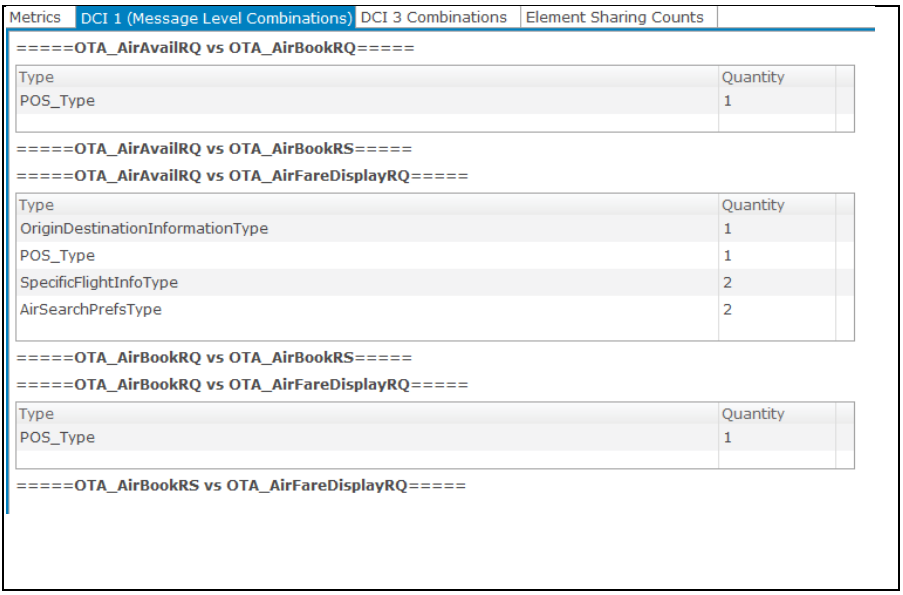

Fig. 3. Multiple element type matches

Computation of MDCI counts the matches between complex elements and their extended versions, as type extensions result in data coupling. As message schemas are constructed by assembling common element types into multi-level hierarchical XML structures, element type matches can occur at different levels of the hierarchy.

We consider two alternatives for the computation of the MDCI index: 1) $MDCI_1$ - counts element matches only at the top-level of the schema, i.e. message level, indicated by the prefix OTA, 2) $MDCI_2$ - counts element matches for all levels of the schema hierarchy. Finally, it is possible for element matches to occur several times for the same message pair as illustrated in Figure 3. The OTA_AirAvailRQ and OTA_AirFareDisplayRQ messages exhibit multiple matches for SpecificFlightInfoType and AirSearchPrefType, indicating that these complex types occur twice in one of the messages. The MDCI tool provides an option to count only unique matches for both $MDCI_1$ and $MDCI_2$ metrics.

3.3 Example Calculations of MDCI

We use a subset of OTA Airline (Air) message schemas (Flattened OTA Schemas version 2011b) shown in Table 1 for the computation of the MDCI index in this paper, but the prototype tool can use message schemas from other sources.

The OTA Air messages implement various business functions related to airline travel, such as checking flight availability, flight booking, etc. For example, the Search and Availability of flights business function is implemented using the Air_AvailabilityRQ/RS request/response message pair [31]. OTA defines common data types (OTA_AirCommonTypes) for the airline messages that form a repository

Table 1. Subset of OpenTravel Air Messages used for calculation of MDCI

Ident.	OpenTravel Message	Business Functionality
AIR01	OTA_AirAvailRQ/RS	Search & Availability
AIR02	OTA_AirBookRQ/RS	Reservation Management: Booking
AIR03	OTA_AirBookModifyRQ	Reservation Management: Modification
AIR04	OTA_AirCheckInRQ/RS	Passenger Check-in & Check-out
AIR05	OTA_AirDemandTicketRQ/RS	Ticket Fulfillment
AIR06	OTA_AirDetailsRQ/RS	Descriptive Information: Flight leg and Codeshare
AIR07	OTA_AirFareDisplayRQ/RS	Fare Search & Display (No Availability)
AIR08	OTA_AirFlifoRQ/RS	Descriptive Information: Flight Operation
AIR09	OTA_AirPriceRQ/RS	Fare Pricing
AIR10	OTA_AirRulesRQ/RS	Fares Rules: Fare Basis & Negotiated Fares
AIR11	OTA_AirScheduleRQ/RS	Descriptive Information: Flight Schedules
AIR12	OTA_AirSeatMapRQ/RS	Seat Availability & Information

of reusable XML Schema components used in the construction OTA Air messages. OTA differentiates between *complex types* (types that contain multiple data elements) and *simple types* (types that contain a single data element).

Using the OTA (Flattened) Air message schemas in Table 1 the value for $MDCI_1$ = 1.51, and 1.18 for multiple and unique element matches, respectively; i.e. on average the selected OTA Air messages share 1.51 complex elements at the top message schema level, and 1.18 complex elements if only unique matches are counted. Corresponding values for $MDCI_2$ (i.e. counting complex element matches for all levels of the message schema hierarchy) are 77.42 and 6.97 for multiple and unique element matches, respectively.

4 Conclusions and Further Work

Standardized XML schemas that define message structures for domain-specific services form the basis for large-scale SOA applications. Complex hierarchical message schemas with overlapping structures that characterize document-centric services result in applications that are difficult maintain and evolve. Reliable metrics that can identify poor design early during system development can significantly reduce maintenance costs.

We have briefly reviewed existing service design metrics and identified their limitations in the context of coarse-grained document-centric services. Following on from our previous proposal of a service design metric that evaluates data coupling between service interfaces [26] we propose a design metric that estimates

orthogonality of a family of message schemas that typically form the basis for the implementation of services for a particular domain of interest (e.g. travel). We argue that the maintenance effort required to accommodate new requirements increases with the extent of overlap (i.e. lack of orthogonality) of message schemas. The proposed MDCI metric relies on evaluating the level of data coupling by counting the number of shared complex schema elements among a set of message schemas. We have described a prototype tool that uses a set of XML schemas as input and provides a number of options for the evaluation of the MDCI metric.

The MDCI metric needs to be empirically validated using different sets of XML schemas and for different design strategies to establish the reliability of MDCI as a measure of design quality. We are currently investigating the of impact service granularity on the orthogonality of the message schemas. We expect that re-designing the schemas to reduce granularity of services will improve the orthogonality of the message schemas and produce a measurable effect on the MDCI index.

Acknowledgments. This research was supported by GAČR (Grant Agency, Czech Republic) grant No. P403/11/0574 and P403/10/0092. Dr George Feuerlicht was supported by the Australian-China Science and Research Fund ACSCRF-01280 from the Australian Department of Innovation, Industry, Science, Research and Tertiary Education (DIISRTE) and the Research Centre for Human Centered Technology Design at the University of Technology, Sydney. We acknowledge the assistance of Enrico Shi with the development of the MDCI prototype tool.

References

1. Basci, D., Misra, S.: Measuring and evaluating a design complexity metric for XML schema documents. Journal of Information Science and Engineering 25(5), 1405–1425 (2009)
2. Bansiya, J., Davis, C.G.: A hierarchical model for object-oriented design quality assessment. IEEE Transactions on Software Engineering 28(1), 4–17 (2002)
3. Etzkorn, L.H., et al.: A comparison of cohesion metrics for object-oriented systems. Information and Software Technology 46(10), 677–687 (2004)
4. Eder, J., Kappel, G., Schrefl, M.: Coupling and cohesion in object-oriented systems. Technical Report, University of Klagenfurt, Austria (1994)
5. Sindhgatta, R., Sengupta, B., Ponnalagu, K.: Measuring the Quality of Service Oriented Design. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 485–499. Springer, Heidelberg (2009)
6. OTA. OTA Specifications (May 6, 2010), `http://www.opentravel.org/Specifications/Default.aspx`
7. Necaský, M.: Conceptual Modeling for XML. Dissertations in Database and Information Systems Series. IOS Press/AKA Verlag (2009)
8. Glushko, R., McGrath, T.: Document engineering: analyzing and designing documents for business informatics and Web services. MIT Press Books (January 2008)

9. Glushko, R., McGrath, T.: Patterns and reuse in document engineering. In: XML 2002 Proceedings (2002)
10. Glushko, R.J., McGrath, T.: Document Engineering for e-Business. In: Proceedings of the 2002 ACM Symposium on Document Engineering (DocEng 2002), McLean, Virginia, USA. ACM Press, New York (2002)
11. ebXML. ebXML - Enabling A Global Electronic Market (December 9, 2007), http://www.ebxml.org/
12. Feuerlicht, G., Lozina, J.: Understanding Service Reusability. In: 15th International Conference Systems Integration 2007, Prague, Czech Republic. VSE Prague (2007)
13. Perepletchikov, M., Ryan, C., Frampton, K.: Cohesion metrics for predicting maintainability of service-oriented software, pp. 328–335. qsic (2007)
14. Papazoglou, M.P., Yang, J.: Design Methodology for Web Services and Business Processes. In: Buchmann, A.P., Casati, F., Fiege, L., Hsu, M.-C., Shan, M.-C. (eds.) TES 2002. LNCS, vol. 2444, pp. 54–64. Springer, Heidelberg (2002)
15. Papazoglou, M.P., Heuvel, W.V.D.: Service-oriented design and development methodology. International Journal of Web Engineering and Technology 2(4), 412–442 (2006)
16. Vinoski, S.: Old measures for new services. IEEE Internet Computing 9(6), 72–74 (2005)
17. Pautasso, C., Zimmermann, O., Leymann, F.: Restful web services vs. big'web services: making the right architectural decision. In: 17th International Conference on World Wide Web. ACM, Beijing (2008)
18. Pautasso, C., Wilde, E.: Why is the web loosely coupled?: a multi-faceted metric for service design. In: 18th International Conference on World Wide Web. ACM, Madrid (2009)
19. Stevens, W.P., Myers, G.J., Constantine, L.L.: Structured Design. IBM Systems Journal 38(2 & 3) (1999)
20. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: Object-oriented modeling and design 1991. Prentice Hall, New Jersey (2000)
21. Chidamber, S., Kemerer, C.: A metrics suite for object oriented design. IEEE Transactions on Software Engineering 20(6), 476–493 (2002)
22. McDowell, A., Schmidt, C., Yue, K.B.: Analysis and metrics of XML schema (2004)
23. Visser, J.: Structure metrics for XML Schema. In: Proceedings of XATA (2006)
24. Necaský, M., Mlýnková, I.: A Framework for Efficient Design, Maintaining, and Evolution of a System of XML Applications. In: Proceedings of the Databases, Texts, Specifications and Objects, DATESO, vol. 10, pp. 38–49
25. Necaský, M., Mlýnková, I.: Five-Level Multi-Application Schema Evolution. In: Proceedings of the Databases, Texts, Specifications and Objects, DATESO, vol. 9, pp. 213–217
26. Feuerlicht, G.: Simple Metric for Assessing Quality of Service Design. In: Maximilien, E.M., Rossi, G., Yuan, S.-T., Ludwig, H., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6568, pp. 133–143. Springer, Heidelberg (2011)
27. Page-Jones, M.: The Practical Guide to Structured Systems Design, 2nd edn. Prentice Hall, New Jersey (1988)
28. Feuerlicht, G.: Design of Service Interfaces for e-Business Applications using Data Normalization Techniques. Journal of Information Systems and e-Business Management, 1–14 (2005)

29. Feuerlicht, G.: System Development Life-Cycle Support for Service-Oriented Applications. In: 5th International Conference on Software Methodologies, Tools and Techniques, SoMet 2006, Quebec, Canada. IOS Press, The Netherlands (2006)
30. Schmelzer: Solving the service granularity challenge (December 13, 2007), http://searchsoa.techtarget.com/tip/0,289483,sid26_gci1172330,00.html
31. Alliance, O.T.: OpenTravel™ Alliance XML Schema Design Best Practices (September 1, 2010), http://www.opentravel.org/Resources/Uploads/PDF/OTA_SchemaDesignBestPracticesV3.06.pdf